Pattern Recognition Letters 123 (2019) 47–52

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Locally linear embedding with additive noise☆

Justin Wang[a,*], Raymond K.W. Wong[b], Thomas C.M. Lee[a]

[a] University of California, Davis, One Shields Ave, Davis, CA 95616, United States
[b] Texas A&M University, 400 Bizzell Street, College Station, TX 77843, United States

ARTICLE INFO

Article history:
Received 12 October 2017
Available online 8 March 2019

Keywords:
Cross validation
Dimension reduction
Regularization

ABSTRACT

Locally linear embedding (LLE) is a nonlinear dimension reduction technique that only relies on the assumption of local linearity. While it is known to produce good results and is computationally efficient, it does not perform well when the observations are distorted by noises, as the fundamental assumption of local linearity becomes violated. In this work, we present a modification of locally linear embedding which is designed to handle such situations. This new modification is termed LLEAN, short for locally linear embedding with additive noise, which has been seen to perform better in the presence of noise distortion. In LLEAN, we seek to recover the noiseless data from the noisy data by exploiting the relationship between local linearity and reconstruction potential, and we then use the recovered noiseless data while performing the dimension reduction. The LLEAN algorithm includes a tuning parameter, and our work includes an automatic selection method for the tuning parameter to remove the burden from the user.

1. Introduction

Dimension reduction refers to the machine learning problem of extracting a lower dimensional set of features from a higher dimensional dataset. It is often used as a form of preprocessing: by extracting relevant features first the dataset becomes easier to work with. One such common dimension reduction technique is known as principal components analysis (PCA) [12]. PCA is limited in that it requires that the data lie on or near a linear subspace, which is an assumption that is often not satisfied.

When the linearity assumption is not met, we turn to nonlinear dimensional reduction techniques, which do not require the linearity assumption and have been successfully adopted in various applications [e.g., [6,15,22,23]]. Popular nonlinear dimension reduction techniques include Kernel PCA [20], Isomap [21], principle curves [8], deep autoencoders [9], diffusion maps [4], and locally linear embedding [19], just to name a few. Our focus in this paper will be extensions to the latter.

Locally linear embedding (LLE) is a powerful alternative to PCA when the data is nonlinear, but is not effective in the presence of noise. When Gaussian noise is added to each data point, the local linearity assumption becomes violated, and LLE no longer handles the dimension reduction well. This appears to be the case even when the variance of the noise is relatively small. We remark that there are other nonlinear dimensional reduction techniques that are robust to noise, especially recent developments in deep learning such as Arpit et al. [1], Jiang et al. [11], Ren et al. [17]. In this paper, our focus is LLE. While LLE has advantages in optimization and tuning, a key limitation of LLE is that the underlying manifold is smooth.

Various extensions to the basic LLE algorithm have already been developed. These include Robust Locally Linear Embedding (RLLE), which was developed to handle outlier points [3]; a supervised version of LLE developed to handle classification tasks [18]; a version of LLE based on Hessian eigenmaps to handle high-dimensional data [5]; and an incremental version of LLE to preserve topology [13].

In this paper, we introduce an alternative to LLE called Locally Linear Embedding with Additive Noise (LLEAN), which is designed to handle the case when the data are corrupted by additive noise. This is not to be confused with the setting of the above-mentioned Robust Locally Linear Embedding, in which additional noise/outlier points are added to the data. For our problem, no additional points are added, but rather the original points are just distorted with noise.

In LLEAN, we modify the minimization criterion to consist of a minimization term and a regularization term. As to be shown below, this modification allows the proposed approach to handle noisy observations. The rest of the paper will be organized in the

☆ **Conflict of interest:** None.

* Corresponding author.
E-mail addresses: jstwang@ucdavis.edu (J. Wang), raywong@stat.tamu.edu (R.K.W. Wong), tcmlee@ucdavis.edu (T.C.M. Lee).

https://doi.org/10.1016/j.patrec.2019.02.030
0167-8655/

following manner: we will first provide some background on the LLE algorithm as well as discuss the shortcomings of the LLE algorithm with respect to noise in Section 2. In Section 3, we introduce the LLEAN algorithm, including an algorithm for automatic selection of the tuning parameter. In Section 4, we will present some simulation results and compare our algorithm performance to LLE. Lastly, concluding remarks are offered in Section 5 while technical details are delayed to the appendix.

2. Locally linear embedding

2.1. Introduction

Locally linear embedding (LLE) involves a minimization term that utilizes the linearity of small neighborhoods formed by data points and its nearest neighbors. The minimization has a closed form solution and therefore LLE is very computationally efficient. LLE first computes the optimal weights needed to form the best linear reconstruction of every data points from its nearest K neighbors, and then computes the set of lower dimensional vectors that are best linearly reconstructed from its K neighbors using these optimal weights. It follows that LLE excels at handling data that are locally linear.

2.2. Algorithm: Locally Linear Embedding (LLE)

Let $x_1, \ldots, x_n$ be a set of vectors in a high dimensional space $\mathbb{R}^d$. Locally linear embedding takes this set of vectors and produces a lower-dimensional embedding $y_1, \ldots, y_n$, which lie in a lower dimensional space $\mathbb{R}^m$, where $m \ll d$. The algorithm can be summarized in the following three steps:

1. Obtain the set of K nearest neighbors for each x_i. Denote this set as $\mathcal{N}_i$.
2. Obtain the weight matrix $W = (w_{ij})_{i,j=1,\ldots,n}$ that minimizes the following error term:

$$E = \sum_{i=1}^{n} \|x_i - \sum_{j \neq i} w_{ij} x_j\|^2,$$

where $w_{ij} = 0$ if $j \notin \mathcal{N}_i$ and $\sum_{i=1}^{n} w_{ij} = 1$.
3. Obtain the low dimensional embedding $y_1, \ldots, y_n$ by minimizing the following cost function:

$$C = \sum_{i=1}^{n} \|y_i - \sum_{j \neq i} w_{ij} y_j\|^2,$$

where $\sum_{i=1}^{n} y_i = \mathbf{0}$ and $Y^T Y = I_m$. Here $Y = (y_1, \ldots, y_n)^T$.

The minimization in Step 2 can be done through solving a constrained least squares problem. Define the matrix

$$Q_i = (x_i \mathbf{1}^T - N_i)^T (x_i \mathbf{1}^T - N_i),$$

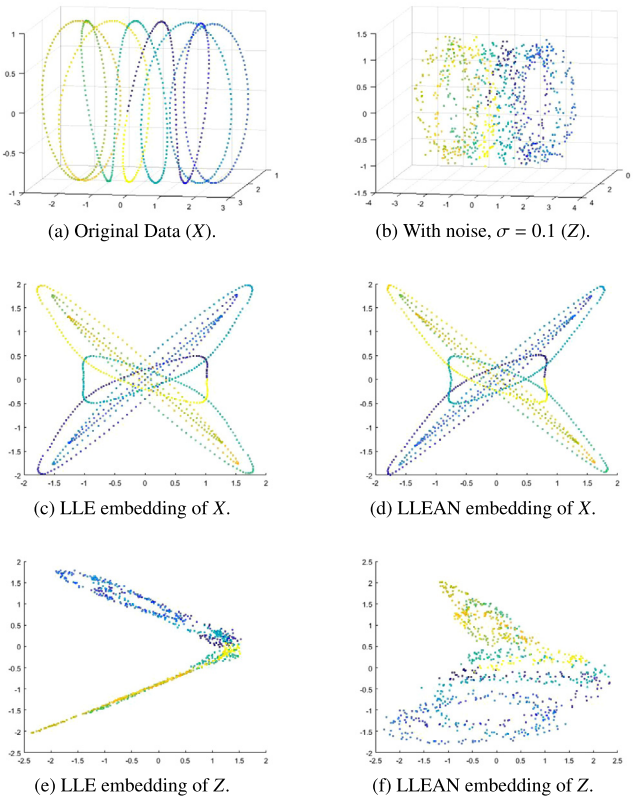
where $\mathbf{1}$ is a column vector of ones and N_i is a $d \times K$ matrix with each of its K columns being a neighbor of x_i. A closed form solution for the weight vector w_i is then given by:

$$w_i = \frac{Q_i^{-1}\mathbf{1}}{\mathbf{1}^T Q_i^{-1} \mathbf{1}}.$$

Computing the above requires inverting Q_i, which may be impractical in very large dimensions. A more efficient solution is to solve the equation $Q_i w_i = \mathbf{1}$, and then normalize the weights to enforce the constraint $\sum_{i=1}^{n} w_{ij} = 1$.

The minimization in Step 3 can be done in the following way. Define the following matrix

$$S = (I - W)^T (I - W).$$



(a) Original Data (X). (b) With noise, $\sigma = 0.1$ (Z). (c) LLE embedding of X. (d) LLEAN embedding of X. (e) LLE embedding of Z. (f) LLEAN embedding of Z.

Fig. 1. 3-dimensional Helix data with noise distortions and 2D embeddings from LLE and LLEAN.

Obtain the $d+1$ eigenvectors of S with the smallest eigenvalues, and then discard the eigenvector with the smallest eigenvalue (this eigenvalue will be zero). The remaining d eigenvectors solve the minimization problem.

2.3. Performance in the presence of noise

As mentioned earlier, locally linear embedding does not perform well in the presence of noise. To illustrate this fact, we generated 800 observations from a helix shaped curve in three dimensions (the "Helix dataset"). We first ran the LLE algorithm on the noiseless data, and then added Gaussian noise to the data (with standard deviation $\sigma = 0.1$), and ran the LLE algorithm on the noisy data. For comparison purposes, we have also ran the LLEAN algorithm on both the non-noisy and noisy datasets as well. The results are illustrated in Fig. 1.

From this figure, we see an improvement in the 2D embedding of the noisy data obtained with LLEAN as compared to LLE. We can see that the points, in particular the purple and blue colored points, are better separated in the former.

3. Local linear embedding with additive noise

3.1. Algorithm: Locally Linear Embedding with Additive Noise (LLEAN)

Let $z_1, \ldots, z_n$ be a set of noisy vectors in high dimensional space $\mathbb{R}^p$. Let $x_1, \ldots, x_n$ denote the underlying (unobserved) noiseless vectors. They are related to $z_1, \ldots, z_n$ in the following way:

$$z_i = x_i + \epsilon_i, \quad i = 1, \ldots, n,$$

where $\epsilon_1, \ldots, \epsilon_n \overset{\text{iid}}{\sim} N_p(0, \sigma^2 I_p)$ represent Gaussian noise.

Notice that the original LLE algorithm applies to the x_i's (which are not observed in the current setting) while the LLEAN algorithm

applies to the z_i's. Another major difference between the LLEAN algorithm and the LLE algorithm is the modification of the minimization criterion in Step 2 of LLE. The LLEAN algorithm is as follows.

1. Obtain the set of K nearest neighbors for each z_i. Denote this set as $\mathcal{N}_i$.
2. Obtain the weight matrix $W = (w_{ij})_{i,j=1,\ldots,n}$ and data matrix $X = (x_1, \ldots, x_n)$ that minimizes the following error term:

$$E = ||X - WX||_F^2 + \frac{1}{\lambda}||Z - X||_F^2,$$

where $w_i = 0$ if $j \notin \mathcal{N}_i$ and $\sum_{i=1}^n w_{ij} = 1$, $||\cdot||_F$ represents the Frobenius norm, and $\lambda > 0$ is a tuning parameter.
3. Obtain the low dimensional embedding $y_1, \ldots, y_n$ by minimizing the following cost function:

$$C = \sum_{i=1}^{n} ||y_i - \sum_{j \neq i} w_{ij} y_j||^2,$$

where $\sum_{i=1}^n y_i = \mathbf{0}$ and $Y^T Y = I_m$. Here $Y = (y_1, \ldots, y_n)^T$.

The minimization in Step 3 is identical to the minimization in Step 3 of LLE. The minimization in Step 2 may be done through block coordinate descent [16]. We will iterate between minimizing W while keeping X fixed, and vice versa. The minimization of W with fixed X is similar to the minimization of W in LLE. For the minimization of X with fixed W, it can be shown that the minimizer is

$$X_{\min} = \left\{\lambda(I_n - W)^T(I_n - W) + I_n\right\}^{-1} Z, \tag{1}$$

where I_n represents an n x n identity matrix. With this explicit expression, the overall minimization of Step 2 is relatively fast. The derivation of $X_{\min}$ will be relegated to Appendix A.

3.2. Intuition

The minimization term of LLEAN resembles that of regularized linear regression, in that it includes a minimization component $\frac{1}{\lambda}||Z - X||_F^2$ and a regularization term $||X - WX||_F^2$. Unlike regularized linear regression, however, we view the regularization term here as the primary minimization criterion and the minimization component as the constraint. In other words, in LLEAN, we are selecting a data configuration X^* along with a corresponding weight matrix W that has the best *overall reconstruction potential*, in that it is the configuration in which its observations are best overall able to be reconstructed by its K neighbors. As the overall reconstruction is closely linked to local linearity, we are effectively attempting to find the data configuration X^* that is the most locally linear among all candidate solutions. The constraint $\frac{1}{\lambda}||Z - X||_F^2$ is imposed to ensure that X^* does not stray too far from the original data Z.

The parameter λ controls how tightly the constraint is enforced. The smaller the value of λ, the more weight we are putting on the constraint term, meaning that the search space for the optimal solution will be restricted to a smaller neighborhood near Z.

Once we have selected X^* in Step 2 of the LLEAN algorithm, we proceed by computing the optimal lower dimensional embedding in the same manner as in LLE. Thus, the primary purpose of the LLEAN is to attempt to recover a data configuration that is close to the original noiseless data X from the observed noisy data Z. Since Z is seen to be less locally linear than X due to the presence of noise, we attempt to recover the original noiseless data X by selecting a data configuration X^* near Z that is more locally linear, and thus, closer to X.

3.3. Choosing a parameter automatically

The tuning parameter λ will need to be chosen. A smaller value of λ penalizes solutions that are further away (in terms of Frobenius norm) from the actual data. Here we present an algorithm to automatically choose the value of λ.

The algorithm is based on cross validation. The idea is to calculate a cross validation score for a candidate parameter λ_j by computing the distance between the original data Z and a reconstruction $\hat{X}$ based on λ_j and cross validation. Each row $\hat{X}_i$ of this reconstruction is obtained by first applying the LLEAN algorithm to Z with the i^{th} observation removed, and then extracting the K nearest neighbors of the removed observation from the resulting X^* and then computing their mean. The candidate parameter with the lowest CV score will then be chosen.

3.4. Algorithm: Automatic Parameter Selection (APS)

Choose a set $\Lambda = \{\lambda_1, \ldots, \lambda_q\}$ to test. Do the following for each λ_j, $j = 1, \ldots, q$:

Repeat Steps 1 and 2 for $i = 1, \ldots, n$:

Let z_i denote the i^{th} row (observation) of the data matrix Z. Remove z_i from Z and denote the resulting $(n-1) \times p$ matrix as Z_{-i}. Apply the LLEAN algorithm with parameter λ_j to Z_{-i}, stopping at the second step where the optimal X is obtained. Denote this result as $\hat{X}_{-i}$.

1. Find the K nearest neighbors of z_i. Extract the rows of $\hat{X}_{-i}$ corresponding to these K neighbors, and denote them as $n_1(\hat{X}_{-i}), \ldots, n_K(\hat{X}_{-i})$. Then define the following

$$\hat{X}_i = \sum_{j=1}^{K} n_j(\hat{X}_{-i}).$$

2. Let $\hat{X}^{(j)}$ denote the matrix whose i^{th} row is $\hat{X}_i$, $i = 1, \ldots, n$. Define the following quantity:

$$CV(\lambda_j) = \sum_{i=1}^{1000} ||Z - \hat{X}||_F^2.$$

The optimal value λ^* is equal to the λ_j with the lowest value of $CV(\lambda_j)$.

Depending on the size of the data etc, this algorithm could be computationally expensive. If necessary, the following method can be employed to speed up the computation. Instead of repeating Steps 1 and 2 for all n indices, we may choose a random subset of indices $I \subseteq \{1, \ldots, n\}$ and iterate through only those indices. Let $N' < n$ denote the size of the set I. For example, if $N' = 0.5n$, the computational time will be halved. In fact, in practice we have observed that setting $N' = 0.5n$ will approximate the result very well. A successful imaging example of applying this approach for speed gain can be found in Hudson and Lee [10].

4. Simulations and examples

4.1. A new metric to measure performance

In order to quantitatively evaluate the performance of LLEAN against LLE in simulated examples, we have developed a metric for performance measurement. We first define the overall distance between two reduced dimension configurations Y and $Y^{(b)}$ as

$$D\left(Y, Y^{(b)}\right) = \sum_{\text{all pairs i,j}} |\delta\left(Y_i, Y_j\right) - \delta\left(Y_i^{(b)}, Y_j^{(b)}\right)|, \tag{2}$$

where $\delta(\cdot, \cdot)$ is a standard distance measure (e.g., Euclidean) and Y_i refers to the i^{th} row (observation) of matrix Y. It can be shown that this is a distance metric; see Appendix B.

This closeness measure assesses how "close" two reduced dimension configurations are by assessing the discrepancy in distance between every pair of observations. If the measure is large, then this implies that, overall, the distances between pairs are not similar among the two configurations. We would like to note that although there are $\binom{N}{2}$ pairs to evaluate, the computational time required for a reasonable sized N (800–1500) is less than half a minute.

We may then define a metric M that compares the closeness of two reduced dimension configurations $Y^{(1)}$ and $Y^{(2)}$ to a baseline configuration $Y^{(b)}$:

$$M\left(Y^{(1)}, Y^{(2)}, Y^{(b)}\right) = D\left(Y^{(1)}, Y^{(b)}\right) - D\left(Y^{(2)}, Y^{(b)}\right).$$

The metric M compares the closeness of $Y^{(1)}$ and $Y^{(b)}$ to that of $Y^{(2)}$ and $Y^{(b)}$. A negative value for the metric implies that $Y^{(1)}$ is closer to the baseline, while a positive value implies that $Y^{(2)}$ is closer to the baseline. We will take $Y^{(1)}$ and $Y^{(2)}$ to be the LLEAN and LLE reduced configurations on the noisy data, respectively, and $Y^{(b)}$ being the LLE reduced configuration on the non-noisy data. Therefore, a smaller value of the metric indicates that LLEAN preserves pairwise distances from the non-noisy data better than LLE does.

In our following experiments, we take $Y^{(b)}$ to be the result of the LLE algorithm on the non-noisy data X, $Y^{(1)}$ to be the result of the LLEAN algorithm on the noisy data Z, and $Y^{(2)}$ to be the result of the LLE algorithm on the noisy data Z. Therefore, negative values of the metric M imply that results from LLEAN are better when the data are corrupted by noise.

4.2. Helix Data

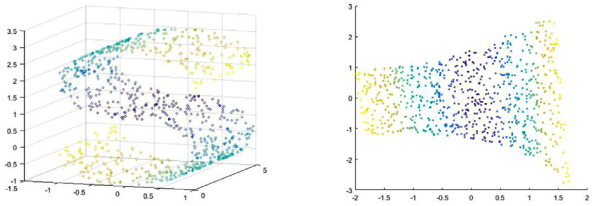
We first tested the performance of LLEAN against LLE using the Helix Dataset. The number of repetitions in was 1000. That is, we generated 1000 datasets a 3-dimensional Helix with $N = 800$. For each dataset, we first ran the LLE algorithm on the non-noisy version of the data, and the resulting output is the baseline configuration $Y^{(b)}$. We then added Gaussian noise to the data with noise standard deviation $\sigma = 0.1$, and ran both LLEAN and LLE on the noisy data, with respective outputs $Y^{(1)}$ and $Y^{(2)}$. Finally, we applied the metric to the results. We repeated this process for all 1000 generated datasets.

For both LLEAN and LLE, we set $K = 15$, a standard choice for the number of neighbors. In addition, we fixed the number of iterations of the block coordinate descent in LLEAN to be 20, as the algorithm has typically been found to have converged by then. We also used the automatic parameter selection method to choose λ, setting $N' = N/2 = 400$.

Finally, we collected the values of the metric for all 1,000 datasets. We then performed a t-test to assess whether the true average value of the metric would be negative, which would provide strong evidence that LLEAN performs better than LLE in the long run. The results are shown in Table 1. We can see that since the value for the test statistic is negative with a low p-value, we can empirically conclude that LLEAN does perform better than LLE in the long run. Note that in the classical theory of statistical hypothesis testing, the p-value is the probability of observing at least as extreme as the experimental outcomes, under the assumption that the null hypothesis is true. Therefore, the smaller the p-value, the more confident we are that the null hypothesis is false.

Table 1
Experimental settings and results for the Helix Data simulation.

Number of repetitions	1,000
Number of observations for each repetition	800
Dimension of the data	3
Number of neighbors K	15
Iterations per LLEAN	20
Standard deviation of the noise	0.1
Number of candidate parameters	16
Test statistic	−2.062
p-value	0.0394



(a) Original Data. (b) 2D Embedding.

Fig. 2. 3-dimensional S Curve data and its 2D embedding using LLE.

Table 2
Experimental settings and results for the S Curve Data simulation.

Number of repetitions	1000
Number of observations for each repetition	800
Dimension of the data	3
Number of neighbors K	15
Iterations per LLEAN	20
Standard deviation of the noise	0.2
Number of candidate parameters	16
Test statistic	−2.771
p-value	0.0058

4.3. S Curve Data

For this experiment, we used the S Curve Dataset. That is, the observations are sampled from an S-shaped curve in three dimensions. A typical data set is shown in Fig. 2. The experimental settings here are essentially the same as for the Helix Dataset simulation, except the standard deviation of the Gaussian noise added to the data was increased to $\sigma = 0.2$. The results for this simulation are reported in Table 2 in a similar manner as before.

Once again, these results strongly suggest that LLEAN is superior to LLE when the observations are corrupted by noise.

4.4. Magic gamma telescope data

The Magic Gamma Telescope Data is Monte Carlo generated data that simulates the registration of gamma particles in a gamma telescope using the imaging technique [2]. For each observation, there are 10 continuous covariates describing various characteristics of the gamma particle. The goal is to predict whether the particle is caused by primary gammas (sigma) or from a hadronic shower (background). Overall, there are 19,020 observations, with 12,332 belonging to the category sigma and the remaining 6688 belonging to the category background.

This dataset is assumed to be noisy data generated from a 10 dimensional manifold. Since non-noisy data is not available, we cannot use the metric from earlier to compare the performance of LLE and LLEAN. Instead, we performed classification with 5-fold cross validation and assessed which method has the overall lower classification error. We first reduced the data into 5 dimensions with $K = 15$ using both LLE and LLEAN (5 iterations on LLEAN). Then each reduced-dimension dataset was divided into 5 equally-sized segments, with one of the segments acting as the test data and the remaining segments acting as the training data. We then trained an artificial neural network on the training data using 10 hidden nodes, a learning rate of 0.05, and 30 iterations. The trained neural network was then fitted on the test data and we obtained

a classification error. This process was repeated 5 times in total, each time with a different segment acting as the test data. Finally, the five resulting classification errors were averaged, and we then compared the performance of LLE and LLEAN on this data based on the classification error of their respective dimension reduced datasets.

Due to the large number of observations, the automatic parameter selection algorithm was not practical here, as it is computationally expensive. Instead, we tested the performance on a list of candidate λ parameters. For all the candidate parameters, we found that the classification error of LLEAN to be lower than that of LLE. The best performing candidate parameter was $\lambda = 1 \times 10^{-6}$, which yielded a 3.8% improvement for the classification error.

4.5. MNIST digit data with additive noise

The MNIST handwritten digit database [14] consists of a set of images of handwritten digits from 0 to 9. Each image is represented as a 28 x 28 matrix with each pixel taking values between 0 and 255. We divided each pixel by 255 to scale the values to lie between 0 and 1. The dimensionality of the data is $28^2 = 784$.

Similarly to the previous example, we will use binary classification of the digit 4 vs. the digit 9 to show the effectiveness of LLEAN compared to LLE. Note that 4 and 9 are two of the most easily confused digits in the database, and that datasets based on MNIST have been specifically created to discriminate between 4's and 9's [7]. To perform our experiment, we took random samples of 4's and 9's from the database and introduced additive noise. An example of a 4 and 9 with additive noise is shown in Fig. 3. We then preprocessed the noisy data using both LLE and LLEAN, and showed that the classification accuracy is higher for the preprocessed data created by LLEAN. More specifically, we used the following process:

1. Take random samples of size 500 each from 4's and 9's.
2. Add a random noise term $\epsilon_{ij}^{(G)} \sim N(0, \sigma^2)$ to each pixel (i, j) of each image G in the above samples.
3. Use LLE to reduce the dimensionality of the data from $n = 784$ to $m = 10$ using $K = 15$. Call the result $\mathrm{D_{LLE}}$.
4. Repeat Step 3 with LLEAN using the same parameters. We ran the algorithm for 3 iterations and found a good performing tuning parameter to be $\lambda = 0.00001$. Call the result $\mathrm{D_{LLEAN}}$.
5. Split $\mathrm{D_{LLE}}$ into 70% training set and 30% testing set. Train random forest with 500 trees and 3 randomly sampled features for each tree on the training set and obtain classification error on the testing set.
6. Repeat Step 5 for $\mathrm{D_{LLEAN}}$.

We repeated the above process 20 times for a total of 10,000 replications each of noisy 4's and noisy 9's. The data was preprocessed and trained on relatively small samples of data at once, as the preprocessing using both LLE and LLEAN becomes exponentially more computationally expensive as the sample size increases. We used the random forest classifier here as it is well-suited to relatively small sample sizes. LLEAN yielded a 5.9% improvement for the classification error over LLE.

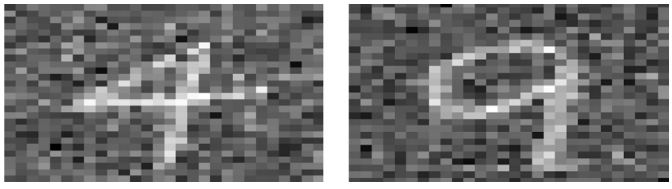


Fig. 3. MNIST 4 and 9 with additive noise.

5. Concluding remarks

Locally linear embedding (LLE) is an effective approach to nonlinear dimension, which only relies on the assumption of local linearity. However, it is limited by its inability to handle data with additive noise. It was observed that even a small amount of noise can drastically alter the results of the dimension reduction step.

In this paper we have introduced an alternative to LLE, LLEAN, which has been seen to have better long run performance than LLE when noise is present in the observations. LLEAN finds the set of vectors X^* with the best "linear reconstruction potential" within a specified neighborhood of the original data Z, and then performs the dimension reduction step of LLE using X^* instead of the original data Z. It is thought that due to the superior locally linear properties of X^* as compared to the original data Z, that the resulting dimension reduction of LLEAN will more closely mirror the dimension reduction of noiseless data than LLE will.

LLEAN also includes a tuning parameter which controls how far from the original data Z the algorithm searches. In order to take the burden of choosing the parameter from the user, we have devised an automatic parameter selection algorithm, which chooses the best parameter from a list based on a criteria, the Cross-Validation (CV) score.

We have included several experiments to present the performance of LLEAN against LLE under noisy data. Through our experiments we have seen that, on average, LLEAN achieves a dimension reduction that more closely mirrors that of noiseless data in the long run.

One limitation of our algorithm is in the computational cost. The block coordinate descent step in the LLEAN algorithm requires around 20 steps before converging to the optimal solution, with each step iterating over each of the N observations in the data. In addition, the automatic parameter selection algorithm requires a significant computational cost when computing the CV scores of each parameter candidate.

Future work will include, first and foremost, a way to reduce the computational cost required for the automatic parameter selection algorithm. While the cost can be reduced significantly by setting $N' = N/2$, it is still significant. Also, while LLEAN has been seen to outperform LLE in the long run, there are still cases where it does not outperform LLE. In the future, we would like to isolate the cases where LLEAN does not perform LLE and figure out under exactly what conditions LLEAN will not outperform LLE.

Acknowledgment

The authors are grateful to the reviewers for their useful comments. This work was supported by the National Science Foundation (Grant nos. DMS-1512945, DMS-1811405 and DMS-1811661).

Appendix A. Derivation of (1)

To derive the minimizer $X_{\min}$, we first rely on the following result.

Proposition 1.

$$||X - WX||_F^2 + \alpha||Z - X||_F^2 = tr\left(Z'Z - \tilde{Z}'\tilde{Z}\right) + ||\tilde{Z} - HX||_F^2,$$

where $\alpha > 0$, $H = [(I_n - W)'(I_n - W) + \alpha I_n]^{1/2}$, *and* $\tilde{Z} = \alpha H^{-1} Z$.

Proof. First note that the left-hand side can be rewritten as follows after expanding out the Frobenius norm:

$$\begin{aligned} &||X - WX||_F^2 + \alpha||Z - X||_F^2 \\ &\quad = (1+\alpha)tr(X'X) + \alpha\left[tr(Z'Z) - 2tr(Z'X)\right] \\ &\quad -2tr(X'WX) + tr\left(X'W'WX\right). \end{aligned}$$

We now show that the right-hand side can be rewritten in the same way:

$$\begin{aligned} &tr\left(Z'Z - \tilde{Z}'\tilde{Z}\right) + ||\tilde{Z} - HX||_F^2 \\ &\quad = tr(Z'Z) - tr\left(\tilde{Z}'\tilde{Z}\right) + tr\left(\tilde{Z}'\tilde{Z}\right) - 2tr\left(\tilde{Z}'HX\right) + tr\left(X'H'HX\right) \\ &\quad = tr(Z'Z) - 2tr\left(\alpha Z(H^{-1})^T HX\right) + tr\left(X'H'HX\right). \end{aligned}$$

Let $K = H^2$. It can be easily seen that K is a symmetric positive definite matrix, so it follows that H is also a symmetric matrix. So we now have

$$\begin{aligned} &tr\left(Z'Z - \tilde{Z}'\tilde{Z}\right) + ||\tilde{Z} - HX||_F^2 \\ &\quad = tr(Z'Z) - 2tr(\alpha ZX) + tr\left(X'KX\right) \\ &\quad = tr(Z'Z) - 2tr(\alpha ZX) + tr\left(X'\left[(1+\alpha)I_n - 2W + W'W\right]X\right) \\ &\quad = tr(Z'Z) - 2tr(\alpha ZX) + tr(X'X) + \alpha tr(X'X) \\ &\quad -2tr(X'WX) + tr\left(X'W'WX\right) \\ &\quad = (1+\alpha)tr(X'X) + \alpha\left[tr(Z'Z) - 2tr(Z'X)\right] \\ &\quad -2tr(X'WX) + tr\left(X'W'WX\right). \end{aligned}$$

Now apply Proposition 1 to the minimization criterion of RLLE, letting $\alpha = \frac{1}{\lambda}$. Note that since we are minimizing with respect to X, we only need to minimize $||\tilde{Z} - HX||_F^2$. Let $\tilde{z}_j$ and x_j represent the jth column of $\tilde{Z}$ and X, respectively. It can be seen through the ordinary least squares problem that the minimizing x_j is given by $(H'H)^{-1}H'\tilde{z}_j$. It therefore follows that the minimizing X is $H^{-1}\tilde{Z} = \{\lambda(I_n - W)^T(I_n - W) + I_n\}^{-1}Z$. □

Appendix B. Closeness measure (2) is a metric

We show here that the closeness measure (2), given by

$$D(Y, Y^{(b)}) = \sum_{\text{all pairs i,j}} |\delta(Y_i, Y_j) - \delta(Y_i^{(b)}, Y_j^{(b)})|,$$

is a distance measure. To do this, we will need to show the three properties of a distance metric: identity, symmetry, and subadditivity.

Identity. Suppose that $Y = Y^{(b)}$. Then $D(Y, Y^{(b)}) = \sum_{\text{all pairs i,j}} |\delta(Y_i, Y_j) - \delta(Y_i^{(b)}, Y_j^{(b)})| = 0$.

Symmetry. $D(Y, Y^{(b)}) = D(Y^{(b)}, Y)$ due to the absolute value within the summation.

Subadditivity. Suppose we have three inputs: Y, $Y^{(b)}$, and $Y^{(c)}$. We wish to show that $D(Y, Y^{(b)}) \le D(Y, Y^{(c)}) + D(Y^{(b)}, Y^{(c)})$.

We will show that $|\delta(Y_i, Y_j) - \delta(Y_i^{(b)}, Y_j^{(b)})| \le |\delta(Y_i, Y_j) - \delta(Y_i^{(c)}, Y_j^{(c)})| + |\delta(Y_i^{(b)}, Y_j^{(b)}) - \delta(Y_i^{(c)}, Y_j^{(c)})|$ for all pairs i and j.

By the triangle inequality, $|(a-c) + (c-b)| \le |(a-c)| + |c-b|$, and thus $|a-b| \le |a-c| + |c-b|$. Now let $a = \delta(Y_i, Y_j)$, $b = \delta(Y_i^{(b)}, Y_j^{(b)})$, and $c = \delta(Y_i^{(c)}, Y_j^{(c)})|$. The result follows.

References

[1] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, et al., A closer look at memorization in deep networks, arXiv:1706.05394 (2017).

[2] R. Bock, A. Chilingarian, M. Gaug, F. Hakl, T. Hengstebeck, M. Jirina, J. Klaschka, E. Kotrc, P. Savicky, S. Towers, A. Vaicilius, W. W., Methods for multidimensional event classification: a case study using images from a cherenkov gamma-ray telescope, Nucl. Instrum. Methods Phys. (2004) 511–528.

[3] H. Chang, D. Yeung, Robust locally linear embedding, Pattern Recognit. 39 (2006) 1053–1065.

[4] R. Coifman, S. Lafon, Diffusion maps, Appl. Comput. Harmon. Anal. (2006) 5–30.

[5] D. Donoho, C. Grimes, Hessian eigenmaps: locally linear embedding techniques for high-dimensional data, Proc. Natl. Acad. Sci. (2003) 5591–5596.

[6] J. Gui, C. Wang, L. Zhu, Locality preserving discriminant projections, in: International Conference on Intelligent Computing, Springer, 2009, pp. 566–572.

[7] I. Guyon, S. Gunn, B.-H. A., G. Dror, Result analysis of the nips 2003 feature selection challenge, NIPS (2004).

[8] T. Hastie, W. Stuetzle, Principal curves, J. Am. Stat. Assoc. (1989) 502–516.

[9] G. Hinton, S. Osindero, Y. Teh, A fast learning algorithm for deep belief nets, Neural Comput. (2006) 1527–1554.

[10] H. Hudson, T. Lee, Maximum likelihood restoration and choice of smoothing parameter in deconvolution of image data subject to poisson noise, Comput. Stat. Data Anal. 26 (1998) 393,410.

[11] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, L. Fei-Fei, Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels, in: International Conference on Machine Learning, 2018, pp. 2309–2318.

[12] I. Jolliffe, Principal Component Analysis, second ed., Springer, 2002.

[13] O. Kouropteva, O. Okun, M. Pietikainen, Incremental locally linear embedding, Pattern Recognit. Lett. (2005) 1764–1767.

[14] Y. Lecun, L. Bottou, B. Y., P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (1998) 2278–2324.

[15] Y.-K. Lei, Y.-M. Xu, J.-A. Yang, Z.-G. Ding, J. Gui, Feature extraction using orthogonal discriminant local tangent space alignment, Pattern Anal. Appl. 15 (3) (2012) 249–259.

[16] J. Nocedal, S. Wright, Numerical Optimization, Springer, 1999.

[17] M. Ren, W. Zeng, B. Yang, R. Urtasun, Learning to reweight examples for robust deep learning, arXiv:1803.09050 (2018).

[18] D. de Ridder, R. Duin, Locally linear embedding for classification, IEEE Trans. Pattern Anal. Mach. Intell. (2002).

[19] S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, Sci. New Ser. 290 (2000) 2323–2326.

[20] B. Scholkopf, A. Smola, K. Muller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (1998) 1299–1319.

[21] J. Tenenbaum, V. de Silva, J. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.

[22] S.-L. Wang, J. Gui, X. Li, Factor analysis for cross-platform tumor classification based on gene expression profiles, J. Circt. Syst. Comput. 19 (01) (2010) 243–258.

[23] Z.-H. You, Y.-K. Lei, J. Gui, D.-S. Huang, X. Zhou, Using manifold embedding for assessing and predicting protein interactions from high-throughput experimental data, Bioinformatics 26 (21) (2010) 2744–2751.