Four or Nine? Handwritten Digit Recognition with High Dimensional Analysis

Kuiyu Zhu, Zhiyu Lin, Tianxing Jiang

kz175@georgetown.edu, zl281@georgetown.edu, tj247@georgetown.edu

INTRODUCTION

Digit recognition is a subcategory of handwriting recognition, which can be applied to a variety of areas such as detecting objects from pictures, powering touch screens, guiding self-driving cars, and labeling image contents with meta tags.

Investigating data of two easily confusable numbers, 4 and 9, is a vital step to test the capability of a image recognition model. Current state-of-the-art methods on image recognition usually utilize deep learning models to capture the intricate relationships between image pixels. A variety of studies and methods have been tested out these digit images with very high accuracy. In this study, we will include some of the best performing models, but focus on providing a dimensionality reduction framework for handwritten digit recognition.

In this study, we visualized a low dimensional embedding of the image data with t-SNE and UMAP. We also trained a variety of classification models and feature selection methods on 6000 handwritten digits and achieved over 97% accuracy. We will in the sections below explain how our dataset is structured, what methods we employed, how the visualizations are produced, and how each prediction model performed with different feature subsets.

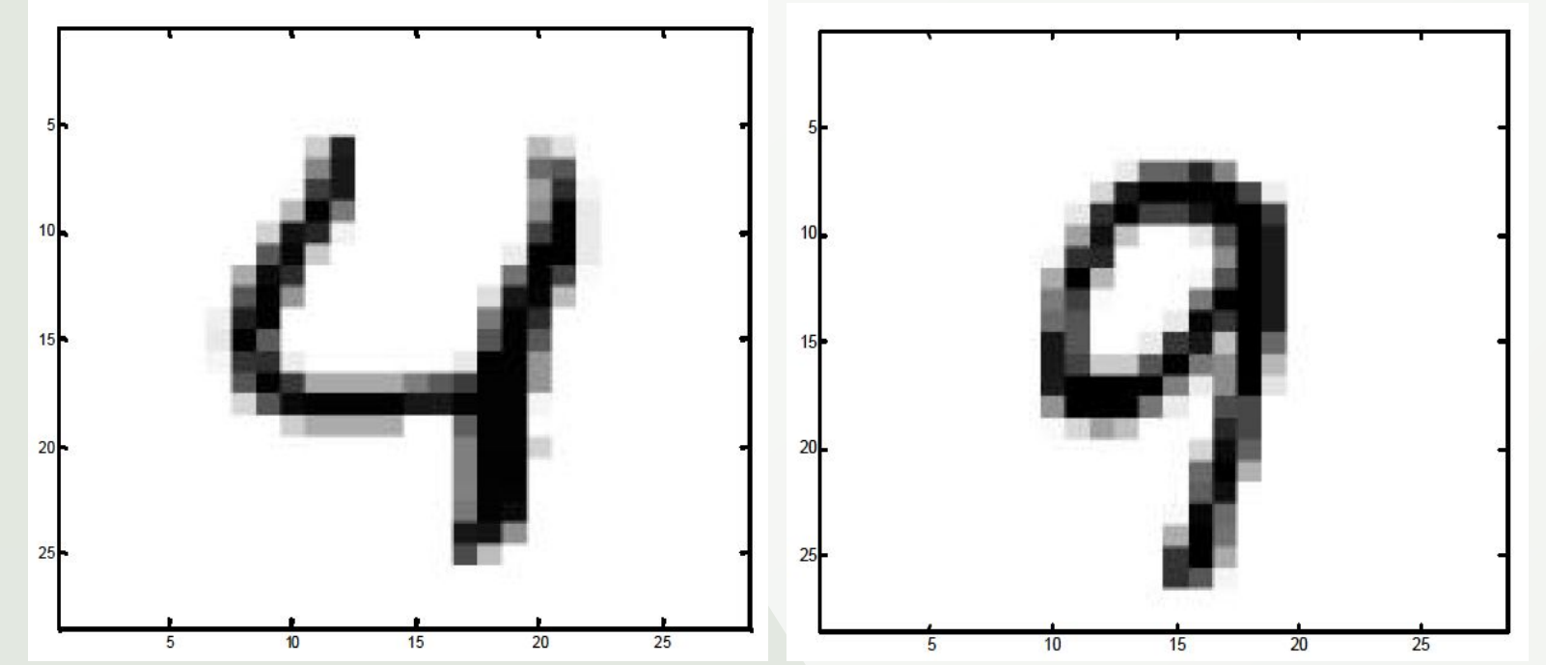


Figure 1. Number 4 and 9 in handwriting taken from the MNIST dataset.

DATA & METHODS

GISETTE is a UCI machine learning dataset constructed from MNIST, which made available by Yann LeCun of the NEC Research Institute in 2003 for a feature selection challenge. MNIST contains 60,000 handwritten digits flattened into 28*28 pixels, among which Gisette subsetted number 4 and number 9, and performed additional feature engineering steps. For example, Gisette normalized the pixel values, added products of pairs of variables, eliminated zero features, and included 2500 additional noise features.

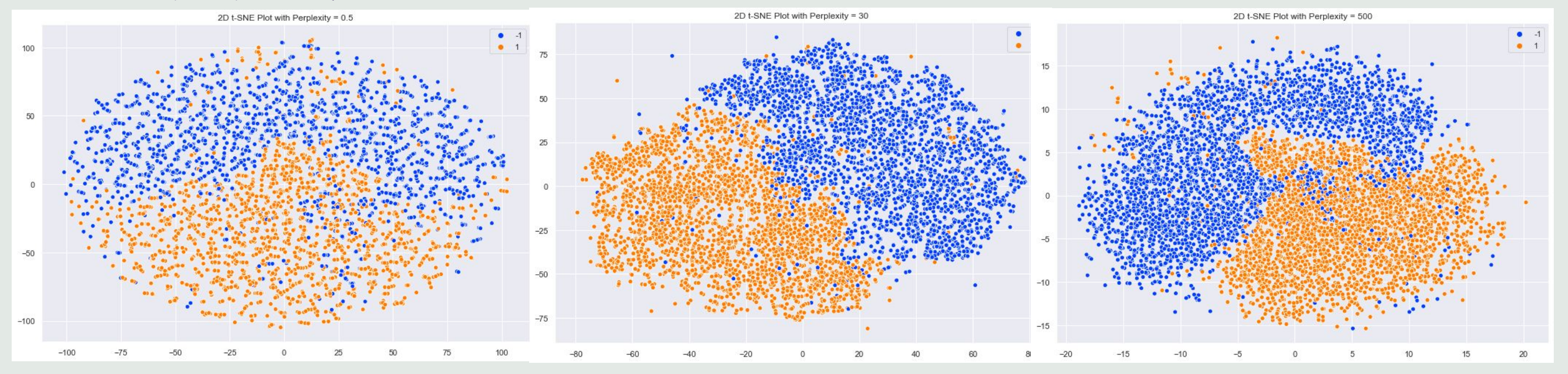
Specifically, the data was can be understood in the following way:

1. Each row can be viewed as a picture of a handwritten digit. Each picture contains 28 * 28 pixels, which are flattened into one dimensional vectors.
2. Each column is a product of some randomly selected pixels in the picture.
3. The label is 0 and 1, representing 4 and 9 respectively.

Overall, this training set has 6000 digits and 5000 features, and the validation set contains another 1000 digits. The high-dimensionality nature of this dataset brought many challenges such as difficulty calculating Euclidean distances, which then dictated some of the methods we employed for the analysis.

VISUALIZATIONS

t-SNE (t-Distributed Stochastic Neighbor Embedding) is a nonlinear dimension reduction technique used to represent high-dimensional data. The core idea is to ensure that the distribution of data in low-dimension is similar to the distribution of the original feature space. In TSNE function of sklearn.manifold library, there is an important parameter, perplexity, which "is related to the number of nearest neighbors that is used in manifold learning algorithms" according to scikit-learn documentation. The figures below are displaying t-SNE embedding in two dimensions with different levels of perplexity.



Perplexity = 0.5, ET: 264 sec   Perplexity = 30, ET: 266 sec   Perplexity = 500, ET: 335 sec

Figure 2. t-SNE 2-dimensional plots with different values of perplexity

We observe that the values of perplexity do make a big difference on the result. While low perplexity corresponds to local perspective (smaller number of neighbors to consider), and high perplexity corresponds to a wider perspective (more neighbors to draw distances to). In addition, t-SNE technique is computationally expensive, and since the cost function is not convex, the outcome plot is produced randomly. So it may take us to run the script multiple times to get the best results.

UMAP (Uniform Manifold Approximation and Projection) is another manifold learning technique for dimension reduction. The plot below shows the two dimensional embedding of UMAP on the Gisette dataset.

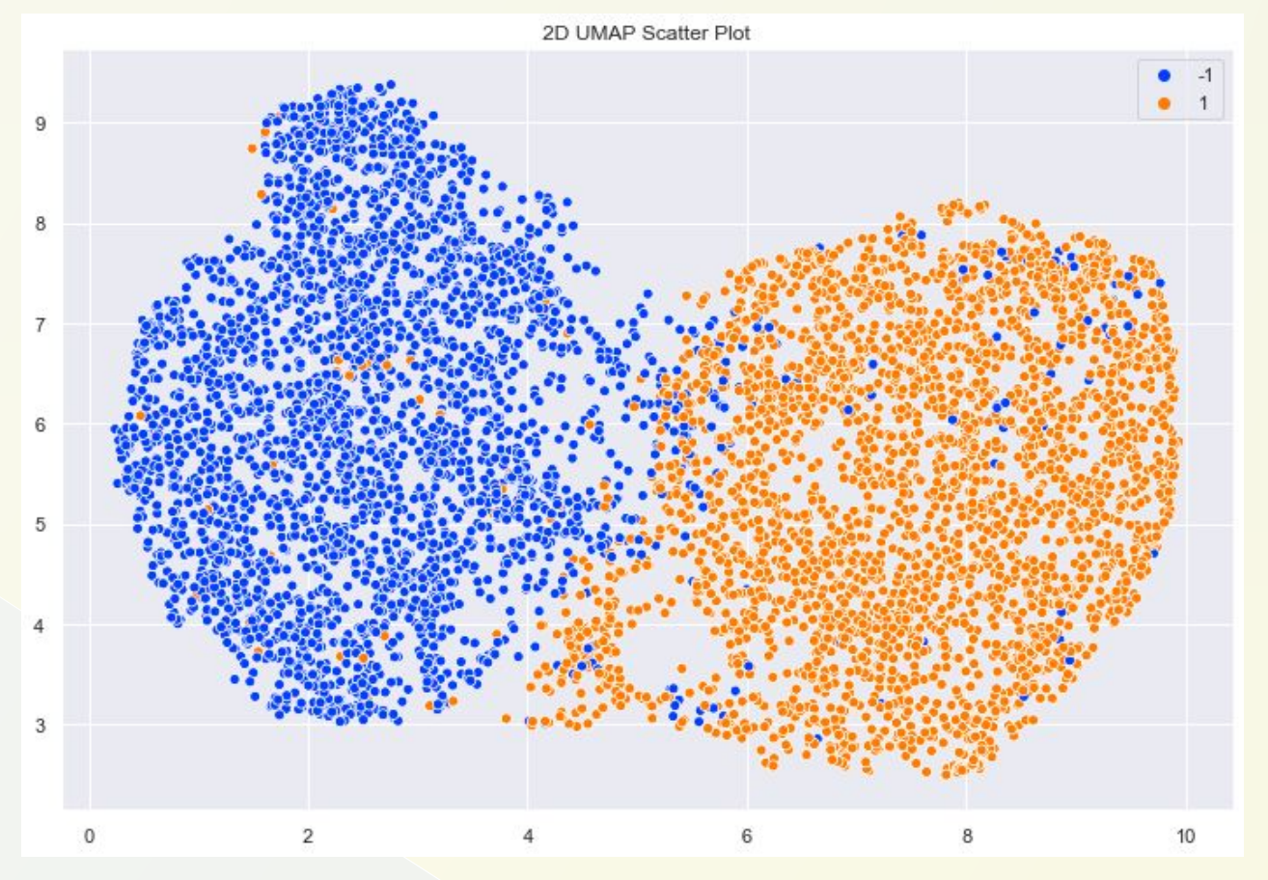


Figure 3. UMAP 2-dimensional plots

While both algorithms exhibit strong local clusters and successfully grouped almost all "4"s together and almost all "9"s together, UMAP produces a much wider separation between these groups. It's also worth noting that the UMAP projection of the dataset took 4 seconds in comparison to 260+ seconds with t-SNE method.

Regardless of the differences between t-SNE and UMAP, they both helped us visualize that the two digits seem to be linearly separable. This observation suggests that we might want to test out some linear methods during classification.

KMeans is a well-known clustering method and it uses Euclidean distances to measure how far each data point lies from the centroids. However, it cannot be applied in high dimensional space since Euclidean distances do not make sense in high dimensions. Here, we utilized UMAP and t-SNE embeddings on the digits data which contains 5,000 features, and we observe that KMeans clustering fails miserably on our data. Below are two graphs obtained by UMAP and t-SNE, respectively, and the data points are color coded such that each color represents a KMeans cluster.

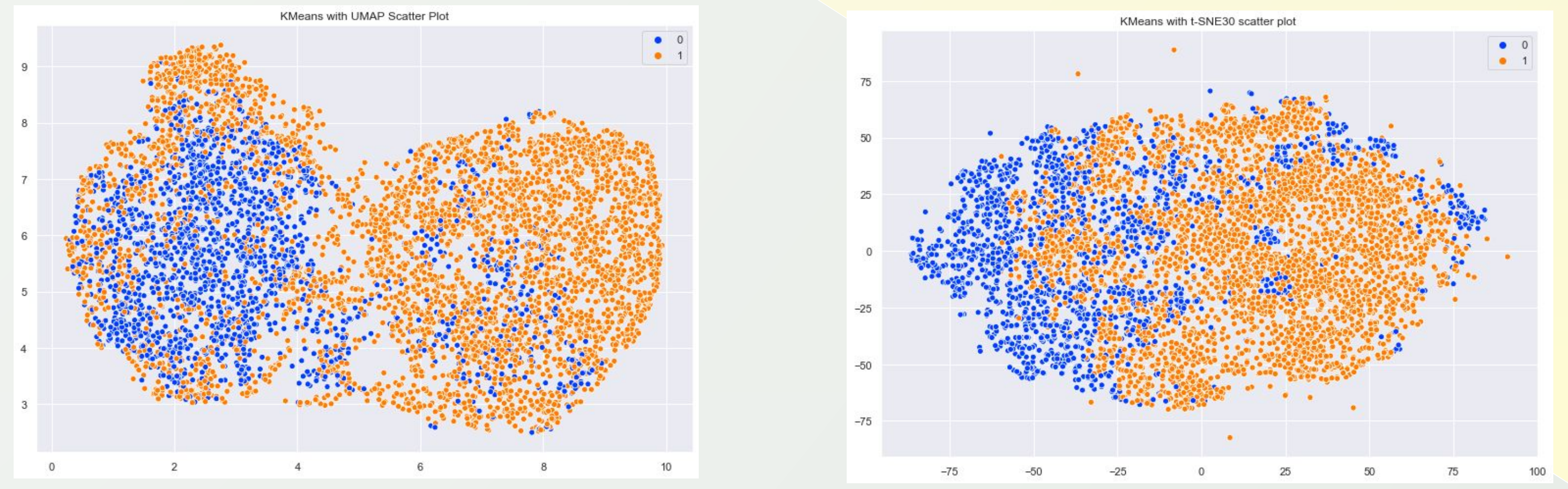


Figure 4. KMeans clustering results on UMAP and t-SNE embeddings.

FEATURE SELECTION & CLASSIFICATION

Classification. Because from the previous section we have visualized that the data seems to be linearly separable, we decided to test out a few standard classification models especially those with linear learners. Here we included Linear SVM, Random Forests, and K Nearest Neighbors with all 5,000 features. We then examined a range of feature selection methods including Lasso regression, Recursive Feature Elimination (similar to backwards stepwise), Bonferroni correction, Benjamini-Yekutieli correction, and Benjamini-Hochberg correction. We also applied a 3-layer Neural Network and a Convolutional Neural Network to the full feature set. The classification results are shown below:

Test Accuracy	All features	Lasso	RFE	Bonferroni	Benjamini Yekutieli	Benjamini Hochberg
Linear SVM	0.977	0.962	0.976	0.973	0.974	0.971
Random Forests	0.968	0.954	0.948	0.967	0.968	0.966
KNN	0.961	0.949	NA*	0.956	0.96	0.96
# features	5000	1738	1738*	4247	4488	4735
** RFE requires we specify the number of variables to be selected, and is benchmarked against Lasso*						
** KNN does not have feature selection implementation in Scikit-Learn*						

Figure 5. Classification with feature selections.

Regardless of feature numbers, Linear SVM with all features performs the best with 97.7% accuracy. With feature size in mind, Linear SVM with RFE selected variables can achieve similar results (97.6%) with only 1738 variables. CNN trained on the original image data also achieves 96.9% accuracy. Overall, we can see that Linear SVM does a really good job across different subsets of features compared to the other methods. We suspect that this is because of the fact that the data is linearly separable in low dimensional settings, as shown with UMAP and t-SNE before.

Recursive Feature Elimination (RFE) is a Scikit-Learn package. It goes through a linear loop where it removes a small number of the weakest features in every step until a specified number of features (in this case 1738 as benchmarked against Lasso) is reached. RFE searches for feature dependencies and collinearity and consider those highly dependent features best candidates to remove.

Multiple testing and Lasso. Comparing Bonferroni, Benjamini-Yekutieli and Benjamini-Hochberg, we observe that Bonferroni is the most conservative procedure where only 4247 variables are selected. Benjamini-Yekutieli is slightly more relaxed (4488), and Benjamini-Hochberg is the most relaxed one, selecting 4735 features.

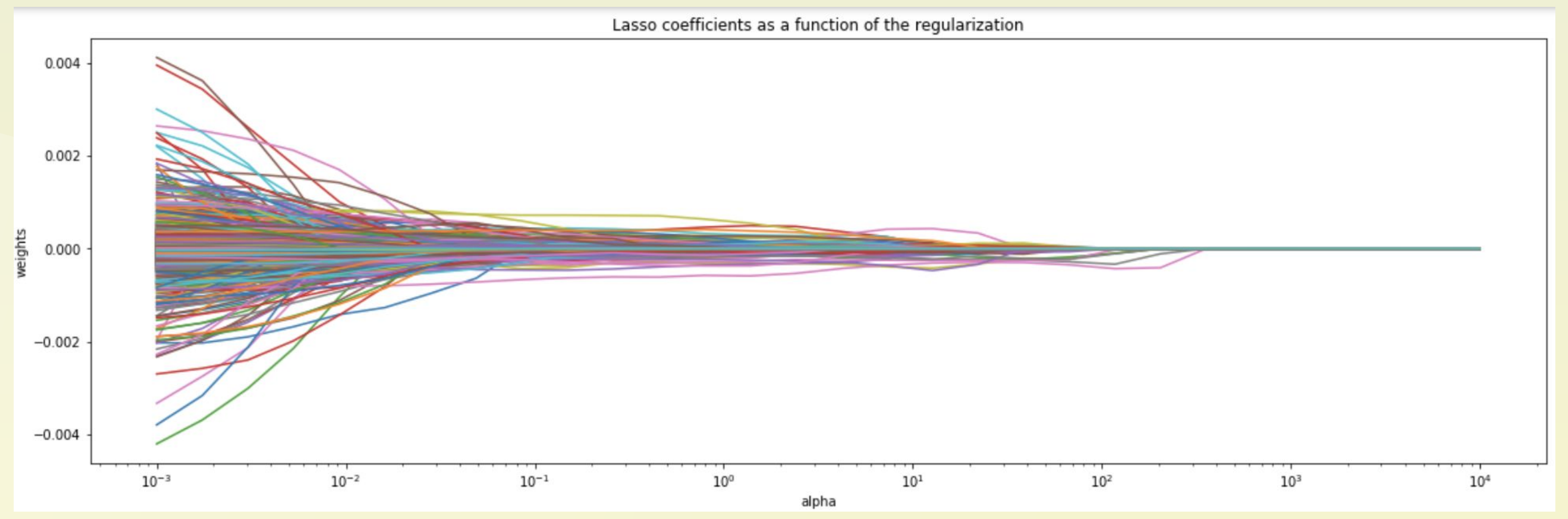


Figure 6. Lasso path

All three multiple testing procedures rejected more variables than Lasso, which only selected 1738 variables with cross validation. This is interesting to observe since Lasso and multiple testing theoretically should produce similar results. As Zhang et al. explained in their paper, because the Benjamini-Hochberg procedure exhibits a Gaussian distribution N(0, 1), we could obtain equivalent results from Lasso and from multiple testing, if we set the lambda value in Lasso to be equal to the Benjamini-Hochberg threshold statistic (2.63 in this case).

CNN & 3-Layer NN. Without feature selection methods, we also implemented a 3-layer Neural Network with Tensorflow. It is trained with 1,000 hidden units on the first layer, 500 units on the second layer, and 10 units on the last layer. It uses an Adam optimizer with 1e-4 learning rate, 1,000 iterations and 0.5 dropout on the last layer to prevent overfitting. It achieved 95.6% accuracy on the test set but does not exceed the results obtained by other simpler methods.

The Convolutional Neural Network is trained on the original "4" and "9" images subsetted from the MNIST dataset with Keras. Without the same feature engineering procedure, it is debatable whether we can compare CNN's accuracy with the other methods trained on Gisette. The reason that CNN did a great job for this data is because it applies a filter while moving the tensors across the image. Therefore it takes into account of the local relationships between pixels while re-assigning the weights in each iteration. In this case, we added two Convolutional layers with filter size 3*3 and ReLU activation function, one Max Pooling layer with filter size 2*2, as well as two dense layers and two dropouts. CNN achieved 96.9% accuracy on the test dataset.

CONCLUSION

We first performed dimensionality reduction with t-SNE and UMAP on the high-dimensional handwritten digits dataset. From the low dimensional embedding, we observed linearly separable clusters, and demonstrated that clustering techniques that utilize the Euclidean distance such as KMeans fails in high dimensional settings.

We then employed Linear SVM, Random Forests and KNN on the subsets of features selected by a few different feature selection methods, including multiple testing, RFE and Lasso. Comparing the accuracy metrics from these classification models, we observed that Linear SVM with RFE selected features maintains high performance with much smaller feature spaces. We also compared Lasso with the Benjamini-Hochberg procedure on their theoretical similarity. In the end, we tested out a 3-layer NN and a CNN and the dataset and achieved comparable results. Future efforts could focus on improving the prediction accuracy with other feature selection methods.

REFERENCE

Sklearn.manifold.TSNE¶. (n.d.). Retrieved May 01, 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

Zhang, B., Cheng, G., Zhang, C., Zheng, S., *Variable Selection Procedures from Multiple Testing, Science China Mathematics, Vol. 62 No. 4: 771–782*, https://link.springer.com/content/pdf/10.1007/s11425-016-9186-x.pdf

Huang, M., Hung, Y., Lee, W., Li, R., Jiang, B., *SVM-RFE Based Feature Selection and Taguchi Parameters Optimization for Multiclass SVM Classifier, TheScientificWorldJournal, 2014, 795624.* https://doi.org/10.1155/2014/795624

https://www.digitalocean.com/community/tutorials/how-to-build-a-neural-network-to-recognize-handwritten-digits-with-tensorflow

https://towardsdatascience.com/a-simple-2d-cnn-for-mnist-digit-recognition-a998dbc1e79a