Statistics Learning Project Report

Yilong Zhao
022033910058

Abstract

The task is to train a model on the training dataset and predict on a noised test set. I try two data processing methods, PCA and Reduced Rank LDA, to reduce the dimension of the samples. After that, I try three models, LDA, SVM, and MLP to fit the training dataset. K-fold cross-validation and AIC are used to evaluate the performance of the models. Results show that PCA is much better than Reduced Rank LDA on the noised test dataset. LDA can achieve the best test accuracy on the test dataset.

1. Data Processing

1.1. Principal Components Analysis (PCA)

The task is to train a model on the training dataset and predict on a noised test dataset. The training set has 6666 pictures, while the test set have 2857 pictures. Each picture has 1536 features. This feature dimension is too large for models. Therefore, I use principal components analysis (PCA) to reduce the dimension of features. The steps of PCA are as follows:

1. Substrate the mean value of each feature.
2. Compute the covariance matrix with python function $np.cov$.
3. Compute the covariance matrix's eigenvalue and eigenvector with python function $np.linalg.eig$, and sort the eigenvalue in descending order.
4. Keep the first N columns of eigenvectors, and use the $1536 \times N$ submatrix to map the features from the original space to the new space.

One key value to decide is the dimension of the new space N. We use the following equation to compute the error of PCA:

$$1 - \frac{\sum_{i=0}^{N} \lambda_i}{\sum_{i=0}^{1536} \lambda_i} < \varepsilon \tag{1}$$

Where λ_i is the i^{th} eigenvalue of the covariance matrix. We set $\varepsilon = 0.3$, and from Figure 1, the corresponding $N = 112$.

1.2. Reduced Rank Linear Discriminant Analysis (Reduced Rank LDA)

Another method to reduce the dimension is Reduced Rank Linear Discriminant Analysis (Reduced Rank LDA). Unlike PCA, Reduced Rank LDA tries to find the discriminant direction which minimizes this overlap for Gaussian data. The step is as follows:

1. Estimate the centroid matrix $M_{k\times P}$, and the variance of the sample $W = \hat{\Sigma}$.
2. Compute $M^* = MW^{-1/2}$.
3. Compute B^*, the covariance matrix of M^*, and compute the eigenvector matrix V^*.
4. The new discriminant variable is $Z_l = v_l^T X$, with $v_l = W^{-1/2} v_l^*$

Note that the Reduced Rank LDA can only reduce the number of dimensions to $L-1$ dimensions, with $L = 20$ being the number of classes. Therefore, compared to PCA, Reduced Rank LDA loses too much information. We will discuss this in the experiment Section 5.

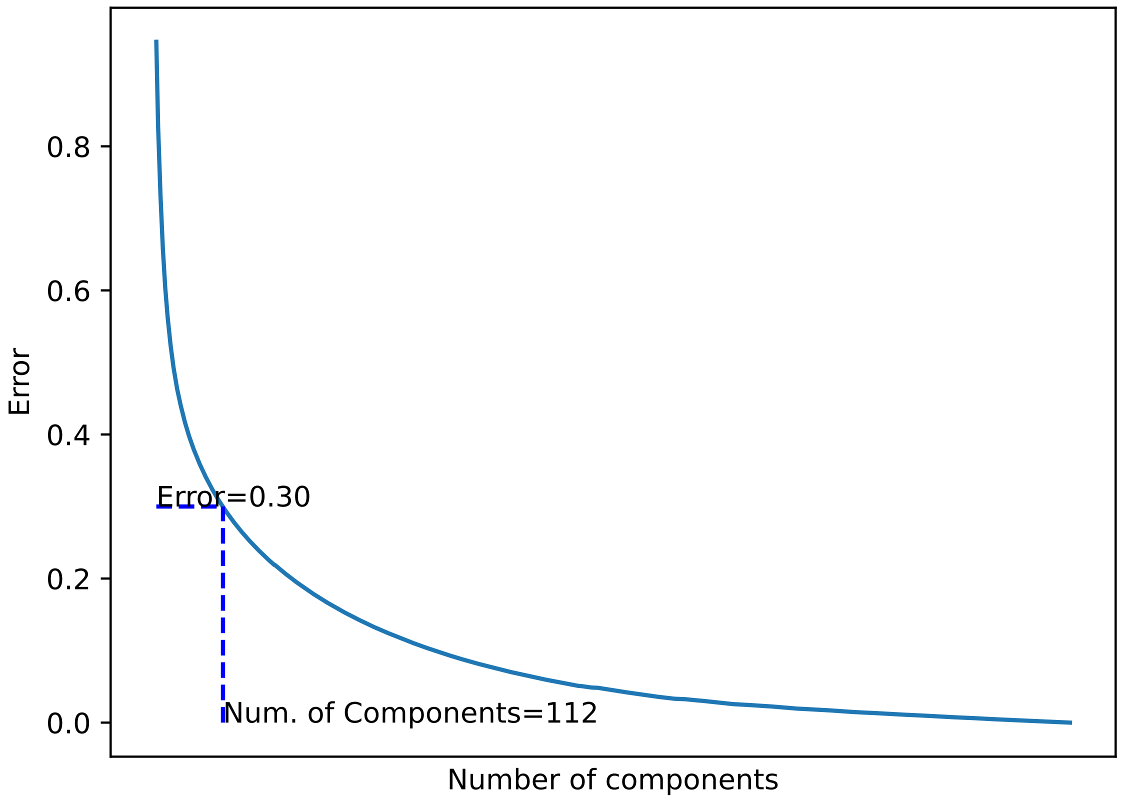


Figure 1. Relationship between PCA error and the dimension of the new space N

2. Linear Discriminant Analysis (LDA)

The first classification model I try is Linear Discriminant Analysis (LDA). This means that we introduce an assumption that the distribution is the same among different classes. A linear line is used to distinguish between the classes. The LDA Rule is:

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k\Sigma^-1\mu_k + \log\pi_k \tag{2}$$

where μ_k is the mean of the k^{th} class, π_k is the proportion of class k, Σ is the variance of the samples. For each category $k = 1, ..., 20$, we compute the score $\delta_k(x)$. The class of the sample is the category with the maximum $delta_k(x)$ value.

The parameters are estimated by:

$$\hat{\pi}_k = \frac{N_k}{N} \tag{3}$$

$$\hat{\mu}_k = \sum_{g_i=k} x_i N_k \tag{4}$$

$$\hat{\Sigma} = \sum_{k=1}^{K}\sum_{g_i=k}(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T / (N-K) \tag{5}$$

3. Support Vector Machine (SVM)

The second classification model I try is the support vector machine (SVM). SVM uses a linear surface to distinguish between two classes. The aim of SVM is to maximize the distance between the linear surface and support vectors. However, the problem is an ill-posed problem. Therefore, instead of solving the original problem, I try to solve the dual problem.

The first step is to solve a quadratic programming (QP) problem:

$$\begin{aligned} \max_{\alpha} \quad & \mathcal{L} = -\frac{1}{2} \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) + \sum_{i=1}^{n} \alpha_i & (6) \\ s.t. \quad & \alpha_i \geqslant 0 & (7) \\ & \textstyle\sum_{i=1}^{n} \alpha_i y_i = 0 & (8) \end{aligned}$$

We solve the QP problem with Python library *cvxopt*. We use the function *cvxopt.solvers.qp(P,q,G,h,A,b)* to solve the QP problem:

$$\begin{aligned} \min \quad & \frac{1}{2} x^T P x + q^T x & (9) \\ s.t. \quad & Gx \leqslant h & (10) \\ & Ax = b & (11) \end{aligned}$$

Therefore, the parameters are:

$$\begin{aligned} P &= yy^T . * K(x,x) & (12) \\ q &= -1 & (13) \\ G &= -\mathcal{I} & (14) \\ h &= 0 & (15) \\ A &= y & (16) \\ b &= 0 & (17) \end{aligned}$$

Then we select the samples with $\alpha > 0$ as support vector x_{t_j}. However, none of the α in the solution is strictly 0 due to the computational error. To address this, we manully set a threshold $\alpha_{tr} = 1e-6$, and reguard $\alpha_i < \alpha_{tr}$ as 0. Then a new sample z can be classified with:

$$\hat{y} = \operatorname{sgn}\left(\alpha_t \cdot y_t \cdot \left(x_t^T z\right)\right) \tag{18}$$

For every two classes, we train an SVM model to classify them. As there are 20 classes, we train 190 SVM models. Given a test sample, each SVM model votes on it. The class with the most votes is regarded as the result of the classification.

The kernel function $K(x_i, x_j)$ is decided using model selection. We consider the polynomial kernel function:

$$K(x_i, x) = \left(x_i^T \cdot x + 1\right)^p \tag{19}$$

In model selection, we test the order $p = 1, 2, 3$ and select the best value of the order.

4. Multilayer Perceptron (MLP)

The third model I try is Multilayer Perceptron (MLP). I directly use the MLP model provided by *sklearn* library. The activation function is ReLU, and the optimizer is Adam. We scan different hidden neuron number in the experiment Section 5.

5. Model Selection

5.1. K-fold Cross Validation

I use K-fold cross-validation to perform model selection. The training set is randomly divided into K sub-sets. Given a model f, we train the model with removing k^{th} fold data, and use the k^{th} fold data to test the model:

$$CV\left(\hat{f}\right) = \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, \hat{f}^{-}k\left(x_i\right)\right) \tag{20}$$

Table 1. Models

Model	Description
Reduced Rank LDA	LDA model; Reduce dimension with Reduced Rank LDA ($N = 19$)
PCA + LDA	LDA model; Reduce dimension with PCA ($N = 112$)
PCA + SVM-1	SVM model with linear kernel; Reduce dimension with PCA ($N = 112$)
PCA + SVM-2	SVM model with kernel $p = 2$; Reduce dimension with PCA ($N = 112$)
PCA + SVM-3	SVM model with kernel $p = 3$; Reduce dimension with PCA ($N = 112$)
PCA + MLP-100	MLP model with 100 hidden neurons; Reduce dimension with PCA ($N = 112$)
PCA + MLP-200	MLP model with 200 hidden neurons; Reduce dimension with PCA ($N = 112$)
PCA + MLP-300	MLP model with 300 hidden neurons; Reduce dimension with PCA ($N = 112$)
PCA + MLP-400	MLP model with 400 hidden neurons; Reduce dimension with PCA ($N = 112$)

Table 2. Cross Validation and AIC Result

Model	AIC	CV	Accuracy
Reduced Rank LDA		0.9541	0.0560
PCA + LDA	0.9649	0.9548	0.9447
PCA + SVM-1	0.7858	0.7790	0.7647
PCA + SVM-2	0.0489	0.0489	0.0378
PCA + SVM-3	0.0489	0.0489	0.0378
PCA + MLP-100	1.0	0.9532	0.9251
PCA + MLP-200	1.0	0.9551	0.9286
PCA + MLP-300	1.0	0.9535	0.9321
PCA + MLP-400	1.0	0.9577	0.9349

5.2. Akaike information criterion (AIC)

The second model selection method is the Akaike information criterion (AIC). The AIC equation is:

$$AIC\left(\alpha\right) = \overline{err}\left(\alpha\right) + 2\frac{d\left(\alpha\right)}{N}\hat{\sigma}_{\varepsilon}^{2} \tag{21}$$

where $\overline{err}\left(\alpha\right)$ is the training error, and $d\left(\alpha\right)$ is the number of parameters.

5.3. Results

The models we compare are listed in Table 1. We compare LDA, three SVM models with different kernels, and 4 MLP models with different hidden neuron numbers. The cross-validation is listed in Table 2. The best model is MLP with 400 hidden neurons. Both LDA and MLP can achieve relatively high accuracy.

Then we compare the two data processing methods, PCA and Reduced Rank LDA. We add noise under standard normal distribution to the validation set. Figure 2 and Figure 3 depicts the relation between cross-validation and noise of reduced rank LDA and PCA. Reduced rank LDA can barely tolerate noise. Even a slight noise with a variance of 0.0001 can severely affect the predict accuracy. While for PCA, the noise does not influence the accuracy much.

6. Accuracy

We test the models in Table 1 on the test set, and get the accuracy from the Kaggle platform. Results are listed in Figure 4 and Figure 5. The LDA model can achieve the best accuracy 94.47%. For reduced rank LDA, the test accuracy is much lower than the cross-validation accuracy. This is because the test set samples are noised, and reduced rank LDA cannot tolerate noise.

[?]

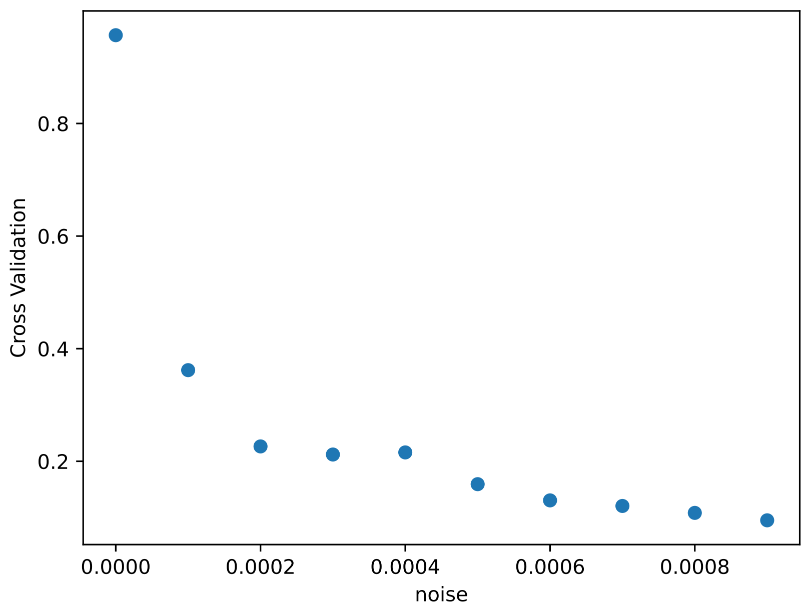


Figure 2. Cross validation of Reduced Rank LDA under different noise variance.

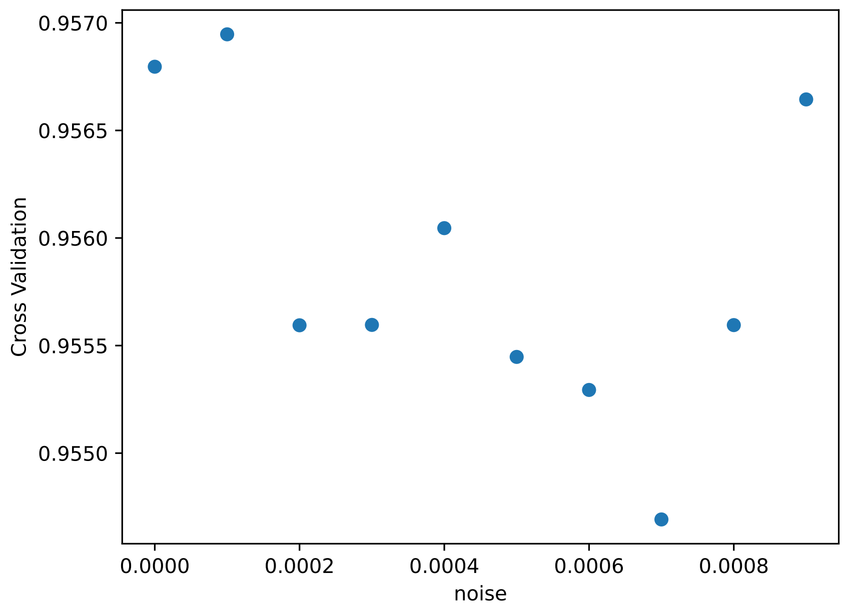
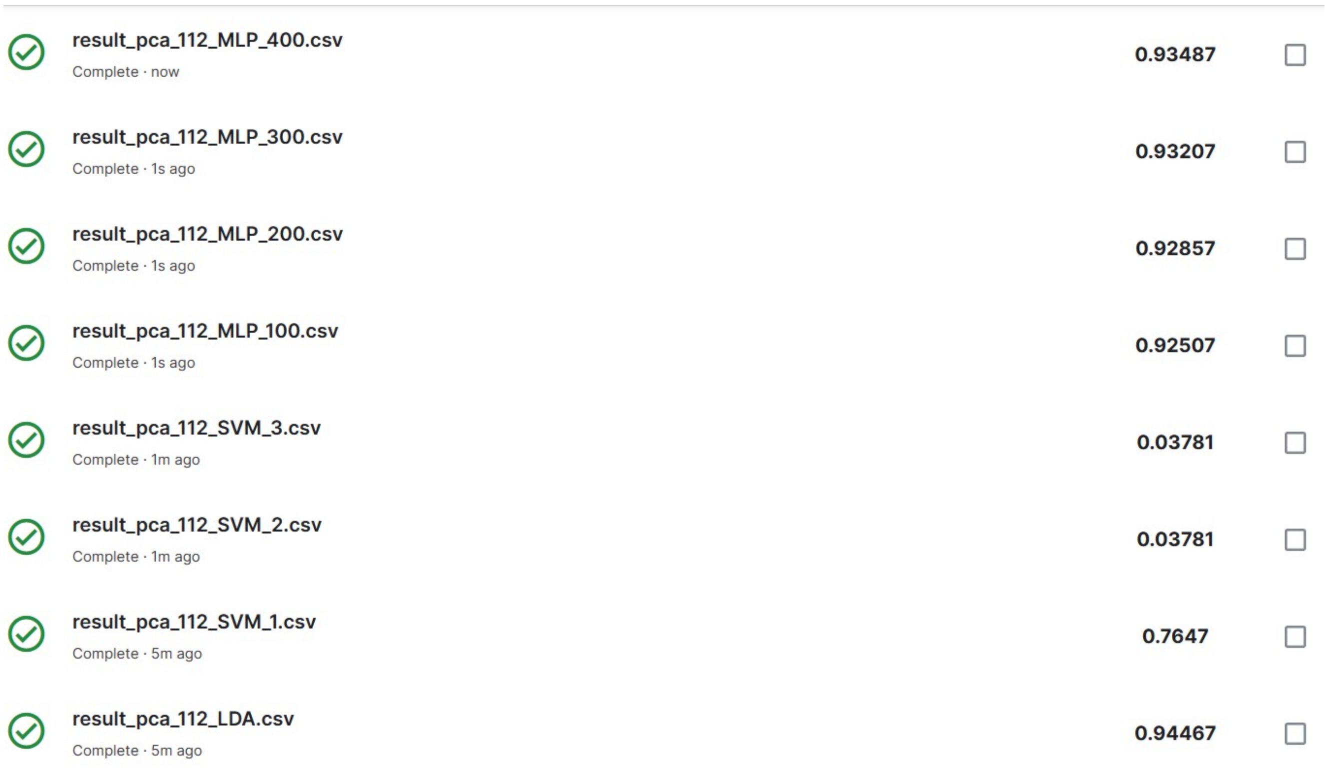


Figure 3. Cross validation of PCA under different noise variance.



result_pca_112_MLP_400.csv
Complete · now
0.93487

result_pca_112_MLP_300.csv
Complete · 1s ago
0.93207

result_pca_112_MLP_200.csv
Complete · 1s ago
0.92857

result_pca_112_MLP_100.csv
Complete · 1s ago
0.92507

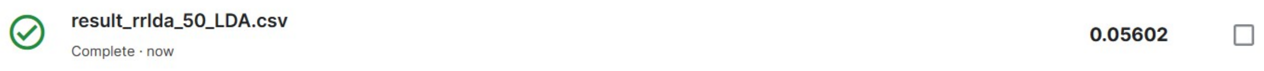
result_pca_112_SVM_3.csv
Complete · 1m ago
0.03781

result_pca_112_SVM_2.csv
Complete · 1m ago
0.03781

result_pca_112_SVM_1.csv
Complete · 5m ago
0.7647

result_pca_112_LDA.csv
Complete · 5m ago
0.94467

Figure 4. Accuracy result from Kaggle.



result_rrlda_50_LDA.csv
Complete · now
0.05602

Figure 5. Accuracy result of Reduced Rank LDA from Kaggle.