STP 498: Machine Learning (Final Project)

Ernest Dela Cruz, Hamza Amjad, Jeremy Liu, Yassine Mazboudi

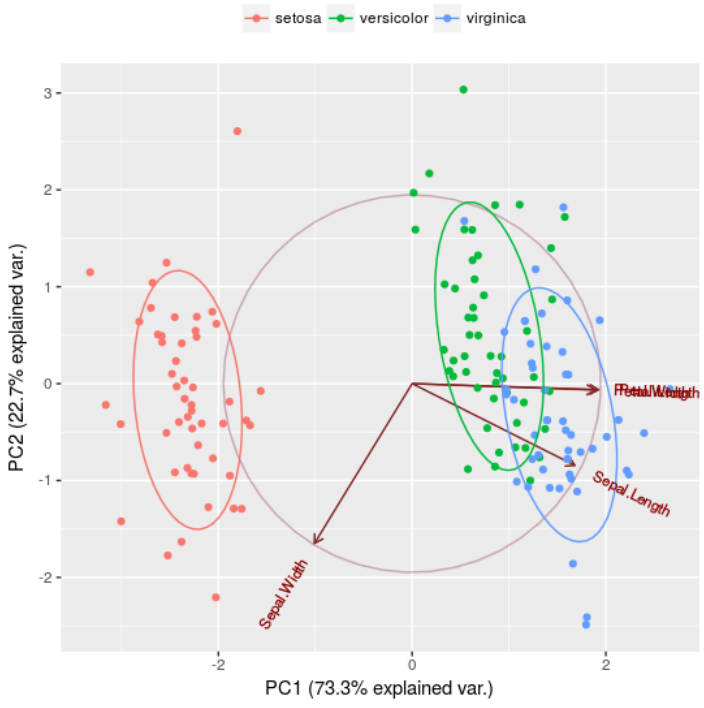
(1) **Introduction**

We begin by introducing the concept of dimensionality reduction, which in the linear space includes techniques (but not limited to) Principle Components Analysis, Singular Value Decomposition, and Factor Analysis. Our project will be focused on Autoencoders used with the MNIST dataset which can be thought of as one of the many non-linear analogs of the previously mentioned techniques, one can also view Autoencoders in the realm of topology and differentiable geometry as finding features in the manifolds of our data. The motivation for all of these tools is for dimensionality reduction. We will refer to the matrix of our data as X where every row corresponds to an observation in p-dimensions $x \in \mathbb{R}^p$, now per the discussion from class and our intuitive understanding of the bias-variance tradeoff it may be beneficial to reduce the number of variables used in a supervised/unsupervised learning scenario, especially when p is large. So what we want then is a new representation of our observations $\tilde{x} \in \mathbb{R}^q$ where $q << p$. There are a multitude of benefits from this representation of our data which includes:

(a) Intuition of Bias-Variance tradeoff as mentioned earlier
(b) Improved Computational Efficiency
(c) Potentially better understanding of the data and best subset for representing our observations

(2) **Dimensionality Reduction (Example and Auto-encoders)**

To start we introduce some definitions. The term "Latent" can mean to be something hidden or unknown, and we have something known as "Latent Variables" within our data. The purpose of finding these Latent Variables is to reduce our data to bare essentials that best represent our data, and this is beneficial for the reasons mentioned prior such as computational efficiency. Below is an example of PCA and a Latent Space. The data for this example is based on the IRIS data set whose four variables were the length and width of the flower, and then the length and width of the petals. These were used to help determine the clusters of the three types of flowers in the data set Iris Sestosa, Iris Versicolor, and Iris Virginica. The R-Code used for this example was sourced from [1].

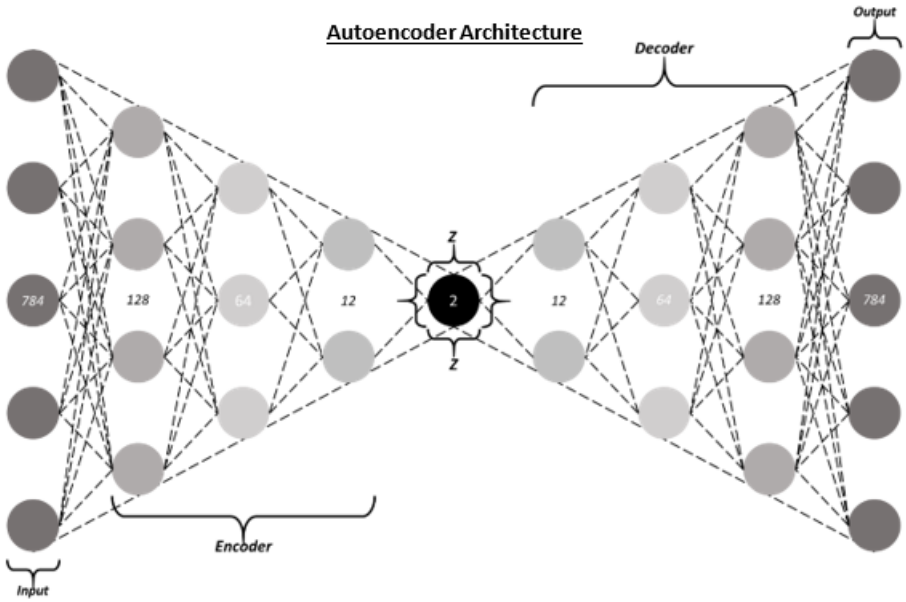


In the above image our representation of the data has been compressed to two Principle Components {PC1, PC2} with our original variables plotted along with all of the observations. With this all of the data points were represented by just two variables and fortunately the various types of Iris within the dataset clustered together in this 2-dimensional representation of a 4-dimensional data set.

We now divert our attention to Autoencoders which was the primary topic of interest for our project. An Autoencoder in summary is an neural network used in unsupervised learning for feature representation. The goal of an autoencoder is very similar to that of the previously mentioned techniques for data reduction, but is not confined to linear subspaces (non-linear subspaces/manifolds). An Autoencoder is composed of three components:
1. Input Layer $X \in \mathbb{R}^p$.
2. A hidden layer that compresses this to $q << p$.
3. An an output layer which is the latent representation of our data.
For our project we used very standard Autoencoder that was just a simple feed forward neural network. The goal is to be able to classify data accurately by utilizing Latent Variables within the data. Often these non-linear subsets of the data are termed as Data Manifolds in that in Manifold Learning the distributions that generate the data are hypothesized to be confined to a lower dimensional space in our full data set. The purpose of feature learning is to find this lower dimensional space and use this subspace rather than the full data set to represent our observations. Further more, we attempt to explore the Latent space as well. Unlike PCA, the Latent space is much more difficult to understand what every latent variable represents because of the non-linearity of the transformations.

(3) **Data**
The Data used for this project was the MNIST Data set provided on the course webpage. The inidividual images are 28 x 28 pixel grey scale coded in the data frame as 784 columns, each column representing a pixel of values 0 to 255, and the 785th column representing the actual digit of the image.

(4) **Method**
Our intent is to train an autoencoder to take a high dimensional image of 784 dimensions and accurately represent it on a 2-dimensional plane. Furthermore, we intend to explore the latent space by taking weighted-averages of the 2-dimensional latent variables that represent the original images and observe the output.
The autoencoder was trained on simple feed-forward neural net composed of 9 layers. Layer 1 takes the input X where $X \in \mathbb{R}^{784}$. Hidden layers 2 to 4 represent the encoder, which reduces the 784 dimensions down to 2 dimensions through a series of non-linear transformations. Hidden layer 5 represents the reduced 2-D latent variables $\hat{z}_{image}$ where $\hat{z}_{image} \in \mathbb{R}^2$. Layers 6 to 8 represent the decoder which takes $\hat{z}_{image}$ and reconstructs $\hat{X}$ where $\hat{X} \in \mathbb{R}^{784}$, the original 784-dimensional representation of the image. The count of nodes in each layer was constructed as: 784, 128, 64, 12, 2, 12, 64, 128, 784. The epoch parameter was set at 100, batch size at 128, learning rate at .001, and a rectifier activation function was applied to each layer. Our autoencoder was trained in Python using the framework PyTorch.

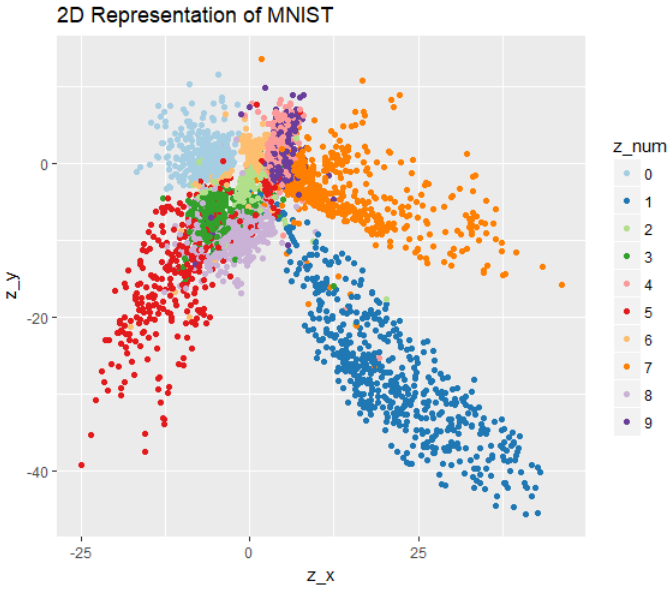


After constructing the autoencoder, we attempted to explore the latent space as they are not simple to understand. Understanding the latent space would imply understanding what the autoencoder found unique of each image to be able to remove the noise and reduce the image down to 2-D. We explored the latent space by taking weighted-averages of the latent variables of 2 images and walked the weight value ω by small increments, observing how the autoencoder decoded the new latent variable $\hat{z}_{average}$ into a new image.

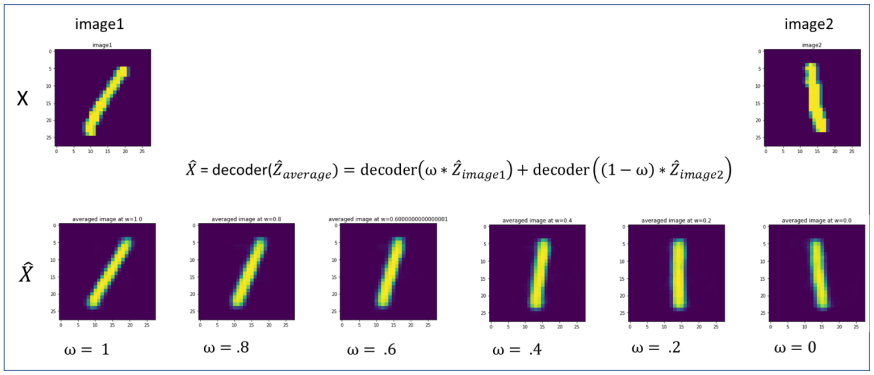
$$\hat{z}_{average} = \omega * \hat{z}_{image1} + (1 - \omega) * \hat{z}_{image2}$$

In the equation, $\omega \in [0, 1]$. We would then pass the new variable $\hat{z}_{average}$ into the decoder and generate our new image $\hat{X}$ We took the weighted-average of 2 different representations of the digit 1, 4, and 2 seeing how the different shapes take form as we incremented ω. We also experimented with taking the weighted-average of 2 separate digits and observed how completely different digits were formed. Essentially, by taking a weighted average, a line is drawn between the 2-D points on the X-Y plane, therefore, any digit on that line could be represented by some combination of the 2 latent variables.

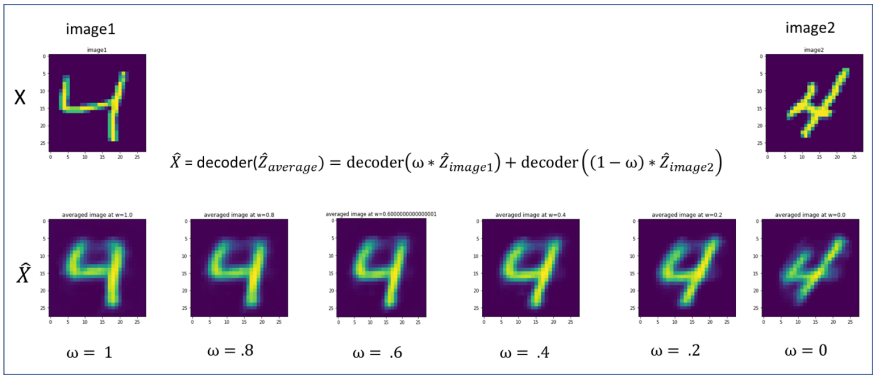
(5) **Results**



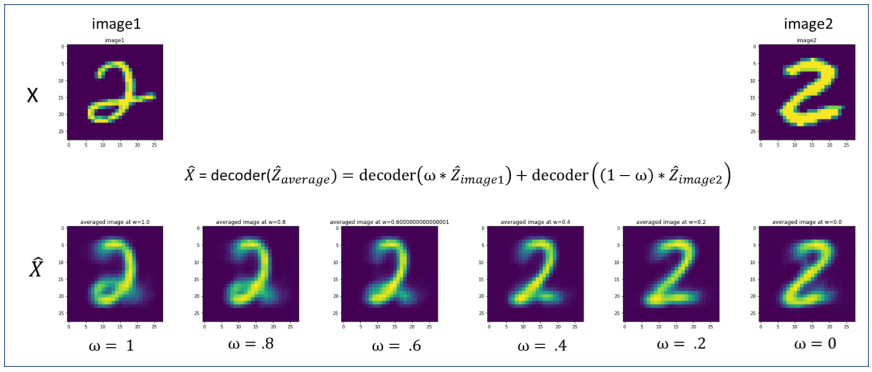
Pictured are 5000 digits from the MNIST dataset and their 2-D latent variables plotted. Clearly, certain clusters representing the different digits formed speaking to the quality of the autoencoder. It appears the autoencoder clustered the digits 0, 5, 7, and 1 particularly well.



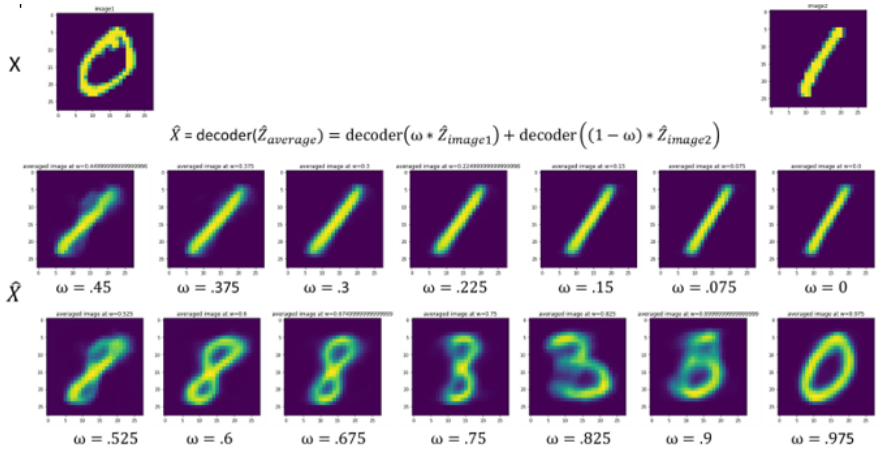
Pictured is the weighted-average of a partially rotated 1 and another partially rotated 1 in the opposite direction walking ω by .2 at each step. Note at $= 1$ it is the autoencoder output of image1 and at $\omega = 0$ is it the autoencoder output of image2. Notice the rotation of the 1 which makes us believe the latent space in between the 2 latent variables is examining the rotation of the digits.



Pictured is a similar weighted-average procedure of two differently-shaped 4s. Notice the autoencoders outputs were fuzzier than those of the digit 1 showing the loss of clarity caused by the reduction in dimensions. It is unclear what to make of the latent spanned by those two latent variables. It is still interesting to see the different shapes take form!



Pictured is a similar weighted-average procedure of two differently-shaped 2s. Again, notice the loss of clarity in the loop of the first image lost in the encoder process of reducing the dimensions. Notice the dissipation of the loop at the bottom of the 2 as the weights transition.



Pictured is a similar weighted-average procedure but with steps of .075 to explore the full extent of the latent space between the two latent vectors. Because taking a weighted average of a 2-D point implies drawing a line of between two points, any point that is on this line can be recreated by a linear combination of the two latent variables. Referencing the 2-D plot of the 5000 MNIST digits, taking a 0 which clustered on one corner of the plot and a 1 which is clustered on the polar end of the plot, we can recreate digits that fall in between them. Without any calculations to extract the exact linear combination needed to recreate an already existing image on the plot, observable images naturally developed! Clearly, the 1 transitions into an 8 which slowly transitions into a 3, which finally transitions to the original image of 0. Fascinating!

(6) **Conclusion**

In conclusion, the MNIST data of 784 dimensions was reduced to 2-dimensions using a simple feed-forward autoencoder. Clusters of digits were clearly visible for some and not clear for others using an autoencoder. Understanding the manifolds and the latent space is extremely difficult as they are essentially random in the training of the autoencoder. If the model was to be retrained, an entirely different latent space would be formed. To explore the latent space of this particular model, we successfully took weighed-averages of differently shaped single digits 2-D latent variables to recreate images that did not already exist in the dataset. Furthermore, we successfully took weighted-averages of two different digits latent variables and created entirely new digits whose image did not exist in the data set either. Although it is unclear what to conclude about what the autoencoder identified as unique per image or add intuition behind the non-linear transformation made, it is clear there are observable patterns the model proxies. Further research can be conducted to identify which features of the images are particularly being targeted per digit and which are ignored as noise. Furthermore, it would be interesting to explore in higher dimensions whether this weighted-average of latent variables would still produce observable digits as we were able to do in this project.

References:

1. https://www.r-bloggers.com/computing-and-visualizing-pca-in-r/