Neural Network for Nonlinear Dimension Reduction Through Manifold Recovery

Jessica Bader*
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA, USA
jabader@iastate.edu

Declan Nelson*
Department of Computer Science
Georgia Tech
Atlanta, Georgia, USA
df.nelson@gatech.edu

Thalia Chai-Zhang*
Department of Computer Science
Bard College
Annandale-on-Hudson, NY, USA
tz8035@bard.edu

Walter Gerych
Department of Data Science
Worcester Polytechnic Institute
Worcester, MA, USA
wgerych@wpi.edu

Elke Rundensteiner
Department of Data Science
Worcester Polytechnic Institute
Worcester, MA, USA
rundenst@wpi.edu

Abstract - The curse of dimensionality is a classic problem in machine learning which states that the number of data points required to achieve a desirable accuracy increases as the square of the dimensionality of the data points. Many dimension reduction techniques have been developed to combat this. Maximum Variance Unfolding (MVU) is one such state-of-the-art nonlinear dimension reduction algorithm that recovers a lower dimensional manifold which most of the data lies near. The algorithm's inability to handle large data sets or those that fail to meet certain distribution criteria are severe detriments to its efficiency and robustness. In this report we have introduced a variant of MVU which utilizes a novel Feed Forward Neural Network to recover the lower dimensional manifold representation. We demonstrate that our method succeeds in recovering lower dimensional manifolds and outperforms MVU in several ways including increased efficiency and flexibility .

Keywords - Dimension reduction, maximum variance unfolding, machine learning, neural network, manifold learning.

I. Introduction

The handling of high dimensional data continues to be a ubiquitous challenge across various fields. Application areas which work with high dimensional data (such as natural language processing, image processing, etc.) suffer from problems caused by high dimensionality. Utilizing more dimensions demands more data points, increases algorithmic run time, and decreases the accuracy of the model [1]. The *curse of dimensionality* also implies that as dimensions increase, the classifier performance achievable tends to decrease exponentially [1]. In this work, we combat this problem through a nonlinear dimension reduction technique which preserves local geometry.

*Work on this project was done while the authors were at Worcester Polytechnic Institute

A. Problem Statement

The manifold hypothesis states that many higher dimensional data sets lie on or near a lower dimensional manifold embedded in the higher dimensional space [2], where there is some isometry between the manifold and a lower-dimensional real space. Recovering these manifolds thus finds a lower dimensional embedding of the data. The goal of this research is to develop a new method to recover these manifolds.

B. State-of-the-Art

Many dimension reduction techniques exist, such as Principal Component Analysis (PCA), a simple linear technique [3]. Examples of nonlinear solutions include Kernel-PCA [4] and Maximum Variance Unfolding (MVU) [5]. MVU is a dimension reduction method motivated by the manifold hypothesis which can recover nonlinear, lower dimensional manifolds through convex optimization problem [6]. The benefit of MVU as a dimension reduction technique is its ability to recover these manifolds while preserving local geometry by preserving the distances between nearest neighbors [5]. The details of MVU will be discused in depth in *Background*.

C. Limitations of the State-of-the-Art

MVU has several disadvantages. It is a constrained optimization problem, and thus more constraints must be added to the optimization problem with the addition of each data point, causing the computational costs to increase exponentially. The number of constraints can be determined by the equation $\frac{k(k-1)}{2} * N$ where N represents the number of data points in the set and k represents the number of neighbors. MVU has a theoretical limit of 2,000 points with 6 neighbors, after which the algorithm becomes too inefficient to reasonably utilize [7]. In addition, MVU cannot resolve two or more disjointed sets

in its neighborhood matrix because this would be a non-convex problem.

D. Proposed Solution

We introduce a deep variant of MVU, which we call Deep Maximum Variance Unfolding (DMVU). DMVU is a novel dimension reduction method based on MVU which utilizes a Feed Forward Neural Network (FFNN) in place of MVU's convex optimizer. The loss function of the FFNN encourages the model to preserve local neighbors and maximize the variance between non-neighbors. Rather than adding constraints, DMVU applies penalties to maximize global variance while encouraging the model to preserve neighbor distance. This makes DMVU far more efficient for large data sets and keeps it from being bound by the theoretical limit of 2,000 points.

II. Background

A. Principal Component Analysis and Kernel Principal Component Analysis

PCA is a linear dimension reduction technique using the least mean-square error [8]. PCA utilizes the most important eigenvectors in a data set to transform it into a lower dimensional space [3]. Kernel-PCA is a variation of PCA which uses the Kernel trick to map data into a new space that allows for nonlinear dimension reduction [9]. One downside to Kernel-PCA is that the kernel is input manually, making it infeasible to find representations of highly complex spaces [4]. Another issue with Kernal-PCA is that it does not ensure that local neighbor distance between points remain consistent, which results in possible loss of accuracy in the lower dimensional space. Given these limitations, MVU is more appropriate than Kernal-PCA when one wishes to preserve local geometry.

B. Maximum Variance Unfolding

MVU is solved using a semi-definite problem which finds a kernel, *K*, for the lower dimensional representation of the dataset D. *K* is the square matrix created by $K_{i,j} = < d_i, d_j > = d_i^T d_j$, for all data points $d_i, d_j \in D$. The best *K* can be found by maximizing the trace of *K* subject to three constraints: *K* must (1) preserve neighbor distance of the original data, (2) be centered at the origin, and (3) be positive semi-definite. The neighborhood is defined by the neighborhood matrix, *Q*, which is a square matrix where $Q_{i,j} = 1$ if points d_i and d_j are neighbors and 0 otherwise. The constraints are described as followed:

$$K_{ii} - 2K_{ij} + K_{jj} = \|d_i - d_j\|^2, \forall i, j, Q_{i,j} = 1 \quad (1)$$

$$\Sigma_{ij} K_{ij} = 0 \quad (2)$$

$$K \succeq 0 \quad (3)$$

Once the optimal kernel has been identified, performing Singular Value Decomposition (SVD) will return the lower dimensional representation of the data that preserves neighbor distance exactly while having maximum global variance. SVD is the same method used in PCA to solve for the best representation.

III. Proposed Method: Deep Maximum Variance Unfolding

DMVU is a FFNN which finds a dataset's optimal lower-dimensional manifold. The key idea of DMVU is to replace the constraints of MVU with a FFNN penalized by a custom loss function. Penalization results in a more flexible and robust model over a constraints based model, as it lacks hard restrictions. It likewise does not need to maintain and check $\frac{k(k-1)}{2} * N$ constraints, having a more consistent resource cost as data set size increases. The loss function encourages the model to approximately preserve local distances while maximizing overall variance. An overview of DMVU can be seen in Fig. 1

A. Methodology

Local Distance Preservation: The first term of the loss function punishes the model for differences between the distances of each neighbor pair and their the distances between those pairs in the lower dimensional representation, known as the latent space. *Q* is applied element-wise to the pairwise-distance matrices of the original and latent spaces to force the model to overlook non-neighbor distances. We calculate the pairwise-distance matrix as the square matrix where $M_{i,j}$ is the Euclidean distance between points d_i and d_j. Then the norm is taken of the difference between the two to find the neighbor distance term *A*. Therefore, the model is encouraged to minimize the difference between high and low dimensional neighbors. For this study, the neighborhood matrix *Q* was defined by the *k*-Nearest Neighbor algorithm [10]. Thus the local distance preservation term is:

$$A = \| (QP - QR) \|_2 \quad (4)$$

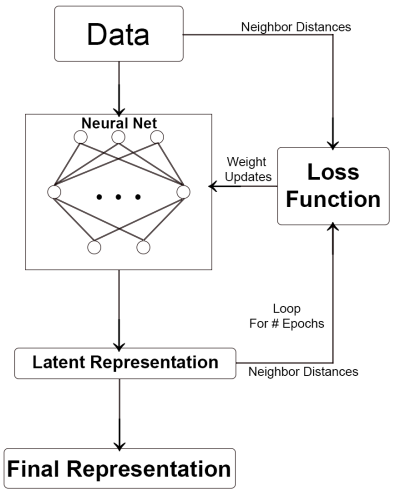
Where *Q* is the neighborhood mask; *P* is the pairwise-matrix of the original space *D*, where $D \in R^{N,M}$, $D = \{d_1, d_2, ..., d_N\}$; *R* is the pairwise-matrix of the latent space *L*, where $L \in R^{N,m}$, $D = \{l_1, l_2, ..., l_N\}$; *N* is the number of data points; *M* is the original dimensionality of the data; and *m* is the final dimensionality of the data.

Variance Encouragement: The second term of the loss function, *V*, encourages the model to maximize the distance between non-neighboring points. It sums the multiplicative inverses of the variance in each dimension. This encourages the latent space to spread in all directions, which increases the variance and decreases its multiplicative inverse. The second term is scaled by λ, which is a hyper-parameter for balancing the two loss terms in case one term becomes much bigger than the other.

$$V = \sum_{i=1}^{d} \left\| \frac{\lambda}{\frac{\sum_{j=1}^{k} R_{ji}}{N}} \right\|_2 \quad (5)$$

Where d is the dimensionality of the latent space, k is the number of neighbors. The full cost equation can be seen below.

Fig. 1: Overview of DMVU



$$C = A + V = \left\| (QP - QR) \right\|_2 + \sum_{i=1}^{d} \left\| \frac{\lambda}{\frac{\sum_{j=1}^{k} R_{ji}}{N}} \right\|_2 \quad (6)$$

Neural Network Architecture: The FFNN we implemented for DMVU used a single hidden layer with ten dimensions to project the points into a higher dimensional space before reducing to two dimensions. Through experimentation, we selected the Adam optimizer and Parametric Rectified Linear Unit (PReLU) activation function [11]. Our FFNN was created in PyTorch [11].

IV. Experimental Study Evaluation

A. Data Sets

1) S Curve: The first data set we employ is the *S* curve data set from the Sci-Kit Learn data set library [12]. This synthetic data set contains 3D data points distributed in a manifold in the shape of an *S*, which can be simply represented in a 2D space as a rectangle. The data set comes with a color-coding scheme which allows it to be graphed with a color gradient according to where it lies along the manifold. This option allows the user to verify the points mapped correctly. The data set also allows any number of points to be requested and generated according to its distribution.

2) Swiss Roll: The Swiss Roll data set is a 3D data set from the Sci-Kit Learn data set library [12], with a manifold in the shape of a Swiss Roll, as shown in Fig. 3. It also allows any number of points to be requested and provides a color-coded scheme for correctness verification. Similar to the *S* Curve, the Swiss Roll can be represented in 2D as a rectangle.

3) Iris: The Iris data set includes information about 150 flowers belonging to three different classes [13]. Each data point has 4 dimensions of numerical information about the plant. We selected this simple data set to test if classifier accuracy would be retained after using DMVU.

Fig. 2: S Curve Data Set

(a) Original 3D S Curve

(b) S Curve unrolled with DMVU

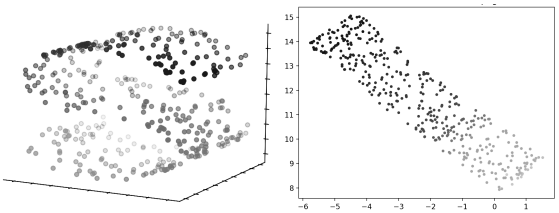


Fig. 3: Original 3D Swiss Roll

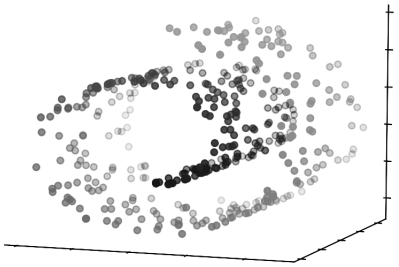


Fig. 4: Swiss Roll Reduction to 2D

(a) 2D Shape Unrolled by MVU

(b) 2D Shape Unrolled by DMVU

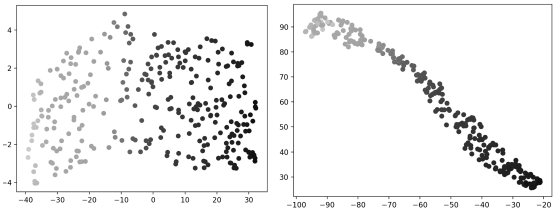
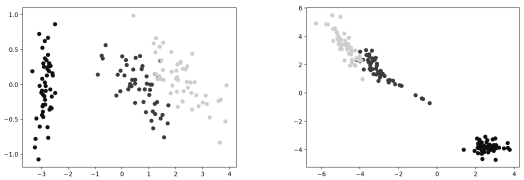


Fig. 5: Iris Data Set Reduction to 2D

(a) 2D Shape Unrolled by MVU and clustered by *k*-Means

(b) 2D Shape Unrolled by DMVU and clustered by *k*-Means



B. Experiments

1) Experiment 1 (Unrolling): We verified DMVU's ability to unroll a 3D shape into a 2D shape with the *S* Curve and the Swiss Roll. For the *S* Curve, we set our hyper-parameters as 400 points, 20,000 epochs, 20 nearest neighbors, and λ to 10. These were found via hyper-parameter tuning. As shown in Fig. 2b, DMVU was able to successfully unroll the curve into a rectangle.

For the Swiss Roll, we used 20,000 epochs, 400 points, 10 nearest neighbors, and λ to 10,000. As presented in Fig. 4b, the algorithm also successfully unrolled the Swiss Roll in 2D.

2) Experiment 2 (Clustering): We ran a common clustering technique, the *k*-Means classifier [12], to cluster 2D representations of 300 Swiss Roll points with 9 neighbors. We found DMVU clustered the data in 25.52 seconds, compared to the 148.71 seconds MVU took, while achieving similar classification accuracy (DMVU 75.82% : MVU 74.19%). The increase in DMVU's accuracy over MVU could be a result of its more flexible model or a tighter fit to its data but further investigation is not within the current scope of this study.

3) Experiment 3 (Runtime): We also compared the runtimes of DMVU and MVU with the Swiss Roll data set. We ran both algorithms on a set of 300 points with a neighborhood mask of 9 neighbors. Fig. 4 shows that the representations they found were acceptable, as they preserved local geometry and maximized global variance. The DMVU, however, only took 85.60 seconds to run in comparison to the 405.36 seconds required by the MVU. Furthermore, we compared the run times of MVU and DMVU on a series of different sized training sets. The results can be seen in Fig. 6.

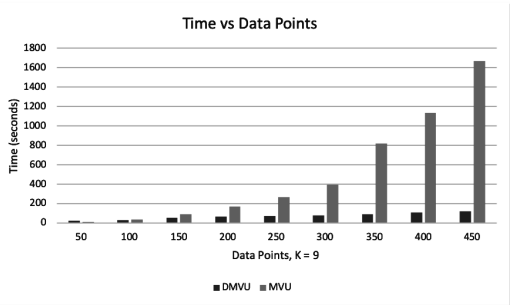
V. Discussion

From the results of the *S* curve and Swiss Roll, we confirmed that DMVU is able to achieve similar lower dimensional representations to MVU. The Iris data set demonstrated that a clustering algorithm would work as well on the results of DMVU as MVU. In addition, we verified several ways DMVU out preformed MVU. First, DMVU was consistently faster than MVU with data sets of 100 points or more. This was because DMVU showed a linear relationship between number of points and run time, as compared to the exponential time increase of MVU. This displayed excellent efficiency improvement for large data sets, particularly because the lower dimensional representation classified equally well. Additionally, DMVU is not limited to 2,000 points; we were able to process 10,000 points in several minutes.

VI. Conclusion

As demonstrated experimentally, DMVU has excellent potential as a dimension reduction technique. It matches the accuracy of the state-of-the-art nonlinear dimension reduction algorithm, MVU, while also out-performing MVU according to several important metrics. First, DMVU is more efficient than MVU when using sets with many data points. It can also efficiently process many more points than MVU.

Fig. 6: Time comparison between MVU and DMVU dimension reduction while varying number of points



Our next steps are to implement improvements which will allow for batch processing in DMVU. We expect that allowing for batch processing will further improve the efficiency of DMVU. There is also the potential for measurable increases in the accuracy of DMVU further over MVU's that we would like to explore and develop fully through future work.

ACKNOWLEDGEMENT

We would like to thank the National Science Foundation for the REU Site: Data Science Research for Healthy Communities in the Digital Age grant under award number #1852498 which funded our research. We would also like to thank the Worcester Polytechnic Institute Department of Data Science for hosting the REU site and providing further resources.

References

[1] R. E. Bellman S. E. Dreyfus "Applied Dynamic Programming" in Princeton University

[2] C. Fefferman, S. Mitter, H. Narayanan. "Testing the manifold hypothesis." Journal of the American Mathematical Society 29, no. 4: 983-1049. 2016

[3] K. P. F.R.S., LIII. On lines and planes of closest fit to systems of points in space, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, vol. 2, no. 11, pp. 559572, 1901.

[4] Y.-J. Liu, T. Chen, and Y. Yao, Nonlinear process monitoring and fault isolation using extended maximum variance unfolding," *SIAM Rev*. vol. 48 no. 4 pp. 681-699 2006.

[5] K. Q. Weinberger, F. Sha, and L. K. Saul, "Learning a kernel matrix for nonlinear dimensionality reduction," *Twenty-first international conference on Machine Learning - ICML 04*, 2004.

[6] J. Sun S. Boyd L. Xiao P. Diaconis "The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem" *J. Process Control*, vol. 24, no. 6, pp. 880891, 2014.

[7] J. Wang, (2012). "Geometric Structure of High-Dimensional Data and Dimensionality Reduction." Berlin, Heidelberg: Springer, pp(181-202).

[8] I. K. Fodor A Survey of Dimension Reduction Techniques June 2002.

[9] B. Schlkopf, A. Smola, and K.-R. Mller, Nonlinear Component Analysis as a Kernel Eigenvalue Problem, *Neural Computation*, vol. 10, no. 5, pp. 12991319, 1998.

[10] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), pp.21-27.

[11] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in Pytorch, 2017.

[12] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[13] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.