Principles of Data Science Project 1
Dimension Reduction

Hongzhou Liu
517030910214
deanlhz@sjtu.edu.cn

Xuanrui Hong
517030910227
hongxuanrui.1999@sjtu.edu.cn

Qilin Chen
517030910155
1017856853@sjtu.edu.cn

Abstract

Datasets in real world often consist of noisies. Those noisies bring some difficulties and complexities to machine learning tasks. It is significant for us to eliminate those noisies, which leads to the topic of this project. In this project, we conduct experiment on different dimension reduction methods, including select-k-best, variance threshold, tree-based selcetion, PCA, LDA, AutoEncoder, t-SNE and LLE. We will observe performances of SVM on those dimension-reducted data and compare the difference among those methods. Also, we will find the optimal dimensions of each method and the optimal dimension reduction method.

1 Method

1.1 Feature Selection

1.1.1 Select-k-best

Select-K-Best is one of the methods of univariate feature selection, which works by selecting the best features based on univariate statistical tests. It removes all but the k highest scoring features. There're different functions to score features:

- χ^2 value: We can get χ^2 value by testing chi-squared statistic from X. The greater the chi-square value means the higher the correlation with the classification.
- ANOVA F value: F value = variance of the group means (Mean Square Between) / mean of the within group variances (Mean Squared Error). When the variance between the groups is much larger than the variance between the groups, that is, the larger the gap between the groups, the larger the F value.

1.1.2 Variance Threshold

VarianceThreshold is a simple approach to feature selection. It removes all features whose variance doesn't meet some threshold. The principle is that features with small variance often contain less data information.

1.1.3 Tree-based Selection

Tree-based selcetion is a kind of embedded method. Embedded methods can learn which features contribute most to the accuracy of the model. Then, we can pick up important features and remove irrelevant features. Here, tree-based estimator is a number of randomized decision trees. The

estimator will fit those trees on various sub-samples of the dataset and use averaging to improve accuracy and control over-fitting problem.

1.2 Feature Projection

1.2.1 PCA

Principal component analysis is one of the most widely used data dimensionality reduction algorithms. It performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized[1]. Formally, the optimization goal is

$$\max_{v} \frac{1}{n} \sum_{i=1}^{n} (v^T x_i)^2 = \frac{1}{n} v^T X X^T v \tag{1}$$

where v is the new axis.

$$s.t. \quad v^T v = 1 \tag{2}$$

Using Lagrange Multiplier we can get

$$X X^T v = \lambda v \tag{3}$$

We can see that v is the eigenvector of XX^T, and λ is the corresponding eigenvalue. Therefore, v can be calculated by performing eigenvalue decomposition to the co-variance matrix XX^T. Then we can get the data after dimensionality reduction.

1.2.2 Kernel PCA

In general, principal components analysis is suitable for linear dimensionality reduction of data. Kernel PCA can achieve nonlinear dimensionality reduction of data and is used to process linear inseparable data sets.

The general idea of Kernel PCA is: for the matrix in the input space, we first use a non-linear mapping to map all samples in a high-dimensional or even infinite-dimensional space, and then perform PCA dimensionality reduction in this high-dimensional space. So Kernel PCA will replace X in PCA by $\phi(X)$ to do this mapping, and $\phi(X)$ is called kernel. However, $\phi(X)$ may not have an explicit form, sometimes it will map X into a infinite dimension space. The kernel trick solves this problem by directly compute the inner product between $\phi(X)$ and $\phi(Y)$ by

$$K(X, Y) = \phi(X)^T \phi(Y) \tag{4}$$

So, using Lagrange Multiplier we will instead get

$$K(X, X) v = \lambda v \tag{5}$$

There're different kinds of kernels

- Linear Kernel: $K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T \boldsymbol{y}$
- Polynomial Kernel: $K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^T \boldsymbol{y} + c)^d$
- Gaussian Kernel (RBF Kernel): $K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2})$
- Sigmoid Kernel: $K(\boldsymbol{x}, \boldsymbol{y}) = \tanh(\alpha \boldsymbol{x}^T \boldsymbol{y} + c)$

1.2.3 LDA

Linear discriminate analysis is commonly used as dimensionality reduction technique. Its main idea is to project the samples on a straight line so that the projections of similar samples are as close as possible, and the projection points of heterogeneous samples are as far as possible. [2]

The key step of LDA:

- Standardize d-dimensional data (d is the number of features).

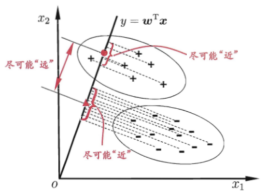


Figure 1: Two-dimensional illustration of LDA.

- For each category, calculate the d-dimensional mean vector.
- Construct inter-class scatter matrix S_B and intra-class scatter matrix S_W.
- Calculate the eigenvalues and corresponding eigenvectors of the matrix $S_W^{-1}S_B$.
- Select the eigenvectors corresponding to the first k eigenvalues to construct a $d \times k$ dimension transformation matrix W, where the eigenvectors are arranged in columns.
- Use the transformation matrix W to map the samples to the new feature subspace.

1.2.4 AutoEncoder

An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner.[3]The purpose of autoencoder is to train a neural network for sample dimensionality reduction, and the sample after dimensionality reduction can reconstruct the original sampel well.

It contains encoder and decoder. The coding network belongs to the dimension reduction part, and the role is to reduce the high-dimensional original data to a low-dimensional nested structure with a certain number of dimensions; the decoding network belongs to the reconstruction part, which can be regarded as the reverse process of the coding network. There is also an intersection between the encoding network and the decoding network, called the "code layer", which is the core of the entire self-encoding network and can determine the essential dimensions of high-dimensional datasets.

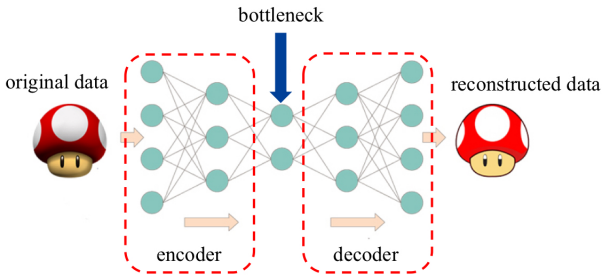


Figure 2: Autoencoder.

1.3 Feature Learning

1.3.1 t-SNE

T-distributed stochastic neighbor embedding is a machine learning algorithm for dimensionality reduction, proposed by Laurens van der Maaten and Geoffrey Hinton in 2008.[4]It is a nonlinear dimensionality reduction technique. T-SNE obtains $\tilde{x}$ from x and maintains the relative distance as much as possible. Its main steps are:

- Evaluate the similarity of data point x_i and data point x_j by

$$p_{j|i} = \frac{(1 + \|x_i - x_j\|^2)^{-}1}{\sum_{k \neq i}(1 + \|x_i - x_k\|^2)^{-}1} \tag{6}$$

- Meature similarity of data point $\tilde{x_i}$ and data point $\tilde{x_j}$ by

$$q_{j|i} = \frac{(1 + \|\tilde{x_i} - \tilde{x_j}\|^2)^{-}1}{\sum_{k \neq i}(1 + \|\tilde{x_i} - \tilde{x_k}\|^2)^{-}1} \tag{7}$$

- Minimize the $Kullback - Leiblerdivergence$ of the distribution Q from the distribution P, that is

$$L = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j p(i|j) log \frac{p_{j|i}}{q_{j|i}} \tag{8}$$

Besides, t-SNE is the deformation of SNE. The difference is that t-SNE uses t-distribution while SNE uses Gaussian distribution.

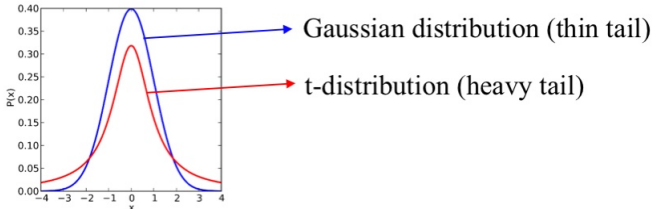


Figure 3: t-SNE and SNE.

1.3.2 LLE

Locally Linear Embedding (LLE) is a member of the manifold learning algorithm and is a nonlinear dimensionality reduction method. LLE was invented in 2000 and published in the journal Sience.[5]LLE algorithm tries to maintain the linear relationship between samples in the neighborhood. As shown in 4, the sample x_i in the high-dimensional space can be reconstructed from its neighbor sample x_j, x_k, x_l through linear combination, i.e.

$$x_i = w_{ij}x_j + w_{ik}x_k + x_{il}x_l \tag{9}$$

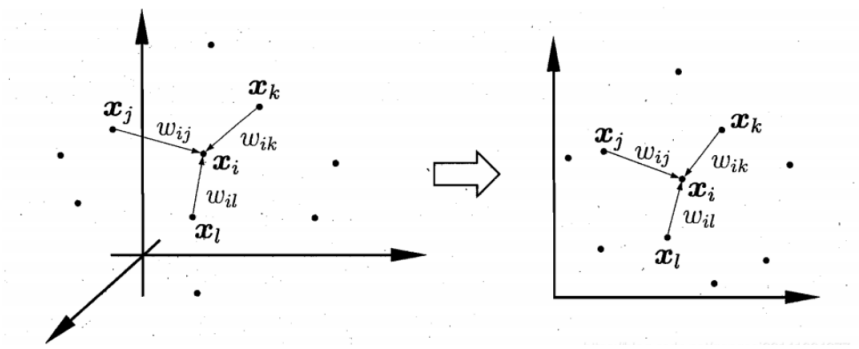


Figure 4: LLE.

Its main steps are:

- Learn the linear relation W between each sample and its neighbors, which can be formalized as:

$$W = \arg\min_W \sum_{i=1}^{m} \|x_i - \sum_{j \in N(i)} w_{ij}x_j\|^2 \tag{10}$$

where $N(i)$ is the k-nearest neighbors of x_i

- Reconstruct the low-dimensional features:

$$Z = \arg\min_{Z} \sum_{i=1}^{m} \|z_i - \sum_{j \in N(i)} w_{ij} z_j\|^2 \tag{11}$$

where z_i is the sample after dimensionality reduction of x_i

2 Experiment

2.1 Baseline

The dataset we use is Animals with Attributes (AwA2), which consists of 37322 images of 50 animal classes. We will do our experiment on pre-extracted deep learning features with 2048 dimensions of AwA2.

We will split 60% of the features and labels into training set and 40% for testing set. We feed the training set without dimension reduction into SVM and test the model on testing set. Then we observe the optimal hyper-parameter by k-fold validation. The optimal hyper-parameters of linear SVM and RBF kernel SVM on AwA2 is 1

Table 1: SVM

	C	Acc.(%)
Linear	0.002	93.2815
RBF Kernel	5.0	93.5160

2.2 Feature Selection

2.2.1 Select-K-Best

Select-K-Best belongs to univariate feature selection. It removes all but the k highest scoring features. There are many options for the score function of this method and we chose the ANOVA F value. In this section, two experiment settings can derive from the rule: We use Select-K-Best on different number of aim component $[2, 5, 10, 20, 50, 100, 200, 500, 750, 1000, 1200, 1500, 2000]$, and we give our results on two types of kernel: linear kernel and radial basis function kernel. We adopt classifiction accuracy as metric in this section.

We summarize the experimental results of Select-K-Best in Tab. 2 and Fig. 5. As is shown, linear Select-K-Best model and RBF Select-K-Best model expectively exhibit the best performances at 1000 components and 750 components, reaching 93.01% and 92.96% on the classification task. We can compare the result with the simple linear SVM and simple RBF SVM, find the Select-K-Best model's performance is worse than the baselines, and we deem the reason is that Select-K-Best reduces the dimension at the cost of reducing a little accuracy. Besides, more dimensions results in higher accuracy.

We give the 2-components scattering figure of different kernel and dataset in Fig. 6, from which we can see that the data reduced to 2 dimensions using the Select-K-Best method are clustered together and cannot be well classified. This can also be inferred from the Tab. 2.

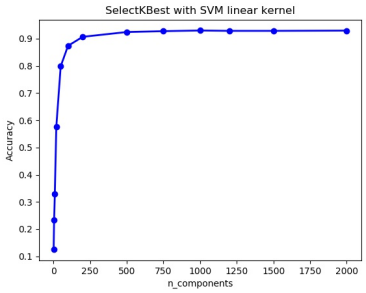
2.2.2 Variance Threshold

VarianceThreshold is one of the simple feature selection methods, removing all features whose variance is less than the threshold. In this section, two experiment settings can derive from the rule: We use Variance Threshold on different number of aim component $[0.1, 0.2, 0.3, 0.5, 0.7, 0.9]$, and we give our results on two types of kernel: linear kernel and radial basis function kernel. We adopt classifiction accuracy as metric in this section.

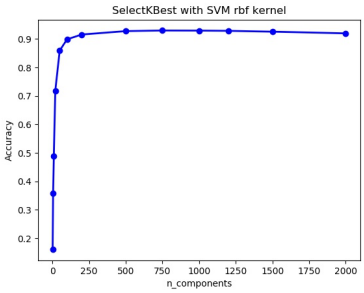
We summarize the experimental results of Variance Threshold in Tab. 3 and Fig. 7. As is shown, the linear Variance Threshold model and the RBF Variance Threshold model exhibit the best performance on 0.1 components and 0.3 components, with data dimensions of 1883 and 1022, and classification accuracy of 92.96% and 93.08% respectively. We can see that Variance Threshold is

Table 2: Comparison of Select-K-Best and baslines in Classification Task

Acc. M / d	Select-K-Best+SVM	
	Linear kernel(%)	RBF kernel(%)
Baseline	93.28	93.52
2	12.45	16.05
5	23.33	35.80
10	32.88	48.84
20	57.59	71.85
50	79.86	85.92
100	87.38	89.85
200	90.69	91.55
500	92.45	92.75
750	92.77	**92.96**
1000	**93.01**	92.92
1200	92.88	92.84
1500	92.88	91.55
2000	92.98	91.96

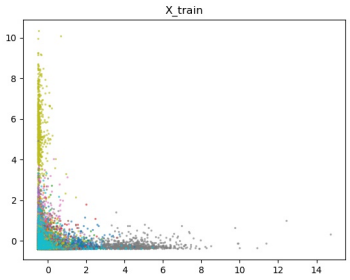


(a) metric accuracy comparison on linear Select-K-Best

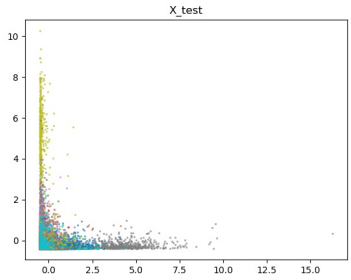


(b) metric accuracy comparison on RBF Select-K-Best

Figure 5: Select-K-Best performance on linear kernel and RBF kernel



(a) Trainset 2D scatter with Select-K-Best



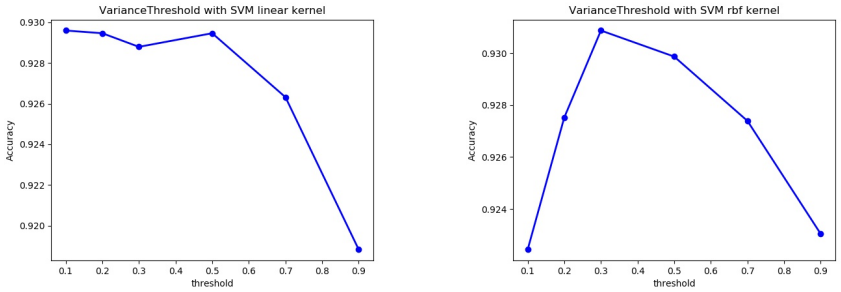
(b) Testset 2D scatter with Select-K-Best

Figure 6: Dataset 2D scatter with Select-K-Best

similar to Select-K-Best because their classification accuracy is not as good as baselines. It also reduces dimensionality at the cost of accuracy.

Table 3: Comparison of Variance Threshold and baselines in Classification Task

Acc. M / θ	Variance Threshold+SVM		
	Dimension	Linear kernel(%)	RBF kernel(%)
Baseline	2048	93.28	93.52
0.1	1883	**92.96**	92.24
0.2	1345	92.94	92.75
0.3	1022	92.87	**93.08**
0.5	620	92.94	92.98
0.7	401	92.63	92.73
0.9	271	91.88	92.30



(a) metric accuracy comparison on linear Variance Threshold

(b) metric accuracy comparison on RBF Variance Threshold

Figure 7: VarianceThreshold performance on linear kernel and RBF kernel

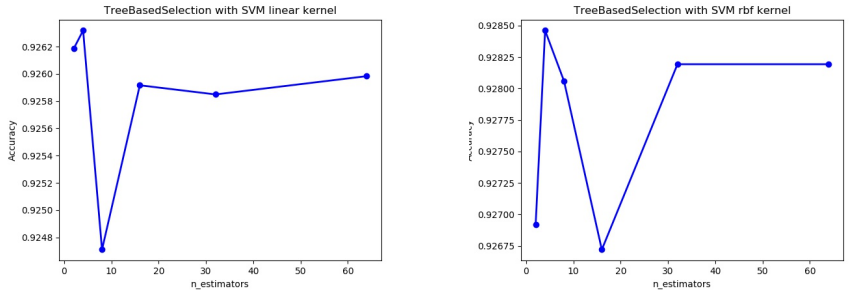
2.2.3 Tree-based Selection

Tree-based predictive models can be used to calculate the importance of features, so they can be used to remove irrelevant features. In this section, two experiment settings can derive from the rule: We use tree-based selection on different number of aim component $[2, 4, 8, 16, 32, 64]$, and we give our results on two types of kernel: linear kernel and radial basis function kernel. We adopt classifiction accuracy as metric in this section.

We summarize the experimental results of tree-based selection in Tab. 4 and Fig. 8. As is shown, the linear Variance Threshold model and the RBF Variance Threshold model exhibit the best performance both on 4 components, with data dimensions of 564, and classification accuracy of 92.63% and 92.84% respectively. We can see that the best classification accuracy of tree-based selection is not as good as Select-K-Best and Variance Threshold. According to the decision tree, the reason may be that for data with inconsistent sample sizes in each category, the information gain is biased towards those features with more values.

Table 4: Comparison of Tree-based selection and baslines in Classification Task

Acc. M / $\#est.$	Tree-based selection+SVM		
	Dimension	Linear kernel(%)	RBF kernel(%)
Baseline	2048	93.28	93.52
2	523	92.61	92.69
4	564	**92.63**	**92.84**
8	563	92.47	92.80
16	609	92.59	92.67
32	609	92.58	92.81
64	600	92.59	92.81



(a) metric accuracy comparison on linear Tree-based selection

(b) metric accuracy comparison on RBF Tree-based selection

Figure 8: Tree-based selection performance on linear kernel and RBF kernel

2.3 Feature Projection

2.3.1 PCA

Principal component analysis (PCA) is the most typical feature projection method based on dimension reduction, since PCA provides a roadmap for how to reduce a complex data set to a lower dimension to reveal the sometimes hidden, simplified dynamics that often underlie it. Kernel principal component analysis (kernel PCA) provides an extension of traditional PCA using techniques of kernel methods, which project source data into a higher dimensional space, providing with better reduction and classification performance. In this section, two experiment settings can derive from the rule: We use kernel PCA on different number of aim component $[2, 5, 10, 20, 50, 100, 200, 500, 750, 1000, 1200, 1500, 2000]$, and we give our results on two types of kernel: linear kernel and radial basis function kernel. We adopt classifiction accuracy as metric in this section.

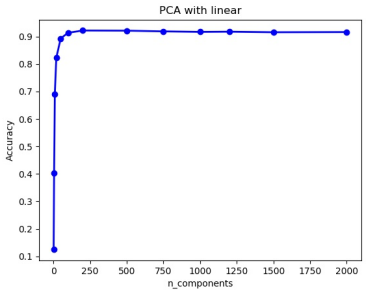
We summarize the experimental results of kernel PCA in Tab. 5 and Fig. 9. As is shown, linear kernel PCA model and RBF kernel PCA model expectively exhibit the best performances at 200 components and 750 components, reaching 92.20% and 92.50% on the classification task. We can compare the result with the simple linear SVM and simple RBF SVM, find the kernel PCA model's performance is worse than the baselines, and we deem the reason is that compoents reduction of PCA remove some effective feature as it remove the most noisy feature. And we can find RBF kernel have better performance than linear kernel in PCA tasks and none-PCA tasks, it proves the provided dataset have some features which can't be handled merely by linear kernel. Besides, in kernel PCA tasks, we can find if we have less components, RBF kernel have better performance than linear kernel.

We give the 2-components scattering figure of different kernel and dataset in Fig. 10, which can support the result that RBF kernel have better performance than linear kernel since the RBF figure can be better splited. What's more, if we compare the trainset and testset, we can find the dataset is randomly splited.

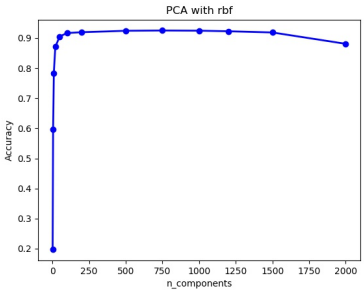
2.3.2 LDA

Linear discriminate analysis (LDA) is a dimension reduction technique that is closely related to principal component analysis (PCA) and factor analysis in that they both look for linear combinations of variables which best explain the data. However, in contrast, LDA explicitly attempts to model the difference between the classes of data. In this section, we use LDA model to project features into [2, 3, 5, 10, 20, 30, 40] components and respectively combine RBF SVM and Linear SVM to finish the classification task. We adopt classifiction accuracy as metric in this section.

We summarize the experimental results of LDA-based model in Tab. 6 and Fig. 11. As is shown, based on LDA method, Linear kernel SVM and RBF kernel SVM have better performance as the compoents number increase and reach accuracy of 90.56% and 91.47% respectively. As PCA-based method, some components loss may occur while we reduce the data dimension, but with the components number increase, we can find the accuracy nearly closed to the traditional SVM

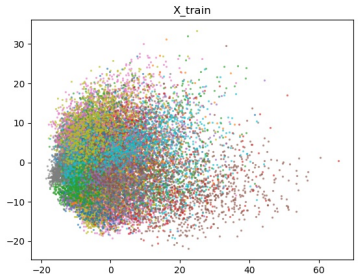


(a) metric accuracy comparison on linear PCA

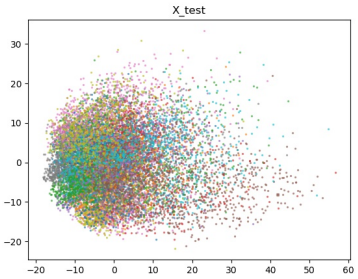


(b) metric accuracy comparison on RBF PCA

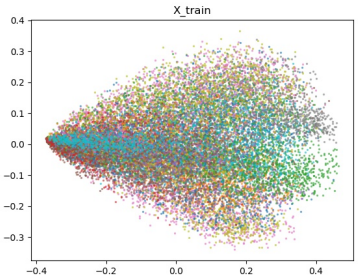
Figure 9: Kernel PCA performance on linear kernel and RBF kernel



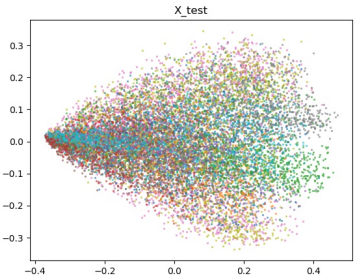
(a) Trainset 2D scatter with linear PCA



(b) Testset 2D scatter with linear PCA



(c) Trainset 2D scatter with RBF PCA



(d) Testset 2D scatter with RBF PCA

Figure 10: Dataset 2D scatter with linear kernel and RBF kernel

Table 5: Comparison of kernel PCA and baslines in Classification Task

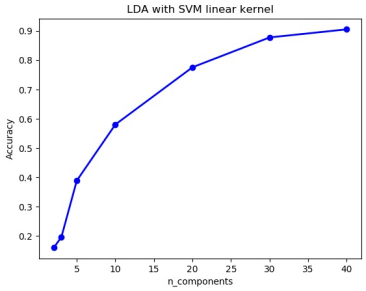
Acc. M / d	PCA+SVM	
	Linear kernel(%)	RBF kernel(%)
Baseline	93.28	93.52
2	12.46	19.68
5	40.31	59.68
10	68.99	78.25
20	82.27	87.07
50	89.31	90.53
100	91.37	91.66
200	**92.20**	91.94
500	92.14	92.42
750	91.90	**92.50**
1000	91.70	92.44
1200	91.78	92.26
1500	91.57	91.85
2000	91.64	88.08

model. Besides, we can find RBF kernel have better performance than linear kernel in LDA tasks and none-LDA tasks, it proves the provided dataset have some features which can't be handled merely by linear kernel. Especially, in LDA tasks, we can find if we have less components, RBF kernel have much better performance than linear kernel.

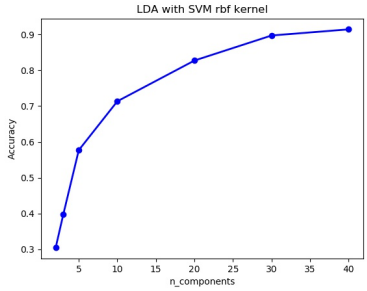
We give the 2-components scattering figure of different kernel and dataset in Fig. 12. Compared with Fig. 10, we can conclude LDA have better classes distance than PCA, which supports the correctness of the experiment. And we can find the random splitness of the dataset is covered.

Table 6: Comparison of LDA-based SVM and baslines in Classification Task

Acc. M / d	LDA+SVM	
	Linear kernel(%)	RBF kernel(%)
Baseline	93.28	93.52
2	16.02	30.39
3	19.63	39.80
5	38.93	57.69
10	58.08	71.36
20	77.61	82.75
30	87.80	89.75
40	**90.56**	**91.47**

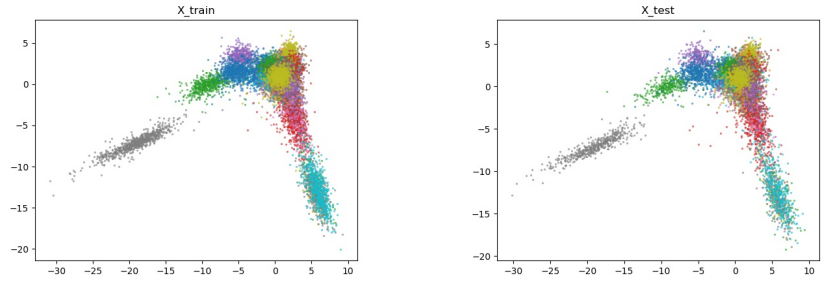


(a) metric accuracy comparison on LDA with linear SVM



(b) metric accuracy comparison on LDA with RBF SVM

Figure 11: LDA-based model performance on linear kernel SVM and RBF kernel SVM



(a) Trainset 2D scatter with LDA (b) Testset 2D scatter with LDA

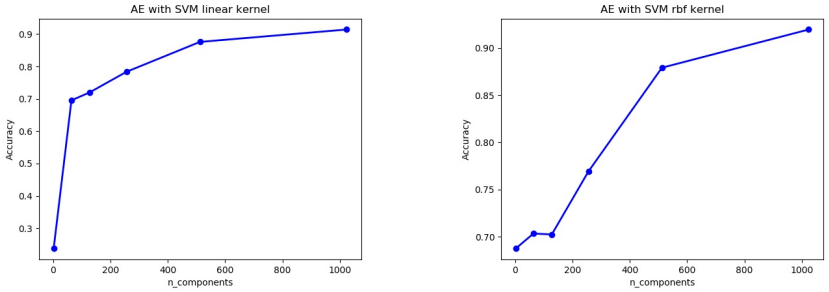
Figure 12: Dataset 2-components scatter with LDA

2.3.3 AutoEncoder

AutoEncoder is a unsupervised method to determine the essential dimensions of high-dimensional dataset, which can construct the complete dataset features with minimun loss, i.e. with encoder reducing the data dimension to 'bottleneck' and decoder to restruct the data, we can get the origin sample data or get the minimun loss. In this section, we use AutoEncoder with MSE loss to get the low-dimensional feature and respectively combine linear SVM and RBF SVM to achieve the classification task. We adopt classifiction accuracy as metric in this section.

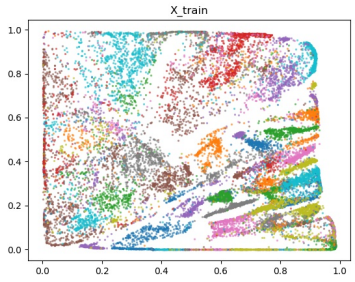
We summarize the experimental results of AutoEncoder-based model in Tab. 7 and Fig. 13. As is shown, with the components number increases, models receive better performance especially for RBF kernel model. We can recognize that when we have less compoents, model with RBF SVM may receive a higher accuracy. AE with linear SVM and with RBF SVM respectively reach 91.42% and 91.95% in classification accuracy when components number comes to 1024.

We give the 2-components scattering figure of different kernel and dataset in Fig. 14. We can find that some classes can't be linearly differed in 2-components, and we deem it's the reason why RBF SVM have better performance than linear one when we have less components.

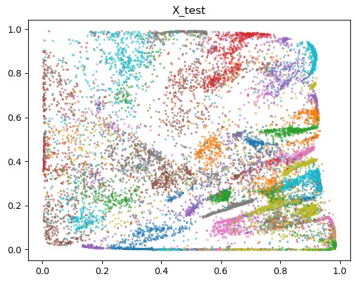


(a) metric accuracy comparison on AutoEncoder with linear SVM (b) metric accuracy comparison on AutoEncoder with RBF SVM

Figure 13: AutoEncoder-based model performance on linear kernel SVM and RBF kernel SVM



(a) Trainset 2D scatter with AutoEncoder



(b) Testset 2D scatter with AutoEncoder

Figure 14: Dataset 2-components scatter with AutoEncoder

Table 7: Comparison of AutoEncoder-based SVM and baslines in Classification Task

Acc. M / d	AutoEncoder+SVM	
	Linear kernel(%)	RBF kernel(%)
Baseline	93.28	93.52
2	23.71	68.75
64	69.52	70.35
128	72.01	70.25
256	78.35	76.93
512	87.59	87.90
1024	**91.42**	**91.95**

2.4 Feature Learning

2.4.1 t-SNE

The technique t-SNE is a common method for high-dimensional data visualizing by giving each data point a location in a two or three-dimensional space. In this section, two experiment settings can derive from the rule: t-SNE used Barnes-Hut approximation to calculate gradient on dimension 2 and 3, and test parameter perplexity in $[10.0, 20.0, 30.0, 40.0, 50.0]$.

tSNE with linear SVM receives its best performance 5.39% when $perplexity = 10.0$ while tSNE with RBF SVM reaches 5.95% when $perplexity = 10.0$. We deem the reason of low accuracy is that aim components number is not enough to extract efficient features to predict.

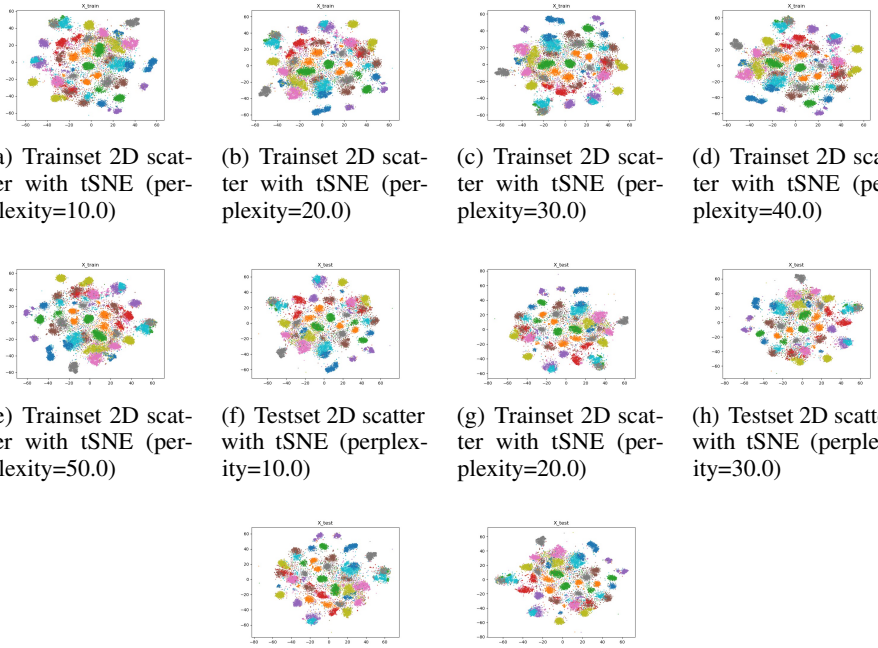
We also give the 2-components scattering result in Fig. 15. tSNE enables different data clusters in low-dimension to be distant enough and the data points belonging to the same cluster to be close, which benefits classification. As is shown, we can differ various classes in 2D scattering. Besides, we can find with the parameter $perplexity$ increase, tSNE model can process more data.

2.4.2 LLE

Locally Linear Embedding (LLE) is a feature learning mathod which tried to keep the linear relation between neighbor samples. In this section, we respectively use 4, 8 and 16 neighbors to rule the experiments and our aim components range is $[2, 3, 50, 100, 500, 1000, 2000]$.

We summarize the experimental results of AutoEncoder-based model in Tab. 7 and Fig. 16. We can find if we maintain more neighbor relations and more aim components, we can get almost better performance, i.e. LLE with Linear SVM receives 90.60% and LLE with RBF SVM receives 90.18%. Performance of LLE is satisfiying beacause original features may have simple structure and the data is randomly scatering in the high-dimension space.

We also give the 2-components scattering result in Fig. 17. When we use 4 neighbors to reconstruct the sample, we can find its poor performance. For data can be regarded as manifold structure, LLE performs while when we give the proper neighbors' number.



(a) Trainset 2D scatter with tSNE (perplexity=10.0)

(b) Trainset 2D scatter with tSNE (perplexity=20.0)

(c) Trainset 2D scatter with tSNE (perplexity=30.0)

(d) Trainset 2D scatter with tSNE (perplexity=40.0)

(e) Trainset 2D scatter with tSNE (perplexity=50.0)

(f) Testset 2D scatter with tSNE (perplexity=10.0)

(g) Trainset 2D scatter with tSNE (perplexity=20.0)

(h) Testset 2D scatter with tSNE (perplexity=30.0)

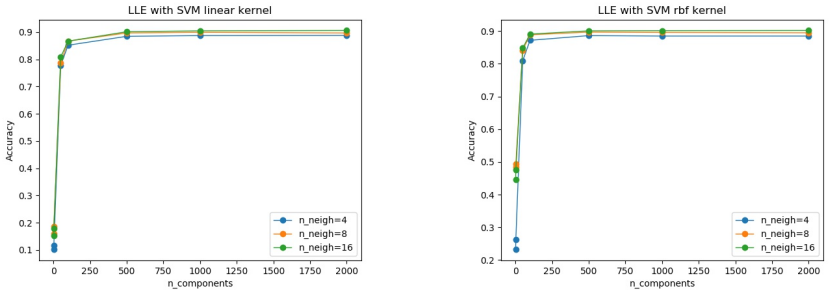
(i) Trainset 2D scatter with tSNE (perplexity=40.0)

(j) Testset 2D scatter with tSNE (perplexity=50.0)

Figure 15: Dataset 2D scatter with tSNE on perplexity of $\{10.0, 20.0, 30.0, 40.0, 50.0\}$

Table 8: Comparison of LLE-based SVM and baslines in Classification Task

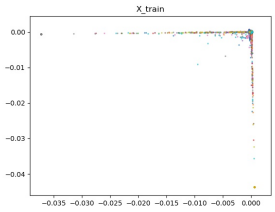
Acc. M / $\#neigh.$	LLE+SVM					
	Linear kernel(%)			RBF kernel(%)		
	4	8	16	4	8	16
Baseline	93.28	93.28	93.28	93.52	93.52	93.52
2	10.12	15.93	15.13	23.19	48.47	44.62
3	11.57	18.55	17.81	26.21	49.43	47.65
50	77.72	78.67	80.82	80.91	84.18	84.79
100	85.20	86.72	86.69	87.18	88.87	89.08
500	88.47	89.70	90.15	**88.62**	**89.69**	90.07
1000	88.74	**89.93**	90.48	88.49	89.58	90.12
2000	**88.82**	89.70	**90.60**	88.49	89.48	**90.18**



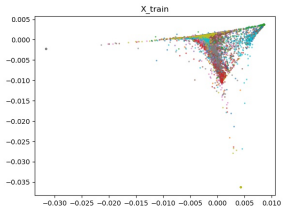
(a) metric accuracy comparison on LLE with linear SVM

(b) metric accuracy comparison on LLE with RBF SVM

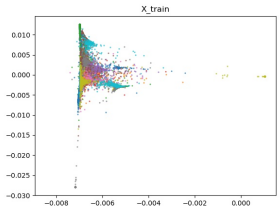
Figure 16: LLE-based model performance on linear kernel SVM and RBF kernel SVM



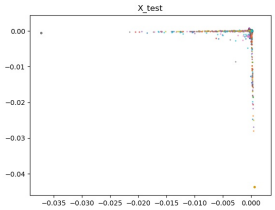
(a) Trainset 2D scatter with LLE (neighbor_num=4)



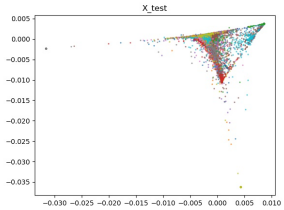
(b) Trainset 2D scatter with LLE (neighbor_num=8)



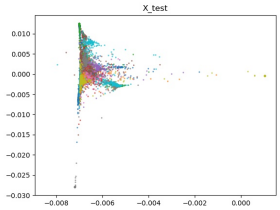
(c) Trainset 2D scatter with LLE (neighbor_num=16)



(d) Testset 2D scatter with LLE (neighbor_num=4)



(e) Testset 2D scatter with LLE (neighbor_num=8)



(f) Testset 2D scatter with LLE (neighbor_num=16)

Figure 17: Dataset 2D scatter with LLE on neighbors of $\{4, 8, 16\}$

3 Conclusion

In this project, we evaluate different dimension reduction techniques by measuring their classification performance when combined with SVM model. In this section, we want to give our perspectives on drawbacks and merits of these dimension reduction method. Importantly, we'll conclude the situations to use these methods. In general, we think feature projection and feature learning have better average performance than feature selection since minimizing effective information loss. We think kernel PCA is a universal method for dimension reduction while LDA receives a lot in the classification tasks. AutoEncoder is a good choice for generative model. In learning model, t-SNE has advantages in data visualization while LLE have great performance for data's manifold structure.

References

[1] Karl Pearson. 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):559–572, 1901.

[2] Zhihua Zhou. *Machine Learning*. Tsinghua University Press, 2016.

[3] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[4] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[5] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.