Project 1 Dimensionality Reduction

CS245 Principles of Data Science

Chen Wang
516021910501

Chenxu Zhu
516021910629

Huayu Wang
516021910503

Kangjie Xu
516021910805

***Abstract*—This document is the report for Dimensionality Reduction, the first project of Principles of Data Science. In this project, we use feature selection (*Genetic algorithm*, *VarianceThreshold*, *ANOVA F-value*, *L1-based feature selection*, *Tree-based feature selection*, *Random forest*), feature projection (*PCA*, *LDA*, *Factor analysis*) and feature learning (*t-SNE*, *Auto-encoder*, *VAE*) to do the dimensionality reduction and record the result separately. We will compare these results and find the optimal dimensionality reduction method and the optimal dimensionality.**

***Index Terms*—dimensionality reduction, feature selection, feature projection, feature learning**

I. FEATURE SELECTION

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. Feature selection techniques are used for four reasons:

1) Simplification of models to make them easier to interpret by researchers/users.
2) Shorter training times.
3) Avoid the curse of dimensionality.
4) Enhanced generalization by reducing overfitting.

Subset selection algorithms can be broken up into Wrappers, Filters and Embedded.

A. Wrapper Methods

Wrapper methods consider the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. We use a predictive model to evaluate a combination of features and assign a score based on model accuracy.

Here we use Genetic algorithm to select features.

1) Genetic Algorithm: Genetic algorithm (GA) is a meta-heuristic inspired by the process of natural selection that belongs to the larger class of *Evolutionary algorithm*. *Genetic algorithms* are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as *mutation*, *crossover* and *selection*. [1]

The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a *generation*. In each generation, the fitness of every individual in the population is evaluated. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. The flow chart is shown below.

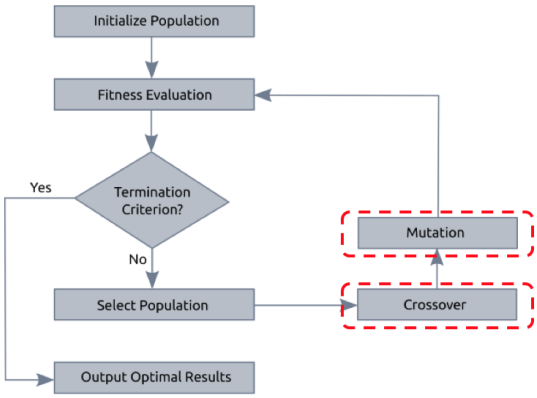


Fig. 1. Genetic algorithm

B. Filter Methods

Filter feature selection methods apply a statistical measure to assign a scoring to each feature. The features are ranked by the score and either selected to be kept or removed from the dataset. The methods are often univariate and consider the feature independently, or with regard to the dependent variable.

Here we use two methods. One is VarianceThreshold, the other is to consider ANOVA (analysis of variance) F-value.

1) VarianceThreshold: VarianceThreshold is a basic method in feature selection. It removes all features whose variance does not meet the threshold. If a feature has low variance, it means that this feature has so few information that we could remove it.

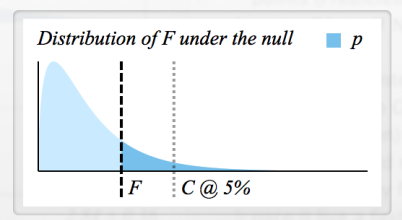
2) ANOVA F-value: Analysis of variance (ANOVA) can determine whether the means of three or more groups are different. ANOVA uses F-tests to statistically test the equality of means. [2] The F value is simply a ratio of two variances.

$$F = \frac{VariationBetweenGroups}{VariationWithinGroups} \tag{1}$$

Here we use *ANOVA* to make feature selection. We assume that the mean between different groups of the same feature is same. So how much probability do I accept this assumption?

Now we computeth F value based on data. If the variance between groups is same as the variance within groups, then F value is 1. If the variance between groups is much larger than the variance within groups, this F value will be larger.

Here we give the relationship between F value and P value.



It can be seen that each F value corresponds to a p value. The larger the F value, the smaller the p value. So it is less likely that I accept my hypothesis.

Therefore, the larger the F value, the more this feature should be preserved.

C. Embedded

Embedded methods learn which features best contribute to the accuracy of the model while the model is being created. The most common type of embedded feature selection methods are regularization methods.

Here we use L1-based feature selection and tree-based feature selection.

1) L1-based feature selection: Linear models penalized with the L1 norm have sparse solutions: many of their estimated coefficients are zero. We could use SVM with L1 norm to train a classifier. Then we could select the non-zero coefficients and remove the other coefficients.

2) Tree-based feature selection: Tree-based estimators can be used to compute feature importances, which in turn can be used to discard irrelevant features.

3) Random forest: Random forest is conducted by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classed or mean prediction of the individual trees. [3]

II. FEATURE PROJECTION

Feature projection transforms the data in the high-dimensional space to a space of fewer dimensions. The data transformation may be linear, as in principal component analysis (PCA), but many nonlinear dimensionality reduction techniques also exist. [4] Here we use PCA and LDA to do feature projection.

A. PCA

Principal components analysis (PCA) is a good way to interpret the data into a lower-dimensional and more meaningful norm in a linear process. The function of *PCA* can be shown as Figure 2.

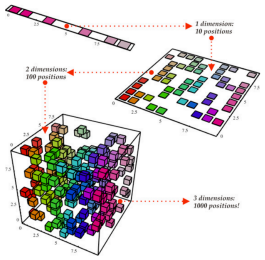


Fig. 2. An illustration for PCA.

We also give the implementation of *PCA* algorithm (X is the data matrix with size $n \times p$, where n is the number of samples and p is the feature dimension):

1) Normalize the X.

$$x = x - \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2}$$

2) Calculate the covariance matrix C.

$$C = \frac{1}{n-1} X^T X \tag{3}$$

3) Compute the eigenvectors and eigenvalues of C by diagonalization.

$$C = VDV^T \tag{4}$$

where D is the diagonal matrix with eigenvalues λ_i and λ_i is sorted in a decreasing order.

4) Select the first k columns of V as W so that

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq Threshold \tag{5}$$

5) Transform data into a lower-dimension space.

$$X' = W^T X \tag{6}$$

B. LDA

Linear Discriminant Analysis (LDA) is most commonly used as dimensionality reduction technique in the pre-processing step. *LDA* is "supervised" and computes the directions ("linear discriminants") that will represent the axes that that maximize the separation between multiple classes. The goal of *LDA* is to maximize the distance in-between-class and minimize the distance within-class. [5] The function of *LDA* can be shown as Figure 3.

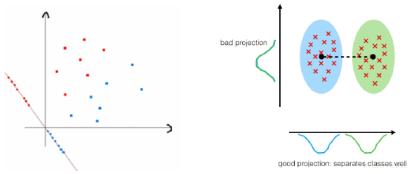


Fig. 3. An illustration for LDA.

We also give the implementation of *LDA* algorithm (reduce the dimensions of a d-dimensional dataset by projecting it onto a k-dimensional subspace):

1) Compute the d-dimensional mean vectors for the different classes from the dataset.
2) Compute the scatter matrices (in-between-class and within-class scatter matrix).
3) Compute the eigenvectors $(e_1, e_2, \cdots, e_d)$ and corresponding eigenvalues $(\lambda_1, \lambda_2, \cdots, \lambda_i)$ for the scatter matrices.
4) Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $d \times k$ dimensional matrix W (where every column represents an eigenvector).
5) Use this $d \times k$ eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the matrix multiplication: $Y = X \times W$ (where X is a $n \times d$-dimensional matrix representing the n samples, and y are the transformed $n \times k$-dimensional samples in the new subspace).

C. Factor analysis

Factor analysis is a theory that the information gained about the interdependencies between observed variables can be used later to reduce the set of variables in a dataset. It helps to deal with datasets where there are large numbers of observed variables that are thought to reflect a smaller number of underlying/latent variables. [6]

Suppose we have a set of p observable random variables, $x_1, \cdots, x_p$ with means $\mu_1, \cdots, \mu_p$. Suppose for some unknown constants l_{ij} and k unobserved random variables $\mathbf{F_j}$ (called *common factors*). We have

$$x_i - \mu_i = l_{i1}F_1 + \cdots + l_{ik}F_k + \epsilon_i \tag{7}$$

where $i \in 1, \cdots, p$, $j \in 1, \cdots, k$ and $k < p$. Here, the ϵ_i are unobserved stochastic error terms with zero mean and finite variance.

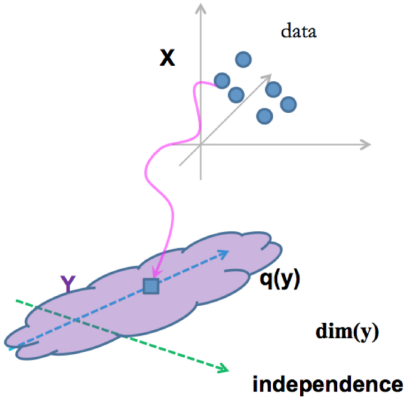


Fig. 4. Factor analysis

III. FEATURE LEARNING

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

A. t-SNE

T-distributed Stochastic Neighbor Embedding (t-SNE) is a nonlinear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. [7] The *t-SNE* algorithm comprises two main stages. First, *t-SNE* constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked while dissimilar points have an extremely small probability of being picked. Second, *t-SNE* defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the *Kullback-Leibler divergence* between the two distributions with respect to the locations of the points in the map.

Given a set of N high-dimensional objects $\mathbf{x}_1, \cdots, \mathbf{x}_N$, *t-SNE* first computes probabilities p_{ij} that are proportional to the similarity of objects $\mathbf{x}_i$ and $\mathbf{x}_j$, as follows:

$$p_{j|i} = \frac{(1 + ||\mathbf{x}_i - \mathbf{x}_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||\mathbf{x}_i - \mathbf{x}_k||^2)^{-1}} \tag{8}$$

Then *t-SNE* aims to learn a d-dimensional map $\mathbf{y}_1, \cdots, \mathbf{y}_N$ (with $\mathbf{y}_i \in \mathbb{R}^d$) that reflects the similarities p_{ij} as well as possible. It measures similarities $q_{j|i}$ between two points in the map $\mathbf{y}_i$ and $\mathbf{y}_j$, using a very similar approach.

$$q_{j|i} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||\mathbf{y}_i - \mathbf{y}_k||^2)^{-1}} \tag{9}$$

The locations of the points $\mathbf{y}_i$ in the map are determined by minimizing the *Kullback-Leibler divergence* of the distribution Q from the distribution P, that is:

$$L = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p(j|i) \log \frac{p_{j|i}}{q_{j|i}} \tag{10}$$

Note that *t-SNE* is heavy-tailed compared to *SNE* with Gaussian distribution.

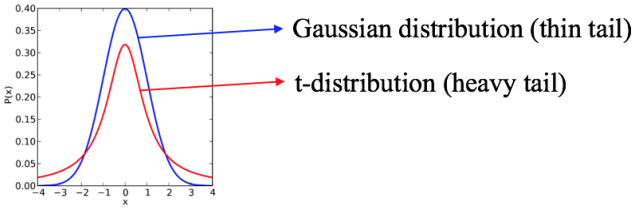


Fig. 5. t-SNE and SNE with Gaussian distribution.

B. Auto-Encoder

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 for a set of data by training the network to ignore signal *noise*. [8] Along with the reduction side, a reconstructing side is learnt, where the *autoencoder* tries to generate from the reduced encoding a representation as close as possible to its original input.

a) Structure: We can define the encoder as transition ϕ and the decoder as ψ, such that:

$$\phi : \mathbb{X} \to \mathbb{F}$$

$$\psi : \mathbb{F} \to \mathbb{X}$$

$$\phi, \psi = \text{argmax}_{\phi,\psi} ||X - (\psi \circ \phi)X||^2$$

The simplest form of an *autoencoder* is a feedforward neural network very similar to the many layer perceptrons (MLP), but with the output layer having the same number of nodes as the inputs layer. That is,

$$\mathbb{F} = MLP_{\theta_1}(\mathbb{X})$$

$$\mathbb{X} = \sigma(MLP_{\theta_2}(\mathbb{F}))$$

Where σ is the activation function such as *sigmoid function* or *rectified linear unit*.

We want to train a model that minimizes reconstruction error. Since the feature space $\mathbb{F}$ has lower dimensionality than the input space $\mathbb{X}$, then the feature vector $\phi(x)$ can be regarded as a compressed representation of the input x, which is shown in Fig.6

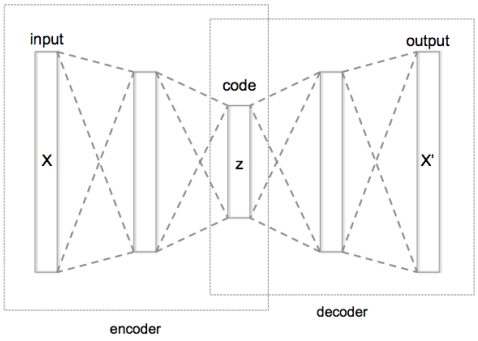


Fig. 6. The structure of Auto-Encoder

b) With Noise: In order to obtain a more robust low-dimensional representation, we add some noise to the model and train the decoder to learn how to remove the noise to reconstruct the input. By doing so, the model can be prevented from overfitting the training data, so that the representation obtained by the encoder has better performance on the test set.

Here, we add noise to the hidden layer. That is,

$$\phi : \mathbb{X} \to \mathbb{F}$$
$$\psi : \mathbb{F} + \delta \to \mathbb{X}$$

Where δ is a noise that $\delta \sim \mathcal{N}(0, 1)$. This definition contains the following implicit assumptions: the higher level representations are relatively stable and robust to the noise.

C. Variational Auto-Encoder (VAE)

Variational Auto-Encoder models inherit the *Auto-Encoder* architecture, but make strong assumptions concerning the distribution of latent variables. It assumes that the data is generated by a directed graphical model $p(x|z)$ and that the encoder is learning an approximation $q_\phi(z|x)$ to the posterior distribution $p\psi(x|z)$ where ϕ and ψ denote the parameters of the encoder (recongnition model) and decoder (generative model) respectively [**?**]. [9] The loss function has the following form:

$$\mathcal{L}(\phi, \psi, x) = D_{KL}(q_\phi(z|x)||p_\psi(z)) - \mathbb{E}_{q_\phi(z|x)}(\log p_\psi(x|z))$$

Commonly, the distribution is chosen such that they are factorized Gaussians:

$$q_\phi(z|x) = \mathcal{N}(\rho(x), \omega^2(x)I)$$

$$p_\psi(x|z) = \mathcal{N}(\mu(z), \sigma^2(z)I)$$

Where $\rho(x)$ and $\omega^2(x)$ are the encoder outputs, while $\mu(z)$ and $\sigma^2(z)$ are the decoder outputs. The structure is shown as Fig.7.

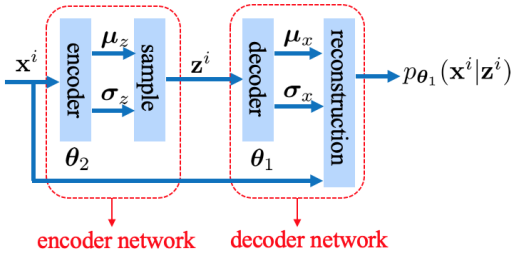


Fig. 7. The structure of VAE

IV. EXPERIMENTS AND RESULTS

In this section, we will introduce something about the experiment procedure and the results of different methods will be displayed and compared.

A. Data processing

1) Download *Animals with Attributes (AwA2)* dataset from $https://cvml.ist.ac.at/AwA2/$, which consists of 37322 images of 50 animal classes with pre-extracted deep learning features for each image.
2) Split the features and labels into 60% for training and 40% for testing.

B. Results of feature selection

1) SVM: We do not reduce the dimensionality of deep learning features and use linear SVM to train a classifier. We try different parameters C. When $C = 0.001$, we get the highest $Accuracy = 0.93304$.

2) Genetic algorithm: In terms of feature selection, we can apply the genetic algorithm here. Fortunately, Distributed Evolutionary Algorithms in Python(DEAP) has provided us with an advanced toolkit. To calculate Feature subset fitness, we utilize logistic Regression and cross validation. As for some hyper-parameters, we set population to 20, generation to 4, considering limited time and computation resources. Commonly, we just apply genetic algorithm to the whole set, however, this way costs too much time.

To be adapted to it, we split features to small groups of 100 features, do feature selection independently and combine the result together. Secondly, we iterate the method above 2 times, and we select 3 subsets of features. The feature subset size and test accuracy is listed in the table below.

After 3 iterations, we get 3 feature subset of different sizes.

TABLE I
DIFFERENT FEATURE SUBSET SIZE TO REDUCE THE DIMENSION

Feature subset size	Accuracy
445	0.9119
728	0.9207
1185	0.9317

3) VarianceThreshold: We filter the features according to their variance. We try different thresholds to do the dimensionality reduction.

TABLE II
DIFFERENT THRESHOLD TO REDUCE THE DIMENSION.

Threshold	Dimension	Accuracy
0.1	1879	0.9325
0.3	1021	0.9313
0.5	624	0.9289
0.7	400	0.9236
1.0	230	0.9142

We could see that VarianceThreshold reduce the dimension with the cost of a little decrease of accuracy.

4) ANOVA F-value: We filter the features according to their F value. We try different dimensions to do the dimensionality reduction.

TABLE III
DIFFERENT REDUCED DIMENSION FOR ANOVA F-VALUE

Dimension	Accuracy
1024	0.93304
512	0.9245
256	0.9159
128	0.8965

We could see that the effect of *ANOVA F-value* is similar to variance. It reduces the dimension with the cost of a little decrease of accuracy. But it exists something interesting that when reduced dimension is 1024, its accuracy does not decrease. It shows that there exists some redundant features in the data.

5) L1-based feature selection: We use SVM with L1 norm to select the features. We could change C to change the dimension reduced.

TABLE IV
DIFFERENT REDUCED DIMENSION FOR L1 NORM

Dimension	Accuracy
1867	0.9324
971	0.9316
321	0.9225
67	0.8649

We could see that the effect of L1 norm is satisfied and similar to variance and F value.

6) Tree-based feature selection: We use tree-based model to select the important features. After selection, the remain dimension and accuracy is shown below.

TABLE V
TREE-BASED FEATURE SELECTION

Dimension	Accuracy
584	0.9276

The tree-based model also get a satisfied result.

7) Random forest: Since the computation cost is acceptable, we test the result with different hyper-parameters.

Different n_estimators($feature\_subset = 200$, $max\_depth = 10$):

TABLE VI
DIFFERENT N_ESTIMATORS TO REDUCE DIMENSION

n_estimators	Accuracy
1	0.8964
2	0.9009
3	0.9027
4	0.9047
8	0.9042
12	0.9038
16	0.9046

Different feature subset size($n\_estimator = 200$, $max\_depth = 10$):

TABLE VII
DIFFERENT FEATURE SUBSET SIZE TO REDUCE DIMENSION

n_estimators	Accuracy
8	0.3931
16	0.5624
32	0.7099
64	0.8297
128	0.8968
256	0.9114
512	0.9213

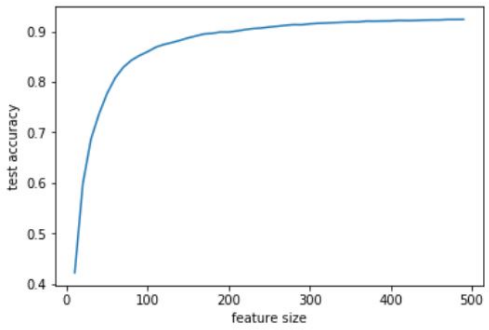


Fig. 8. test set accuracy – feature subset size

TABLE VIII
THE RESULT OF PCA

Reserved Variance	Dimension	Accuracy
0.95	848	0.9253
0.9	467	0.9255
0.8	189	0.9196

C. Results of feature projection

1) PCA: We use LDA to reduce the dimension. Here we try different reserved variances.

Here we could see that the data with a lower reserved variance and fewer dimensions has a higher accuracy. That is amazing. In my opinion, the first example may have some redundant or even interfering information.

2) LDA: We use LDA model to map the data into a low-dimension space and we show the result below.

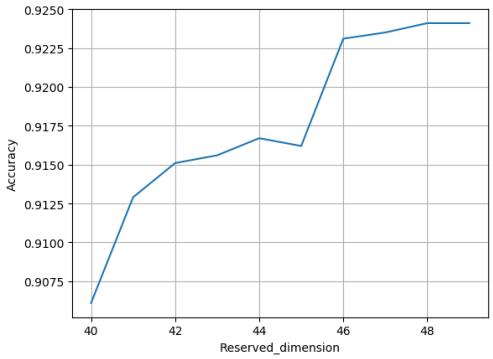


Fig. 9. The result of LDA.

We could see that the data mapped by LDA could have a satisfied accuracy with a low dimension.

3) Factor analysis: The observations are assumed to be caused by a linear transformation of lower dimensional latent factors and added Gaussian noise. Without loss of generality the factors are distributed according to a Gaussian with zero mean and unit covariance. The noise is also zero mean and has an arbitrary diagonal covariance matrix.

We change hyper-parameters $n\_components$ and here follows a table and a line-plot graph below.

TABLE IX
THE RESULT OF FACTOR ANALYSIS

n_components	Accuracy
8	0.4645
16	0.6709
32	0.7976
64	0.8601
128	0.8929
256	0.9087
512	0.9203

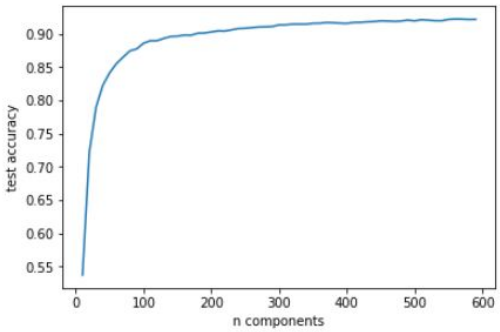


Fig. 10. test set accuracy – n_components

D. Results of feature learning

1) t-SNE: We change the parameter $n\_components$ between 2 and 3 to reduce the dimension and test the accuracy. The result is shown in the following table.

TABLE X
THE RESULT OF T-SNE

n_components	Accuracy
2	0.9017
3	0.9042

The results of 2 and 3 components do not have large difference. It can been seen that both results of *t-SNE* are not so satisfying. I guess that many related information has been lost during the reduction to 2 or 3 components. Since the process of *t-SNE* is more likely in order to be visualized, the result is understandable.

2) Auto-Encoder: We want to know the effect of the dimension of the feature data. So we first compress the raw data to different dimensions through *Auto-Encoder*, then the Linear SVM is used to classify the dimension-reduced data and we record the best results on test set. The parameters we used in our experiment is shown in Tab.XII.

The accuracy based on different dimensionality is show in Fig.6. From the graph we can conclude that as the dimension increases, the accuracy increases at the beginning stage and the tendency will stop when reach a certain threashold.

TABLE XI
THE PARAMETER IN AUTO-ENCODER

Parameter	Value
batch size	200
learning rate	0.001
train epoch	10
hidden dim	[32,64,128,256,512,1024]

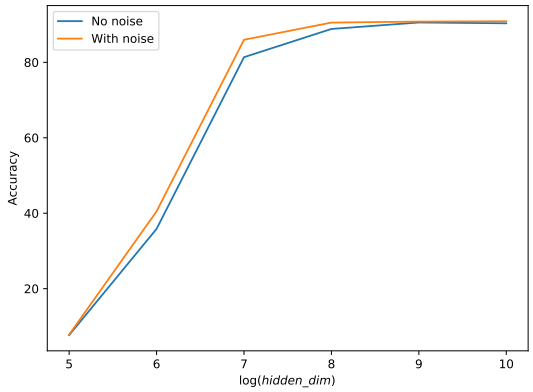


Fig. 11. The best accuracy on different dimensionality of Auto-Encoder

At the same time, it can be seen that adding noise to the model can get better classification accuracy on the test set. This means we can let the model learn to de-noise while learning to reconstruct the input, so that we can get a more robust intermediate representation. The lower target dimensionality is, the greater the benefit of adding noise is.

However, using SVM to classify the vectors after dimension reduction through *Auto-Encoder*, the result is inferior to the baseline which is without dimension reduction. We attribute this to the fact that dimensionality reduction will lose some of the original information, but simplifies the complexity and computation of the model, which is a trade off.

3) VAE: In order to compare the dimension reduction effects of *VAE* and *Auto-Encoder*, we choose the same hyper-parameters as *autoencoder* in this experiments. However, as the loss function of *VAE* contains more terms, the convergence rate will slow down. We increase the training epoch to 50. The result of *VAE* is shown in Fig.12.

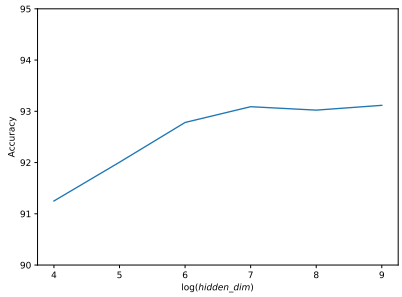


Fig. 12. The best accuracy on different dimensionality of VAE

The accuracy of *VAE* is much higher than that of *autoencoder*, especially when the target dimension is low. When the dimension is reduced to 32, the accuracy of *VAE* can exceed 92 on the test set, but the representation obtained by *Auto-Encoder* cannot be classified by SVM. This means that *VAE* can get low-dimensional representations with better quality. We consider that the constraint of KL divergence is very effective for dimensionality reduction.

Also, we can see that the results of *VAE* can benefit from the increase of dimensions. When the $hidden\_dim$ is 512, the result obtained by *VAE* can exceed the performance of the raw data which is without dimension reduction.

V. DISCUSSIONS

We list the highest accuracy of different methods and their corresponding reserved dimension.

TABLE XII
THE HIGHEST ACCURACY OF DIFFERENT METHODS AND THEIR CORRESPONDING RESERVED DIMENSION

Methods	Accuracy	Dimensions
GA	0.9317	1185
VarianceThreshold	0.9325	1879
ANOVA	0.9330	1024
L1	0.9324	1867
Tree	0.9276	584
Random forest	0.9213	512
PCA	0.9255	467
LDA	0.9242	48
FA	0.9203	512
t-SNE	0.9042	3
Auto-encoder	0.9213	512
VAE	0.931	512

It can be seen that the optimal method of dimensionality reduction in our experiment is *ANOVA* and its corresponding optimal dimensionality is 1024.

It can be concluded that the accuracy will be higher with the increasement of dimensionality. Feature selection can maintain a high accuracy with high reserved dimensions. However, when the dimensionality is below 200, since it is not involved in the procedure of advanced data processing, feature selection will not remain endurable. Feature projection can still keep a good performance around the dimensionality of 100. Meanwhile, feature learning is remarkable under some extremely low dimensions. For example, the accuracy of *VAE* can still exceed 92 even when the dimension is reduced to 32.

Therefore, if exceedingly low dimension is not required, we can choose methods of feature selection, which are in low computing complexity. If it is required, methods of feature projection and feature learning will be better choices.

ACKNOWLEDGMENT

This experiment is successfully conducted thanks to the effort of teammates and the guidance from Li Niu.

REFERENCES

[1] Mitchell, Melanie. An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press. ISBN 9780585030944, p. 2. 1996

[2] Belle, Gerald van. Statistical rules of thumb (2nd ed.). Hoboken, N.J: Wiley. ISBN 978-0-470-14448-0. 2008

[3] Ho, Tin Kam (1995). Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, pp. 278282. 1416 August 1995.

[4] Hotelling, H. Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology, 24, 417441, and 498520. 1993

[5] McLachlan, G. J. Discriminant Analysis and Statistical Pattern Recognition. Wiley Interscience. ISBN 978-0-471-69115-0. 2004

[6] Jolliffe I.T. Principal Component Analysis, Series: Springer Series in Statistics, 2nd ed., XXIX, 487 p. 28 illus. ISBN 978-0-387-95442-4. Springer, NY, 2002

[7] van der Maaten, L.J.P.; Hinton, G.E. "Visualizing Data Using t-SNE". Journal of Machine Learning Research. 9: 25792605. Nov 2008

[8] Liou, Cheng-Yuan; Cheng, Wei-Chen; Liou, Jiun-Wei; Liou, Daw-Ran. "Autoencoder for words". Neurocomputing. 139: 8496. 2014

[9] Liou, Cheng-Yuan; Huang, Jau-Chi; Yang, Wen-Chie. "Modeling word perception using the Elman network". Neurocomputing. 71 (1618): 3150. 2008