# Applications of Discrete Mathematics—Machine Learning

Matthias Rathbun

University of Central Florida

# APPLICATIONS OF DISCRETE MATHEMATICS—MACHINE LEARNING

## INTRODUCTION

With the rapid improvement of consumer computer technology that rival enterprise-level performance, high-level data analysis is becoming more accessible to the everyday person. With libraries such as Scikit-Learn both handling the background mathematics and providing excellent usage documentation for machine learning algorithms, running a once hard-to-grasp algorithm has become a simple task. Yet behind the simple code is an intricate dive into discrete mathematics. Models like K-Nearest-Neighbors (KNN) and K-Means Clustering apply graph theory to predict and classify clusters while Principal Component Analysis (PCA) and T-Distributed Stochastic Neighbors Embedding (TSNE) seek to reduce the dimensionality of a dataset with entirely different approaches. This discussion seeks to explain the mathematics behind the aforementioned algorithms while providing real-world applications of their uses.

## DIMENSIONALITY REDUCTION

Performance of machine learning algorithms often degrades when there are too many input dimensions. Given a dataset of size M rows by N columns, dimensionality reduction seeks to reduce the N-space dataset to a visualizable space, often 2 or 3-space. Having high dimensionality can clutter the feature space, making it tougher for models to distinguish between points in the dataset (Brownlee, 2020). This can lead to overfitting on the training dataset, causing the model to fail test-validation. The two most prominent methods for dimensionality reduction are PCA and TSNE, with PCA utilizing Singular Value Decomposition (SVD) to linearly reduce the feature space and TSNE non-linearly reducing dimensionality.

Given a dataset $X \in \mathbb{R}^{(m*n)}$ and a transformation $Y = XP$ where $Y \in \mathbb{R}^{(m*r)}$ and $P \in \mathbb{R}^{(n*r)}$, PCA reduces a n-space dataset to a r-space dataset. In order to find similarities and correlations within the dataset, PCA utilizes the Covariance of the feature space.

$$C = \begin{bmatrix} Var(V_1) & Cov(V_1,V_2) & \cdots & Cov(V_1,V_n) \\ Cov(V_2,V_1) & Var(V_2) & \cdots & Var(V_n) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(V_n,V_1) & \cdots & \cdots & Var(V_n) \end{bmatrix} = \frac{1}{m} X^T X$$

For the transformed data Y, the covariance $D = \frac{1}{m} Y^T Y$ can be rewritten as $\frac{1}{m}(XP)^T XP = P^T CP$.

Since Y is linearly independent, $D = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \lambda_n \end{bmatrix}$ and through diagonalization,

$D = P^T CP, CP = PD, C = PDP^T$. Applying SVD on X yields $X = U\Sigma V^T$, thus

$X^T X = (U\Sigma V^T)^T U\Sigma V^T = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T \Sigma V^T = V\Sigma^2 V^T$ (Note: $U^T U = I$ and Σ is symmetric). Thus $C = \frac{1}{m} X^T X = \frac{1}{m}\, V\Sigma^2 V^T$. Comparing to the previous equation for C, we can conclude $P = V$ and $D = \frac{\Sigma^2}{m}$. Once P is obtained, dataset X can be transformed into Dataset Y using the transformation $Y = XP$ (Peng, 2019).

Contrary to the linear transformation and variances PCA uses, TSNE utilizes a probabilistic methodology to reduce dimensionality. The first part of the TSNE algorithm calculates the Euclidean distance of each point in the dataset from others. These distances are then used to calculate how likely each set of point are neighbors using the Gaussian Distribution. Given 2 points $x_i$ and $x_j$

$$P_{j|i} = \frac{exp(-||x_i - x_j||^2 / 2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2 / 2\sigma_i^2)}$$

Dividing by the sum of all other points allows for the algorithm to deal with clusters of different densities. TSNE then uses the joint probability $P_{ij} = \frac{P_{j|i}+P_{i|j}}{2n}$. TSNE then creates a random set of N points with K features, with K representing the target dimensionality of the dataset and N being the number of points in the dataset. For this set of points, TSNE uses the T-Distribution, which has heavy tails, causing small distances in high dimensionality to become extreme at lower dimensions. Marking the lower dimensional points with y and probabilities with q, $q_{ij} = \frac{(1+||y_i-y_j||^2)^{-1}}{\sum_k \sum_{l \neq k} (1+||y_k-y_l||^2)^{-1}}$. TSNE then uses the Kullback-Leiber Divergence (KL Divergence) as the cost function, $C = KL(P||Q) = \sum_i \sum_j P_{ij} log \frac{P_{ij}}{q_{ij}}$. This function is then minimized through gradient descent. The set of lower dimension data with the minimum cost thus becomes the output of the TSNE algorithm (Soroker, 2020).

To compare between PCA and TSNE, both algorithms are tested on the MNIST dataset, which contains 70000 28 by 28 Images (784 features) of hand drawn numbers 0-9.

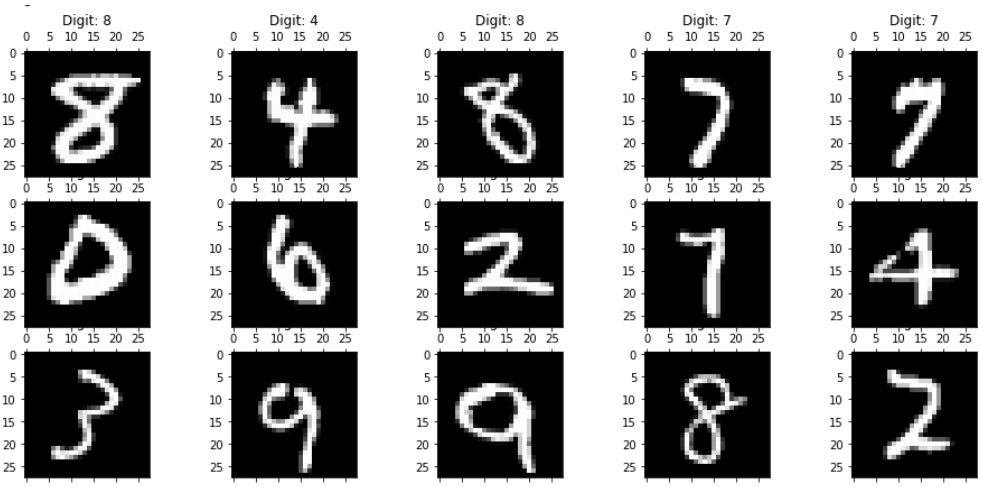


Figure 1: Examples of the Images in the MNIST Dataset

Figure 2 shows the stark differences in the outputs of the algorithms, with TSNE showing more discrete differences between the groups.

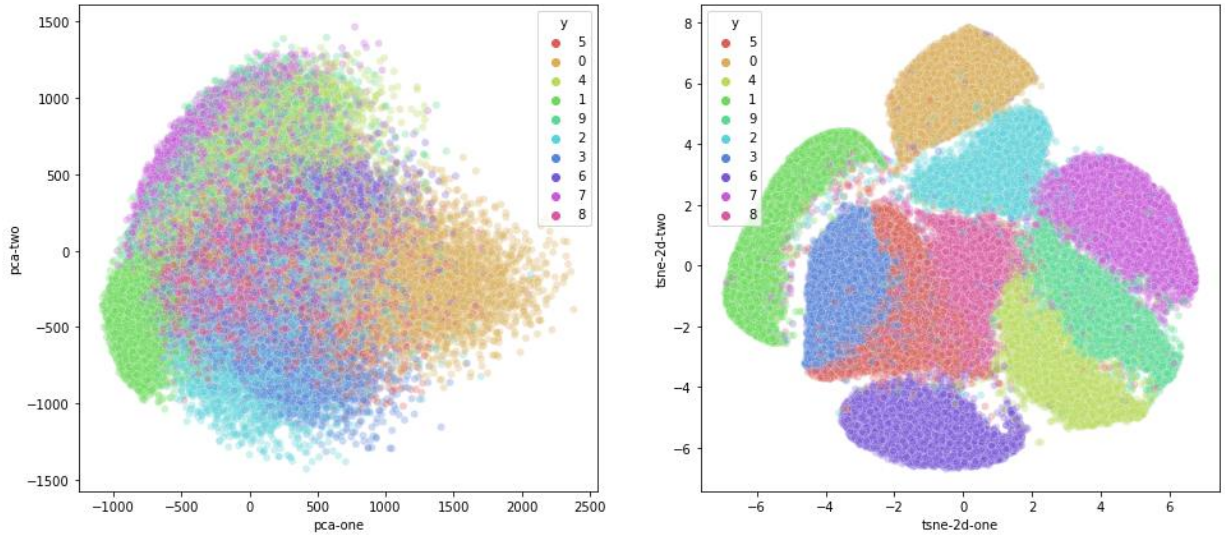


Figure 2: Comparison Between PCA (left) and TSNE (right)

This is due to both the nonlinear relationships considered by TSNE and the T-Distribution used by TSNE, which allows for small differences at higher dimensions to be shown on lower dimensions. While for visualization TSNE is preferable, TSNE is not used for clustering, as unlike PCA, TSNE does not preserve the inputs. This is due to TSNE randomly generating lower dimensional points rather than transforming them like PCA (Violante, 2018).

## CLUSTERING

Clustering can be applied through both supervised (with labels) and unsupervised (without labels) methods. Both the KNN and K-Means Clustering methods assume similar classes exist in close proximity. Given a K value, for each point in the dataset, KNN selects the K nearest points using Euclidean distances and finds the mode of the labels (Harrison, 2018). Since the K value is predetermined by the user, it is best to run the algorithm for a reasonable

range of K values in order to minimize the error and find the optimal K value. Band finds that the optimal K value lies around $\sqrt{N}$ which is in line with KNNs ran on the Iris dataset (2020).

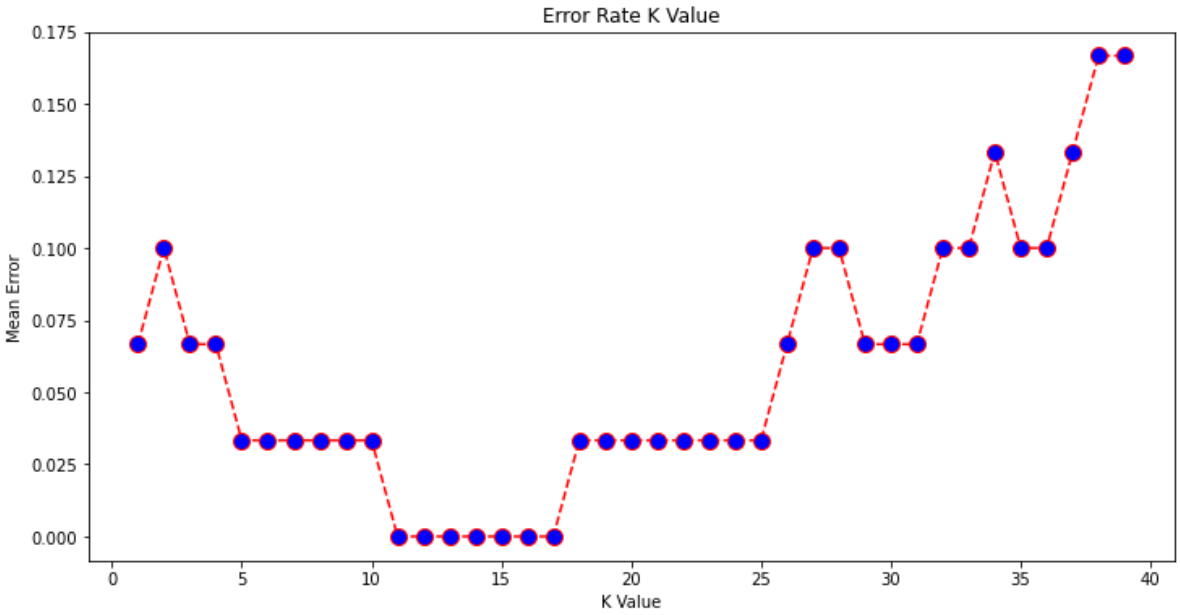


Figure 3: Error as a Function of K

With 150 samples, the optimal K value would be 12, which sits nicely at the minima of Figure 3. K-means works similarly, yet due to its unsupervised nature, it randomly assigns centroids at first (Garbade, 2018). The goal of K-means clustering is to divide the data into K clusters in a way that each clusters' points are similar, and other clusters are dissimilar. K-means operates in five steps:

1. Randomly select centroids.
2. Calculate the distance of all points to each centroid.
3. Assign each point to the cluster with the closest centroid.
4. Find new centroids by averaging the datapoints in each cluster.
5. Repeat 2-4 until a solution converges (centroids stop moving)

The major drawback to K-Means is that the number of clusters must be predetermined, as no where in the algorithm allows it to detect new clusters (Yildirim, 2020). Fortunately, this is where the aforementioned dimensionality reduction techniques are applied. On unsupervised data, PCA and TSNE can be utilized to find distinct clusters.

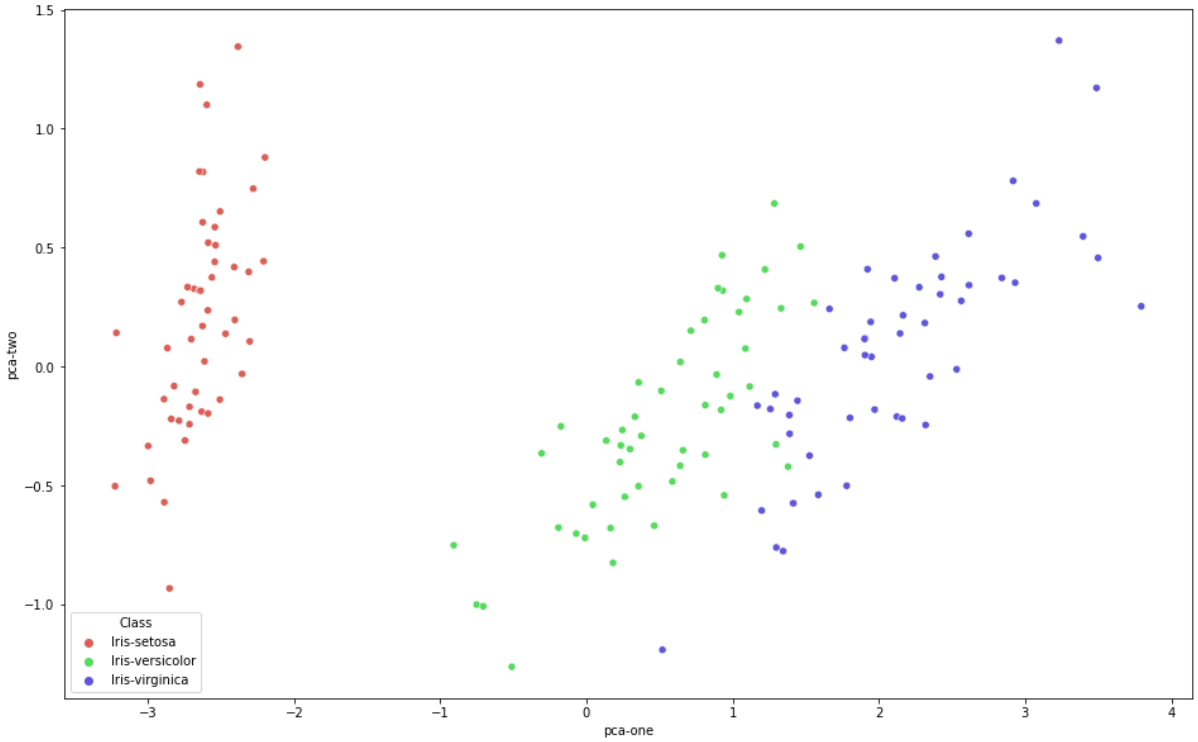


Figure 4: 2D-PCA on the Iris Dataset

Even if the Iris dataset were unlabeled, Figure 4 shows 3 distinct groupings in the dataset, which would allow for the determining of the number of clusters used in k-means clustering.

## CONCLUSION

Machine learning, a once convoluted, and now simple task any can take is deeply rooted in discrete mathematics. Testing on dimensionality reduction and clustering algorithms validate past literature and show the purpose of the algorithms. Analyzing the mathematics behind PCA shows that it preserves the data by making linear transformations, making it excellent for clustering. Its counterpart TSNE attempts to exemplify smaller differences utilizing the T-Distribution, making it better for data visualization as it creates more discrete groupings (Figure 2). KNN and K-Means both operate on the same principle, utilizing graph theory to minimize the difference between points in each cluster while maximizing difference from other clusters.

Code for examples of each of the algorithms discussed may be found at https://github.com/MatthiasRathbun/COT3100-Final-Project

# REFERENCES

Band, A. (2020, May 23). *How to find the optimal value of K in KNN?* Retrieved from Towards Data Science.

Brownlee, J. (2020, June 30). *Introduction to Dimensionality Reduction for Machine Learning*. Retrieved from Machine Learning Mastery.

Garbade, M. J. (2018, September 12). *Understanding K-means Clustering in Machine Learning*. Retrieved from Towards Data Science.

Harrison, O. (2018, September 10). *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. Retrieved from Towards Data Science.

Peng, M. L. (2019, May 25). *The Math Behind Principal Component Analysis*. Retrieved from Medium.

Soroker, A. (2020, August 4). *T-SNE Explained — Math and Intuition*. Retrieved from Medium.

Violante, A. (2018, August 29). *An Introduction to t-SNE with Python Example*. Retrieved from Towards Data Science.

Yildirim, S. (2020, March 3). *K-Means Clustering — Explained*. Retrieved from Towards Data Science.