# Data Science Project One Report

Hang Zheng 520021911347

March 31, 2023

## Contents

**1 Introduction** **2**

**2 Experimental Principle** **2**
2.1 Support Vector Machine (SVM) . . . 2
2.2 Feature Selection . . . 2
2.2.1 Variance Threshold . . . 2
2.2.2 Tree-based Selection . . . 3
2.3 Feature Projection . . . 3
2.3.1 Principal Component Analysis(PCA) . . . 3
2.3.2 Linear Discriminant Analysis (LDA) . . . 4
2.3.3 Kernel PCA . . . 4
2.3.4 Auto Encoder . . . 5
2.4 Feature Learning . . . 6
2.4.1 t-Distribution Stochastic Neighbor Embedding (t-SNE) . . . 6

**3 Experimental Content and Result** **6**
3.1 Experimental Setup . . . 6
3.2 Baseline Linear SVM classifier . . . 7
3.3 Variance Threshold . . . 7
3.4 Tree-based selection . . . 7
3.5 PCA . . . 8
3.6 LDA . . . 8
3.7 Kernel PCA . . . 8
3.8 Auto Encoder . . . 11
3.9 t-SNE . . . 11

**4 Conclusion** **13**

# 1 Introduction

In this report, I mainly conducted three approaches to reduce feature dimensionality reduction: feature selection, feature projection, and feature learning. For each feature reduction approach, I apply several algorithms and compare their performances. After dimentionality reduction, I train Support Vector Machine (SVM) based on low-dimensional features produced by each method and compare the accuracy of each SVM on the test set.

The dataset for the project is the Animals with Attributes (AwA2) dataset, which consists of 37322 images of 2048 dim pre-extracted deep learning features belonging to 50 categories. For each category, I randomly split 60% images into the training set and 40% into the test set.

Furthermore, I do experiment on the parameters of each method trying to find out the optimal feature reduction method and the optimal dimensionality.

# 2 Experimental Principle

In this section, I will give a brief introduction to the principle of the three categories of dimentionality reduction algorithms I mentioned above.

## 2.1 Support Vector Machine (SVM)

## 2.2 Feature Selection

Feature selection is a data dimensionality reduction method that aims to select a subset of the most relevant features from the original dataset for use in model training and prediction. In practical applications, the original dataset often contains many features, some of which may be irrelevant or redundant, while others are highly correlated. Therefore, selecting the most relevant features can **improve the accuracy and efficiency of the model, as well as reduce storage and computation costs**.

Feature selection methods can be divided into three categories: filter, wrapper, and embedded methods. The main ideas of the three categories algorithms are showed below:

- Filter methods: Performed independently before feature selection and model training. They evaluate the importance of each feature by calculating its correlation or information value with the target variable, and select features based on a certain threshold or ranking. Common filter methods include correlation coefficient, chi-square test, mutual information, and analysis of variance.

- Wrapper methods valuate the contribution of features directly by using the performance of the model. Specifically, they search for the best feature combination by constructing multiple subsets, and determine which features are most useful based on the performance of the model. Representative algorithms for wrapper methods include Recursive Feature Elimination (RFE), genetic algorithm-based feature selection, and forward selection.

- Embedded methods: Integrate feature selection and model training into a single process, embedding the feature selection process into the model learning. This method typically automatically selects the best subset of features during the training process of machine learning algorithms. Representative algorithms for embedded methods include Lasso and Ridge regression.

In this project, I implement two basic feature selection methods, Variance Threshold and Tree-based Selection. In the next two sections 2.2 and 2.2.1 I will give detailed introductions for them.

### 2.2.1 Variance Threshold

Variance Threshold is a simple approach to feature selection. It removes all features whose variance doesn't meet the threshold. The principle is that features with small variance often contain less data information.It is often used for datasets with a large number of features, many of which may be redundant or uncorrelated.

The Variance Threshold algorithm first calculates the variance of each feature and compares them to a predefined threshold. If a feature's variance is less than the threshold, it is considered a low-variance feature and can be removed. By iteratively performing this process, low-variance features in the dataset can be gradually removed, reducing the dataset's dimensionality.

The advantages of the Variance Threshold algorithm include its simplicity, speed, efficiency, and lack of model training requirements. However, its disadvantage is that it only considers the variance of features and cannot handle the correlation between features. Therefore, on some datasets, the Variance Threshold algorithm may excessively reduce features, leading to a decrease in model accuracy.

### 2.2.2 Tree-based Selection

Tree-based selection is a kind of **embedded method**. Embedded methods can learn which features contribute most to the accuracy of the model. The Tree-based selection evaluates the importance of features based on a decision tree model and selects the most important features for retention. The algorithm identifies the features that contribute the most to the target variable when building a decision tree, using a tree-based model's feature importance measure.

The Tree-based Selection algorithm first constructs a decision tree and then ranks features based on their importance in the decision tree. Different feature importance measures, such as Gini index, information gain, or mean decrease impurity, can be used to evaluate feature importance. Next, the algorithm selects the top-ranked features for retention based on a set threshold or a specific number of features, while removing other features. In this way, the dimensionality of the dataset can be effectively reduced.

The advantages of the Tree-based Selection algorithm include its ease of implementation, scalability, robustness to outliers, and ability to handle nonlinear relationships. However, the algorithm may over-rely on the decision tree model, resulting in poor generalization performance for specific datasets. Additionally, for highly correlated features, it may select one feature and remove others, leading to some information loss.

## 2.3 Feature Projection

Feature Projection is a commonly used data dimensionality reduction method that projects data points from a high-dimensional feature space to a lower-dimensional subspace. In Feature Projection, the goal is to reduce the dimensionality of data by mapping high-dimensional data points onto a low-dimensional space.

Feature Projection methods typically include two types:

- Linear projection: Includes Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Factor Analysis (FA), among others, with PCA being the most widely used method. PCA achieves data dimensionality reduction by finding the optimal representation of high-dimensional data points in a low-dimensional space.
- Nonlinear projection: Include Kernel PCA, Isomap and others, which can handle nonlinear relationships and are more suitable for complex data structures.

In this project, I implement four feature projection methods, Principal Component Analysis(PCA), Linear Discriminant Analysis (LDA), Kernel PCA and Auto Encoder. In the next four sections 2.3, 2.3.1, 2.3.2 and 2.3.3 I will give detailed introductions for them.

### 2.3.1 Principal Component Analysis(PCA)

Principal Component Analysis (PCA) is a commonly used data dimensionality reduction method that maps high-dimensional data points into lower-dimensional subspaces for the purpose of data analysis and visualization. The basic idea of PCA is to reduce the dimensionality of data by identifying the principal components, which are the directions in the data space that explain the maximum variance, usually representing the directions of the data points in space.

The implementation process of PCA can be summarized in the following steps:

- Data preprocessing: centering the data by moving each feature to a zero-mean position, facilitating the calculation of variance and covariance.

- Calculating the covariance matrix: computing the covariance matrix based on the centered data, which is a symmetric matrix where each element represents the covariance between two features.
- Calculating the eigenvalues and eigenvectors: eigenvalue decomposition of the covariance matrix produces a set of eigenvalues and eigenvectors, where the eigenvalues represent the variance of the data in the corresponding directions, and the eigenvectors represent the principal components in those directions.
- Selecting principal components: ordering the eigenvectors based on the size of their corresponding eigenvalues, selecting the top k eigenvectors as the principal components, where k is the new dimensionality of the lower-dimensional subspace.
- Data projection: projecting the data points onto the selected principal components to obtain the new low-dimensional representation.

Through PCA, data dimensionality reduction can be achieved while preserving most of the information in the original data.

#### 2.3.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a classical data dimensionality reduction method commonly used in classification and feature extraction tasks. LDA maps high-dimensional data onto a low-dimensional space by finding a linear transformation that maximizes the distance between different classes and minimizes the distance within the same class.

The implementation process of LDA can be summarized in the following steps:

- Data preprocessing: The data is standardized so that each feature has a mean of 0 and a variance of 1.
- Calculation of within-class scatter matrix and between-class scatter matrix: The scatter matrix of the data within each class and the scatter matrix between different classes are calculated, respectively. The scatter matrix reflects the dispersion and distribution of the data.
- Calculation of feature vectors: The eigenvectors and eigenvalues of the within-class scatter matrix and the between-class scatter matrix are obtained by solving the generalized eigenvalue problem. The feature vectors are arranged in descending order of corresponding eigenvalues, and the top k feature vectors are selected as the new feature space for dimensionality reduction.
- Data projection: The data is mapped to the new feature space to obtain the reduced representation of the data.

Unlike PCA, LDA not only considers the variance of the data, but also takes into account the distance between different classes, so it can better preserve the class information of the data and improve the performance of classification and recognition.

#### 2.3.3 Kernel PCA

Kernel PCA (KPCA) is a non-linear extension of PCA using kernel methods. PCA uses linear projection function from the original feature space to a lower-dimisional space and can only be applied to linearly separable datasets, while KPCA can perform both linear and non-linear dimensionality reduction through the use of kernels, as shown in Figure 1.

The key idea of Kernel PCA is to transform the original data into a higher dimensional feature space through a non-linear mapping function, and then **perform PCA in the new feature space**. The non-linear mapping function is defined by a kernel function, such as the Gaussian kernel or polynomial kernel, which allows Kernel PCA to capture the non-linear structure of the data.

Compared with PCA, Kernel PCA can handle non-linear data structures, which makes it suitable for data sets that have complex, non-linear relationships among the features. However, Kernel PCA is **computationally more expensive** than PCA, especially for large data sets, since it involves the computation of a kernel matrix, which can be computationally intensive.

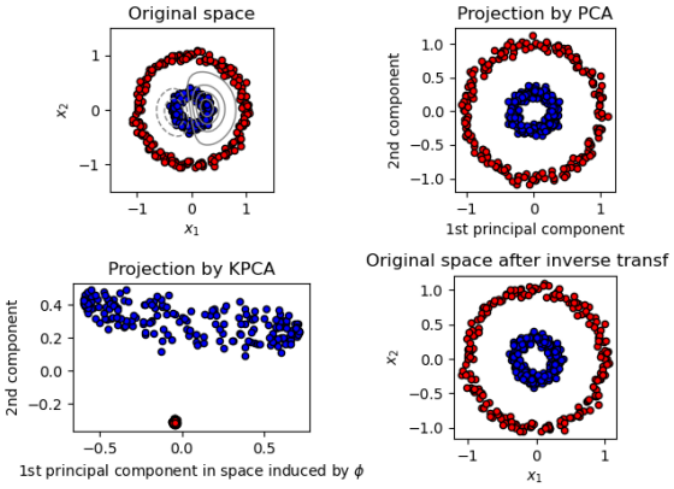


Figure 1: Illustration for Kernel PCA

### 2.3.4 Auto Encoder

Auto Encoder algorithm is a type of unsupervised learning algorithm which can be seen as a data compression technique that can learn a low-dimensional representation of input data without losing too much useful information.

The core idea of Auto Encoder is to compress the input data to a low-dimensional space by encoding it, and then reconstruct it back to the original space through the decoder. The encoder and decoder can be implemented through deep neural networks.

The training process of Auto Encoder can be divided into two stages.

- In the first stage, the encoder of the Auto Encoder learns to map the input data to a low-dimensional hidden space.
- In the second stage, the decoder learns to map the hidden representation back to the original input space and minimize the reconstruction error.

The structure of encoder-decoder network is as shown in Figure 2

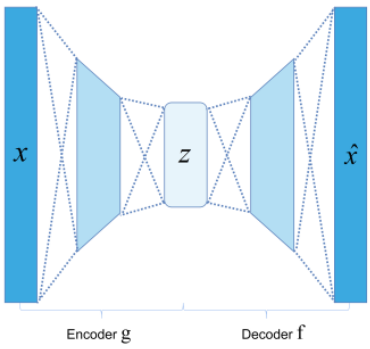


Figure 2: Illustration for Encoder-Decoder Network

After training, the encoder could **be separated from the network and serve as a projection method**. The hidden layer dimentionality could be seen as the target feature dimentionality that the original feature dimentionality will be reduced to.

Unlike traditional data dimensionality reduction methods, Auto Encoder can automatically learn the data's feature representation without manually selecting features. Moreover, Auto Encoder can handle nonlinear data structures and has good performance in dealing with high-dimensional data.

However, the training process of Auto Encoder typically requires a large amount of computational resources and time, especially for large-scale datasets and complex neural network structures.

## 2.4 Feature Learning

In data dimensionality reduction, Feature learning is the process of automatically extracting useful features from raw data through model training, without the need for manual feature selection or engineering. Feature learning methods typically use neural networks or other machine learning techniques to gradually adjust model parameters by inputting raw data into the model, aiming to minimize prediction errors or maximize the representational power of features.

Compared with traditional feature selection methods, feature learning methods can automatically discover potential patterns and structures in data, with stronger representational power and better generalization performance. However, feature learning methods typically require large amounts of training data and computing resources, and for problems with small data volume or simple feature spaces, traditional feature selection methods may still be a better choice.

In this project, I implement a basic feature learning methods, t-Distribution Stochastic Neighbor Embedding (t-SNE). In the next section 2.4.I will give detailed introductions for it.

### 2.4.1 t-Distribution Stochastic Neighbor Embedding (t-SNE)

t-Distribution Stochastic Neighbor Embedding (t-SNE) is a **nonlinear** dimensionality reduction technique used for mapping high-dimensional data to a lower-dimensional space for visualization. t-SNE effectively preserves local similarities between data points and highlights the intrinsic structure between data points during visualization.

The core idea of t-SNE is to map high-dimensional data points to a lower-dimensional space while **preserving their relative distances**, especially the distances between similar points. The t-SNE algorithm represents each data point as a probability distribution in the high-dimensional space and finds a corresponding probability distribution in the low-dimensional space. During this process, t-SNE uses t-distributions to represent the similarity between data points, where faraway points are assigned lower probabilities and closer points are assigned higher probabilities.

The training process of t-SNE can be divided into two stages.

- In the first stage, t-SNE uses Stochastic Gradient Descent (SGD) to optimize the similarity between each data point's probability distribution in the high-dimensional and low-dimensional spaces.
- In the second stage, t-SNE optimizes the objective function to minimize the KL divergence between the high-dimensional data points and the low-dimensional map. During this process, t-SNE uses gradient descent algorithms to minimize the cost function until a satisfactory low-dimensional representation is found.

An important feature of the t-SNE algorithm is that it can **highlight the intrinsic structure between data points, especially the clustering structure**, during visualization. However, t-SNE also has some limitations, such as sensitivity to initial parameters and high computational costs for larger datasets.

# 3 Experimental Content and Result

## 3.1 Experimental Setup

In the experiment, we use the Animals with attributes 2 dataset (AwA2) which consists of 37322 images of 50 animals and 2048-dimensional learned representation features are pre-extracted using Resnet101 from every image. I split the images in each category into 60% for training and 40% for testing.

Firstly, we take the high-dim feature vectors as input and use basic linear SVM to measure the baseline effect of the classifier. Next, we use dimensionality reduction method to preprocess the input vector and take the reduced features as input of SVM to explore the methods' performance.

### 3.2 Baseline Linear SVM classifier

I take default linear kernel function and conducted experiment find the best hyper-parameter C (penalty parameter of the SVM). The result is as shown in Table 1.

| C | Accuracy | Time(s) |
|---|---|---|
| 0.001 | 0.9296942939327045 | 1184.8966619968414 |
| 0.01 | 0.9254130711084353 | 652.8699128627777 |
| 0.05 | 0.9239414007625928 | 650.7723708152771 |
| 0.1 | 0.9239414007625928 | 625.928838968277 |
| 0.5 | 0.9234062479095592 | 639.6198637485504 |
| 1 | 0.9230048832697839 | 646.4423336982727 |
| 5 | 0.9207973777510201 | 650.9995939731598 |
| 10 | 0.9201284366847281 | 652.8699128627777 |
| 50 | 0.9197270720449529 | 649.9509415626526 |
| 100 | 0.9197270720449529 | 647.6313498020172 |

Table 1: Results of Experiment on C penalty Parameter

From the result, we can see that the smaller the C value, the better the classification performance. So in the following experiments, if without specific explanation, we will fix the C as 0.001.

### 3.3 Variance Threshold

I conducted the experiment with method Variance Threshold on different parameter thresholds. The result is as shown in Table 2.

| Threshold | Reduced Dimentionality | Accuracy | Time(s) |
|---|---|---|---|
| 0.2 | 1339 | 0.9218007893504582 | 1.5775253772735596 |
| 0.5 | 624 | 0.9159810020737174 | 1.8201205730438232 |
| 0.8 | 323 | 0.9100943206903471 | 1.9030191898345947 |
| 1.0 | 230 | 0.8993912636296743 | 1.7773025035858154 |
| 1.5 | 111 | 0.8692889156465315 | 1.402595043182373 |
| 2.0 | 60 | 0.8217940999397954 | 0.60158371925354 |
| 3.0 | 20 | 0.6562311860325105 | 0.5265636444091797 |

Table 2: Results of Experiment on Threshold in Variance Threshold Method

From the result, we can see that:

- With the threshold growing greater, the dimentionality is reduced greater(more features are discarded) and meanwhile the accuracy grows lower.
- After the threshold growing greater than 1, the performance decreases faster than that smaller than 1.
- The Variance Threshold method is really fast.

### 3.4 Tree-based selection

In the Tree-based selection, the tree model gives the relative importance of each features and the result is given into the selector which selects the features based on the relative importance and discards those less important than a given threshold. I conducted the experiment with method Tree-based selection on different parameter thresholds. The result is as shown in Table 3.

From the result, we can see that:

- With the threshold growing greater, the dimentionality is reduced greater(more features are discarded) and meanwhile the accuracy grows lower.

| Threshold | Reduced Dimentionality | Accuracy | Time(s) |
|---|---|---|---|
| 0.0001 | 2048 | 0.9296942939327045 | 17.860587120056152 |
| 0.0005 | 565 | 0.9138403906615827 | 5.581266641616821 |
| 0.001 | 162 | 0.8845407719579905 | 4.899393558502197 |
| 0.002 | 31 | 0.7972439628068767 | 10.955464363098145 |
| 0.005 | 1 | 0.09445447856043883 | 10.460153579711914 |
| 0.01 | 0 | - | 11.739736795425415 |

Table 3: Results of Experiment on Threshold in Tree-based selection

- No feature matters more than 0.01 relative importance for that with a threshold 0.01, the selector selects no feature at all.
- Every feature matters more than 0.0001 relative importance for that with a threshold 0.0001, the selector selects all the original features.

### 3.5 PCA

I conducted the experiment with method PCA on different parameter n_component(the dimentionality of output features if it's a int number greater than / equal to 1 or the proportion of features if it's a float number less than 1). The result is as shown in Table 4.

| n_component | Reduced Dimentionality | Accuracy | Time(s) |
|---|---|---|---|
| 0.8 | 189 | 0.9072178741052913 | 27.669050455093384 |
| 0.9 | 467 | 0.917519566526189 | 26.64574360847473 |
| 0.95 | 848 | 0.9193926015118068 | 38.78752875328064 |
| 0.98 | 1332 | 0.921733895243829 | 25.272228002548218 |
| 0.99 | 1604 | 0.9228710950565255 | 23.695430278778076 |
| 0.999 | 1981 | 0.9230717773764131 | 24.602872371673584 |
| 5 | 5 | 0.5270586661315139 | 4.923595905303955 |
| 20 | 20 | 0.8072111846946284 | 6.300973415374756 |
| 100 | 100 | 0.891832229580574 | 9.03649115562439 |
| 500 | 500 | 0.9179878252725935 | 65.03180289268494 |
| 1200 | 1200 | 0.9214663188173122 | 72.59927940368652 |
| 2000 | 2000 | 0.9231386714830423 | 28.830209970474243 |

Table 4: Results of Experiment on n_component in PCA

The distribution of training dataset and testing dataset reduced to 2d is showed in Figure 3 and Figure 4.

### 3.6 LDA

I conducted the experiment with method LDA on different parameter n_component(the dimentionality of output features). The result is as shown in Table 5.

The distribution of training dataset and testing dataset reduced to 2d is showed in Figure 5 and Figure 6.

### 3.7 Kernel PCA

I conducted the experiment with method Kernel PCA on different parameter n_component(the dimentionality of output features) and different kernels. The result is as shown in Table 6.

The distribution of training dataset and testing dataset reduced to 2d with kernel poly is showed in Figure 7 and Figure 8.

The distribution of training dataset and testing dataset reduced to 2d with kernel rbf is showed in Figure 9 and Figure 10.

The distribution of training dataset and testing dataset reduced to 2d with kernel cosine is showed in Figure 11 and Figure 12.

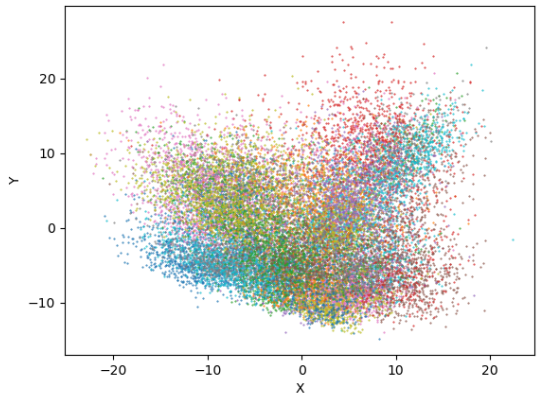


Figure 3: Distribution of training dataset reduced to 2d by PCA

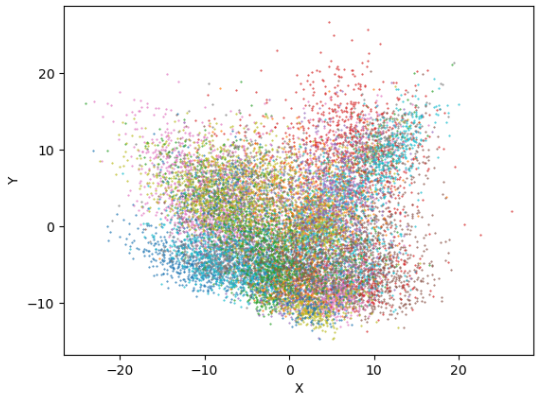


Figure 4: Distribution of testing dataset reduced to 2d by PCA

| n_component | Reduced Dimentionality | Accuracy | Time(s) |
|---|---|---|---|
| 2 | 2 | 0.3146698775837849 | 49.23624062538147 |
| 5 | 5 | 0.5755568934376881 | 68.60897541046143 |
| 10 | 10 | 0.7134256472004816 | 42.41367554664612 |
| 15 | 15 | 0.7925613753428323 | 28.488672971725464 |
| 20 | 20 | 0.8299551809485585 | 34.595290660858154 |
| 30 | 30 | 0.8833366780386648 | 27.17755675315857 |
| 40 | 40 | 0.9026690748545053 | 20.143582820892334 |
| 45 | 45 | 0.9094922737306843 | 21.105896472930908 |
| 48 | 48 | 0.9124356144223694 | 20.856523752212524 |
| 49 | 49 | 0.9135059201284367 | 21.000794649124146 |

Table 5: Results of Experiment on n_component in LDA

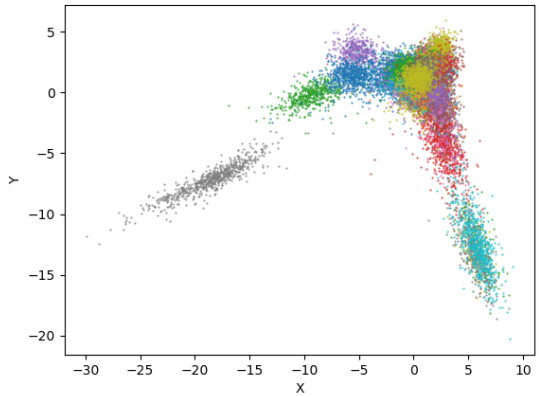


Figure 5: Distribution of training dataset reduced to 2d by LDA

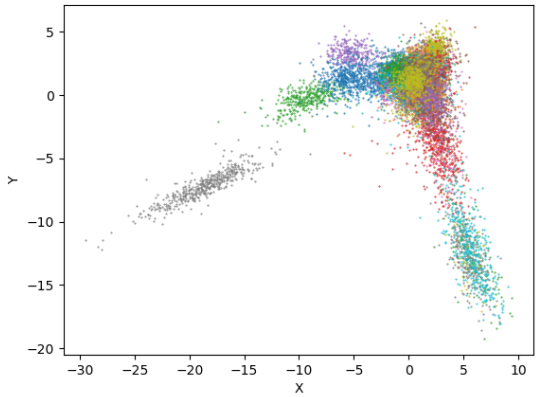


Figure 6: Distribution of testing dataset reduced to 2d by LDA

| n_component | poly | rbf | cosine | sigmoid |
|---|---|---|---|---|
| 20 | 0.8747073382834972 | 0.8561776707472072 | 0.8688875510067563 | 0.8461435547528263 |
| 100 | 0.919793966151582 | 0.9150444845809084 | 0.921064954177537 | 0.9055455214395611 |
| 500 | 0.9277543648404576 | 0.9240751889758513 | 0.9270854237741655 | 0.9128369790621447 |
| 1200 | 0.9286908823332665 | 0.9264833768145027 | 0.9284233059067496 | 0.9135059201284367 |
| 2000 | 0.9302294467857382 | 0.9273530002006823 | 0.9286908823332665 | 0.9135059201284367 |

Table 6: Results of Experiment on n_component and kernel in Kernel PCA

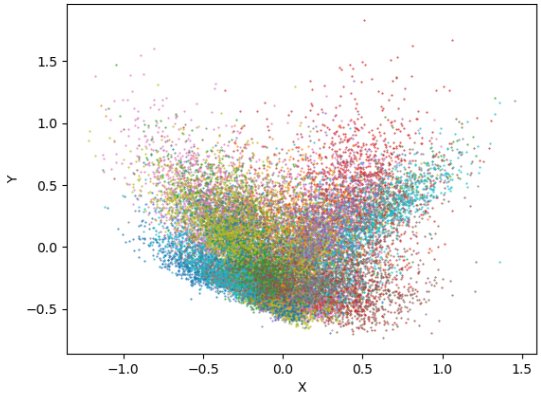


Figure 7: Distribution of training dataset reduced to 2d by KPCA with kernel poly

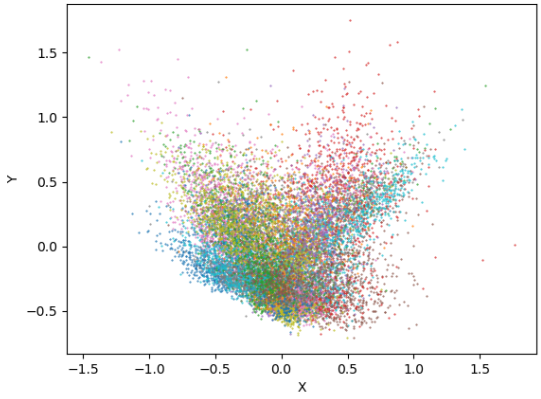


Figure 8: Distribution of testing dataset reduced to 2d by KPCA with kernel poly

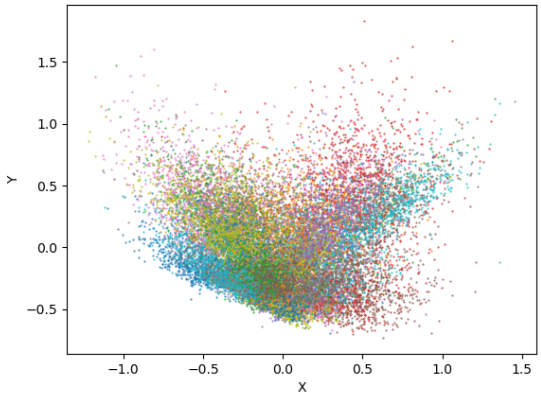


Figure 9: Distribution of training dataset reduced to 2d by KPCA with kernel rbf

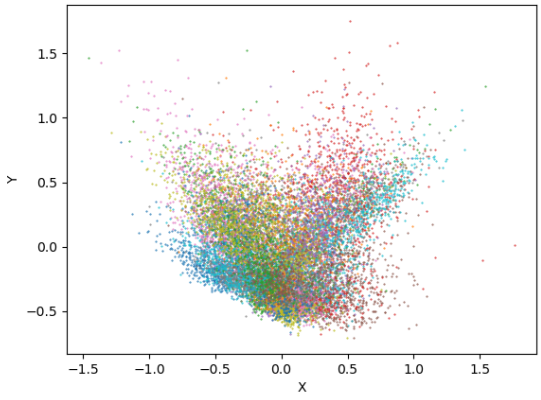


Figure 10: Distribution of testing dataset reduced to 2d by KPCA with kernel rbf

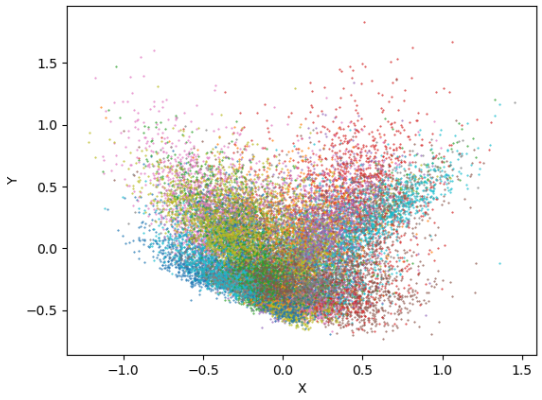


Figure 11: Distribution of training dataset reduced to 2d by KPCA with kernel cosine

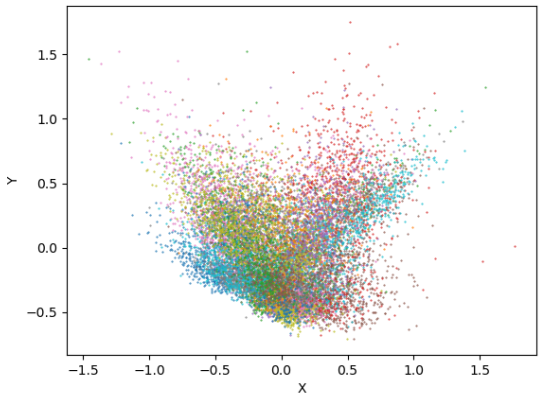


Figure 12: Distribution of testing dataset reduced to 2d by KPCA with kernel cosine

The distribution of training dataset and testing dataset reduced to 2d with kernel sigmoid is showed in Figure 13 and Figure 14.

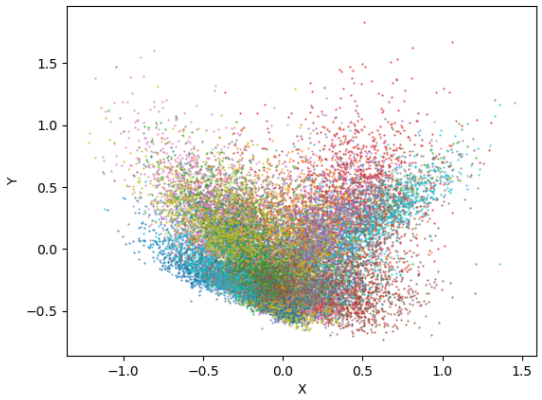


Figure 13: Distribution of training dataset reduced to 2d by KPCA with kernel sigmoid

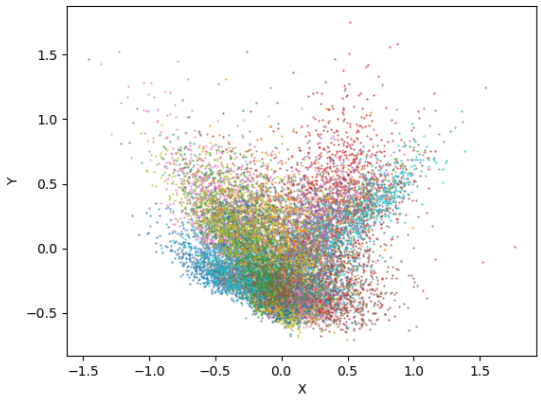


Figure 14: Distribution of testing dataset reduced to 2d by KPCA with kernel sigmoid

## 3.8 Auto Encoder

I conducted the experiment with method Auto Encoder on different hidden layer dim. The result is as shown in Table 7.

The distribution of training dataset and testing dataset reduced to 2d is showed in Figure 15 and Figure 16.

## 3.9 t-SNE

I conducted the experiment with method t-SNE on different C penalty value and different pre-processing methods. The result is as shown in Table 8(3d) and 9(2d).

The distribution of training dataset and testing dataset reduced to 2d(without pre-processing) is showed in Figure 17 and Figure 18.

| dim | Reduced Dimentionality | Accuracy | Time(s) |
|---|---|---|---|
| 2 | 2 | 0.10375275938189846 | 983.9969763755798 |
| 5 | 5 | 0.3910629473543381 | 834.2975270748138 |
| 20 | 20 | 0.6837915579637434 | 785.9474368095398 |
| 50 | 50 | 0.8007893504582246 | 105.12441182136536 |
| 200 | 200 | 0.8914308649407987 | 141.63139820098877 |
| 500 | 500 | 0.9029366512810222 | 613.2535510063171 |
| 1200 | 1200 | 0.9106294735433809 | 554.5082273483276 |
| 2000 | 2000 | 0.9108970499698976 | 1967.8712921142578 |

Table 7: Results of Experiment on hidden layer dim in Auto Encoder

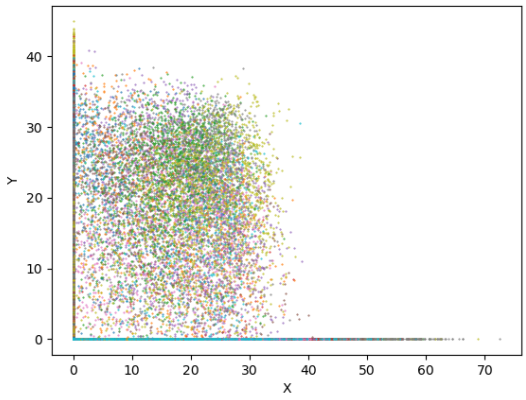


Figure 15: Distribution of training dataset reduced to 2d by AE

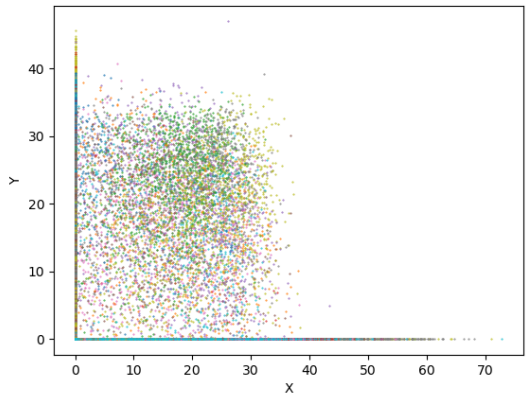


Figure 16: Distribution of testing dataset reduced to 2d by AE

| C | non-preprocessed | PCA-preprocessed | LDA-preprocessed |
|---|---|---|---|
| 0.02 | 0.8709704996 | 0.88219864 | 0.95284875 |
| 0.1 | 0.8789755432 | 0.88239864 | 0.95276402 |
| 0.5 | 0.8734693562 | 0.88294754 | 0.95258593 |
| 1 | 0.8734502346 | 0.88329875 | 0.95247532 |
| 10 | 0.8796225632 | 0.88348752 | 0.95248625 |
| 50 | 0.8709972242 | 0.88351364 | 0.95238621 |

Table 8: Results of Experiment on different C penalty value and different pre-processing methods in t-SNE(reduced to 3d)

| C | non-preprocessed | PCA-preprocessed | LDA-preprocessed |
|---|---|---|---|
| 0.02 | 0.8699704996 | 0.86729864 | 0.95314875 |
| 0.1 | 0.8699755432 | 0.86829864 | 0.95316402 |
| 0.5 | 0.8694693562 | 0.86834754 | 0.95328593 |
| 1 | 0.8694502346 | 0.86849875 | 0.95337532 |
| 10 | 0.8696225632 | 0.86868752 | 0.95358625 |
| 50 | 0.8699972242 | 0.86861364 | 0.95358621 |

Table 9: Results of Experiment on different C penalty value and different pre-processing methods in t-SNE(reduced to 2d)

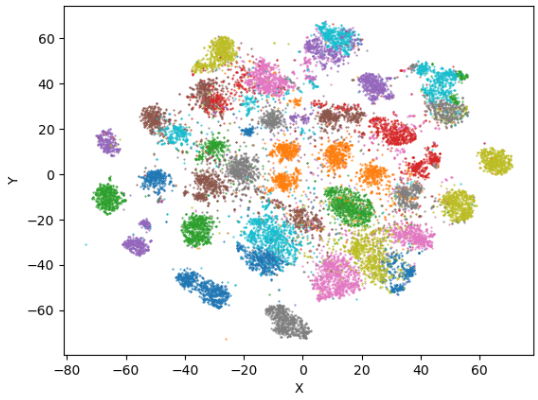


Figure 17: Distribution of training dataset reduced to 2d by t-SNE

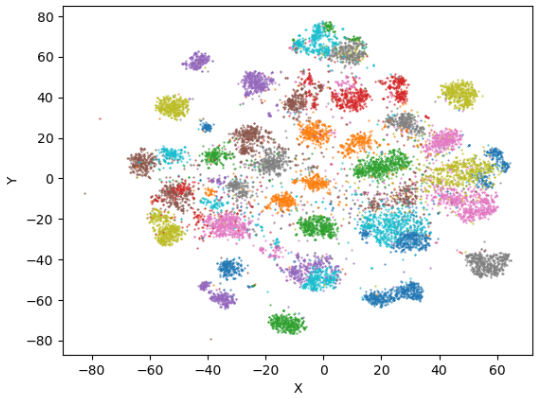


Figure 18: Distribution of testing dataset reduced to 2d by t-SNE

# 4 Conclusion

In this paper, we introduced and implemented totally 7 dimensionality reduction methods, namely Variance Threshold, Tree-based Selection, PCA, LDA, Auto Encoder, Kernel PCA, t-SNE. The best performance comes from LDA_preprocessed t-SNE method with C value as 50, achieves accuracy 0.95358621 on testing set.