Project 2 - Dimensionality Reduction

Yuyang Zhang

Electrical and Computer Engineering, University of Florida

EEL 5934 Applied Machine Learning

Gainesville, U.S.

yuyangzhang@ufl.edu

*Abstract*— This article uses machine learning dimensionality reduction technology to reduce the dimensionality of image data. Applying dimensionality reduction techniques to high-dimensional data can save resources and running time, and maintain high performance while processing information quickly. By using these dimensionality reduction techniques on the labeled character image dataset and evaluating the model performance after dimensionality reduction, we can conclude that the dimensionality reduction techniques on the training set can help us process high-dimensional data more efficiently than without using dimensionality reduction techniques.

# Introdunction

In order to prevent the impact of high-dimensional data on the model, we use feature selection, feature extraction, supervised linear dimensionality reduction and manifold learning to reduce the dimensionality of the data. By visualizing the model dimensionality reduction step, we can analyze the weight of different dimensionality reduction techniques on the pixels of the image data, and further obtain the feature ranking in the data. In addition, based on the model performance, we can also find out the dimensionality reduction method suitable for the data set.

# Methodology

In this experiment, the dimensionality reduction methods we used include: Recursive Feature Elimination (RFE), Principal Component Analysis (PCA), Fisher Linear Discriminant Analysis (LDA), and Manifold Learning. Among them, manifold learning includes multidimensional scaling (MDS), isometric feature map (ISOMAP), locally linear embedding (LLE) and t-distributed stochastic neighbor embedding (t-SNE). In addition, some classic classifiers such as logistic regression and support vector are also used to classify data pixels and perform performance comparisons.

The training dataset is saved as a numpy array and contains a total of 6720 images from 10 classes. Each image is in grayscale and of size 300 \* 300. The 10 classes and its label encoding are:

Character: a | b | c | d | e | f | g | h | $ | #

Label: 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

# implementation steps

In this article, we need to complete the following tasks:

1. Implement RFE to select the subset of features
2. Implement PCA to select the number of components that explain at least 90% of the explained variance. Train a classifier on the original dataset and the reduced dataset.
3. Use Fisher's Linear Discriminant Analysis (LDA) and t-SNE to reduce the dataset to 2-dimensions and visualize it.
4. Implement at least 3 manifold learning algorithms for reducing the dimensionality of the feature space. Utilize the new lower-dimensional feature space to build a classifier.

# Experiment

Before starting the task, we need to reshape the image dataset because 300\*300 is too large to conduct the following experiments. By using OpenCV's resize function, we reshape all images to 50\*50. Then, we split data in 80% as training dataset and 20% as test dataset.

## RFE

We choose two estimators, one is logistic regression and the other is random forest.

1. *Logistic Regression*

We create a sklearn's 'LogisticRegression' with default, then put it into sklearn's 'RFE' as an estimator and fit the training data. By displaying the selected pixel indices in the model we can know which pixels were selected to contribute (selected pixel indices are displayed in test.ipynb). In addition, we can also display the original image and its mask examples for comparison. Some examples are as follows:

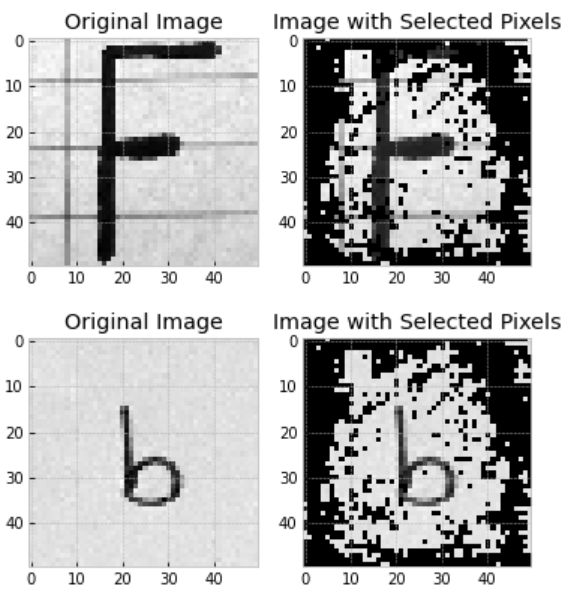
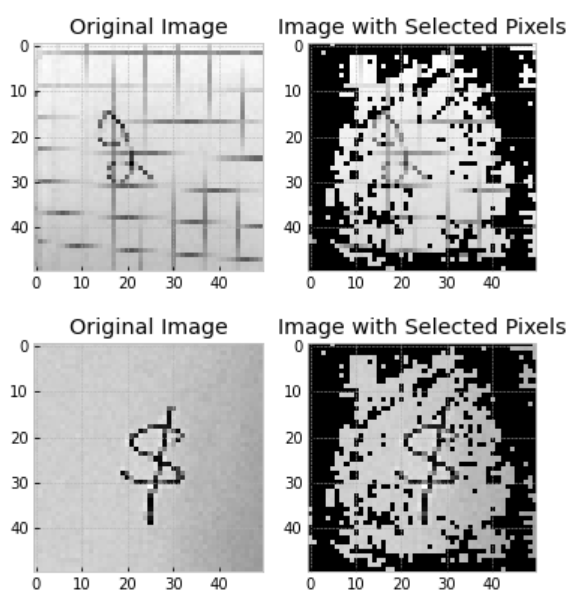
 

Fig. 1 Samples of the original image and masked image from RFE with logistic regression

RFE using logistic regression as estimator has about 39% accuracy (the rest of the performance metrics can be found in test.ipynb). By comparison, we can find that RFE circles the character in the middle of the image and masks the remaining unimportant background parts.

1. *Random Forest*

Similarly, we create sklearn’s ‘RandomForestClassifier’ with default and apply it as an estimator on RFE, then let it fit the training set. We also show its selected pixel indices (displayed in test.ipynb) and some examples to compare the original image and the masked image.

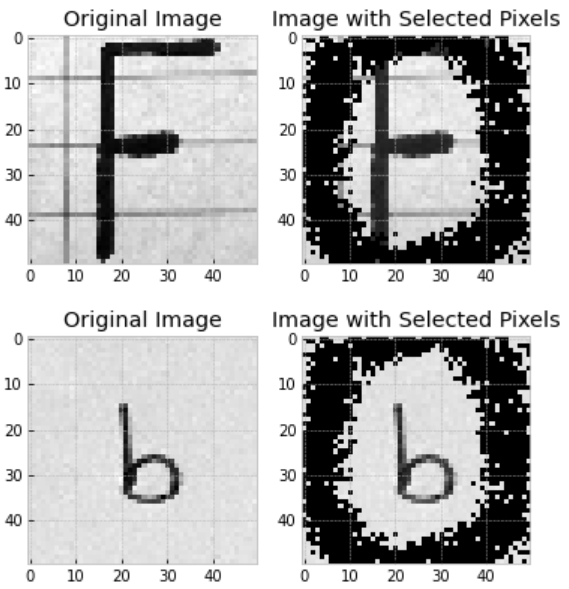
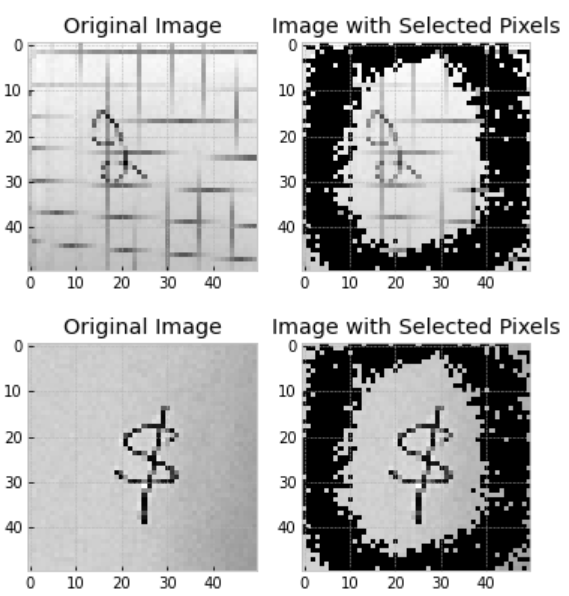
 

Fig. 2 Samples of the original image and masked image from RFE with random forest

RFE using random forest as estimator has about 42% accuracy (the rest of the performance metrics can be found in test.ipynb). By comparison, we can find that RFE circles the characters in the middle of the image and masks the remaining unimportant background parts. Compared with the RFE whose estimator is logistic regression, the RFE whose estimator is random forest masks more background points and tends to retain more central parts. This may be one of the reasons for its higher accuracy.

## PCA

In order to use PCA to select the number of principal components that explain at least 90% of the explained variance, we need to first create a PCA that uses all principal components and then look for the number of principal components equal to 0.9 explained variance. Since we resized the image to 50\*50 before, we use 2500 as principal components on this model. By showing principal components numbers at 0.9 explained variance and the plot of Cumulative Explained Variance Ratio vs. Number of Principal Components, we find that 161 can satisfy 0.9 explained variance.

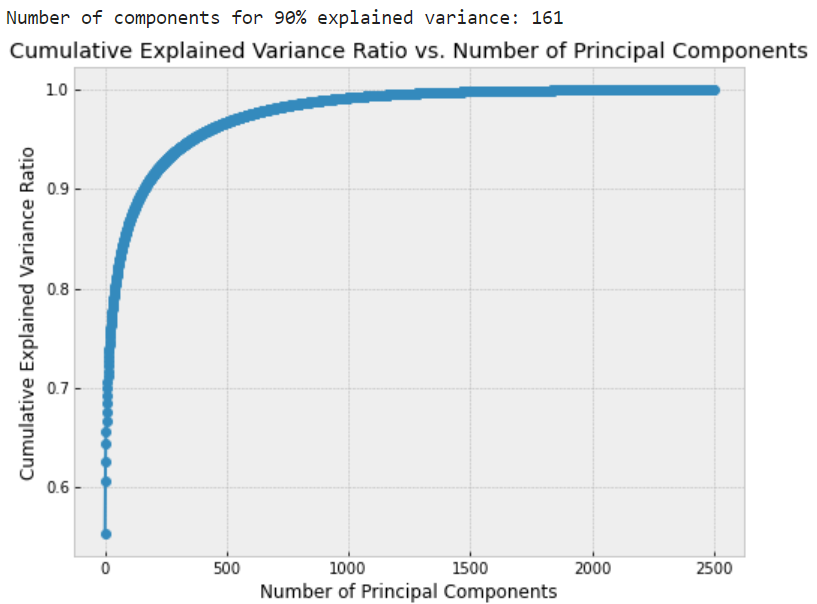


Fig. 3 Cumulative Explained Variance Ratio vs. Number of Principal Components Plot for PCA

Next, we set the principal components to 161 and recreate a PCA model, which can select 161 components to reduce the dimensionality of the image and reconstruct the image. Here are the original images and their reconstructed images:

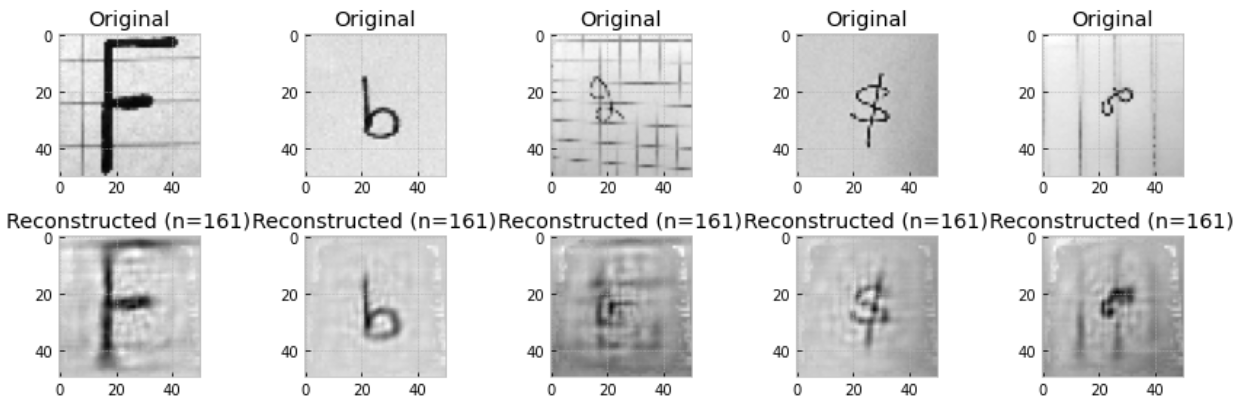


Fig. 4 Samples of the original image and reconstructed image from PCA with 90% explained variance

We also want to analyze the significance represented by the most important eigenvectors revealed by PCA of the data. So we display them:

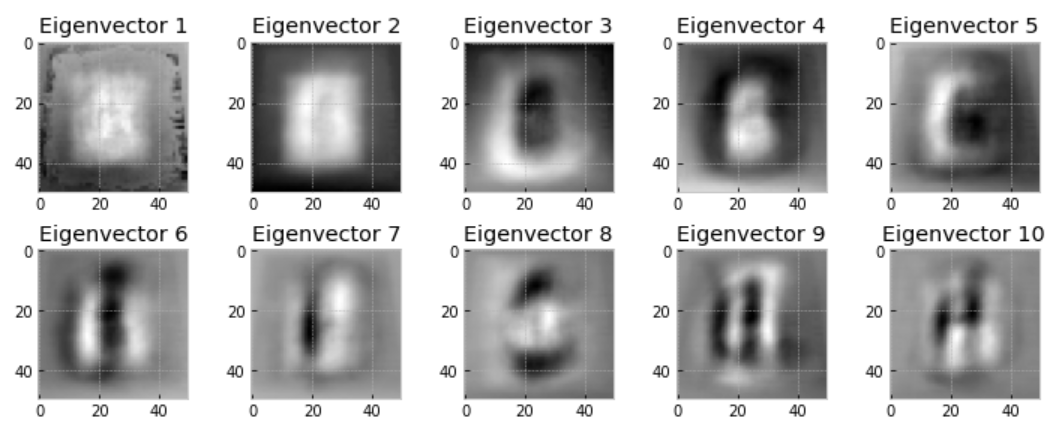


Fig. 5 Top 10 PCA eigenvectors

Eigenvector 1 expresses that the middle position of the data image and a part of the background containing a certain amount of information will be used as features. In Eigenvector 2, it can be found by observation that it only observes whether there are characters in the middle of the image as a feature. Both Eigenvector 3 and 5 focus more on a circle around the middle of the image, which may represent some characters with a hollow center. Eigenvector 4 represents lowercase characters or characters of small size. Eigenvector 6, 7, 9, and 10 all display a horizontal 'E' form. This may be because many characters have vertical lines (b, d, g, h, #) on the left and right of the image. Eigenvector 8 shows an '8' form, which can interpret certain characters such as 'B', 'E', '$'.

Then, we need to know what benefits this PCA model can bring to our classification task. We create two pipelines, one pipeline consists of PCA model, minmax scaler and default logistic regression classifier; the other only has minmax scaler and default logistic regression classifier. By fitting them to the training set, we obtained two classification models, one of which applied PCA that achieved 90% explained variance and the other which did not. We also observe and record their respective running times. Based on the data we found, based on multiple run time records we found that model with PCA costs less than 1 minute to complete fitting. But the model without PCA costs 9 minutes on fitting. This indicates that the model with PCA has extremely faster fitting time than the model without PCA.

Next, let the two models predict using the test set, and output their accuracy and confusion metrics to express performance. According to the data, we found that the model using PCA has about 3% higher accuracy than the model without PCA. This shows that after using PCA to reduce the dimensionality of the training data set, the model pays more attention to important features with more information. This enables the model to gain more generalized predictive capabilities.

## LDA and t-SNE

We need to create an LDA model and a t-SNE model to convert the dataset to 2-dimensions. When creating the pipeline of the LDA model, we create a sklearn's 'LinearDiscriminantAnalysis' and set its hyperparameter components to 2, because we want to convert data to 2-dimensions. Then we also need a minmax scaler to join the pipeline. When creating the pipeline of the t-SNE model, we create 'TSNE' of sklearn and set its hyperparameter components to 2, learning\_rate to auto, and init to 'random'. This is because the performance of t-SNE is more easily affected by its hyperparameters, so we set some of its hyperparameters to ensure its performance.

After the pipelines are completed, we let the model classify with the test set and display the model's plot with color-code each point to its corresponding target label. In the plot, each color represents a feature.

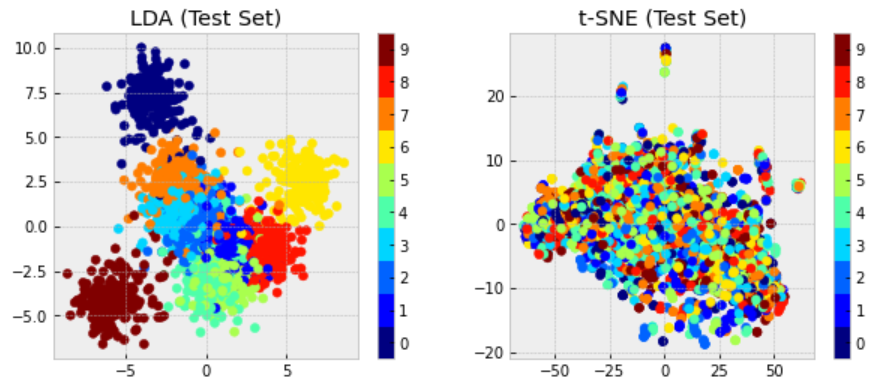


Fig. 6 LDA and t-SNE 2-dimensions dataset

Now, we want to know how many features to select for this dataset. Since plots of different colors of t-SNE overlap together and t-SNE does not support more components, we choose based on the information provided by LDA. According to the LDA dataset plot, we can find that the data points are roughly divided into 8 colors, which implies that the best number of features is 8. However, we have a better way to determine its number.

We create a new LDA and set its components to the maximum value of 9, use it to classify the test set. As in the PCA task, we show its Cumulative Explained Variance Ratio vs. Number of Principal Components Plot again. Through plot, we can find that 8 components can achieve the highest ratio. So for this dataset, we select the 8 features that are the best.

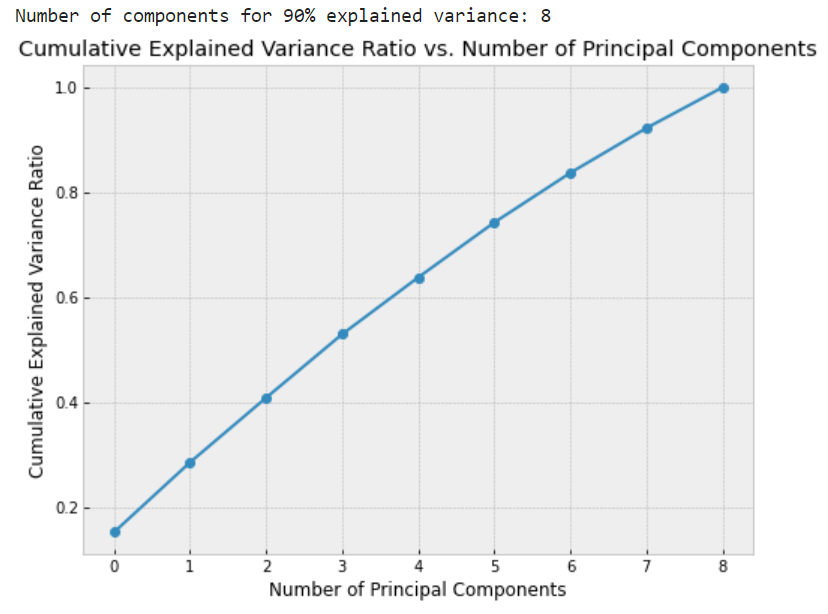


Fig. 7 Cumulative Explained Variance Ratio vs. Number of Principal Components Plot for LDA

We also hope to explore the principles and appropriate application scenarios of these three methods by comparing the relationships and differences between the two-dimensional datasets of LDA, t-SNE and PCA. The LDA plot can distinguish each feature relatively clearly. Because LDA often results in better class separation compared to PCA, making it suitable for classification tasks. Neither t-SNE's plot nor PCA's plot can intuitively distinguish features. This is because t-SNE focuses on preserving local similarities between data points. It is well-suited for visualizing clusters and capturing complex, nonlinear structures in the data. While PCA is unsupervised, it is useful for retaining overall variance in data. It may be preferred when we are more interested in data compression or noise reduction and less concerned with class separability or cluster preservation.

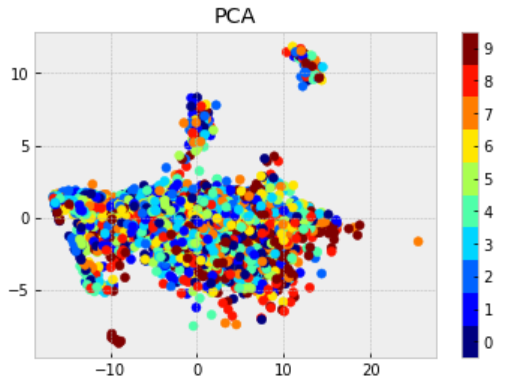


Fig. 8 PCA 2-dimensions dataset

## Manifold learning

In this task, we hope to explore the dimensionality reduction performance of different manifold learning on data sets and visualize their two-dimensional projections. We choose MDS, ISOMAP and LLE as the experimental models because they are all techniques that do not assume the manifold is linear. For convenience, we uniformly set the components of each model to 60. We choose the Support vector classifier as the applied classifier.

Next, apply these three techniques to the training set and save the resulting transformed training set. Create three identical SVC pipelines, each containing a default SVC and minmax scaler. Then use grid search to find the hyperparameters C and gamma of the SVC model. The grid search in this step requires fir transformed training set. After completing the grid search on all three models, we refit the model with the best hyperparameters.

Similarly, we use MDS, ISOMAP and LLE to obtain the reduced dataset of the test set respectively. And use the three SVC models with the best parameters just created to predict using the transformed test set. Compare the performance of the three models by outputting accuracy and Classification Report (see test.ipynb). We found that the MDS model has the highest accuracy (0.1125), followed by LLE (0.1038), while ISOMAP has the lowest accuracy (0.0889). The main reason for the low accuracy is that we did not perform unique optimization for each model, but the three dimensionality reduction techniques were all tested under the same non-optimization conditions. Through accuracy, we selected MDS as the most suitable manifold learning technique for this data.

By visualizing the first 2 dimensions in those 3 manifold learning algorithms, we expect to interpret the features according to which the dimensionality reduction algorithm distinguishes data points. We first show the reduced feature space of MDS, ISOMAP and LLE:

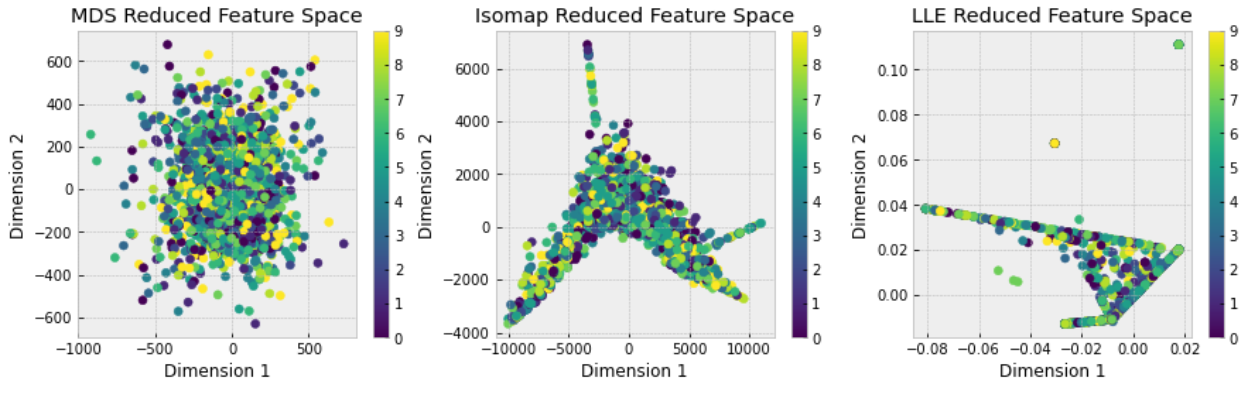


Fig. 9 Reduced feature space for MDS, ISOMAP, LLE

Then, we display the image data of random data points in the reduced feature space so that we can interpret them:

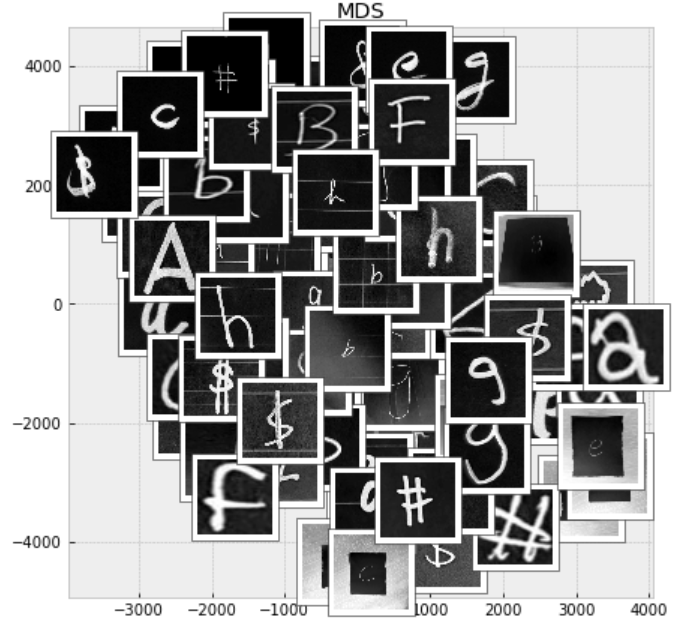
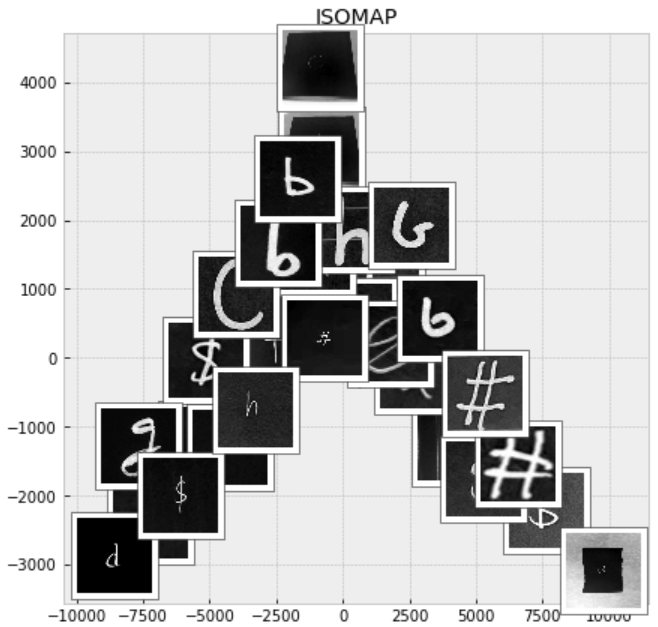
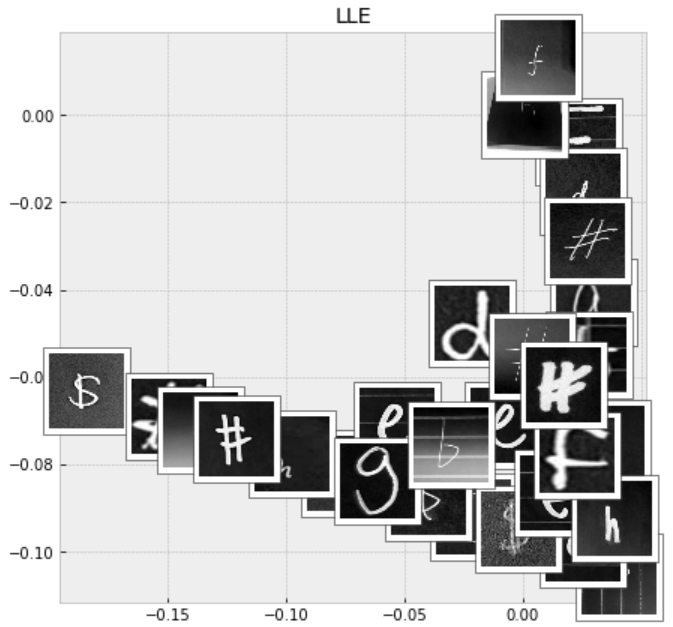
  

Fig. 10 Random images in 2-dimension feature space for MDS, ISOMAP, LLE

From the MDS plot, we can find that it is in the shape of a circle or ellipse as a whole. It places the characters with smaller images at the center of the circle, and the characters with larger characters will be placed further away from the center of the circle. This may also depend on the line thickness of the characters.

The plot of ISOMAP is centered on (0, 0) and extends directly above, lower left and lower right of the plot, forming an upward arrow shape. Its lower left and lower right branches all follow that the farther away from (0, -3000), the larger the characters and the thicker the font; while the closer to (0, -3000), the smaller the characters and the thinner the font.

The LLE plot is in the shape of a triangle and has two extended "lines", forming an arrow shape toward the lower right corner. The characters closer to the upper right corner of the picture will be skewed to the left, while the characters closer to the lower left corner will be skewed to the right.

# conclusion

We can find that by using feature selection technique RFE and feature extraction technique PCA when training the model, we can reduce its dimension while ensuring the important features of the data set. This can help us train models more efficiently and save resources. And when the number of features is reduced, models perform better because they focus on the most informative dimensions. This gives the model better generalization and prediction capabilities. We all know that visualizing and interpreting high-dimensional data is challenging. Dimensionality reduction techniques project data into a low-dimensional space to help identify patterns, clusters, and outliers that may be hidden in the original high-dimensional space. By visualizing dimensionality reduction techniques such as PCA, LDA and manifest learning, we can interpret some of the criteria for feature selection by these techniques and help us decide to use the most appropriate dimensionality reduction technique and its hyperparameter adjustment.

##### References