Report for Final Project from group J

Tasks addressed:	0
Authors:	Wenbin Hu (03779096) Yilin Tang (03755346) Mei Sun (03755382) Daniel Bamberger (03712890)
Last compiled:	2023–07–17
Source code:	https://github.com/HUWENBIN2024/TUMCrowdModelingGroupJ/tree/main/final_proj

The work on tasks was divided in the following way:

Wenbin Hu (03779096)	Task 1	1/3
	Task 2	1/3
	Task 3	1/3
	Task 4	1/3
Yilin Tang (03755346)	Task 1	1/3
	Task 2	1/3
	Task 3	1/3
	Task 4	1/3
Mei Sun (03755382)	Task 1	1/3
	Task 2	1/3
	Task 3	1/3
	Task 4	1/3
Daniel Bamberger (03712890)	Task 1	0
	Task 2	0
	Task 3	0
	Task 4	0

Abstract

The **Curse of Dimensionality** long stills in Data Science and Machine Learning caused by the sparseness of data. It will be beneficial to reduce the data dimension for learning efficacy and efficiency. **Manifold Hypothesis** suggests that high-dimensional data often lies on or near low-dimensional manifolds within the larger space. Motivated by it, researchers have developed some dimensionality reduction methods. In this report, we show some popular methods (**PCA, Diffusion MAP, VAE**) and compare them with theoretical analysis and comprehensive experiments.

1 Datasets

1.1 Non-linear manifold: Swiss Roll

The Swiss Roll dataset is a well-known example in machine learning and computer graphics. It consists of a three-dimensional dataset that is rolled into a spiral shape, as shown in Fig.1(a). It is used to evaluate dimensionality reduction algorithms and assess their ability to separate the spirals while preserving local structure. It poses challenges for traditional analysis techniques due to its non-linear nature. Researchers utilize the Swiss Roll dataset to explore algorithm capabilities and compare different approaches in machine learning. It serves as a valuable benchmark for handling complex, non-linearly separable data.

1.2 Word2Vec

Word2vec[5] represents words as word embeddings in a continuous space, which can preserve semantic information. The main idea behind Word2Vec is to learn distributed representations of words based on their co-occurrence patterns in large text corpora. The key insight is that words with similar meanings tend to appear in similar contexts. Thus, Word embeddings with similar meaning should be close in the vector space. There are 2 popular ways to train word embeddings: CBOW and Skip-Gram. The CBOW model aims to predict a target word from its surrounding context words, and the Skip-Gram model, on the other hand, takes a single word as input and tries to predict the context words around it. We introduce more details about Skip-Gram. We denote word embeddings as $w_1, w_2, \ldots, w_T$ for T words. The objective of Skip-Gram is to maximize the log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq} logP(w_{t+j}|w_t)$$

, where c is the size of the context window. The probability $P(w_i|w_j)$ is defined as

$$P(w_i|w_j) = \frac{exp(w_i^\intercal w_j)}{\sum_{k=1}^{K} exp(exp(w_k^\intercal w_j)}$$

, where K is the size of the vocabulary. After training, similar words are represented by vectors that are closer in the vector space, and analogies can be represented as vector arithmetic (e.g., "king" - "man" + "woman" results in a vector representation close to "queen"). More examples are shown in Fig.1(b).

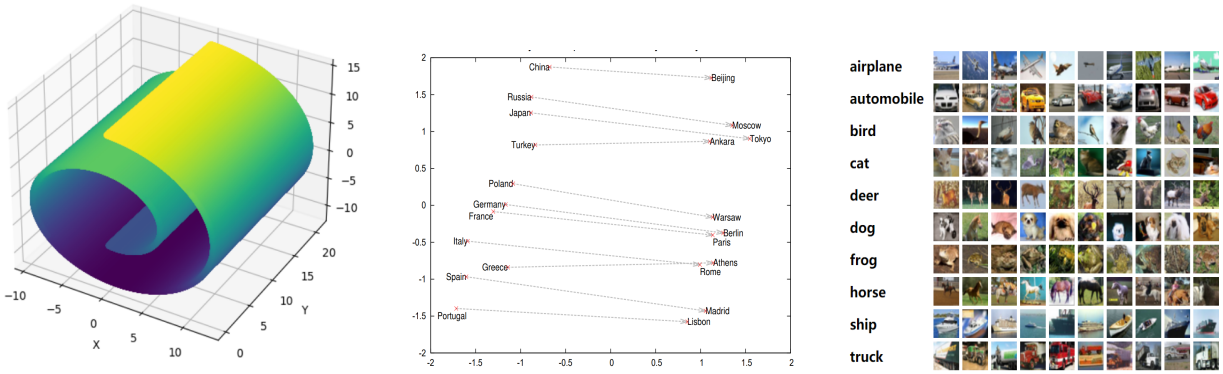
In this project, we adopt pre-trained embeddings trained on a part of the Google News dataset (about 100 billion words). The vocabulary contains 3 million words which are represented as 300-dimension embeddings. We select a part of the embeddings and conduct dimensionality reduction on them.

1.3 Image Dataset: CIFAR10

The **CIFAR-10** (Canadian Institute for Advanced Research) dataset is a widely used dataset in computer vision. It comprises a collection of 60,000 images, each of size 32x32x3, representing color images. The dataset is divided into ten distinct classes, with 6,000 images per category. The classes include airplanes, cars, deer, dogs, frogs, horses, ships, and trucks. The dataset is further split into a training set of 50,000 images and a test set of 10,000 images. As Fig.1(c) shows 10 random images from each category.

In the case of images in the CIFAR-10 dataset, each image is represented by a 32x32x3 pixel, resulting in a high-dimensional feature space of 3,072. However, the curse of dimensionality states that as the number of dimensions increases, the data points tend to become more sparse and separate from each other. This can pose challenges for data analysis and machine learning tasks. By reducing the dimensionality, we can reveal

the intrinsic structure and relationships within the data, making it easier to interpret and extract meaningful insights.



(a) (b) (c)

Figure 1: [a] The Swiss Roll dataset [b] PCA projection of Word2Vec embeddings [c] 10 random images from each class.[3]

2 Methodology

2.1 PCA: Principal Component Analysis

Motivation. Principal Component Analysis (PCA) is a powerful technique used for data representation and manifold learning. It linearly decomposes the data and assumes the data points follow a multi-variate normal distribution or are close to a hyperplane. PCA decomposes the data into principal components that capture the most significant variations. As Fig.2 shows, the data is transferred from the original space to the principal component space with PCA. It is helpful for dimensionality reduction, feature extraction, data compression, and data visualization, providing valuable insights into complex datasets.

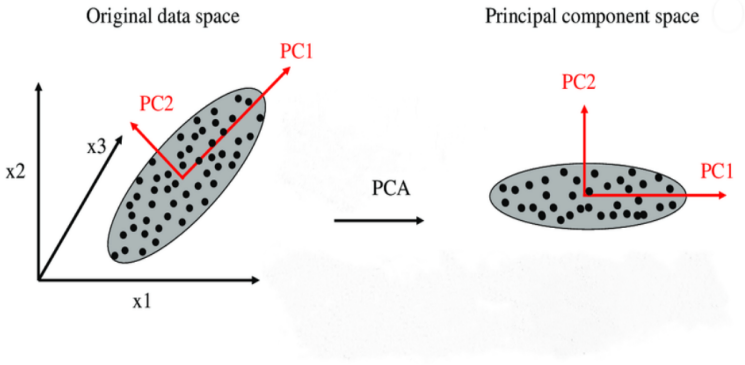


Figure 2: Transferring data from original space to principal component space with PCA

Algorithm. The PCA algorithm can be summarized as follows:

Given a dataset with N data points represented by a d-dimensional feature matrix $\mathbf{X}$, the goal is to find a lower-dimensional projection matrix $\mathbf{U}$ that maps the data onto a k-dimensional subspace, where $k < d$.

1. Center the data: Subtract the mean of each feature from the corresponding data points to center the data around the origin.

$$\mathbf{X}_{\text{centered}} = \mathbf{X} - \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{1}$$

2. Compute the covariance matrix: Calculate the covariance matrix $\mathbf{S}$ as the inner product of the centered data matrix.

$$\mathbf{S} = \mathbf{X}_{\text{centered}}^T \mathbf{X}_{\text{centered}} \tag{2}$$

3. Perform eigenvalue decomposition: Compute the eigenvectors $\mathbf{U}$ and eigenvalues λ of the covariance matrix $\mathbf{S}$.

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i, \quad i = 1, 2, \ldots, d \tag{3}$$

4. Select the top-k eigenvectors: Choose the k eigenvectors corresponding to the largest eigenvalues to form the projection matrix $\mathbf{U}$.

5. Transform the data: Project the centered data onto the lower-dimensional subspace using the projection matrix.

$$\mathbf{X}_{\text{pca}} = \mathbf{X}_{\text{centered}} \mathbf{U} \tag{4}$$

where $\mathbf{X}_{\text{pca}}$ represents the transformed data.

The PCA algorithm enables dimensionality reduction by retaining the most informative features of the data in the lower-dimensional subspace defined by the selected eigenvectors.

2.2 DMAP: Diffusion Maps

Motivation. Diffusion Maps are a non-linear algorithm, which focuses on discovering the underlying manifold. Real-world data is likely to form a highly nonlinear manifold, this nonlinear property allows it has the ability to process complex nonlinear data. Diffusion maps leverage the concept of random walks to uncover the underlying geometric structure within a dataset. By performing this random walk for multiple timesteps, a measure of proximity between data points is derived. This measure gives rise to the diffusion distance, which quantifies the similarity between data points in a low-dimensional representation. The diffusion distance is designed to capture the pairwise relationships between data points in the reduced-dimensional space. It is achieved by considering all possible paths through the data graph and integrating information from these paths. This diffusion process makes the diffusion distance more resilient to short-circuiting phenomena compared to other distance metrics. Below Fig.3 shows that the red line captures the local geometry assigned with high probability and the green line ignores the geometry structure assigned with lower probability. Using this technique allows one to discover lower dimensional manifolds as well as disconnected clusters in data.

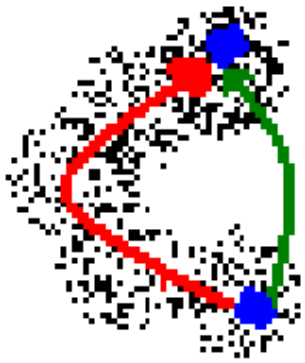


Figure 3: Paths along the true geometric structure of the dataset have high probability[1]

Mathematical Deduction

2.2.1 Connectivity

In the simulation of a random walk on our data, the connectivity between two data points x and y is defined as the probability of jumping from x to y in one step of the random walk. $connectivity(x, y) = p(x, y)$. we could also define a symmetric quantity $k(x, y)$ often known as kernel to obtain a local measure of similarity within a certain neighborhood. So, outside the neighborhood, the connectivity quickly goes to zero. One popular kernel is considered the Gaussian Kernel.

$$k(x, y) = exp\left(-\frac{d(x, y)^2}{\epsilon}\right) \tag{5}$$

Where d(x,y) is usually the Euclidean distance and ϵ is the range of the neighborhood. Then the relation between the kernel k(x,y) and the connectivity p(x,y) is then:

$$p(x,y) = \frac{1}{d_X} K(x,y), d_X = \sum_{y \in X} K(x,y) \tag{6}$$

Then we could define a matrix P with entries $p_{ij} = p(x_i, x_j)$,. Suppose we have two data points in a dataset, the matrix will look like the following:

$$\begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix} \tag{7}$$

This matrix P simply summarizes the probability of jumping from one point to another in one timestep. If we take the square to P, then

$$P^2 = \begin{pmatrix} P_{11}P_{11} + P_{12}P_{21} & P_{11}P_{12} + P_{12}P_{22} \\ P_{21}P_{11} + P_{22}P_{21} & P_{22}P_{22} + P_{21}P_{212} \end{pmatrix} \tag{8}$$

The first row and the first column of P^2 correspond to the probability of starting at point 1 and ending at point 1 in two jumps, then we have two choices to implement this: either just stay at point1 ($p_{11}p_{11}$) or jumps to point 2 and then backs ($p_{12}p_{21}$). Generally, P^t_{ij} sums all paths of length t from point i to point j.

2.2.2 Diffusion Process

When we increased the value of t (path length or timestep), the probability of following a path along the underlying geometric structure of the data set increases. This happens because, along the geometric structure, points are dense and therefore highly connected. Pathways form along short, high-probability jumps. On the other hand, paths that do not follow this structure include one or more long, low-probability jumps, which lowers the path's overall probability.[1] Illustrated at Figure3. This means that even if two points are far from Euclidean distance, we hope that their diffusion distance (next subsection) should be small if they are well connected. By doing so, the diffusion process can reveal the global geometric structure of a given dataset.

2.2.3 Diffusion Distance

Based on this structure, a diffusion matrix is defined to fulfill the above criterion.

$$D_t(x_i, x_j)^2 = \sum_{u \in X} |p_t(x_i, x_u) - p_t(x_j, x_u)|^2 = \sum_{u \in X} |P^t_{iu} - P^t_{ju}|^2 \tag{9}$$

The diffusion distance is small if there are many high probability paths of length between two data points, Since it sums all possible paths of length t it's robust to noise perturbation. The below Figure 4 shows the different cases for the computation of diffusion distances.

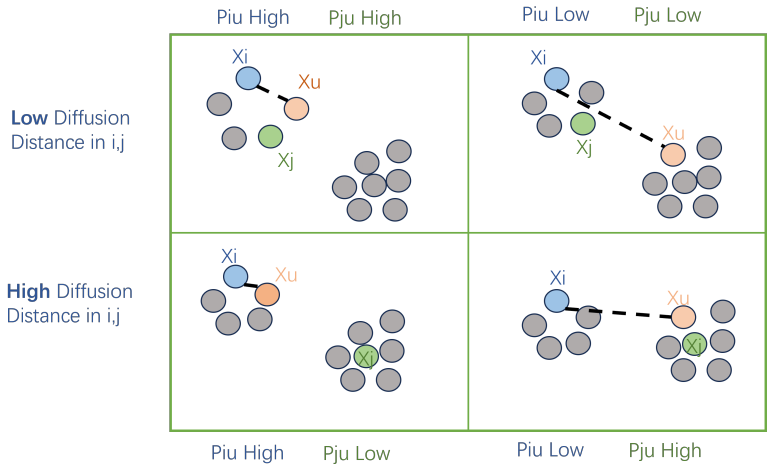


Figure 4: 4 different cases for computing diffusion distances

2.2.4 Diffusion MAP

Calculating diffusion distances is computationally expensive. Therefore it's efficient to map data points into Euclidean space according to the diffusion metric. We define vectors Y_i as the collection of all connectives leading to point x_i.

$$Y_i = \begin{pmatrix} p_t(x_i, x_1) \\ p_t(x_i, x_2) \\ ... \\ p_t(x_i, x_N) \end{pmatrix} \tag{10}$$

After that, we notice that the Euclidean distance $\|\cdot\|$ between these vectors corresponds to the diffusion distance between the original data point,

$$\|Y_i - Y_j\|_E^2 = \sum_{u \in X} \left|p_t\left(x_i, x_u\right) - p_t\left(x_j, x_u\right)\right|^2 = \sum_k \left|P_{ik}^t - P_{kj}^t\right|^2 = D_t\left(x_i, x_j\right)^2 \tag{11}$$

By doing so, we finish the map between the Euclidean distance in diffusion space (Y) and the diffusion distance in the original data space (X).
Finally, the last step is the implementation of dimensionality reduction. Now we have to consider how to compute the matrix P^t. we could first diagonalize the matrix (like SVD.) $P = Q^{-1}\Lambda Q$ where Λ is the diagonal matrix with eigenvalues of P, Q contains the eigenvectors of P. Then $P^t = Q^{-1}\Lambda^t Q$.
Thus, the diffusion distances could be expressed as:

$$Y_i = \begin{pmatrix} \lambda_1^t \Psi_{1,i} \\ \lambda_2^t \Psi_{2,i} \\ ... \\ \lambda_N^t \Psi_{N,i} \end{pmatrix} \tag{12}$$

where $\Psi_{1,i}$ is the i'th element of the first eigenvector of P. λ_1 is the corresponding eigenvalues of P. Based on some mathematical computation, the eigenvalues of P all lie in the range of 0 to 1. In fact, usually only a few eigenvalues are close to 1 whereas all others are small. By dropping the relatively small eigenvalues and keeping the top largest eigenvalues the dimensionality reduction is implemented.

2.3 VAE: Variational Autoencoder

Motivation. Autoencoders are unsupervised learning models that aim to learn a compact representation of the input data. They consist of an encoder network that maps the input data to a latent space and a decoder network that reconstructs the input data from the latent representation. However, traditional autoencoders have certain limitations, such as the lack of control over the latent space and the generation of unrealistic samples. **Variational Autoencoders (VAEs)**[2] were introduced to address these flaws and offer a more powerful generative modeling framework. The key idea behind VAEs is to introduce probabilistic modeling into the latent space of autoencoders. Instead of directly mapping the input data to a fixed latent representation, VAEs learn the parameters of a probability distribution in the latent space. This allows for a more flexible and expressive representation of the input data.

In this project, we use VAE with a low-dimension prior for dimensionality reduction. We use pytorch to implement VAE, and the code is in our repository.
Model Architecture. VAE model contains two parts: An encoder $q_\phi(z|x)$ and a decoder $p_\theta(x|z)$, which can be modeled by neural networks. The details are shown in Fig.5.

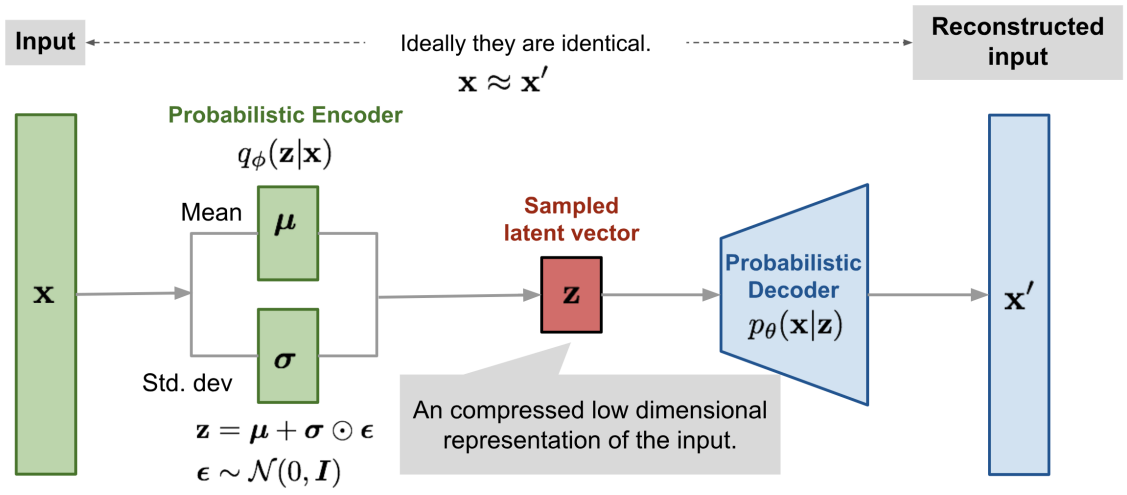


Figure 5: An illustration of the Variational AutoEncoder model.[6]

Model Training. During training, VAEs optimize two objectives simultaneously. The first objective is the reconstruction loss, which measures the difference between the input data and the reconstruction generated by the decoder network. The second objective is the Kullback-Leibler (KL) divergence, which encourages the learned latent distribution to resemble a predefined prior distribution (usually a multivariate Gaussian). The KL divergence regularizes the latent space and helps prevent overfitting. Specifically, the loss of VAE is:

$$\begin{aligned} L(\theta, \phi) &= -\log p_\theta(\mathbf{x}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\ &= -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \\ \theta^*, \phi^* &= \arg\min_{\theta, \phi} L(\theta, \phi) \end{aligned}$$

Mathematical Deduction
Our goal is to derive $p_\theta(x)$, which is hard to be solved directly. An idea is to match it to another distribution $p(z)$, where z can be easily used to control the generation process. By Bayes formula, we can find an easy association between x and z:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$$

Though the math is simple and beautiful, this formula is usually intractable since it is computationally expensive to sample lots of z from $p(z)$. Researchers find that we can derive a low bound of $p(x)$ and optimize the low bound, which is called **Evidence Lower BOund(ELBO)**. Now we show how to derive it.

$$\begin{aligned} \log p_\theta(x) &= \log \frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \\ &= \log \frac{p_\theta(x|z) p_\theta(z) q_\phi(z|x)}{p_\theta(z|x) q_\phi(z|x)} \\ &= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p_\theta(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)} \end{aligned}$$

Thus,

$$\begin{aligned} \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(x) &= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(x|z) - \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{q_\phi(z|x)}{p_\theta(z)} + \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{q_\phi(z|x)}{p_\theta(z|x)} \\ &= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(x|z) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x}) \end{aligned}$$

Then, we can get this equation:

$$\log p_\theta(x) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(x|z) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$$

The LHS of the equation is exactly what we want to optimize: $\log p_\theta(x)$ is the log-likelihood of the data we need to maximize, and $D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})$ represents the difference between the real and the estimated posterior, which we need to minimize. We can also derive the inequality:

$$\log p_\theta(x) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(x|z) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) = L_{ELBO}$$

, since any KL-divergence is greater or equal to 0. This inequality reveals that L_{ELBO} is a lower bound of the data distribution $p_\theta(x)$. Thus we can use L_{ELBO} to optimize the generation of data.

KL-divergence of 2 Gaussian Distribution. The KL-divergence of 2 Gaussian Distribution $N(\mu_1, \sigma_1)$, $N(\mu_2, \sigma_2)$ is:

$$D_{\mathrm{KL}}(N(\mu_1, \sigma_1)\|N(\mu_2, \sigma_2)) = \log \frac{\sigma_2}{\sigma_1} - \frac{1}{2} + \frac{\sigma_1^2}{2\sigma_2^2} + \frac{(\mu_1 - \mu_2)^2}{2\sigma_2^2}$$

For VAE, we usually use $N(0, 1)$ as the prior $p_\theta(x)$. With the equation showed above, the KL-divergence of $N(\mu, \sigma)$ and $N(0, 1)$ is:

$$D_{\mathrm{KL}}(N(\mu, \sigma)\|N(0, 1)) = -\log \sigma - \frac{1}{2} + \frac{\sigma^2}{2} + \frac{\mu^2}{2}$$

Empirical Loss. Practically, we use the loss:

$$L = \sum_{X\in B} [\|X - \hat{X}\|_2^2 + (-\log \sigma - \frac{1}{2} + \frac{\sigma^2}{2} + \frac{\mu^2}{2})]$$

, where B the set of batches of data, $\hat{X}$ is the output of VAE, σ and μ is approximated in the encoder of VAE.

3 Experiments

In this section, we systematically compare and evaluate the performance of the three mentioned dimensionality reduction techniques through empirical analysis. The comparison is conducted by assessing the visualization result, embedding time, and performance on ML(VAE) by using three types of datasets: (1) Non-linear manifold: Swiss roll, (2) natural language datasets. and (3) image datasets: CIFAR10.

3.1 Experimental Setup

We conduct 9 experiments, the corresponding parameter settings for each as followings. To visualize the dataset after dimension reduction, we will keep two important dimensions. To have a comparison for these three methods, for each dataset we select 5000 samples to perform the task.

- PCA

  Tool. We used scikit-learn to implement PCA in Python. So we imported the necessary libraries by:

```
from sklearn.decomposition import PCA
```

  Since we want to reduce the data to 2 dimensions, so we create a PCA object by:

```
pca = PCA(n_components=2)
```

- Diffusion Maps

  Tool. The Algorithm is performed by Datafold [4] which is a Python package that provides operator-theoretic, data-driven models for effectively identifying dynamical systems from time series data and inferring geometric structures in point clouds. Three necessary libraries are imported.

```
from datafold.dynfold import DiffusionMaps
import datafold.pcfold as pfold
from datafold.utils.plot import plot_pairwise_eigenvector
```

  - $\epsilon = 0.0018332392635542376$
  - $cut - off = 0.18376483668322122$

- VAE

  Tools. We use pytorch to implement VAE's architecture. Here is the way to import it:

```
import torch
```

  Here are some important parameters in our implementation:

  - batch size = 32
  - hidden size = 256
  - latent size = 2
  - number of layers in encoder, decoder: 2 (MLP)

3.2 Experimental Result

3.2.1 Time Comparison and Maximum Input Size

Time Comparison We can find the processing time taken by these three methods to process these three datasets in the bar chart as Fig.6. We want to find which method is the most efficient by a preliminary review. Only considering the run time, it is obvious that VAE is always the most efficient method because VAE has the shortest processing time for all three datasets. But the Diffusion Maps is just the opposite, it uses much longer time than the other two methods. Only comparing the time is not enough to analyze the pros can cons of the three methods in big dataset processing. So we need to pay more attention to the visualization of the results.

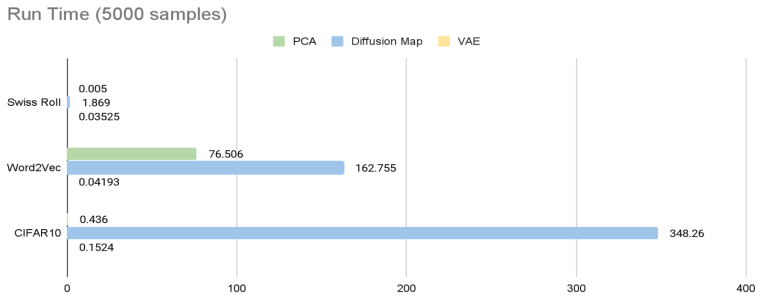
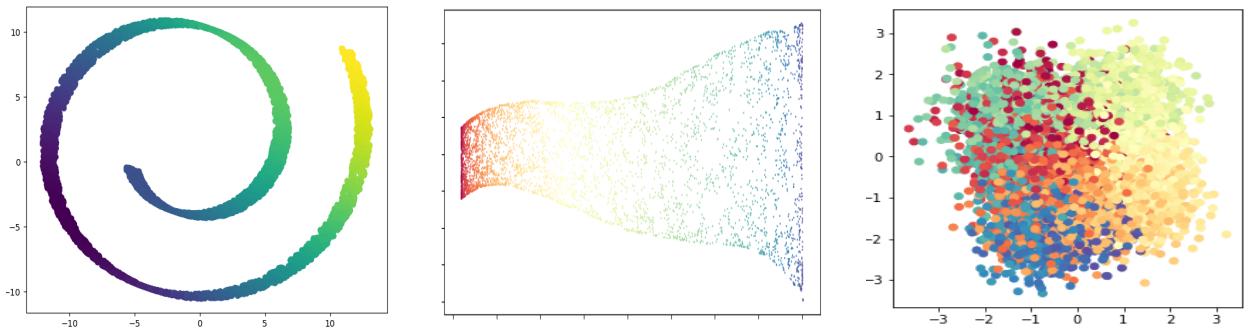


Figure 6: Run Time Comparison

Maximum Input Size The maximum input size for algorithms: PCA, Diffusion Maps, and VAE can vary depending on the hardware and software infrastructure available for computation. Different computers may have varying capacities for loading and processing large datasets. For example, when running Diffusion Maps on my computer, I find that the maximum input size you can handle is limited to 10,000 samples for the Swiss Roll dataset, 5,000 samples for Word2Vec, and 5,000 samples for CIFAR-10. (Once exceed this size, my computer

will restart the kernel.) However, my teammate's computer may have more powerful hardware, allowing her to handle larger input sizes, such as over 50,000 samples for the Swiss Roll dataset. So, the maximum input size is not solely determined by the algorithm itself but is also influenced by factors such as available memory, computational power, and software efficiency. Computers with more memory can handle larger datasets, while faster processors or specialized hardware accelerators can expedite computations.

3.2.2 Dataset-SwidsRoll



(a) (b) (c)

Figure 7: [a] PCA with 5000 samples [b] Diffusion Maps [c] VAE.

PCA. From Fig.7(a), we can observe that PCA does not expand the original Swiss roll dataset, but simply flattens it and caused overlapping the data with each other. PCA can be used in the dimensionality reduction task of a dataset while maintaining the essential relationships between the points. PCA is flexible, fast, and easily interpretable, but it does not perform so well when there are nonlinear relationships within the data. In many cases, the subspace may twist and turn, such as in the Swiss roll dataset represented.

Diffusion Maps. From above Fig.7(b), we can observe that the Diffusion Maps Algorithm effectively unrolls the original Swiss Roll dataset, transforming it into a flat gradient color band. This algorithm exhibits improved performance due to its robustness against short-circuiting phenomena. Now, let's delve deeper into the reasons why it outperforms other methods.
The key idea behind the Diffusion Maps Algorithm lies in preserving distances in a low-dimensional space through the computation of diffusion distances. To illustrate this, let's consider the red, orange, and blue regions of the Swiss Roll. In Euclidean space, the orange region may appear closer to the blue region than the red region. However, in the Diffusion space, the orange region is locally connected to the red region. This implies that the orange region can be reached from the red region through just one or a few "jumps" in the diffusion space. On the other hand, to reach the blue region from the orange region, one needs to take larger steps (represented by the yellow, mint green, and green regions) in order to navigate through the dataset.
Consequently, the Diffusion Maps Algorithm is particularly well-suited for handling nonlinear datasets like the Swiss Roll. It effectively captures the underlying manifold structure by emphasizing local connectivity, enabling the preservation of important geometric properties in the low-dimensional representation. Below showcases additional reduction results obtained using different eigenvectors, providing further evidence of the algorithm's efficacy.

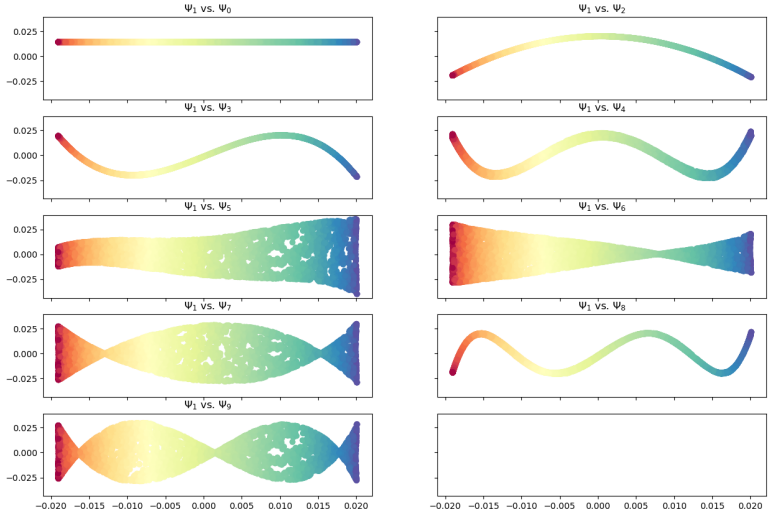


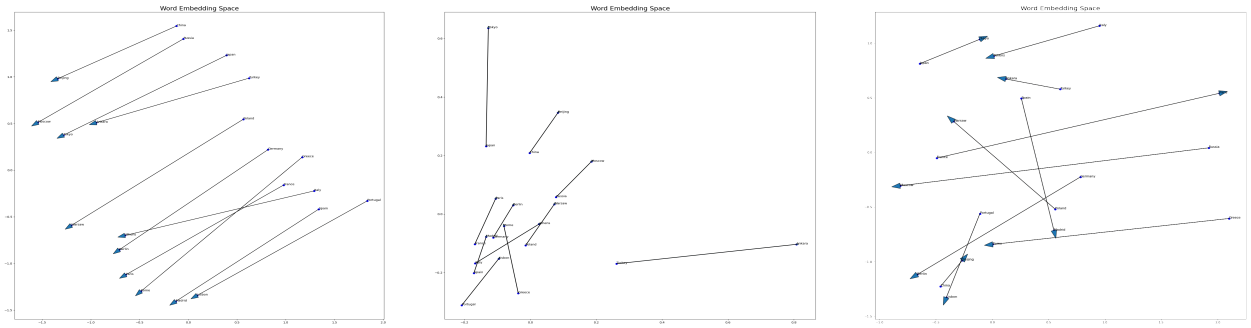
Figure 8: dimensionality reduction with 5000 samples

VAE. There exists a trade-off in VAE: reconstruction and generation. If the reconstruction of data has good quality, then the diversity of the generation will be decreased. In the experiment for VAE on the Swiss Roll dataset, it preserves some locality of data points while it is not isotropic in latent space. In Fig.7(c), adjacent points in the data space are close in the latent space.

3.2.3 Dataset-Word2Vec

We choose a pre-trained word embedding model from Google as our dataset with 300 dimensions. To have a better understanding of the visualization result, we test the dimensionality reduction performance by a word set as below shows:

- Countries: ['China', 'Russia', 'Japan', 'Turkey', 'Poland', 'Germany', 'France', 'Italy', 'Greece', 'Spain', 'Portugal']
- Capitals: ['Beijing', 'Moscow', 'Tokyo', 'Ankara', 'Warsaw', 'Berlin', 'Paris', 'Athens', 'Rome', 'Madrid', 'Lisbon']



(a) (b) (c)

Figure 9: [a] PCA [b] Diffusion Maps [c] VAE.

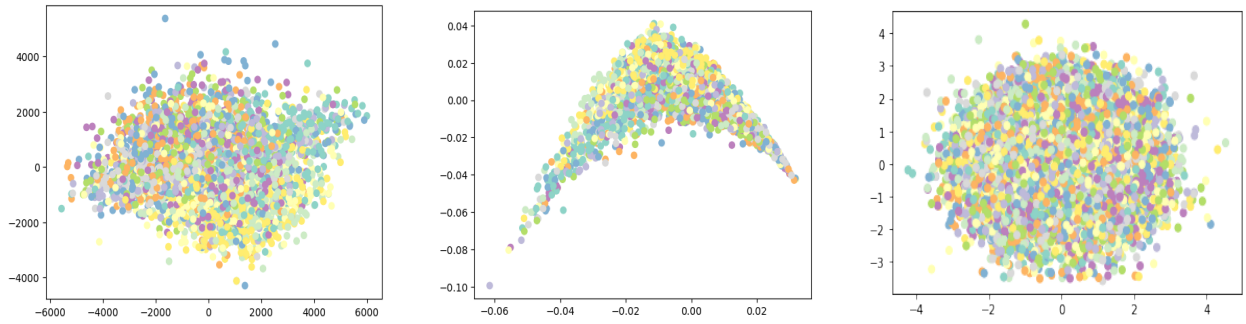
PCA. From the visualization shown in Fig.9(a), From the visualization, the words associated with 'country' are placed closer and also 'city' as they have the same feature. Meanwhile, Due to the distance between countries in the real world, the distance between the words "China" and "Portugal" is also farther. PCA can analyze the Word2Vec dataset well because Word2Vec embeddings are often structured in a way that captures meaningful

relationships between words in a linear manner. PCA is particularly effective at capturing linear relationships and patterns in data, making it a suitable choice for analyzing Word2Vec embeddings and reducing their dimensionality while preserving important semantic information. So, This method fully presents the ideal result.

Diffusion Maps. Upon initial inspection of Fig.9(b), one might perceive it as somewhat chaotic or cluttered. However, upon closer examination, the visualization reveals a distinct radioactive effect. In the lower-left corner of the plot, the European countries are clustered together, indicating their proximity in the diffusion space. On the outer ring, we can observe that China, Japan, Türkiye (Turkey), and Russia are positioned closer to each other. This arrangement is expected since China and Japan are both located in Asia, while Türkiye and Russia share borders with Asia.

VAE. A property of the Word2vec dataset is that embeddings preserve some semantic information, and we can operate vector arithmetic on them, which is explained in 1.2. Unlike PCA and diffusion maps, VAE can't preserve the geometry of data explicitly. Fig. 9(c) shows that VAE preserves some relations among latent vectors, however it still slightly messes up those vectors.

3.2.4 Dataset-CIFAR10



(a) (b) (c)

Figure 10: [a] PCA with 5000 samples [b] Diffusion Maps [c] VAE.

PCA. The Cifar10 dataset contains complex and highly non-linear structures, making it challenging for PCA to capture the intricate patterns and preserve the data's local relationships effectively. The result is just randomly distributed points. There is no effective classification of images. The result as shown in Fig.10(a).

Diffusion Maps. When reducing the dimensionality of the CIFAR-10 dataset, to a lower dimension like 6, it's crucial to acknowledge that there will inevitably be a loss of information. The original CIFAR-10 dataset consists of images with three color channels and a higher-dimensional feature space, which captures more fine-grained details.
To compare and contrast the performance before and after dimensionality reduction, a test was conducted. As anticipated, the accuracy of the classification task significantly decreased after reducing the dimensionality. (Accuracy from 0.421875 reduced to 0.1015625) This decrease in accuracy is expected due to the loss of information during the dimensionality reduction process. The result as shown in Fig.10(b)

VAE. The reconstruction-generation trade-off still exists in the experiment for vAE on the Cifar10 dataset. As we can see in Fig.10(c), data points are randomly distributed in the latent space, which means that it is very close to a Gaussian distribution. The reason is our trained VAE is better at generating more new diverse samples than at reconstruction. What we suppose to see is that data points with the label would be in a cluster in the latent space, which needs the model to be better for reconstruction. However, our result is not the case.

4 Conclusion

PCA PCA is a linear dimensionality reduction method that works well when the underlying data distribution can be approximated with a linear manifold. PCA can analyze the Word2Vec dataset well because Word2Vec

embeddings are often structured in a way that captures meaningful relationships between words in a linear manner. PCA is particularly effective at capturing linear relationships and patterns in data, making it a suitable choice for analyzing Word2Vec embeddings and reducing their dimensionality while preserving important semantic information.

Diffusion Maps Diffusion Map is more suitable for handling non-linear datasets. It considers the diffusion distance to preserve the local connectivity and capture the intrinsic geometry of the data. Diffusion Map is robust to the "short-circuiting" phenomenon that occurs in datasets with non-linear structures. It is particularly beneficial in scenarios where linear methods like PCA may fail to capture the underlying relationships. That's why it works better on Swiss Roll than the other two methods.

VAE The latent space of VAE somehow reveals relations among data points. So it is well-suited for datasets with intricate non-linear structures. But, the locality of the data manifold is hard to preserve. So it may depend on the latent space used to train the model.

References

[1] Jacqueline Delaporte, Ben M. Herbst, Willy A. Hereman, and Van der Walt Stéfan. An introduction to diffusion maps. 2008.

[2] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[3] Alex Krizhevsky. The cifar-10 dataset.

[4] Daniel Lehmberg, Felix Dietrich, Gerta Köster, and Hans-Joachim Bungartz. datafold: data-driven models for point clouds and time series on manifolds. *Journal of Open Source Software*, 5(51):2283, 2020.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[6] Lilian Weng. From autoencoder to beta-vae. *lilianweng.github.io*, 2018.