Project 1 of CS245: Dimensionality Reduction

515030910369, Sicheng Zuo, Ginga479726995@sjtu.edu.cn
516030910044, Yujie Yang, yangyujie@sjtu.edu.cn
516030910021, Hongxiang Yu, sjtu.yuhongxiang@sjtu.edu.cn
5130309210, Zheng Gong, 573965625@qq.com

April 3, 2019

1 Introduction

In this project, we tried several different methods of feature reduction. The dataset we used is Animals with Attributes(AwA2), which is composed of 37322 images with 50 animals classes. Additionally, each image has a deep learning feature vector(dimension 2048) extracted from ResNet. There are two parts in this project. Firstly, we used SVM for image classification based on the deep learning feature. Secondly, we used several different methods to reduce the dimensonality of deep learning features and performed image classification again based on the obtained low-dimensional features. What's more, we also tried to figure out the optimal dimensionality reduction method and the optimal dimensionality.

2 SVM using deep learning features

In this question we used the origin deep learning features to train the SVM and use the result as a baseline. To find a set of suitable hyper parameters, we splited the data in each category into 60% for training and 40% for testing. In addition, 5-fold cross-validation was used within the training set to determine hyper parameters.

Though the project specified linear SVM, we tried two kernels(*linear*, *rbf*) of SVM. Because the process of searching hyper parameters is time consuming, we used **GridSearchCV** to do the job in parallel. Here in Table 2 and Table 2 shows the parameters grid and corresponding 5-fold cross-validation mean score.

Table 1: Parameters grid of linear kernel SVM

C	10E-5	10E-4	0.001	0.01	0.1	1	10	100	1000
score	68.36%	90.60%	92.97%	92.62%	92.40%	92.40%	92.40%	92.40%	92.40%

Table 2: Parameters grid of rbf kernel SVM

gamma \ C	10E-5	10E-4	0.001	0.01	0.1	1	10	100	1000
0.001	4.41%	4.41%	4.41%	36.07%	87.07%	92.76%	93.03%	93.02%	93.02%
0.01	4.41%	4.41%	4.41%	4.41%	5.75%	20.54%	24.09%	24.09%	24.09%
0.1	4.41%	4.41%	4.41%	4.41%	4.41%	4.41%	4.41%	4.41%	4.41%
1	4.41%	4.41%	4.41%	4.41%	4.41%	4.41%	4.41%	4.41%	4.41%

As shown in the tables, $C = 0.001$ is the best parameter for the linear kernel SVM, and $C = 10, gamma = 0.001$ is the best parameter for the rbf kernel. After we got the best hyper parameters, we applied corresponding model to the testing set. Here is the accuracy on testing set:

Table 3: SVM accuracy on testing set

	C	gamma	accuracy
linear	0.001	\	93.19%
rbf	10	0.001	93.28%

We will use the result in Table 2 as a baseline to compare with the results of following tests.

3 Feature selection

Feature selection means for selecting a subset of components in the origin features. By this way, we can not only improve estimators' accuracy scores, but

also boost their performance on very high-dimensional datasets. In this part, we tried four different feature selection methods, which are *Variance Threshold*, *Select-K-best*, *Tree-based feature selection* and *Genetic Algorithm*.

3.1 Variance Threshold

Variance Threshold is a simple baseline approach to feature selection. It removes all features whose variance doesn't meet some threshold. In this method, the value of threshold can be adjusted. The range of features' variance is from 0.16 to 2.6. Therefore, we tried different value of threshold from 0.2 to 2.0 with a strip of 0.1. The results of our experiments is shown in Figure-1.

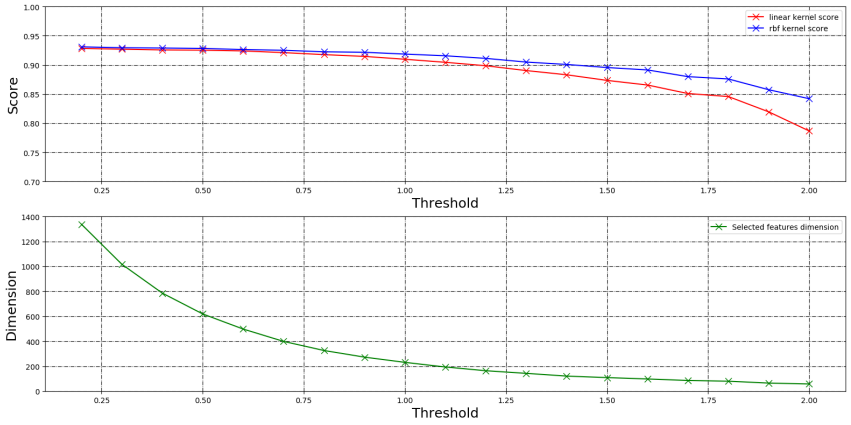

Figure 1: 5-fold cross-validation Score and Dimension VS Threshold(Variance Threshold)

Apparently, with the increasing of threshold value, the accuracy and the dimension of selected features decreases. When the threhold is 0.2, we achieves the best accurary. The results of Variance Threshold is shown in Table-3.1. The dimension of subset features is 1338, only about half of origin feature dimensions, but the performance is very close to the baseline.

Table 4: Variance Threshold Results

kernel	threshold	dimension	accuracy
linear	0.2	1338	93.12%
rbf	0.2	1338	93.25%

3.2 Select-K-best

Select-K-best is a method belongs to univariate feature selection, which works by selecting the best features based on univariate statistical tests. Select-K-best removes all but the k highest scoring features. Since the origin feature dimension is 2048, we chose the value of k from 100 to 2000 with a strip of 100. The results is shown in Figure-2.

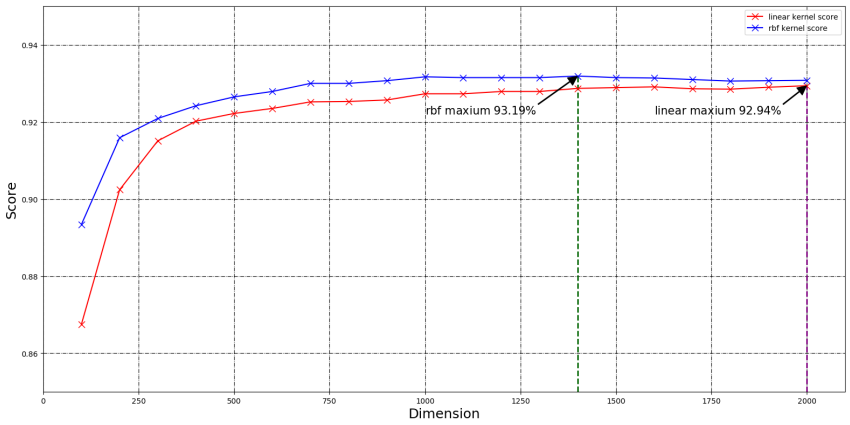

Figure 2: 5-fold cross-validation Score VS dimension(Select-K-best)

According to the figure-2, the optimal value of dimensions are different for the linear kernel and rbf kernel. For the linear kernel, the best dimension value is 2000, but for the rbf kernel, the best dimension value is 1400. By doing tests on the testing set with these configurations, we got the results below.

Table 5: Select-K-best Results

kernel	dimension	accuracy
linear	2000	93.19%
rbf	1400	93.36%

From table-3.2, we can see that the dimension is reduced without harming the performance. The performance of linear kernel is the same as baseline. The rbf kernel's performance is slightly better than the baseline, and the dimension of subset features is only 1400.

3.3 Tree-based feature selection

In this part, we used two components which are *SelectFromModel* and *ExtraTreesClassifier*. *SelectFromModel* is a meta-transformer that can be used along with any estimator that has a *coef_* or *feature_importances_* attribute after fitting. The features are considered unimportant and removed, if the corresponding *coef_* or *feature_importances_* values are below the provided threshold parameter. Here in this experiment, we use the default threshold value which is empirically optimal value in most cases. *ExtraTreesClassifier* is tree-based estimator can be used to compute feature importances, which in turn can be used to discard irrelevant features. By using the combination of *SelectFromModel* and *ExtraTreesClassifier*, we did the tests on testing datasets and got following results. We can see from table-3.3 that both two kernels' perfomance is lower than the baseline, and the effect is not as good as Variance-Threshold and Select-K-best.

Table 6: Tree-based feature selection Results

kernel	accuracy
linear	92.56%
rbf	92.78%

3.4 Genetic Algorithm

Genetic Algorithm is a method used to generate high-quality solution to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection. In this problem, we are looking for an optimal subset features. In other words, we are looking for an optimal binary mask. The binary mask has the same dimension as the features. By applying different binary masks to origin features, we can get different feature subsets. We all know that the key point to design a genetic algorithm is the fitness function. In this problem, we use the score generated by SVM as the fitness. But there is a problem that traditional genetic algorithm is serial, since the SVM is time consuming, it may need a large mount time to run the genetic algorithm. To solve this problem and make full use of the multi-cores, we developed a multi-threads parallel framework of genetic algorithm by ourselves. The multi-threads parallel genetic algorithm is 40 times faster than the traditional genetic algorithm. In addition, the flow chart of the genetic algorithm is shown in Figure-3.

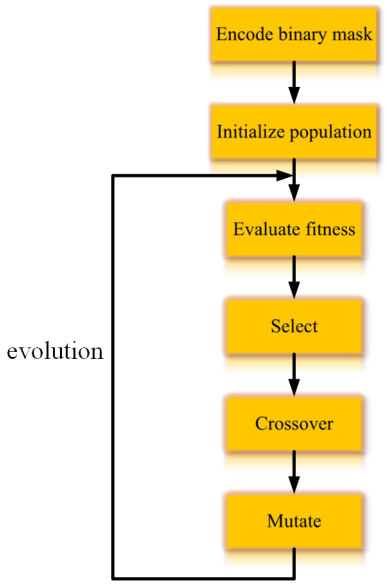

Figure 3: Flow chart of Genetic algorithm

According to the flow chart, there are several steps in our algorithm:

- Initialize 80 individuals with randomly generated binary masks as the initial population.
- Evaluate each individual's fitness with SVM model.
- Select top 30% from the population and eliminate the remaining.
- Create new individuals by randomly selecting two individuals and crossovering with a probability
- Each newly created individual mutate with a probability.
- The new generation is created.

We run the algorithm with 80 population size and iterate for 100 rounds. For the linear kernel, we show the best score per generation and number of selected features per generation in Figure-4. The same thing is shown in Figure-5 for rbf kernel.

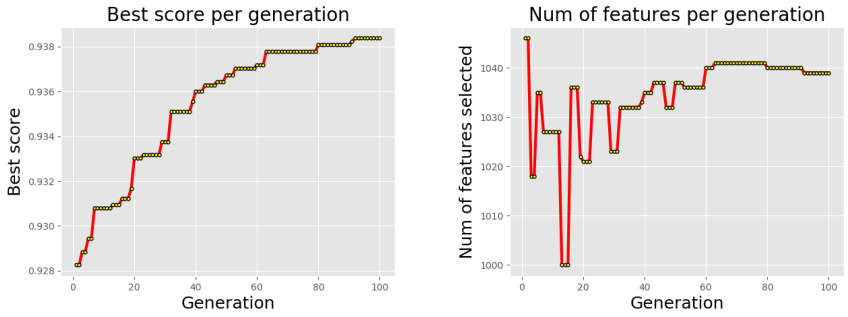

Figure 4: Best score per generation and number of selected features per generation for linear kernel

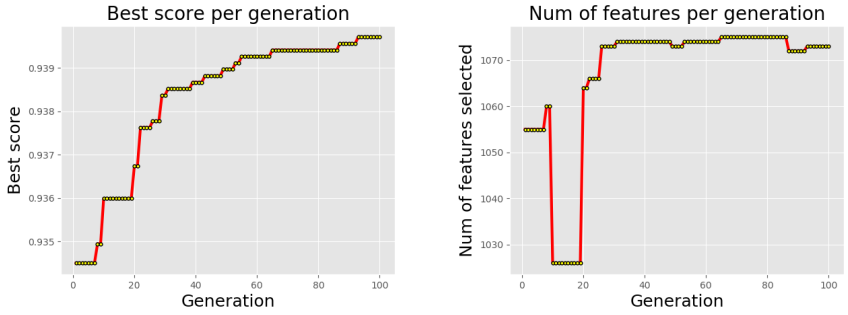

Figure 5: Best score per generation and number of selected features per generation for rbf kernel

After iterating for 100 rounds, we used the best individual's gene(the binary mask) to generate a feature subset and then fed the feature subset to SVM. The results of evaluation on testing dataset are in Table-3.4

Table 7: Genetic Algorithm results

kernel	dimension	accuracy
linear	1039	92.63%
rbf	1073	92.95%

The results of Genetic algorithm is not bad, but there are still some problems.

- The cardinality of search space for this problem is 2^{2048}, a population of

80 is too small. If the population size is larger, the algorithm will works better.

- Since we generate the initial population randomly, most of the binary masks have number of ones around 1000. As a result, the search scope of solution space is limited, we may not get the optimal solution.
- As we can see from the figure-5, the best score is still increase with the generation, maybe iterating for 100 rounds is not enough.
- In order to reduce running time

As we know, the calculation of fitness is really time consuming. Although we developed multi-threads parallel genetic algorithm, we could not solve the problem perfectly due to the limitation of time and hardware resources. However, Genetic Algorithm is still a highly promising method for feature selection.

4 Feature Projection

Feature projection means for transforming the data in high-dimensional space to a space of fewer dimensions. In this part, we tried the feature projection with two frequently used methods, which are *LDA(linear discriminant analysis)* and *PCA(principal component analysis)*.

4.1 LDA

LDA is a classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using bayes' rule. It can reduce the dimensionality of the input by projecting it to the most discriminative directions. In this method, we mainly adjusted number of components for dimensionality reduction. What's more, the number of components should be small than *n_classes - 1*. As a result, we chose the value of parameter *n_components* from 1 to 49. The result of LDA is shown in Figure-6

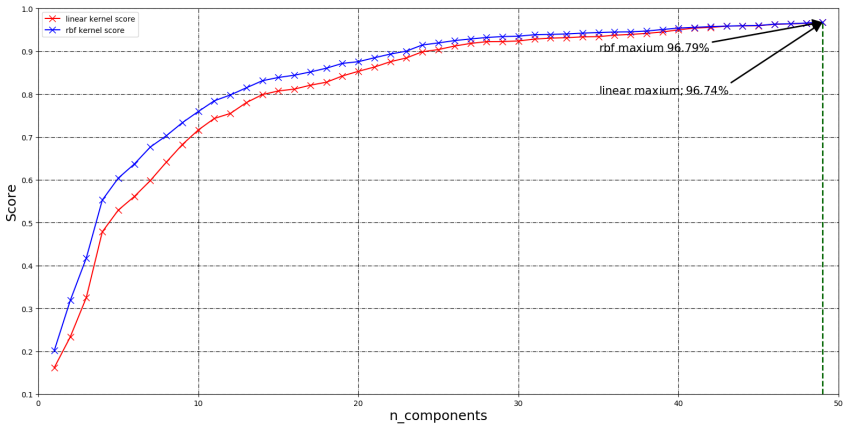

Figure 6: 5-fold cross-validation Score VS n_component(LDA)

As we can see from the figure-6, both the linear kernel and rbf kernel achieve highest score when $n\_components = 49$. By applying this configuration to the testing set, we can get the results shown in Table-4.1.

Table 8: LDA Results

kernel	component	accuracy
linear	49	93.07%
rbf	49	92.73%

Obviously, LDA reduces the dimension of features from 2048 to 49 with little harm to the performance, which means that LDA is really a effective method to dimension reduction.

4.2 PCA

PCA is a commonly used linear dimensionality reduction using singular value decomposition of the data to project it to a lower dimensional space. We mainly tuned the parameter *n_components* to get better performance. In the python's PCA module, we can set $n\_components =' mle'$, so that the program will automatically guess the reduced dimension using MLE(maximum likelihood estimate). If $0 < n\_components < 1$, the program will select the number of components such that the amount of variance that needs to be explained is greater than the percentage specified by *n_components*. In this problem, we

tuned the value of *n_component* from 0.9 to 1.0 with a step of 0.01. When we use $n\_components =' mle'$, the dimensionality of projected feature is 2047, which is almost the same with origin dimensionality. Furthermore, the results of *n_components* range from 0.1 to 1.0 is shown in Figure-7.

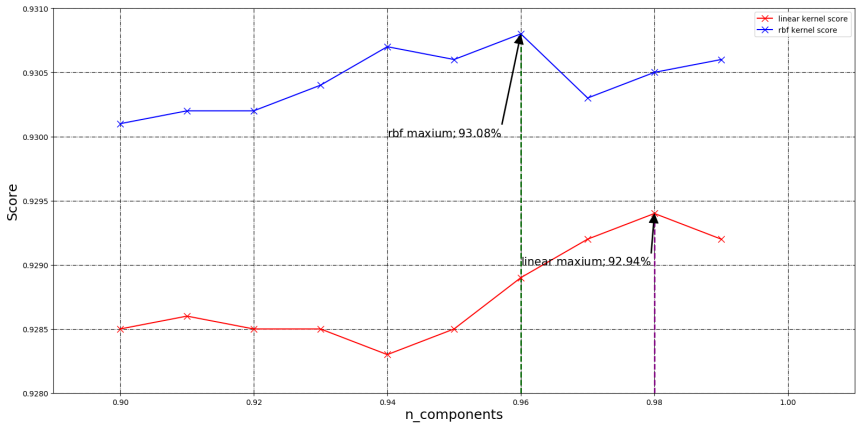

Figure 7: 5-fold cross-validation Score VS n_component(PCA)

The optimal value of *n_components* is clearly showed in the figure-7. By Applying this configuration to the testing dataset, we can get the Accuracy. Here in the Table-4.2 shows the effectiveness of PCA.

Table 9: PCA Results

kernel	component	dimension	accuracy
linear	0.98	1305	93.17%
rbf	0.96	952	93.33%

For the linear kernel SVM, the PCA reduces dimensionality from 2048 to 1305, with performance nearly equal to the baseline. For the rbf kernel SVM, the PCA reduces dimensionality from 2048 to 952, and the performance is better than the baseline.

5 Feature Learning

Feature learning is a set of techniques that allows a system to automatically discover the representations needed for classification from raw data. In this

part, we implemented an autoencoder framework to reduce the dimensionality and used T-SNE to visualize all the samples in a plot.

5.1 T-SNE

T-distributed Stochastic Neighbor Embedding(T-SNE) is a tool to visualize high-dimensional data. It coverts similarities between data points to joint probabilities and tries to minimize the KL divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. In this experiment, we first tried the parameter $n\_components = 2$ and $n\_components = 3$, then we followed the instruction of official document to use LDA to reduce number of dimensions to a reasonable amount(e.g 49) before using T-SNE method. Here is the visualization after using T-SNE directly.

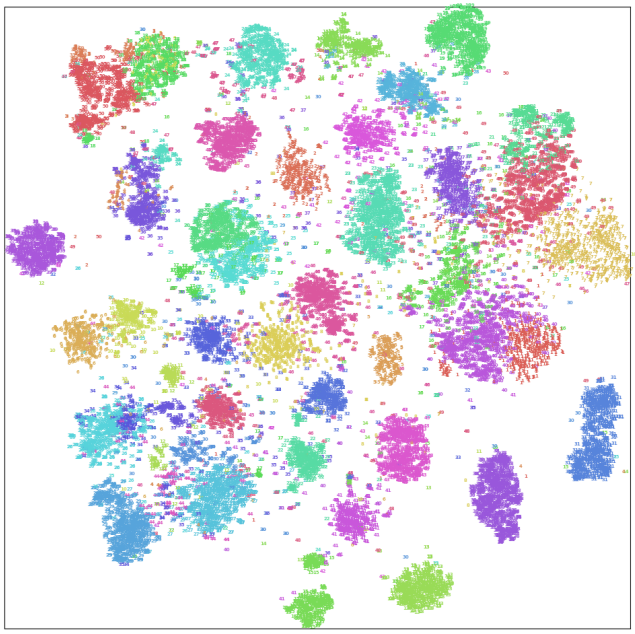

Figure 8: T-SNE visualization

From figure-8, we can see that most of the samples with the same label are gathered together, but there are still some sample with different labels overlap. However, we failed to use the 2-dimensional feature vector after T-SNE reduction to do classification, and the accuracy is very low, only about 8.6%. We can

not find the reason now, may there is some problems with the code.

The official document of T-SNE suggests us to use another dimensionality reduction method to reduce the number of dimensions to a reasonable amount. Hence, we applied LDA to the dataset first and then used the T-SNE to visualize. Here is the visualization after applying LDA.

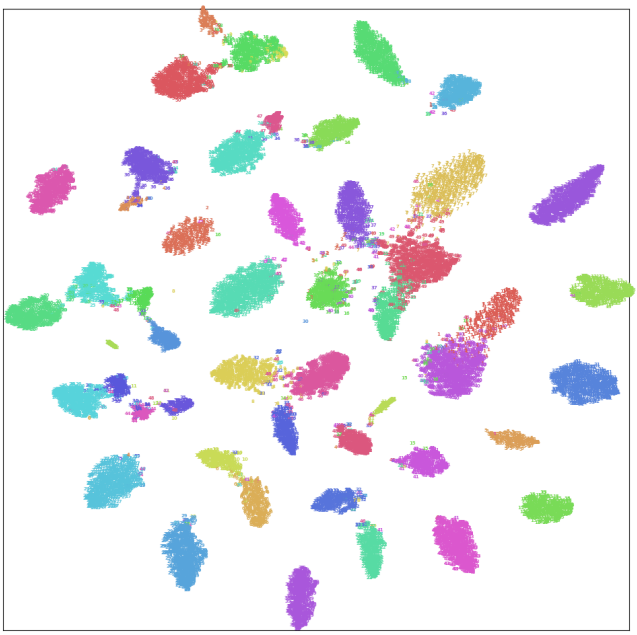

Figure 9: T-SNE visualization after applying LDA

In the figure-9, we can see that the classification visualization is obviously better after applying LDA, but the accuracy is still only 58.52%. Maybe there are still some sample with different labels overlap.

5.2 Autoencoder

An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. The aim of an autoencoder is to learn a representation(encoding) for a set of data, typically for dimensionality reduction. In this experiment, we first build a autoencoder via keras, here is the architecture of the autoencoder in figure-10.

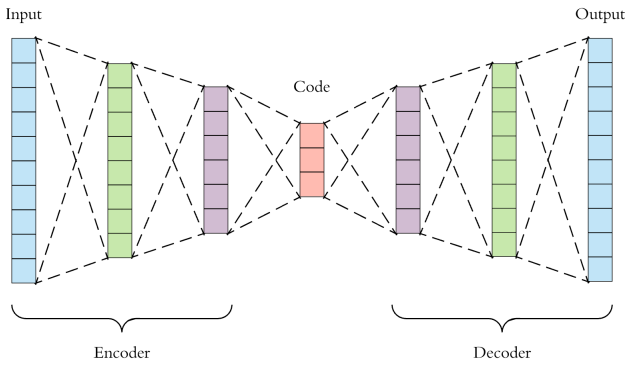

Figure 10: Structure of autoencoder

It is worth noting that the encoding length of the encode layer is adjustable. We selected 5 values(64, 128, 256, 512, 1024) to set as the encoding length to study the influence of encoding length to the accuracy.

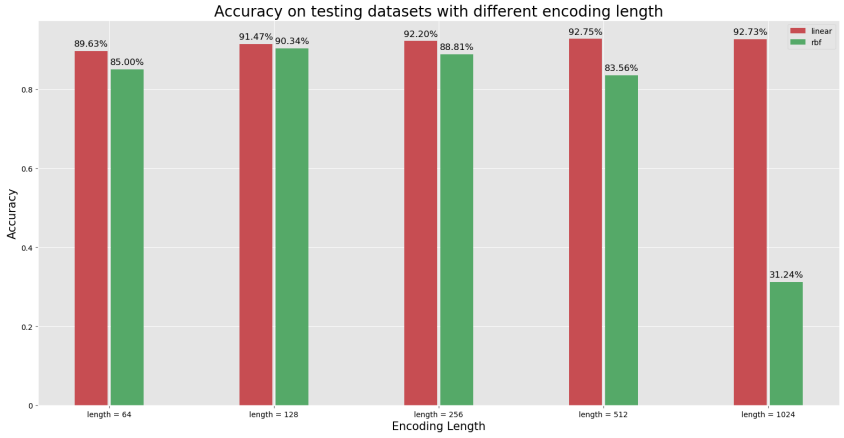

Figure 11: Results of autoencoder

Obviously, the best performance of linear kernel is achieved when encoding length is 512, the corresponding optimal encoding length of rbf kernel is 128. The results is summarize in Table-5.2. The effect of autoencoder is not so good as the methods of feature projection.

Table 10: Autoencoder Results

kernel	encoding length	accuracy
linear	512	92.75%
rbf	128	90.34%

6 Summary

Here, we will summarize the results of all the methods we tried in the Table-6.

Table 11: Results of all methods

	linear-accuracy	linear-dim	rbf-accuracy	rbf-dim
baseline	93.19%	2048	93.28%	2048
Variance-Threshold	93.12%	1338	93.25%	1338
Select-K-best	93.19%	2000	93.36%	1400
Tree-based selection	92.56%	\	92.78%	\
Genetic Algorithm	92.63%	1039	92.95%	1073
LDA	93.07%	49	92.73%	49
PCA	93.17%	1305	93.33%	952
T-SNE with LDA	58.52%	2	\	\
Autoencoder	92.75%	512	90.34%	128

From the table-6, we can clearly see that the most effective methods of feature selection, feature projection, feature learning are Select-K-best, PCA and autoencoder. These three methods can both effectively reduce the feature dimension while maintaining accuracy. However, we still have a lot of work to do like tuning the network structure of autoencoder and using larger population size to iterate more rounds in Genetic Algorithm. In this problem, considering the accuracy and dimension of features after reduction, we chose PCA as the most effective method for dimension reduction temporarily.