LEARNING OUT-OF SAMPLE MAPPING IN NON-VECTORIAL DATA REDUCTION USING CONSTRAINED TWIN KERNEL EMBEDDING

Yi Guo¹, Junbin Gao², Paul W. Kwan³
¹School of Math, Stat. & Computer Science, University of New England, Armidale, NSW 2351, Australia
²School of Computer Science, Charles Sturt University, Bathurst, NSW 2795, Australia
E-MAIL: ygao@turing.une.edu.au, jbgao@cstu.edu.au, kwans@turing.une.edu.au

Abstract—Twin Kernel Embedding (TKE) is a powerful non-vectorial data reduction algorithm proposed recently for advanced applications in clustering and visualization, manifold learning, etc. Due to the requirement of online processing in many cutting edge research problems involving highly structured data like DNA, protein sequences and biometric features that are non-vectorial in nature, learning the out-of-sample (OOS) mapping becomes a necessity. To address this, we propose Constrained TKE, which is an OOS extension of TKE capable of learning such a mapping function. This is achieved by including the mapping in the objective function optimized by the TKE algorithm. More broadly, this mapping function can be applied in other data reduction methods as an OOS extension. Furthermore, to improve the accuracy of predictions in case where new samples are present in batch, a refinement strategy is introduced by exploiting the similarity between new samples which is often ignored by other methods. Experimental results on the Reuters-21578 text collection confirmed the usefulness of the proposed method.

Keywords—Dimensionality reduction; TKE; Out-Of-Sample

I. INTRODUCTION

Dimensionality reduction (DR) and manifold learning for highly structured or so-called non-vectorial data [6] like texts, protein sequences, surveillance video clips, etc. have attracted considerable attention recently due to increasing demands from applications in biometrics [16], bioinformatics [23], information retrieval [9]. Lacking efficient vectorial representation, non-vectorial data pose great challenges to existing dimensionality reduction approaches.

Though recent years have witnessed rapid advances of DR algorithms for global or local similarity preservation operating in linear or nonlinear fashion such as ISOMAP [22], Locally Linear Embedding (LLE) [18], Laplacian Eigensmaps (LE) [1], LTSA [25] etc., these methods were mainly designed for vectorial input. A compromised solution would be to vectorize data first and then use as input to a DR algorithm. However, it is unclear how much information would be lost through the vectorization process resulting possibly in positively biased results.

Observing that most relationships among non-vectorial data can be expressed by a kernel gram matrix [5] (a widely accepted similarity measure), recently we proposed Kernel LE (or KLE) [6] that matches the similarity structure in high dimensional space with the lower dimensional space using Euclidean distance. This provides a solution for the DR problem of non-vectorial data directly without the need for performing any vectorization. Further, we proposed another method called Twin Kernel Embedding (TKE) [7] that uses a kernel gram matrix instead of simple Euclidean distance to reproduce the pairwise similarities among input data in the latent space, thereby capturing possible nonlinear structure in the high dimensional input space. These two algorithms of ours, along with few others (like KPCA [21] and “kernelized” LPP [10]), are the only available methods at present for dealing with non-vectorial data directly by means of kernel.

An unavoidable yet critical problem for DR involves the so-called Out-Of-Sample (OOS) mapping [3, 4]. That is to say, the prediction of output in the latent space for new samples or inputs. It is a practical problem in online applications like human motion tracking that is operating in real time with new data flooding in continuously. Basically the DR methods mentioned earlier were designed for batch processing that can take a substantial amount of time for completion. The absence of an efficient mechanism for handling new samples means that the DR algorithm must start all over again when new data are being presented. Other than the unnecessary computational cost involved, it is wasteful that previous results were simply discarded whether or not they would be beneficial to the prediction of new samples.

To address this OOS problem, many extensions have been proposed for existing methods like revising or appending additional components to the original algorithms. As an example of this approach, a continuous version of ISOMAP was proposed in [24] by using integral operators. A set of eigenfunctions instead of eigenvectors that forms the coordinates of the latent space are computed. Similarly, [3] revised some graph-based DR methods that automatically fall into the framework of spectral analysis by considering the convergence of eigenvectors and eigenvalues as the number of training samples increases by incorporating the Nyström formula and replacing eigenvectors by eigenfunctions [2].

Although it is feasible to provide OOS extensions through modifying the existing algorithms, special treatment to the new samples might be more intuitive. In [12], a three-stage strategy was introduced in order to obtain the incremental version of ISOMAP. It updates the neighborhood graph induced by the presentation of new input. Next, it refreshes the geodesic distances among the data (including the new sample) which are then used to estimate the location of the corresponding
in the latent space. Finally, all embeddings of the input data are updated accordingly by solving an incremental eigenvector problem. This three-stage strategy can be seen as an additional component of the original ISOMAP. Interested readers are referred to these papers and the references therein for more details on these methods.

In spite of the fact that OOS extension is possible through different means, several algorithms are able to process new samples in their design such as Locality Preserving Projections (LPP) [10] and Neighborhood Preserving Embedding (NPE) [8], etc. These algorithms actually impose a mapping function from the input data to their corresponding embeddings in the latent space. The target is to learn the mapping function instead of the embeddings. Once the mapping function is determined through training, the prediction for new data is extremely simple. Essentially, this mapping function can also be regarded as an additional component independent of the original DR algorithms to solve the OOS problem. As such, it can be naturally applied to other DR methods to extend them for handling new samples.

A. Locality Preserving Projections

Locality Preserving Projections (LPP) is a variant of Laplacian Eigenmaps (LE) since the mapping function is constructed on LE. In LE, a neighborhood graph $G$ is constructed by referring to the $k$-nearest neighbor criterion. That is, if $y_j$ is one of the $k$-nearest neighbors of $y_i$ in the Euclidean distance sense, there will be an edge connecting $y_i$ and $y_j$. Further, the weights of the edges are evaluated by an exponential decay function that reveals the relations among the data. Then the weight matrix $W$ containing the proximity information is included in an objective function to be minimized with respect to the embeddings $X$.

$$\sum_{ij} W_{ij} ||x_i - x_j||^2$$

where $W_{ij}$ is the $ij$-entry of the weight matrix $W$. In LPP, a mapping function is defined as

$$x_i = A^\top y_i,$$

where $A$ is the so-called transformation matrix. This relation is substituted into equation (1) and the optimization problem turns to finding an optimal transformation matrix $A$ instead of $X$. Since the objective function is still of quadratic form, LPP can easily find the the solution by solving an eigenvector problem. When new data come, equation (2) can give their embeddings easily once the transformation matrix has been learned from training data.

B. Neighborhood Preserving Embedding

Similarly, Neighborhood Preserving Embedding (NPE) is the result of LLE plus a predefined mapping function. The objective function of LLE is

$$\sum_i ||x_i - \sum_j W_{ij}x_j||^2,$$

where $W_{ij}$ is the $ij$-entry of the weighted adjacency matrix derived from a neighborhood graph $G$ which is constructed in the same way as that in LE. In NPE, an identical mapping function as presented in equation (2) is introduced and substituted into objective function (3) to get an optimal transformation matrix $A$ by a minimization procedure.

C. Mapping Functions

Both NPE and LPP exhibit locality preserving ability. This naturally arouses a interesting question about the choice of mapping functions. Does any mapping function have this attracting property? In fact, only smooth functions can fulfill this task for their neighborhood preserving ability. In LPP and NPE, they are linear mapping which is a typical smooth function. More generally, we can consider other mapping functions having the following form:

$$x_i = f(y_i) + \epsilon_i,$$

where $f$ is a mapping function.
where $f$ is any kind of smooth functions and $\epsilon_i$ is an error which can be zero. Theoretically, the mapping functions can be integrated into any DR algorithm and they can be arbitrarily complex provided that the optimization problem is still solvable with respect to the parameters of the functions after equation (4) is plugged into the objective function of the original algorithm. A better understanding of the original DR algorithm is necessary for the application of this component.

III. TWIN KERNEL EMBEDDING

Twin Kernel Embedding (TKE) preserves the similarity structure of input data in the latent space by matching the similarity relations represented by two kernel gram matrices, i.e. one for input data and the other for embedded data. It simply minimizes the objective function

$$- \text{Vec}K_y \cdot \text{Vec}K_x = -\text{tr}(K_y K_x),$$

where Vec is the vec operator on matrices, and $K_y$ and $K_x$ are the kernel Gram matrices derived from the kernel functions $k_y(\cdot, \cdot)$ and $k_x(\cdot, \cdot)$ which are defined on input data and embeddings, respectively. The logic is to preserve the similarities among input data and reproduce them in lower dimensional latent space expressed again in similarities among projected data. To make the optimization problem proper, two regularization terms on the kernel and embedded data are introduced and the objective function becomes

$$L = -\text{tr}(K_y K_x) + \lambda_x \text{tr}(K_x K_x) + \lambda_x \text{tr}(X X^T).$$

(5)

The second term is a ridge regularizer on the kernel to make sure that the norm of the kernel is controlled. The third term is a penalty of too big norm of the embedded data to make sure that the coordinates of the embeddings are relatively small. $\lambda_x$ and $\lambda_x$ are tunable parameters to control the strength of the regularization.

In order to capture the nonlinear structure in input data, $k_y(\cdot, \cdot)$ is usually nonlinear, so there is no closed form solution for (5) and hence a gradient-based algorithm for optimization has to be employed based on the prerequisite that $k_x(\cdot, \cdot)$ is differentiable. The initialization of $X$ is also required to start the optimization. KLE, KPCA and other methods which can deal with non-vectorial data can be applied here. The dimension of the embeddings is assigned according to the requirement of the application, which is normally 2. A byproduct of this optimization process is that we can get the optimal hyper-parameters for the kernel function $k_x(\cdot, \cdot)$ as well. It ensures that the kernel we picked is well adjusted.

TKE is also designed to preserve local information. This is done by the choice of the entries in $K_y$. Not all the entries are included in the optimization process but those that convey most of the similarity information of the input data. This selection process is fulfilled by the $k$-nearest neighboring. Given an object $y_i$ in the input space, only those objects whose similarities (in the sense of kernel values) to $y_i$ are in $k$ nearest neighbors of $y_i$ are selected. The variable $k(> 1)$ in $k$-nearest neighboring controls the locality that the algorithm will preserve. This process is a kind of filtering that retains what we are interested while leaving out minor details. It can also be interpreted as constructing a weighted adjacency graph which performs in feature space and the weights on edges are evaluated by kernel as that in KLE [6]. However, the algorithm also works if a full kernel gram matrix is input in which case TKE turns to be a global approach.

Other than the fact that TKE outperforms other methods such as KPCA and KLE etc, an elegant feature of TKE is that it can handle non-vectorial data since in its objective function, it only requires the kernel gram matrix of the input data that contains the similarity information. Through TKE, any kind of data can be visualized in lower dimensional space as long as an appropriate kernel is defined on them. So a kernel gram matrix on input data $K_y$ and initialization for $X$ will be adequate for TKE to find the optimal embeddings.

IV. OOS EXTENSION FOR TKE AND REFINEMENT STRATEGY

By analyzing the objective function of TKE and its optimization procedure, it is apparent that the partial derivative $\frac{\partial L}{\partial X}$ is the object for our gradient-based optimization. Because we want to apply the mapping function to TKE, it must be tractable. In other words, the function should be parameterized by a set of differentiable parameters. On the other hand, addition to the smoothness requirement for the neighborhood preserving property mentioned earlier, it should also be able to accept non-vectorial data.

A. Mapping Function for Non-vectorial Data

Apparently, the transformation matrix in LPP or NPE is not applicable since it requires the vectorization of input data. The non-vectorial data handling requirement influences our choice of a mapping function. In other words, we have to seek other solutions that can directly use the non-vectorial data. One possible candidate is the kernel function. Here, we construct a mapping function that only uses kernel functions as its building blocks by exploiting the well known representer theorem (see [20] for its general form)

$$f(y) = \sum_{i=1}^{N} \alpha_i k(y_i, y)$$

(6)

which defines a family of smooth functions as clarified in [17]. $k(\cdot, \cdot)$ can be any kernel function and thus its input can be non-vectorial. In the representer theorem, the codomain of $f$ in (6) is $R$, so we can define our mapping function as follows

$$f_j(y_i) = \sum_{n=1}^{N} \alpha_{mn} k(y_i, y_m)$$

(7)

and let

$$x_{ij} = f_j(y_i)$$

(8)

where $x_{ij}$ is the $j$-th component of $x_i$. The kernel function could be any valid kernel. Interestingly, if $k(\cdot, \cdot)$ is RBF kernel, equation(7) will take the form of the RBF neural network used in GPLVM with back constraints [13]. It may also be interpreted in other possible forms if other kernel functions are
being applied here. Note that there is another kernel defined on the input data \(k_p(\cdot, \cdot)\) in TKE. Although the kernel in (7) is not necessarily the same as \(k_p(\cdot, \cdot)\), for convenience sake, we let it be \(k_p(\cdot, \cdot)\) in order to take advantage of the non-vectorial handling capability. Thus, the mapping function can be rewritten in matrix form as

\[
X = K_p A \tag{9}
\]

where \(A\) is a matrix formed by stacking all parameters \(\alpha_{mj}\)'s in (7). As a result, the mapping function is parameterized by \(A\) which is differentiable.

We then substitute equations (7), (8) and (9) back in the objective function of TKE, equation (5), and minimize it with respect to \(A\), namely \(\alpha_{mj}\)'s, to learn the mapping function which will be used subsequently for new samples. Because the mapping function serves as a constraint on the embeddings, we call this new algorithm Constrained TKE or CTKE in short.

B. Refinement Strategy

Here, we consider the situation when new samples are being presented in batch. To facilitate our explanation, we call the training set \(Y_T\) and the new data samples \(Y_N\), and the kernel gram matrix for the whole set as

\[
K = \begin{bmatrix} K_T & K^T_{NT} \\ K_{NT} & K_N \end{bmatrix} \tag{10}
\]

where \(K_T = k_p(Y_T, Y_T)\), \(K_N = k_p(Y_N, Y_N)\), \(K_{NT} = k_p(Y_N, Y_T)\) and \(k_p(R, S)\) is the short notation for \(|R| \times |S|\) kernel gram matrix evaluated at all pairs of points from set \(R\) and set \(S\) using kernel \(k_p(\cdot, \cdot)\). \(|R|\) is the cardinality of the set \(R\).

Up to now, we know that we can use the training data \(Y_T\) for learning the mapping function as described in the last section, i.e. use \(K_T\) as \(K_N\) in mapping function (9) and TKE (5) for training. Once we have \(A\), we can define the embeddings of \(Y_T\) as

\[
X_T = K_T A
\]

and correspondingly when new samples \(Y_N\) arrive, we can predict their coordinates in latent space by

\[
X_N = K_{NT} A \tag{11}
\]

where \(X_T\) and \(X_N\) are the embeddings of the training set and new samples respectively.

Apparently, the similarity information among \(Y_N\) expressed by \(K_N\) is not fully exploited in the algorithm. Because this information depicts the inner relationships among the new samples, it should be utilized to further refine the coordinates predicted by the mapping function. This motivates the following refinement strategy. Since both the kernel gram matrix \(K_N\) and the starting value \(X_N\) are available, we can use the original TKE to accomplish this task. In this case TKE only runs on the new samples without referring to the training set. As a result, it will further adjust the embeddings according to the pairwise similarity information only for the new samples. This refinement procedure can compensate for the errors introduced by the samples or kernel values and hence improve the accuracy of the prediction. It should be emphasized that when the number of new inputs is less than 2, \(K_N\) is actually a scalar rather than a matrix, the refinement is impractical and unnecessary. However, the refinement is still applicable after a number of new samples have accumulated. This is very useful in multi-step searching in large databases where the coarse prediction in the initial fast indexing step is enough for accelerating the query speed [11].

In the following, we restate the OOS extension for TKE and the refinement strategy concisely:

1) Perform CTKE on training set to learn the mapping function (9);
2) Predict \(X_N\) using (11);
3) Start TKE with \(K_N\) and \(X_N\) to refine the results.

V. EXPERIMENTAL RESULTS

Experiments were conducted on Reuters-21578 Text Categorization Test Collection\(^3\) to compare CTKE with and without refinement, and the results of original TKE and related methods including KLE and KPCA. Two data sets were extracted from the database using different splitting strategies. The first data set is composed of 10 selected topics from TOPICS category (contains economics related topics and is most frequently used in the literature) in Reuters-21578. In the training set, there are 300 pieces of news (30 pieces for each topic) extracted randomly from the training split of the database, and in the test set there are 200 pieces (20 pieces for each topic likewise) from the test split. The second data set includes 3 topics: cocoa, interest and corn from TOPICS category. The training set contains all the text related to coffee and interest while the test set comprises all the texts from corn. In both cases, the samples in the test sets are unknown to the algorithm. Also, in the second case, the test set is made up of a totally new topic differing from those in the training set.

Although by applying methods like VSM [19], the text data could be vectorized, we simply treat the text data as strings of characters, and the String Subsequence Kernel (SSK) [15], [14] is selected as \(k_s\) to evaluate the similarity between texts. Thus, the text data are actually non-vectorial. The texts from the same topic are expected to be close in the 2-dimensional space with overlappings indicating the texts with same or very similar meanings. An RBF kernel is used as \(k_s\) for the embedded data in TKE due to its simple derivative formulation and ability to capture nonlinear structure in the data. KPCA provides the initialization for \(X\). For CTKE, the parameters for kernel regularization \(\lambda_k\) and variable regularization \(\lambda_v\) are set to 0.05 and 0.001 respectively. The \(k\) in \(k\)-nearest neighbor graph to filter \(K_y\) is 13. In the refinement stage, \(k = 6\), \(\lambda_k\) is still 0.05 while \(\lambda_v = 0.01\) in order to encourage the adjustment to the coordinates and to fit less inputs.

Figure 1 plots the results of the proposed algorithm on the first data set. The result of CTKE on the training set (a)

---

\(^3\)Reuters-21578 distribution 1.0 test collection is available from http://www.daviddewitt.com/resources/testcollections/reuters21578
Fig. 1: The results of the proposed algorithm on the first data set. All the figures use the same symbols as shown in the legend of figure (a).

Fig. 2: The results of the proposed algorithm on the second data set with circles for coffee, pluses for interest and arrows for corn.

Fig. 3: The results of other algorithms on the same data set. (a), (b) are the TKE and KPCA on the first data set (treating training set and test set altogether, see (a) in figure 1 for symbols); (c), (d) are the TKE and KLE on the second data set using same symbols as figure 2.

reveals clear cluster structure as expected. But the prediction on the test set (b) are not so satisfactory with indistinguishable overlappings. Through the refinement step, the embeddings of new samples are reorganized to the clusters they belong to which are shown in (c) and (d). One can observe some of these cases such as coffee, jobs and so on. Figure 3 (a) and (b) are the results of TKE and KPCA on the entire first data set. As one can observe, the result of TKE is neat with clear clusters whereas KPCA fails to give any useful information. KLE’s result is not shown for this case since it projects all the input data onto a line which is useless in practice. An interesting observation is that the results of the texts from topic cocoa and corn are almost the same because the corresponding kernel values are very close or equal to 1. That means, the underlying SSK does not significantly distinguish topics cocoa and corn. This also confirms that both the proposed algorithm and TKE correctly reflect the underlying mechanism, namely, if two objects are similar in the sense of given kernel, they will stay close together in the latent space.

Figure 2 demonstrates the results on the second data set. The difference is that the test set contains a topic which is absent in the training set. To the algorithm, the training set provides no information about this new category. The relationship between the new samples and training data is only reflected in $K_{NT}$
in equation (10). The result of the prediction must be biased since the algorithm will have no knowledge about it after training. 2(b) shows the situation where the new embeddings (arrows in the figure) overlap the training data totally. By using the information embedded in $K_\mathcal{Y}$, the refinement optimizes the coordinates of the new samples further. 2(c) displays the result after refinement in which the new category can be easily identified. Intuitively, the information of the kernel on input data is fully exploited in our proposed algorithm because of the symmetry of the kernel gram matrix. Moreover, it is fulfilled in an incremental way thereby learning the OOS mapping for the original TKE. 2(d) is the result of KPCA on the whole set with considerable overlappings. Figure 3(c) and 3(d) are TKE and KLE applied to the entire dataset. It is clear that the result of the CTKE has comparable result to the original TKE.

VI. CONCLUSIONS

Twin Kernel Embedding (TKE) is a powerful tool for dimensionality reduction and visualization of non-vectorial data. The out-of-sample extension for TKE is of significant value in real life applications. By integrating a mapping function that is applicable to non-vectorial data in TKE, we endow the original algorithm the mechanism to learn the low dimensional embeddings for new samples. Furthermore, a refinement strategy was introduced to further improve the result of the prediction. The information contained in the kernel gram matrix is fully exploited by this strategy and therefore can rectify the bias in this new class of learning algorithms. It is worth mentioning that because the mapping function proposed purely involves kernel through the representer theorem, not only can it handle non-vectorial data but it also favors locality preserving due to its smoothness.

More broadly, the mapping function can be incorporated into other dimensionality reduction or manifold learning methods. This additional component can be considered as a common technique for other DR methods to solve the OOS problem.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (NSFC 60373090), the ARC DP Development Grant from Charles Sturt University and the University Research Grant from the University of New England.

REFERENCES