Evaluating Dimensionality Reduction Techniques

Diana Sanchez

Undergraduate Student

University of Florida

***Abstract*—This report describes and evaluates the process used to implement dimensionality reduction and manifold learning techniques for visualization and subsequent classification tasks on a custom handwritten character dataset. The observations made during training and performance evaluation are outlined to determine which techniques produce the most optimal results in terms of computational time and performance.**

I. INTRODUCTION

Dimensionality reduction is often used to work with datasets that have high-dimensional data spaces. When working in large feature spaces, more data is needed to explain the relationships between the data. If the dimensions of the data are not reduced, the data under different algorithms may behave unusually. To prevent this, feature extraction techniques such as Principal Component Analysis (PCA) and Manifold Learning techniques, like Multi-Dimensional Scaling (MDS), Isometric Mapping (ISOMAP), and Locally Linear Embedding (LLE), could be used. The goal of this project is to reduce the dimensions feature selection of the custom handwritten character dataset using Recursive Feature Elimination (RFE), PCA, and different manifold learning techniques. The custom handwritten data set consists of multiple images with 10 different samples of the following characters: [‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’, ‘h’, ‘$’, ‘#’]. After performing dimensionality reduction, estimators and classifiers are used to classify the data and make predictions. Please note that although the images are 300x300, they were down-sampled to 50x50 for quicker computational time.

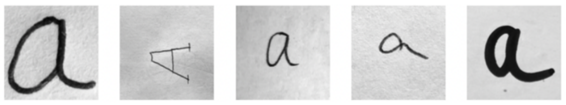

Fig.1 Examples of samples for class ‘a’ in the dataset

II. PROBLEM 1: RECURSIVE FEATURE ELIMINATION

A. *Implementing RFE with Logistic Regression as an estimator to select a subset of features*

One of the first tasks done is implementing recursive feature elimination (RFE) with Logistic Regression as an estimator. This was done by creating a pipeline where logistic regression is the classifier and is passed into the RFE function as an estimator. Recursive feature elimination is a classical sequential algorithm that aims to reduce the dimensionality of the initial feature space to improve computational efficiency. RFE removes features, which in our case are pixel locations, from the full feature subset. To determine which features are removed, RFE uses the estimator (Logistic Regression) to differentiate the performance of the classifier before and after the removal of a specific feature.

After applying the pipeline to the dataset, 1250 features (pixels) were kept out of 2500 features. To display the areas where the features were selected, I applied a binary mask to the dataset and displayed the areas that were selected or not selected. In Figure 2, you can see that the white areas are the pixels that were selected, and the black areas are pixels that were disregarded.

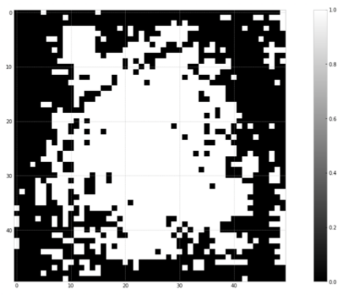

Fig.2 Binary Mask for RFE with Logistic Regression model

As we can see, the pixels in the center of the image were selected as important features, whereas the pixels near the border were not. This makes sense because, in most of the images, the characters are somewhat centered (as seen in Figure 1).

To get a better understanding of the importance of each pixel, I displayed the weight of importance of each pixel by reshaping the “ranking_” feature of RFE that returns the rank of each feature in terms of importance, where 1 is marked as most important.

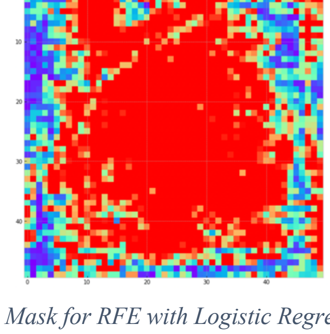

Fig.3 Ranking Mask for RFE with Logistic Regression model

In Figure 3, red is associated with a ranking of 1, and purple is associated with the lowest ranking. As we can see, the red areas are all selected as features when compared to figure 2, which makes sense since those features are marked with more importance. As the pixels move away from the center of the image, the ranking starts to decrease, which correlates with the results from the binary mask since the features around the border were disregarded.

B. *Implementing RFE with a Decision Tree as an estimator to select a subset of features*

I applied a Decision Tree Classifier to RFE as an estimator to compare the results to RFE with Logistic Regression. To do this, I made a pipeline with the decision tree as the classifier, RFE with the decision tree as the estimator, and scaled the data using Standard Scaler. When the pipeline was applied to the training set, 1250 features were selected out of 2500, which is the same amount as RFE with Logistic Regression.

To display the selected features, I applied the binary mask to the supporting features of the original dataset (as shown in Figure 4).

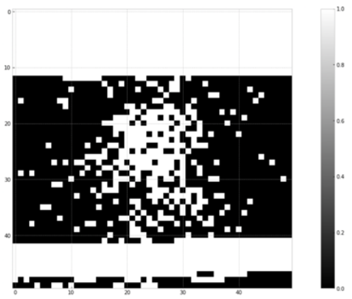

Fig.4 Binary Mask for RFE with a Decision Tree model

As we can see, the selected pixels are vastly different from the first RFE model, where most of the selected pixels were in the center. For this RFE model with a decision tree as the estimator, a lot of the upper pixels were selected as features to keep, and the pixels selected in the middle are sparser. To understand the data more, Figure 5 below displays the rank distribution of each pixel.

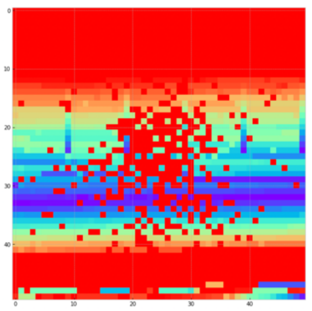

Fig 5. Ranking Mask for RFE with a Decision Tree model

One of the reasons for the unusual distribution of the importance of features is the estimator being a decision tree. The decision tree classifier used in my model had a maximum depth of 40, meaning it is possible that not all the pixels got classified into their group since the data is large.

C. *Evaluating and Comparing Performances of both RFE models*

After building both models, they were evaluated based on their accuracy, precision, and recall scores. The summary of the performance of both models is shown in Table 1.

	Model 1 Training	Model 1 Test	Model 2 Training	Model 2 Test
Accuracy	50.41%	40.76%	58%	22%
Precision	0.51	0.42	0.59	0.23
Recall	0.50	0.42	0.58	0.23

Table 1. Performance Scores where Model 1 is RFE with Logistic Regression and Model 2 is RFE with Decision Tree

The RFE with Logistic Regression model performed better in the test set compared to the RFE with Decision Tree model, whereas the RFE with Decision Tree did better in training. This is an indication that the Decision Tree model was overfitted since it performed better with the training set and has low accuracy for the test set. Based on these results, using RFE with Logistic Regression as the estimator is the most optimal out of the two models experimented with.

III. Problem 2: Principal Component Analysis

In this section, our task was to implement principal component analysis (PCA) to select the number of components that explain at least 90% of the explained variance and then train a classifier on the original dataset (without PCA) and reduced dataset (with PCA) and compare performances.

A. *Implementing Principal Component Analysis*

Principal component analysis (PCA) takes data from sensor coordinates to data-centric coordinates using linear transformation to perform dimensionality reduction. This technique finds the directions of maximum variance in high-dimensional data and projects it onto a new subspace with equal or fewer dimensions. In order to find how many components are needed to explain at least 90% of the explained variance, I applied principal component analysis on the dataset using 2500 n_components, which is the number of components to keep, because that is the number of pixels of each image. After applying the PCA model to the training set, I was able to find that 185 components are needed to explain at least 90% of the explained variance in the data using the PCA-explained variance ratio feature. This value is used to create a reduced dataset using PCA in the next section.

The next thing I did was visualize the top 10 eigenvectors where each one represents a different amount of variance. The First eigenvector has the most variance and is the first principal component. Eigenvectors describe the regions that explain the most variance.

Looking at the first eigenvector in Figure 6, we can see that the regions near the center of the image carry the most variance, which is where most of the characters lie in the image, but it considers the outer regions for any outliers. The second eigenvector is similar to the first, but it is more condensed, which could accidentally consider pixels in the center that are just empty spaces in the characters.

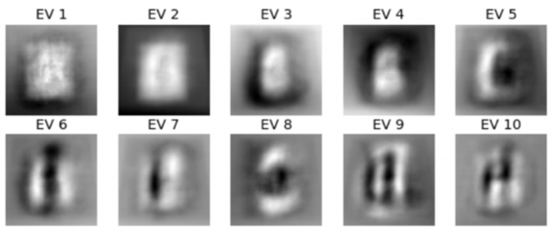

Fig.6 Top 10 eigenvectors for PCA model

To test the effectiveness of the principal component analysis and the selected number of components, we reconstructed some images from the PCA projections using the PCA model with 185 selected components. The results are shown in Figure 7.

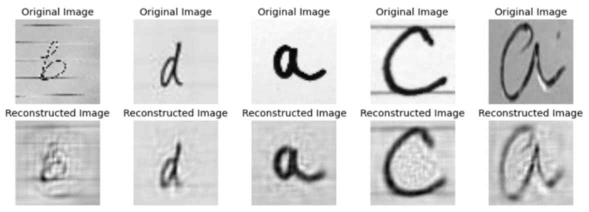

Fig.7 Image reconstruction from PCA projections.

To evaluate the performance, I compared the reconstructed image to the original image. It can be seen that there is some compression loss in the reconstructed images, but they are fairly similar to the original images. If the number of components was reduced, the compression loss would be greater because to get a 90% explained variance, at least 185 components are needed, as seen from our performance.

B. *Training classifier on original dataset and reduced dataset Performance Evaluation*

In this task, I trained an SVM classifier on the original dataset (without PCA) and a reduced dataset with PCA. The performance of both models is shown in Table 2, where model 1 is the original dataset and model 2 is the reduced dataset model. It can be seen that the original dataset without PCA took longer to make predictions and has a slightly lower accuracy score than the reduced dataset with PCA during testing. The precision is also higher for the reduced dataset with PCA.

	Model 1 Training	**Model 1 Test**	**Model 2 Training**	**Model 2 Test**
Accuracy	62.02%	42.47%	55.67%	43.85%
Precision	0.64	0.45	0.58	0.46
Recall	0.62	0.45	0.56	0.44
Time	43 sec	19 sec	7.9 sec	3.5 sec

Table 2. Performance Scores where Model 1 is the original dataset and Model 2 is the reduced dataset with PCA

Based on these results, I would choose the model with a reduced dataset using PCA because it is less computationally expensive, faster, and performs better during testing. Also, the original dataset seems to be overfitting since the accuracy is significantly lower during testing compared to the training results.

IV. PROBLEM 4: MANIFOLD LEARNING ALGORITHMS

In this section, our goal was to implement 3 different manifold learning algorithms for reducing the dimensionality of the feature space and then use these new lower-dimensional feature spaces to build a classifier. The three-manifold algorithms I implemented are Isometric Mapping (ISOMAP), Locally Linear Embedding (LLE), and Multi-Dimensional Scaling (MDS). Note: I attempted to do a Grid Search CV for all models in this problem, but the notebook kept breaking when computing, so the results may not perfectly represent the optimal results.

A. *Reducing the dimensionality of feature space using ISOMAP*

To perform cross-validation, I created 3 different ISOMAP models, each with a different number of coordinates for the manifold (2, 8, and 1250 respectively). I then built the classifier using all three lower-dimensional spaces and evaluated the accuracy score. The first ISOMAP with a reduced dimension of 2 had an accuracy score of 11.26%. The second ISOMAP with a reduced dimension of 8 had an accuracy score of 13.45%. Finally, the third ISOMAP with a reduced dimension of 1250 had an accuracy score of 31%. Based on these results, the larger number of coordinates used for the ISOMAP manifold, the better it performs when applied to a classifier, so the ISOMAP with 1250 was kept out of the three. The classifier used was a Logistic Regression classifier.

To visualize the behavior of the manifold learning algorithm, I visualized the first 2 dimensions of ISOMAP with 1250 dimensions, as shown in Figure 8. Looking at the first two dimensions, the data is a bit sparse and not perfectly unfolded, but we can see the relationship between points. Along the x-axis, the characters range in size and orientation. As we move further to the right, the characters are getting larger and turned at a 90-degree angle. Also, the lines on the paper are more apparent. Along the y-axis, the characters change orientation from top to bottom, and the thickness of the characters increases as you move downwards. This shows that the orientation, background, and thickness of the characters are what carry the most variance for ISOMAP and are most likely considered the top eigenvectors.

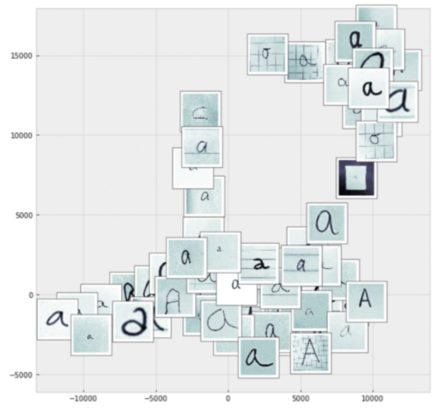

Fig.8 Visualization of the 1st two dimensions of ISOMAP

B. *Reducing dimensionality of feature space using LLE*

To perform cross-validation, I followed the same process used for ISOMAP where I created 3 different LLE models, each with a different number of coordinates for the manifold (2, 10, and 1250 respectively). After I built the Logistic Regression classifier using all three models separately, I was able to evaluate the results of each and pick the most optimal number of coordinates, which ended up being LLE with 1250 components, having an accuracy score of 13.69%.

To visualize the behavior of the manifold learning algorithm, I visualized the first 2 dimensions of LLE with 1250 dimensions, as shown in Figure 9.

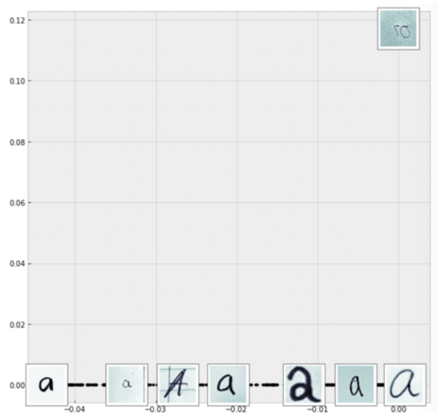

Fig.9 Visualization of the 1st two dimensions of LLE

Looking at the first two dimensions of LLE, the points are significantly sparser than ISOMAP. This is partially because LLE preserves the local intrinsic relationships between points rather than the global relationships. One of the main outliers seen is an 'a' character turned at a 90-degree angle, which makes sense since it wouldn't be a local neighbor to the other characters. Moving down the y-axis, the characters go from a turned 90-degree angle to a normal orientation, which means the orientation is an eigenvector that carries a lot of variance for LLE. The characters also seem to increase in thickness as you approach the middle of the x-axis, meaning the thickness also carries high variance for LLE.

C. *Reducing dimensionality of feature space using MDS*

In this section, I reduced the dimensionality of the feature space using MDS with 2 coordinates for the manifold, which might affect the effectiveness of the model since it is smaller than the dimensions used for ISOMAP and LLE. When I built the logistic regression classifier using the new 2-dimensional space, the accuracy score for the classifier was 10.71%, which is lower compared to ISOMAP and LLE. This is because of the smaller number of dimensional spaces used by MDS.

To visualize the behavior of the manifold learning algorithm, I visualized the first 2 dimensions of MDS with 2 dimensions, as shown in Figure 10. When looking at the first two dimensions, MDS seems to unfold the data more than the other two manifold learning algorithms. As you move downwards along the y-axis, the images range from dark to light, meaning the lightness of an image carries high variance and is an eigenvector for MDS. Moving downwards along the y-axis, the thickness of the characters decreases. Towards the middle of the x-axis, the characters seem to get smaller in size and there are more apparent lines on the paper. Overall, from this visualization, it is apparent that the lightness, thickness of characters, background, and size of characters are features that carry the most variance.

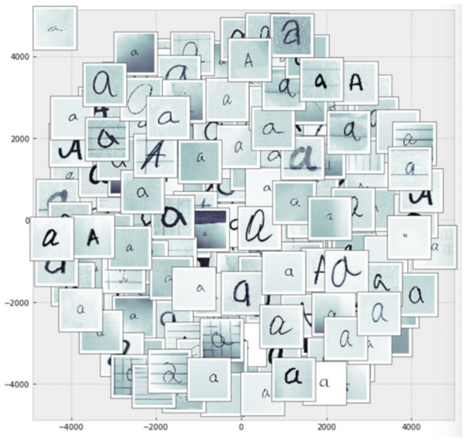

Fig.10 Visualization of the 1st two dimensions of MDS

D. *Manifold Learning Algorithms Performance Evaluation*

To evaluate and compare the three-manifold algorithms used in this section, I had to apply the dimensionality reduction algorithms to the test set, export the reduced test sets to the test notebook, and applied the classifier pipeline to the new lower dimensional space to make predictions. The performance is shown in Table 3.

	ISOMAP	LLE	MDS
Accuracy (prediction)	9.58%	10.83%	9.76%
Precision	0.10	0.11	0.07
Recall	0.09	0.11	0.10
Accuracy (transformed)	50.76%	21.67%	11.11%

Table 3. Performance Scores for manifold learning algorithms.

Based on the results, I would select ISOMAP because it performed the best out of the three algorithms since the accuracy score after the test set was transformed is the highest, and its accuracy during predictions is not that different from MDS. Also, it was faster to apply ISOMAP during training and testing.

References

[1] "Powerful feature selection with recursive feature elimination (RFE) of ..." [Online]. Available: https://towardsdatascience.com/powerful-feature-selection-with-recursive-feature-elimination-rfe-of-sklearn-23efb2cdb54e. [Accessed: 15-Nov-2022].

[2] K. Kelley, "Recursive feature elimination (RFE) in python: Simplilearn," *Simplilearn.com*, 17-Oct-2022. [Online]. Available: https://www.simplilearn.com/recursive-feature-elimination-article. [Accessed: 14-Nov-2022].

[3] "A step-by-step explanation of principal component analysis (PCA)," *Built In*. [Online]. Available: https://builtin.com/data-science/step-step-explanation-principal-component-analysis. [Accessed: 14-Nov-2022].

[4] "Manifold learning [T-sne, lle, isomap, +] made easy." [Online]. Available: https://towardsdatascience.com/manifold-learning-t-sne-lle-isomap-made-easy-42cfd61f5183. [Accessed: 15-Nov-2022]