Empirical study of manifold learning techniques on forecasting stock price trend

Xinghao Niu[1]

In this work, five popular manifold learning techniques, PCA, ISOMAP, Locally Linear Embedding, Laplacian Eigenmaps and t-SNE, are examined on improving prediction accuracy of stock price trend. Effect of examined manifold learning techniques on classification and clustering task is proved to be different. Examined techniques tend to often worsen performance in clustering task. In classification task, observed improvement by all methods is slight, usually less than 1 percent. And only Laplacian Eigenmaps can more often stably improve classification accuracy at all number of components while other methods can't. Experiment results also suggest that there is no general effective technique for different stock price data set.
Keywords: dimension reduction, manifold learning, stock price trend, classification, clustering, neural network, k-means

1. Introduction

Real-world data usually has high dimensionality. In order to handle such real-world data adequately, its dimensionality needs to be reduced [3]. Manifold learning techniques are used to project original higher dimensional data to lower dimensional manifold by keeping structure of original dataset. It's also reasonable to represent financial market data as high-dimensional data. So, reducing high dimensionality of financial market data can be important for improving prediction accuracy.

State of the art manifold learning techniques include PCA (Principal Component Analysis), Locally Linear Embedding, multidimensional scaling (MDS) [11], diffusion maps, ISOMAP, Laplacian Eigenmaps, Maximum Variance Unfolding, t-SNE and autoencoder. Principal Component Analysis ([8, 7]) is aimed to project original high dimensional data to lower dimensional manifold based on main directions of datapoints. ISOMAP [10] determines neighborhood information on the manifold to obtain a weighted graph for data points based on certain distance measurement. Graph is then updated by calculating geodesic distances among data points. Lower-dimensional embedding is determined by solving optimization problem. Disadvantage of ISOMAP and MDS is higher computation requirement. Locally Linear Embedding [12] generally consists of three steps. First step is selecting

[1]Niu Xinghao — Ph.D. student, Department of economic informatics, Faculty of Economics, Lomonosov Moscow State University

neighbors, from which reconstruction occurs and weights are computed by solving optimization problem in second step. In final step, lower dimensional embedding is determined by minimizing error of reconstructing from weights. Laplacian Eigenmaps [13] also generally consists of three steps. In first step, adjacency matrix is constructed for determining whether data points are connected. In second step, weights among connected data points are calculated (other weights are set zero). In third step, Laplacian matrix is calculated and solving related eigenvector problem is needed for determining low dimensional manifold. Spectral clustering ([14, 15]) incorporating k-means usually uses Laplacian Eigenmaps as dimension reduction method. How Laplacian matrix needs to be derived from data set remains a question because structure of data set can be various. Diffusion maps was firstly proposed in [17]. And diffusion semigroups are used to generate multiscale geometries in order to organize and represent complex structures. Later, in ([18, 19]), the proposed diffusion map is integrated into one framework with the eigenmaps and random walk. Stiefel manifold from Grassmannian is used to project data to lower-dimensional space in the work of proposing visClust [16], which is a newly proposed clustering algorithm. Effectiveness of this novel clustering algorithm tends to decrease with rising of dimension while good result can be achieved on low dimensional data. Stochastic Neighbor Embedding (SNE) was initially proposed in [27]. SNE uses conditional probabilities to represent similarities. Final low dimensional embedding is a result of updating initialized low dimensional embedding by using gradient of sum of Kullback-Leibler divergences. For solving certain problems including difficulty of optimization in original SNE, t-SNE [26] is proposed with simpler gradient and computing the similarity by student-t distribution rather than a Gaussian distribution. Autoencoder ([20, 28]) is a multilayer neural network with a small central layer for reconstructing high-dimensional input vectors. Thus, by training the neural network, it's assumed that high-dimensional data can be effectively converted to low-dimensional representation. The idea of maximizing the trace of the inner product matrix has the effect of maximizing the variance of the low dimensional representation [23] was originally proposed as "semidefinite embedding" in ([25, 22]). Later "semidefinite embedding" is described as "maximum variance unfolding" [24].

There are works for comparing different manifold learning techniques. In the work [3], a comprehensive experiment for comparing twelve manifold learning techniques is conducted. In experiment manifold learning techniques are tested on five artificial datasets and five natural datasets. Main conclusion is that nonlinear techniques for dimensionality reduction are often not capable of outperforming traditional linear techniques such as PCA. In another work [5], PCA is examined on clustering gene expression data.

Experiment shows that clustering with the PCs does not necessarily improve, and often degrades, clustering quality.

So far, few works have focused on investigating effectiveness of manifold learning techniques on financial market data, especially stock price data. In [1], experiment is carried out on examining PCA, kernel PCA (KPCA) [31] and FRPCA [4]. Main reached conclusion is that three PCAs do not give significantly different results in average and standard PCA performs slightly better than FRPCA, and FRPCA performs slightly better than KPCA in average based on the P-values. One advantage of the work is the idea for construction of feature space. Limitation of the work is that only methods relating with PCA are examined. And in experiment there is no comparison for result obtained without dimension reduction. In work of [2], principal component analysis (PCA), autoencoder, deep belief network [28] are examined on stock selection. Authors argue that except for fitting nonlinear relations, autoencoder, deep belief network show no superiority to principal component analysis on Sharpe ratio and the advantage of dimensionality reduction is mainly reflected in trend situations (trend of moving up or down). Limitation of the work is related with the fact that risk free rate is not considered and only limited methods are examined.

It's very difficult to make any sufficient conclusion based on conducted researches on financial market data. Effect of manifold learning techniques on reducing dimensionality of financial market data is very unclear.

In this work, five popular manifold learning techniques, PCA, ISOMAP, Locally Linear Embedding, Laplacian Eigenmaps and t-SNE are compared for stock price trend clustering and classification task. The idea in [1] for construction of feature space is also used in this work. Manifold learning techniques are tested on datasets consisting of stock prices of five companies and high dimensional feature spaces including important macroeconomic indicators (for example, non-farm payroll, CPI, M1, M2, etc.) and other important indicators. Original datasets (without dimension reduction) and projected datasets are set as input for fully connected neural network and k-means to reveal effect of five popular manifold learning techniques.

The paper is organized into four sections. Section 1 is introduction part of total work. Section 2 reviews mathematical formulation of five manifold learning techniques. Section 3 presents experiment for comparison and discussion of experiment result. Section 4 is conclusion.

2. Manifold learning techniques

2.1. PCA

Goal of Principal Component Analysis [9] is to extract the important information from the original high dimensional data set and to express this information based on main directions of datapoints. It is likely to be the oldest manifold learning technique.

High dimensional data set with N datapoints is denoted as $\boldsymbol{X}$, with $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_N \end{bmatrix} \in R^{D\times N}$.Vector $\boldsymbol{\mu_X}$ is the central point of all data points:

$$\boldsymbol{\mu_X}=\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i,\ \boldsymbol{\mu_X} \in R^{D\times 1} \tag{1}$$

Matrix $\boldsymbol{M}$ from $\boldsymbol{\mu_X}$, with $\boldsymbol{M} = \begin{bmatrix} \boldsymbol{\mu_X} & \cdots & \boldsymbol{\mu_X} \end{bmatrix} \in R^{D\times N}$, is built. Original high dimensional data $\boldsymbol{X}$ needs to be centered firstly:

$$\boldsymbol{X}_c=\boldsymbol{X}-\boldsymbol{M} \tag{2}$$

Based on centered data $\boldsymbol{X}_c$, main directions need to be found through Singular Value Decomposition (SVD):

$$\boldsymbol{X}_c = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \tag{3}$$

Low dimensional representation $\boldsymbol{D}$ is obtained through:

$$\boldsymbol{D} = \boldsymbol{U}^T\boldsymbol{X}_c \tag{4}$$

2.2. ISOMAP

Original high dimensional data set $\boldsymbol{X}$, with $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_N \end{bmatrix} \in R^{D\times N}$. Dimension of low dimensional manifold is denoted as d. Initially neighborhood relations of data points are represented as a weighted graph $\boldsymbol{G}$, with distances $d_X(i,j)$ between pairs of points i, j in the input space $\boldsymbol{X}$.

$$\boldsymbol{G} = (d_G(i,j)) = (\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) \tag{5}$$

By sorting every column of $\boldsymbol{G}$, if distance between pairs of points i, j is smaller than certain threshold, then two points are considered as neighbors. On the contrary, two points are not considered as neighbors. Next step for updating weight is conducted based on:

$$d_G(i,j)=\begin{cases} d_X(i,j), if\ \|\boldsymbol{x}_i - \boldsymbol{x}_j\| < \epsilon \\ \infty, \qquad if\ \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq \epsilon \end{cases} \tag{6}$$

For $k \in \{1, 2, \ldots, N\}$, geodesic distances are calculated as followings:

$$d^c{}_G(i,j) = \min\{d_G(i,j),\ d_G(i,k) + d_G(k,j)\} \tag{7}$$

For finding the shortest path between every two nodes, Floyd's algorithm can be used ([29, 30]). Apply MDS to $\boldsymbol{D}_G = \{d_G(i,j)\}$, $Y = \{\boldsymbol{y}_i\}$,$\boldsymbol{y}_i \in R^d$ and minimize the cost function by setting the coordinates $\boldsymbol{y}_i$ to the top d eigenvectors of the matrix $\tau(\boldsymbol{D}_G)$.

$$E = \|\tau(\boldsymbol{D}_G) - \tau(\boldsymbol{D}_Y)\|_{L^2} \tag{8}$$

where: $\tau(\boldsymbol{D}_G) = -\frac{\boldsymbol{HSH}}{2}$
$\boldsymbol{S} = (s_{ij}) = \left(d^c{}_G(i,j)^2\right)$
$\boldsymbol{H} = (h_{ij}),\ h_{ij} = \delta_{ij} - \frac{1}{N}$
p-th eigenvalue: λ_p.
p-th eigenvector: $\boldsymbol{v}_p$.
p-th component of coordinate of $\boldsymbol{y}_p$, with $\boldsymbol{y}_p = \boldsymbol{v}_p\sqrt{\lambda_p}$. Low dimensional manifold is then: $\boldsymbol{Y} = \begin{bmatrix}\boldsymbol{y}_1 & \cdots & \boldsymbol{y}_d\end{bmatrix}^T$.

2.3. Locally Linear Embedding

High dimensional input data, N samples with dimensionality D is denoted as $\boldsymbol{X}$, with $\boldsymbol{X} = \begin{bmatrix}\boldsymbol{x}_1 \\ \cdots \\ \boldsymbol{x}_N\end{bmatrix} \in R^{N\times D}$.

For first step in Locally Linear Embedding, the set of neighbors for each data point can be determined by: choosing the K nearest neighbors by Euclidean distance, or choosing points within a fixed radius, or using prior knowledge [12].

For second step in Locally Linear Embedding, thus reconstruction with weights, matrix of weights $\boldsymbol{W}$ is calculated by minimizing reconstruction error:

$$\varepsilon(\boldsymbol{W}) = \sum_i \left|\boldsymbol{x}_i - \sum_j W_{ij}\boldsymbol{x}_j\right|^2 \tag{9}$$

Each data point is reconstructed from its neighbors and sum of each row of the weight matrix is one:

$$\sum_j W_{ij} = 1 \tag{10}$$

In the final step of Locally Linear Embedding, low dimensional manifold $\boldsymbol{Y}$ is determined by minimizing the embedding cost function:

$$\Phi(\boldsymbol{Y}) = \sum_i \left|\boldsymbol{y}_i - \sum_j W_{ij}\boldsymbol{y}_j\right|^2 \tag{11}$$

2.4. Laplacian Eigenmaps

Data in high dimensional space is denoted as $\boldsymbol{X}$, with $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_k \end{bmatrix} \in R^{l \times k}$.After constructing adjacency graph, a weight matrix is needed and denoted as $\boldsymbol{W}_{k \times k}$, with $\boldsymbol{W}_{k \times k} = (W_{ij})$.Weight between two points can be calculated either by a heat kernel or simple rule. By heat kernel, weight is calculated as:

$$W_{ij} = \begin{cases} e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{t}}, if \ \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 < \epsilon \\ 0 \quad , if \ \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \geq \epsilon \end{cases} \tag{12}$$

By simple rule, weight is calculated as:

$$W_{ij} = \begin{cases} 1 \, , if \ \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 < \epsilon \\ 0 \, , if \ \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \geq \epsilon \end{cases} \tag{13}$$

Diagonal weight matrix is denoted as $\boldsymbol{D}_{k \times k}$ and:

$$D_{ii} = \sum_j W_{ji} \tag{14}$$

Laplacian matrix is obtained by following way:

$$\boldsymbol{L}_{k \times k} = \boldsymbol{D}_{k \times k} - \boldsymbol{W}_{k \times k} \tag{15}$$

Eigenvectors $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{k-1}$ (corresponding eigenvalues $\lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{k-1}$) are solutions of equation 15, ordered according to their eigenvalues:

$$\boldsymbol{L}\boldsymbol{v} = \lambda \boldsymbol{D}\boldsymbol{v} \tag{16}$$

Where $\boldsymbol{v} \in R^{k \times 1}$. By removing $\boldsymbol{v}_0$ corresponding to $\lambda_0 = 0$, next m eigenvectors are used for embedding. Then dimensionality of k datapoints is reduced to m and $F_{k \times m} = (\mathbf{f}_\lambda)$,$\lambda \in \{1, 2, \ldots, m\}$.

2.5. t-SNE

As mentioned above, t-SNE is proposed based on SNE. For solving certain problems, such as difficulty of optimization in original SNE, t-SNE is proposed with computing the similarity by student-t distribution bot not Gaussian distribution. And a simpler gradient is used for minimizing sum of Kullback-leibler divergences.

Data in high dimensional space is denoted as $\boldsymbol{X}$, with $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_n \end{bmatrix} \in R^{h \times n}$. Low dimensional representation at step t is denoted

as $\boldsymbol{Y}^{(t)}$, with $\boldsymbol{Y}^{(t)} = \begin{bmatrix} \boldsymbol{y}_1^{(t)} & \cdots & \boldsymbol{y}_n^{(t)} \end{bmatrix} \in R^{d \times n}$. The similarity of datapoint $\boldsymbol{x}_j$ to datapoint $\boldsymbol{x}_i$ in high dimensional space is defined as:

$$p_{j|i} = \frac{\exp(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma_i^2})}{\sum_{k \neq i} \exp(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_k\|^2}{2\sigma_i^2})} \tag{17}$$

Where σ_i is the variance of the distribution that is centered on datapoint $\boldsymbol{x}_i$. Further:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{18}$$

The perplexity is defined as:

$$Perp\left(P_i\right) = 2^{H(P_i)} \tag{19}$$

Shannon entropy is:

$$H\left(P_i\right) = -\sum_j p_{j|i} \log_2 p_{j|i} \tag{20}$$

The similarity of $\boldsymbol{y}_i$ to $\boldsymbol{y}_j$ in manifold:

$$q_{ij} = \frac{(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)} \tag{21}$$

The sum of Kullback-leibler divergences over all datapoints is:

$$C = \sum_i KL(P|\,|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{22}$$

Gradient for minimizing C is then:

$$\frac{\partial C}{\partial \boldsymbol{y}_i} = 4 \sum_j (p_{ij} - q_{ij})(\boldsymbol{y}_i - \boldsymbol{y}_j)(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1} \tag{23}$$

For first step in t-SNE, compute p_{ij} of datapoint $\boldsymbol{x}_j$ to datapoint $\boldsymbol{x}_i$, and initiate $\boldsymbol{Y}^{(0)}$ randomly from $\mathcal{N}(0, 10^{-4}\boldsymbol{I})$. Then for $t \in \{1, 2, \ldots, T\}$, at every iteration, firstly, compute q_{ij}, then result is used for computing C and $\frac{\partial C}{\partial \boldsymbol{Y}^{(t)}}$, following update is then made on $\boldsymbol{Y}^{(t)}$:

$$\boldsymbol{Y}^{(t)} = \boldsymbol{Y}^{(t-1)} + \eta \frac{\partial C}{\partial \boldsymbol{Y}^{(t)}} + \alpha(t)(\boldsymbol{Y}^{(t-1)} - \boldsymbol{Y}^{(t-2)}) \tag{24}$$

Where:
η: learning rate,
T: number of iterations,
$\alpha(t)$: momentum.

3. Experiment

3.1. Methodology

3.1.1. Data sets

Stock prices of five companies, Amazon (AMZN), CME Group (CME), Microsoft (MSFT), Netflix (NFLX), Texas Instruments Incorporated (TXN) are chosen to build data sets. Totally, 5 datasets are used for conducting experiment. Raw data from 2021.05.26 to 2023.12.29, is obtained from following sources:

Raw data	**Source**
Stock price: Stock high prices Stock low prices	https://finance.yahoo.com/
Stock index: Dow Jones Industrial Average NASDAQ Composite	https://finance.yahoo.com/
Monetary and interest rate data: Federal Funds Effective Rate Monetary Base Bank Prime Loan Rate M1 M2 90-Day AA Financial Commercial Paper Interest Rate 30-Day, 60-Day Nonfinancial Commercial Paper Interest Rate 4-Week, 3-Month, 6-Month, 1-Year Treasury Bill Secondary Market Rate, Discount Basis Market Yield on U.S. Treasury Securities at 1-Month, 3-Month, 6-Month, 1-Year, 2-Year, 3-Year, 5-Year, 7-Year, 10- Year, 30-Year Constant Maturity	https://fred.stlouisfed.org/series

Market Yield on U.S. Treasury Securities at 5-Year, 7-Year, 10-Year, 20-Year, 30-Year Constant Maturity, Quoted on an Investment Basis, Inflation-Indexed	
Inflation, labor data: Consumer Price Index Unemployment Rate All Employees, Total Nonfarm	https://fred.stlouisfed.org/series
Exchange rate: EUR/USD GBP/USD USD/JPY USD/CNY	https://finance.yahoo.com/
Commodities: Gold Dec 24 (future) Crude Oil Dec 24 (future)	https://finance.yahoo.com/
Net income and Earnings per share	https://www.sec.gov/submit-filings

Table 1: Raw data and sources.

Five data sets are created based on collected raw data. Description of data sets is provided as the following:

Data set	number of samples	dimension
Stock AMZN	948	141
Stock CME	948	141
Stock MSFT	948	141
Stock NFLX	948	140
Stock TXN	948	141

Table 2. Description of data sets.

TimeSeriesSplit from sklearn is used for splitting original dataset to training and test data. For experiment, train test data size split is 80-20.

Data set	number of samples for training	number of samples for test
Stock AMZN	758	190
Stock CME	758	190
Stock MSFT	758	190
Stock NFLX	758	190

Stock TXN	758	190

Table 3: Training and test data.

Price trend label at day t is determined by:

$$s_t = \begin{cases} 1, \ if \ p_t^h > p_{t-1}^h, \ p_t^l > p_{t-1}^l \\ 0, \ if \ \left(p_t^h - p_{t-1}^h \right) \left(p_t^l - p_{t-1}^l \right) \leq 0 \\ -1, if \ p_t^h < p_{t-1}^h, \ p_t^l < p_{t-1}^l \end{cases} \tag{25}$$

Where:
p_t^h: high price at day t,
p_t^l: low price at day t.

3.1.2. Computation

Experiment is conducted on google Colab under configuration T4 GPU with system RAM 51.0 GB and GPU RAM 15.0 GB.

3.1.3. Sources of codes

Following Python implementations are used for conducting experiment. PCA: Python implementation from scikit-learn is used. Assessment on number of components for PCA : Python implementation (cross_val_score, FactorAnalysis) from scikit-learn is used. ISOMAP: Python implementation from scikit-learn is used. Locally Linear Embedding: Python implementation from scikit-learn is used. Laplacian Eigenmaps: Python implementation from scikit-learn is used. T-SNE: Python implementation from developer's site[2] is used with one modification for avoiding appearance of error. In function Hbeta, type is set as numpy.float128. When sumP is equal to zero, value is replaced by 10^{-10}. When sumP is positive infinity, value is replaced by 10^{10}. When sumP is negative infinity, value is replaced by -10^{10}. K-means: Python implementation from scikit-learn is used. Fully connected neural network: Python implementation from Keras is used.

3.1.4. Process of conducting experiment

For every original dataset, k-means and fully connected neural network are used to conduct clustering and classification on daily trend of stock price. Then by applying PCA, ISOMAP, Locally Linear Embedding, Laplacian Eigenmaps and t-SNE, dimension of original dataset is reduced. And newly

[2]https://lvdmaaten.github.io/tsne/

projected datasets are then used as input for k-means and fully connected neural network. Number of clusters for k-means is set as 3, because of price trend definition above. Architecture of fully connected neural network is configured as following:

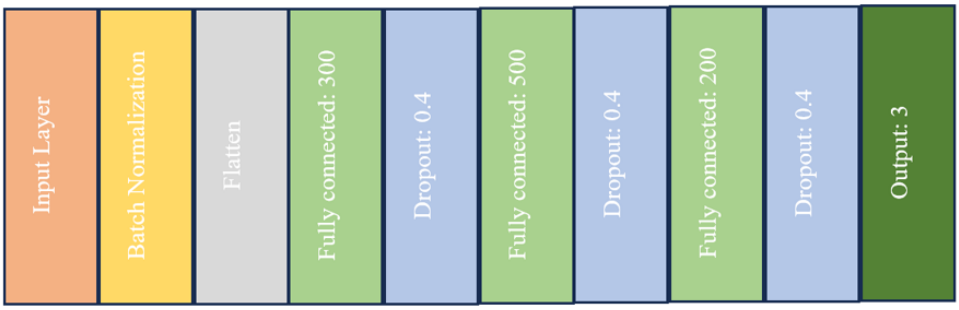

Fig. 1. Architecture of fully connected neural network

Neural network is trained by using method of gradient descent. Number of epochs for training neural network is 2000. Learning rate is 1e-06.

In experiment, dimension of manifold (n_c) is usually configured as 2, 50, 100, 140 except for PCA. Before conducting experiment for PCA, number of components is calculated based on two methods in 3.1.3. Calculated number of components is marked bold and represents recommended configuration of number of components on PCA. Choosing 2, 50, 100, 140 as dimension of manifold is because, firstly, for most examined manifold techniques, 2 is default configuration. This is because most manifold techniques are designed to achieve the goal of projecting high dimensional data on a 2D plane. Secondly, other dimension settings are used for revealing effect of manifold techniques more comprehensively. Every result on ACC (mean ± standard deviation) in experiment is obtained through 20 independent runs.

3.1.5. Metrics

ACC is used both for clustering and classification task. For clustering task, ACC is used instead of using ARI [6], which is a standard measurement for performance of clustering task. But for more straightforward comparison between clustering and classification task on prediction, ACC is used as a common measurement for performance. For clustering, best match between permutation of predicted labels and target labels is used for calculating ACC.

3.2. Experiment result

3.2.1. Result of k-means

Datasets	No dimension reduction	PCA			
Stock AMZN	0.47±0.008	**n_c = 3**	**n_c = 28**	n_c = 100	n_c = 140
		0.46±0.024	0.46±0.018	0.45±0.019	0.46±0.021
Stock CME	0.43±0.020	**n_c = 3**	**n_c = 29**	n_c = 100	n_c = 140
		0.42±0.017	0.43±0.017	0.43±0.019	0.43±0.018
Stock MSFT	0.45±0.013	**n_c = 3**	n_c = 28	n_c = 100	**n_c = 140**
		0.44±0.004	0.44±0.009	0.45±0.013	0.45±0.012
Stock NFLX	0.44±0.014	**n_c = 3**	**n_c = 31**	n_c = 100	n_c = 140
		0.44±0.012	0.44±0.016	0.44±0.012	0.45±0.007
Stock TXN	0.43±0.029	**n_c = 3**	**n_c = 29**	n_c = 100	n_c = 140
		0.43±0.022	0.46±0.018	0.45±0.024	0.44±0.033

Table 4. Clustering result on no dimension reduction and PCA.

Datasets	ISOMAP			
	n_c = 2	n_c = 50	n_c = 100	n_c = 140
Stock AMZN	0.44±0.008	0.43±0.029	0.43±0.034	0.42±0.032
Stock CME	0.38±0.005	0.39±0.014	0.39±0.017	0.38±0.016
Stock MSFT	0.43±0.010	0.45±0.016	0.44±0.019	0.44±0.018
Stock NFLX	0.45±1.110	0.42±0.033	0.42±0.027	0.42±0.027
Stock TXN	0.41±0.005	0.42±0.029	0.43±0.025	0.42±0.020

Table 5. Clustering result on ISOMAP.

Datasets	Locally Linear Embedding			
	n_c = 2	n_c = 50	n_c = 100	n_c = 140
Stock AMZN	0.44±0.004	0.43±0.011	0.43±0.006	0.43±0.005
Stock CME	0.39±1.110	0.40±0.009	0.40±0.011	0.40±0.006
Stock MSFT	0.41±0.001	0.43±0.018	0.43±0.011	0.43±0.006
Stock NFLX	0.41±0.004	0.41±0.011	0.41±0.007	0.41±0.007
Stock TXN	0.41±0.002	0.41±0.010	0.41±0.006	0.41±0.004

Table 6. Clustering result on Locally Linear Embedding.

Datasets	Laplacian Eigenmaps			
	n_c = 2	n_c = 50	n_c = 100	n_c = 140
Stock AMZN	0.45±0.002	0.43±0.014	0.43±0.013	0.43±0.011
Stock CME	0.41±5.551	0.40±0.015	0.40±0.014	0.39±0.014
Stock MSFT	0.44±0.002	0.42±0.021	0.41±0.019	0.42±0.023
Stock NFLX	0.42±0.001	0.40±0.014	0.41±0.019	0.41±0.014
Stock TXN	0.45±0.002	0.41±0.012	0.41±0.012	0.41±0.012

Table 7. Clustering result on Laplacian Eigenmaps.

Datasets	t-SNE			
	n_c = 2	n_c = 50	n_c = 100	n_c = 140
Stock AMZN	0.39±0.010	0.42±0.026	0.42±0.023	0.41±0.022
Stock CME	0.40±0.004	0.39±0.019	0.39±0.020	0.39±0.019
Stock MSFT	0.36±0.013	0.40±0.023	0.41±0.018	0.41±0.016
Stock NFLX	0.36±0.007	0.41±0.024	0.41±0.022	0.40±0.026
Stock TXN	0.39±0.014	0.42±0.024	0.43±0.029	0.43±0.020

Table 8. Clustering result on t-SNE.

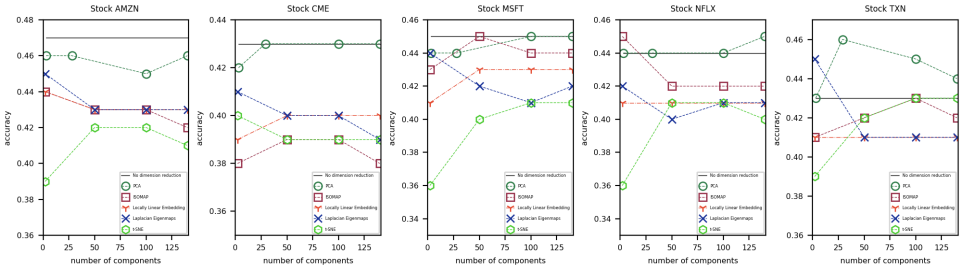

Fig. 2. Visualization of clustering results.

3.2.2. Results of neural network

Datasets	No dimension reduction	
	training	test
Stock AMZN	0.702±0.007	0.705±0.014
Stock CME	0.677±0.005	0.664±0.012
Stock MSFT	0.688±0.006	0.661±0.009
Stock NFLX	0.674±0.006	0.658±0.010
Stock TXN	0.684±0.007	0.689±0.011

Table 9: Classification result on no dimension reduction.

Datasets	PCA							
	training	test	training	test	training	test	training	test
Stock AMZN	**n_c = 3**		**n_c = 28**		n_c − 100		n_c − 140	
	0.670±0.003	0.667±0.001	0.682±0.009	0.679±0.008	0.694±0.007	0.698±0.012	0.697±0.008	0.707±0.011
Stock CME	**n_c = 3**		**n_c = 29**		n_c − 100		n_c − 140	
	0.668±0.002	0.663±0.000	0.670±0.006	0.668±0.007	0.674±0.006	0.664±0.009	0.675±0.007	0.659±0.008
Stock MSFT	**n_c = 3**		n_c − 28		n_c − 100		**n_c = 140**	
	0.666±0.002	0.664±0.003	0.676±0.008	0.643±0.009	0.686±0.006	0.657±0.009	0.688±0.008	0.657±0.008
Stock NFLX	**n_c = 3**		**n_c = 31**		n_c − 100		n_c − 140	
	0.666±0.002	0.664±0.001	0.667±0.006	0.654±0.006	0.671±0.007	0.659±0.008	0.675±0.007	0.656±0.009
Stock TXN	**n_c = 3**		**n_c = 29**		n_c − 100		n_c − 140	
	0.665±0.003	0.666±0.000	0.677±0.006	0.674±0.008	0.684±0.007	0.681±0.010	0.684±0.005	0.684±0.016

Table 10. Classification result on PCA.

Datasets	ISOMAP							
	n_c − 2		n_c − 50		n_c − 100		n_c − 140	
	training	test	training	test	training	test	training	test
Stock AMZN	0.670±0.003	0.667±0.003	0.677±0.006	0.666±0.012	0.684±0.007	0.666±0.011	0.684±0.007	0.668±0.007
Stock CME	0.666±0.002	0.666±0.002	0.663±0.008	0.660±0.007	0.666±0.005	0.657±0.008	0.664±0.007	0.656±0.008
Stock MSFT	0.669±0.003	0.662±0.002	0.671±0.006	0.637±0.012	0.672±0.007	0.640±0.010	0.672±0.006	0.638±0.011
Stock NFLX	0.667±0.003	0.664±0.003	0.664±0.008	0.648±0.008	0.666±0.008	0.648±0.009	0.668±0.010	0.645±0.007
Stock TXN	0.666±0.002	0.667±0.002	0.675±0.006	0.671±0.009	0.677±0.007	0.671±0.01	0.678±0.006	0.672±0.013

Table 11. Classification result on ISOMAP.

Datasets	Locally Linear Embedding							
	n_c − 2		n_c − 50		n_c − 100		n_c − 140	
	training	test	training	test	training	test	training	test
Stock AMZN	0.666±0.001	0.667±0.001	0.669±0.007	0.664±0.010	0.682±0.008	0.654±0.017	0.681±0.008	0.649±0.019
Stock CME	0.667±0.001	0.667±0.000	0.667±0.006	0.666±0.005	0.670±0.007	0.662±0.008	0.672±0.007	0.661±0.012
Stock MSFT	0.668±0.001	0.668±0.005	0.661±0.005	0.636±0.012	0.666±0.007	0.650±0.010	0.672±0.006	0.649±0.016
Stock NFLX	0.666±0.002	0.667±0.000	0.659±0.006	0.660±0.008	0.667±0.007	0.656±0.010	0.670±0.007	0.655±0.012
Stock TXN	0.665±0.001	0.667±0.001	0.666±0.006	0.666±0.010	0.669±0.007	0.662±0.012	0.666±0.008	0.649±0.014

Table 12. Classification result on Locally Linear Embedding.

Datasets	Laplacian Eigenmaps							
	n_c − 2		n_c − 50		n_c − 100		n_c − 140	
	training	test	training	test	training	test	training	test
Stock AMZN	0.667±0.000	0.667±0.000	0.666±0.002	0.667±0.000	0.662±0.006	0.667±0.000	0.658±0.008	0.666±0.000
Stock CME	0.667±0.000	0.667±0.000	0.666±0.001	0.667±0.000	0.666±0.001	0.667±0.000	0.664±0.003	0.667±0.000
Stock MSFT	0.667±0.000	0.667±0.000	0.666±0.003	0.666±0.001	0.662±0.005	0.666±0.002	0.663±0.006	0.667±0.001
Stock NFLX	0.667±0.000	0.667±0.000	0.666±0.002	0.667±0.000	0.664±0.004	0.667±0.000	0.663±0.004	0.667±0.000
Stock TXN	0.667±0.000	0.667±0.000	0.666±0.002	0.667±0.000	0.662±0.005	0.667±0.000	0.662±0.005	0.667±0.000

Table 13. Classification result on Laplacian Eigenmaps.

Datasets	t-SNE							
	n_c − 2		n_c − 50		n_c − 100		n_c − 140	
	training	test	training	test	training	test	training	test
Stock AMZN	0.664±0.004	0.667±0.000	0.681±0.010	0.678±0.011	0.682±0.009	0.684±0.009	0.688±0.009	0.685±0.008
Stock CME	0.665±0.002	0.667±0.000	0.661±0.007	0.666±0.010	0.662±0.007	0.669±0.013	0.665±0.005	0.658±0.012
Stock MSFT	0.664±0.004	0.666±0.004	0.668±0.006	0.654±0.010	0.670±0.006	0.647±0.012	0.673±0.006	0.651±0.010
Stock NFLX	0.666±0.003	0.665±0.005	0.669±0.007	0.637±0.011	0.669±0.007	0.638±0.008	0.667±0.006	0.634±0.011
Stock TXN	0.664±0.004	0.667±0.002	0.668±0.006	0.676±0.01	0.673±0.007	0.676±0.010	0.670±0.007	0.672±0.009

Table 14. Classification result on t-SNE.

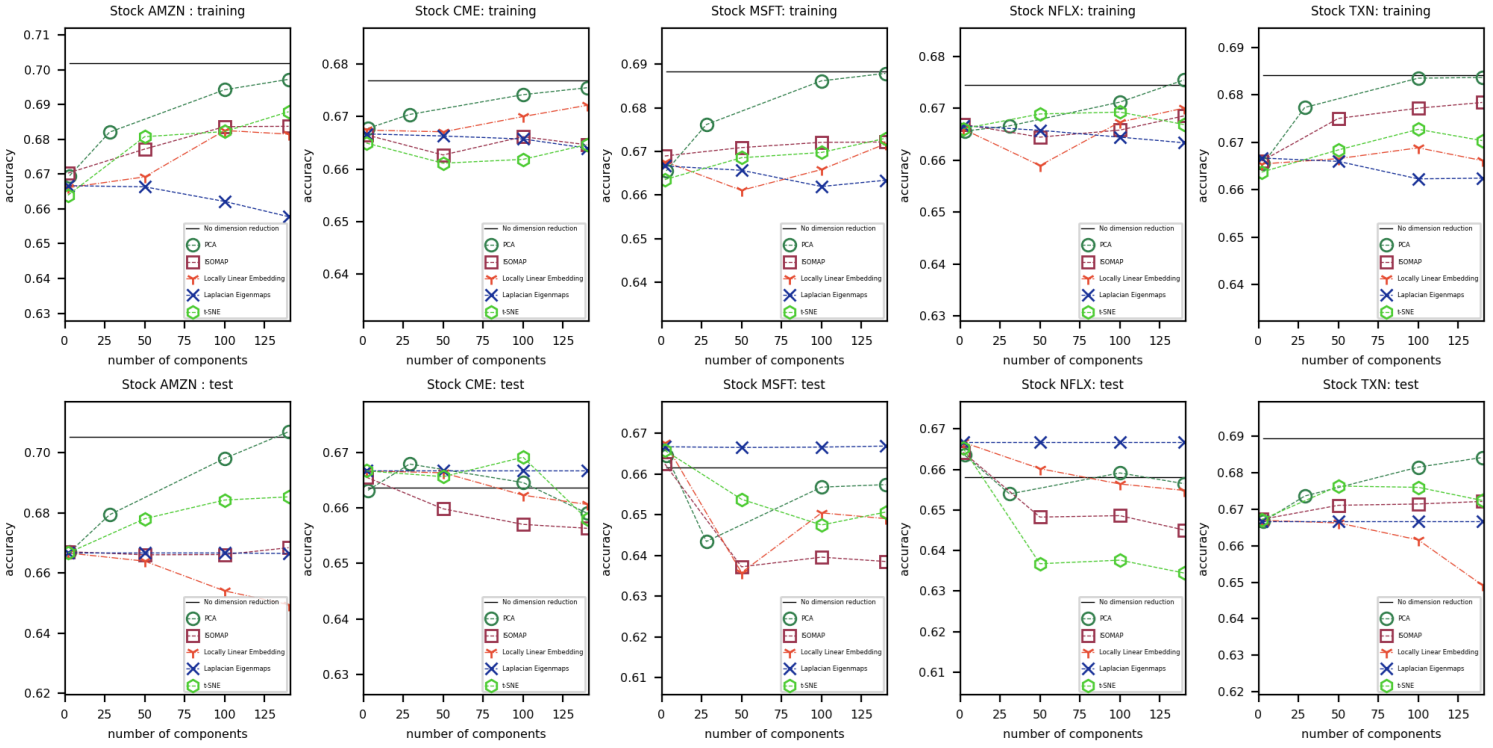

Fig. 3. Visualization of classification results

3.3. Discussion

3.3.1. K-means

Clustering results obtained from PCA generally can surpass other dimension reduction techniques. But it should be noticed that best result of PCA doesn't always correspondent to assessed number of components by methods mentioned in 3.1.3. Generally speaking, experiment results have shown that dimension reduction techniques often tend to worsen clustering performance of k-means. For all methods, there is at least one observation for not worsening clustering performance except Locally Linear Embedding. There is no observation for not worsening clustering performance for Locally Linear Embedding.

3.3.2. Neural network

For dataset AMZN, almost all methods tend to decrease neural network performance. Only PCA slightly improves neural network performance with number of components equal to 140. For dataset CME, most methods tend to slightly improve neural network performance. Improvement is generally less than 1 percent. For PCA, improvement of neural network performance is achieved at n_c= 29 and n_c = 100. For ISOMAP, improvement of performance is achieved at n_c = 2. Improvement of performance for Locally Linear Embedding is achived at n_c = 2 and n_c= 50. Improved performance for Laplacian Eigenmaps is achieved at all number of components. T-SNE improves performance of classification at n_c = 2, 50, 100 with best improvement among all method at n_c = 100. ISOMAP tend to more often decrease performance than other methods. For dataset MSFT, all methods tend to decrease classification accuracy except Laplacian Eigenmaps. Laplacian Eigenmaps can stably improve performance at all number of components. Other methods can all improve performance at lowest number of components but to worsen performance at other number of components. For dataset NFLX, all methods can all improve performance at lowest number of components. And PCA and Locally Linear Embedding can archive improvement correspondingly at n_c = 100 and n_c = 5. Again, Laplacian Eigenmaps can stably improve performance at all number of components. For dataset TXN, all methods worsen performance without exceptions. Generally speaking, improvement achieved by all methods is slight, usually less than 1 percent. For different dataset, improved performance by different methods is different. For best method, Laplacian Eigenmaps should be mentioned, as Laplacian Eigenmaps can stably improve classification accuracy for three datasets at all number of components.

3.3.3. Total result

The conclusion on PCA from clustering results of k-means supports the findings in [5]. Although result from PCA generally can be better than other dimension reduction techniques, but it's also observed that dimension reduction techniques often tend to worsen clustering performance of k-means. For classification task, observed improvement achieved by all methods is slight, usually less than 1 percent. Laplacian Eigenmaps can stably improve classification accuracy for three datasets at all number of components while other methods aren't able to achieve such performance. One assumption based on experiment can be raised: structures of data sets containing different stock prices can be very different. And there is no proof for that there exists one effective method for different stock data sets. Even the dimensions of data sets used in this experiment are almost same (except stock NFLX). And macroeconomic features of data sets are largely same. Thus, performance of examined techniques on stock data sets looks like the general performance of machine learning algorithms on different natural data sets.

3.3.4. Advantages and limitations

This work provides a more comprehensive study on manifold leaning techniques for financial data than previous works. Firstly, more methods are examined. Secondly, study has appropriately organized advantages of previous works, so conclusions are more convincing. In previous works, disadvantage on organizing experiment exists either in methodology of building dataset (for example, risk free rate is not considered) or type of examined methods (for example, only methods that are related with PCA are considered). Disadvantage also exists on the fact that only one of two tasks (clustering and classification) has been involved on drawing conclusion, but not both. These insufficiencies have negatively affected reached conclusions by these works. All mentioned insufficiencies have been properly addressed in this work. Future work can be improved further by involving more manifold leaning techniques that haven't been examined in this work.

4. Conclusion

In this work, effect of manifold learning techniques on predicting stock price trend is more carefully examined than previous works. Effect of manifold learning techniques on improving classification task result is slightly more obvious than that on clustering task. In clustering task, manifold learning techniques tend to often worsen clustering performance. In classification task, observed improvement achieved by all methods is slight, usually less than 1 percent. And only Laplacian Eigenmaps can more often stably

improve classification accuracy at all number of components while other methods aren't able to achieve such performance. There is no general effective technique for different stock data set. This result reminds about general performance of machine learning algorithms on different natural data sets, dimension of which can be very different.

Code and data availability

Code and data are available upon request.

Acknowledgments

This work is supported by the China Scholarship Council (grant no. 202308090103). I want to thank my scientific supervisor (Сидоренко Владимир Николаевич, к.ф.-м.н., к.э.н., к.ю.н.) for advices on improving quality of this work.

References

[1] Zhong, Xiao and Enke, David, "Forecasting daily stock market return using dimensionality reduction", *Expert Systems with Applications*, **67** (jan 2017), 126–139, http://dx.doi.org/10.1016/j.eswa.2016.09.027.

[2] Han, Jingti and Ge, Zhipeng, "Effect of dimensionality reduction on stock selection with cluster analysis in different market situations", *Expert Systems with Applications*, **147** (jun 2020), 113226, http://dx.doi.org/10.1016/j.eswa.2020.113226.

[3] Van Der Maaten, Laurens and Postma, Eric O and Van Den Herik, H Jaap and others, "Dimensionality reduction: A comparative review", *Journal of machine learning research*, **10**:66-71 (2009), 13.

[4] Luukka, Pasi, "A New Nonlinear Fuzzy Robust PCA Algorithm and Similarity Classifier in Classification of Medical Data Sets.", *International Journal of Fuzzy Systems*, **13**:3 (2011).

[5] Yeung, K. Y. and Ruzzo, W. L., "Principal component analysis for clustering gene expression data", *Bioinformatics*, **17**:9 (sep 2001), 763–774, http://dx.doi.org/10.1093/bioinformatics/17.9.763.

[6] Hubert, Lawrence and Arabie, Phipps, "Comparing partitions", *Journal of Classification*, **2**:1 (dec 1985), 193–218, http://dx.doi.org/10.1007/bf01908075.

[7] Hotelling, H., "Analysis of a complex of statistical variables into principal components.", *Journal of Educational Psychology*, **24**:6 (sep 1933), 417–441, http://dx.doi.org/10.1037/h0071325.

[8] Pearson, Karl, "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*, **2**:11 (nov 1901), 559–572, http://dx.doi.org/10.1080/14786440109462720.

[9] Abdi, Hervé and Williams, Lynne J., "Principal component analysis", *WIREs Computational Statistics*, **2**:4 (jul 2010), 433–459, http://dx.doi.org/10.1002/wics.101.

[10] Tenenbaum, Joshua B. and Silva, Vin de and Langford, John C., "A Global Geometric Framework for Nonlinear Dimensionality Reduction", *Science*, **290**:5500 (dec 2000), 2319–2323, http://dx.doi.org/10.1126/science.290.5500.2319.

[11] Cox, M. A. A. and Cox, T. F., "Multidimensional Scaling on the Sphere", *Compstat*, Physica-Verlag HD, 1988, 323–328, ISBN: 9783642469008, http://dx.doi.org/10.1007/978-3-642-46900-8_44.

[12] Roweis, Sam T. and Saul, Lawrence K., "Nonlinear Dimensionality Reduction by Locally Linear Embedding", *Science*, **290**:5500 (dec 2000), 2323–2326, http://dx.doi.org/10.1126/science.290.5500.2323.

[13] Belkin, Mikhail and Niyogi, Partha, "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation", *Neural Computation*, **15**:6 (jun 2003), 1373–1396, http://dx.doi.org/10.1162/089976603321780317.

[14] von Luxburg, Ulrike, "A tutorial on spectral clustering", *Statistics and Computing*, **17**:4 (aug 2007), 395–416, http://dx.doi.org/10.1007/s11222-007-9033-z.

[15] Ding, Ling and Li, Chao and Jin, Di and Ding, Shifei, "Survey of spectral clustering based on graph theory", *Pattern Recognition*, **151** (jul 2024), 110366, http://dx.doi.org/10.1016/j.patcog.2024.110366.

[16] Breger, Anna and Karner, Clemens and Ehler, Martin, "visClust: A visual clustering algorithm based on orthogonal projections", *Pattern Recognition*, **148** (apr 2024), 110136, http://dx.doi.org/10.1016/j.patcog.2023.110136.

[17] Coifman, R. R. and Lafon, S. and Lee, A. B. and Maggioni, M. and Nadler, B. and Warner, F. and Zucker, S. W., "Geometric diffusions as

a tool for harmonic analysis and structure definition of data: Diffusion maps", *Proceedings of the National Academy of Sciences*, **102**:21 (may 2005), 7426–7431, http://dx.doi.org/10.1073/pnas.0500334102.

[18] Lafon, S. and Lee, A.B., "Diffusion maps and coarse-graining: a unified framework for dimensionality reduction, graph partitioning, and data set parameterization", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**:9 (sep 2006), 1393–1403, http://dx.doi.org/10.1109/tpami.2006.184.

[19] Nadler, Boaz and Lafon, Stéphane and Coifman, Ronald R. and Kevrekidis, Ioannis G., "Diffusion maps, spectral clustering and reaction coordinates of dynamical systems", *Applied and Computational Harmonic Analysis*, **21**:1 (jul 2006), 113–127, http://dx.doi.org/10.1016/j.acha.2005.07.004.

[20] DeMers, David and Cottrell, Garrison, "Non-linear dimensionality reduction", *Advances in neural information processing systems*, **5** (1992).

[21] Hinton, G. E. and Salakhutdinov, R. R., "Reducing the Dimensionality of Data with Neural Networks", *Science*, **313**:5786 (jul 2006), 504–507, http://dx.doi.org/10.1126/science.1127647.

[22] Weinberger, Kilian Q. and Saul, Lawrence K., "Unsupervised Learning of Image Manifolds by Semidefinite Programming", *International Journal of Computer Vision*, **70**:1 (may 2006), 77–90, http://dx.doi.org/10.1007/s11263-005-4939-z.

[23] Weinberger, Kilian Q. and Sha, Fei and Zhu, Qihui and Saul, Lawrence K., "Graph Laplacian Regularization for Large-Scale Semidefinite Programming", *Advances in Neural Information Processing Systems 19*, The MIT Press, sep 2007, 1489–1496, ISBN: 9780262256919, http://dx.doi.org/10.7551/mitpress/7503.003.0191.

[24] Sun, Jun and Boyd, Stephen and Xiao, Lin and Diaconis, Persi, "The Fastest Mixing Markov Process on a Graph and a Connection to a Maximum Variance Unfolding Problem", *SIAM Review*, **48**:4 (jan 2006), 681–699, http://dx.doi.org/10.1137/s0036144504443821.

[25] Weinberger, Kilian Q. and Sha, Fei and Saul, Lawrence K., "Learning a kernel matrix for nonlinear dimensionality reduction", *Twenty-first international conference on Machine learning - ICML '04*, ACM Press, 2004, 106, http://dx.doi.org/10.1145/1015330.1015345.

[26] Van der Maaten, Laurens and Hinton, Geoffrey, "Visualizing data using t-SNE.", *Journal of machine learning research*, **9**:11 (2008).

[27] Hinton, Geoffrey E and Roweis, Sam, "Stochastic neighbor embedding", *Advances in neural information processing systems*, **15** (2002).

[28] Hinton, Geoffrey E. and Osindero, Simon and Teh, Yee-Whye, "A Fast Learning Algorithm for Deep Belief Nets", *Neural Computation*, **18**:7 (jul 2006), 1527–1554, http://dx.doi.org/10.1162/neco.2006.18.7.1527.

[29] Floyd, Robert W., "Algorithm 97: Shortest path", *Communications of the ACM*, **5**:6 (jun 1962), 345–345, http://dx.doi.org/10.1145/367766.368168.

[30] Warshall, Stephen, "A Theorem on Boolean Matrices", *Journal of the ACM*, **9**:1 (jan 1962), 11–12, http://dx.doi.org/10.1145/321105.321107.

[31] Schölkopf, Bernhard and Smola, Alexander and Müller, Klaus-Robert, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem", *Neural Computation*, **10**:5 (jul 1998), 1299–1319, http://dx.doi.org/10.1162/089976698300017467.