BOĞAZİÇİ UNIVERSITY

CMPE 462

Project 3

İNCI MELIHA BAYTAŞ

BETAZERO

BARAN DENIZ KORKMAZ - 2015400183

DOĞUKAN KALKAN - 2015400132

SPRING 2020

Contents

1 Introduction

In the field of Machine Learning, there are several types of learning which are speciliazed according to their purpose. The main categories are the following:

- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

In our course, we focus on the two types of learning: Supervised Learning and Unsupervised Learning.

Supervised Learning is a learning technique where we use labeled data to train a model to come up with a function that maps our data points to their corresponding labels(or targets). In other words, we try to build a model that explains the relationships and dependencies between our data and labels.

In the case of Unsupervised Learning, we do not have such labeled data. We only have the data set itself, and our aim to find undetected patterns in the data set without supervision. In other words, we let the algorithm discover information all by itself.

In this project, we are asked to implement one of the popular Unsupervised Learning methods, K-Means Clustering and to implement a Principal Component Analysis(PCA) which is a dimensionality reduction technique used in Machine Learning. Both will be thoroughly explained below.

2 Task 1: K-Means Clustering

So far in our course, we have only worked with algorithms that involve supervised learning. In supervised learning, models that we build are to learn from labeled training data, then the models are tested with test data that is not introduced previously to measure the accuracy of them.

However, in unsupervised learning, we do not have such labeled data. The model we desire to build works on its own to discover and obtain information related to the data.

One of the most popular unsupervised learning methods is clustering. There are two types of clustering: Partitional Clustering and Hierarchical Clustering. In this project, we are asked to design a K-Means Clustering which is a type of Partitional Clustering. Below, you can find explanations for both K-Means Algorithm and our implementation of the algorithm.

2.1 Visualization of Clusters

Below, you can see the Ground Truth for Clusters. Even though there are no labels present in K-Means Clustering, it is necessary for us to compare our result with the correct results.

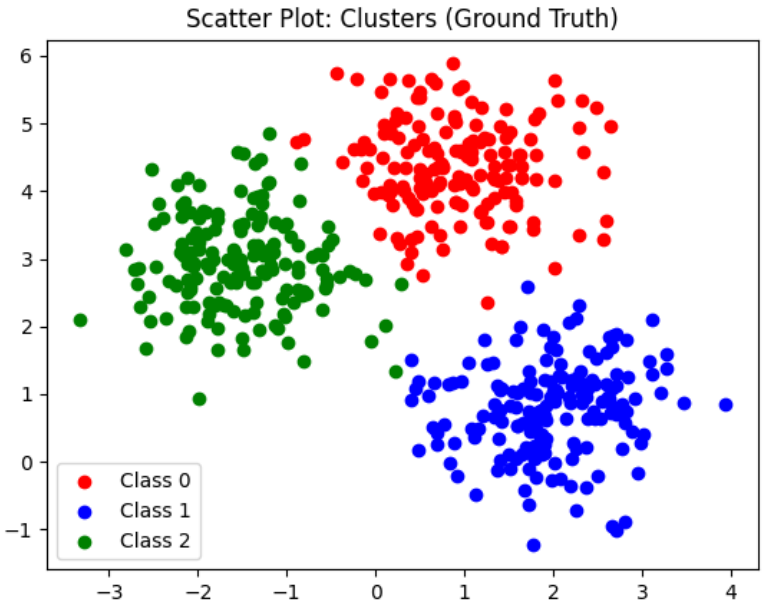

Figure 1: Ground Truth for Clusters

2.2 K-Means Algorithm

K-Means Algorithm partitions the data into K clusters each of which is associated with a centroid which is the center of that cluster. Each cluster is formed in a way that the sum of Euclidean distances of each point to the centroid of the cluster to which that point belong is minimized. More formally, it is given by the formula:

$$\arg\min_{c_j, m_{i,j}} \sum_{j=1}^{K} \sum_{i=1}^{n} m_{i,j} \|x_i - c_j\|$$

where m_i,j = 1 if i^{th} data point belongs to j^{th} cluster, otherwise 0 and c_j is the centroid of j^{th} cluster. Also K must be specified beforehand.

Below, you can find the pseudo code for K-Means Algorithm and explanation for each line.

Algorithm 1: K-Means Algorithm

```
Select K points as the initial centroids.;
while The centroids change do
    Form K clusters by assigning all points to the closest centroid.;
    Recompute the centroid of each cluster.;
end
```

Note that the algorithm above is the general K-Means Algorithm. In our case, the stopping condition is different. Our algorithm executes for number of iterations which is given as a parameter. The reason is that the data set we work with is relatively smaller compared to the data sets dealt with in the field of Machine Learning. Thus, our algorithm converges after a couple of iterations which suggest that using a maximum number of iterations as a stopping condition makes sense.

We initially start with creating initial centroids for clusters. For this purpose, we have a function called ***get_initial_centroids*** which takes 5 parameters: *K, x_min, x_max, y_min, y_max*. For each cluster, we choose a centroid whose coordinates are between the minimum and the maximum coordinates.

After initializing the centroids, we proceed by by assigning each data point to the correct cluster. Then, as written in the pseudo algorithm, we recompute the centroids of each cluster.

Here is how the assignment of each data point to the correct cluster works: We have a function named ***get_clusters*** which takes 2 parameters: *data, centroids*. The algorithm computes the Euclidean distance of each data point to all the centroids, and assigns the data point to the centroid that is closest to the data point.

Here is how the recomputation of the centroids works: We have a function named ***get_centroids*** which takes 3 parameters: K, data, labels. The algorithm computes the sum of each coordinate of the data points for each cluster and sets that coordinate of the centroid of that cluster to the average of the sum.

The whole procedure explained above is applied N times, which is given as a parameter as well.

Below, you can see a clustering example with N=12.

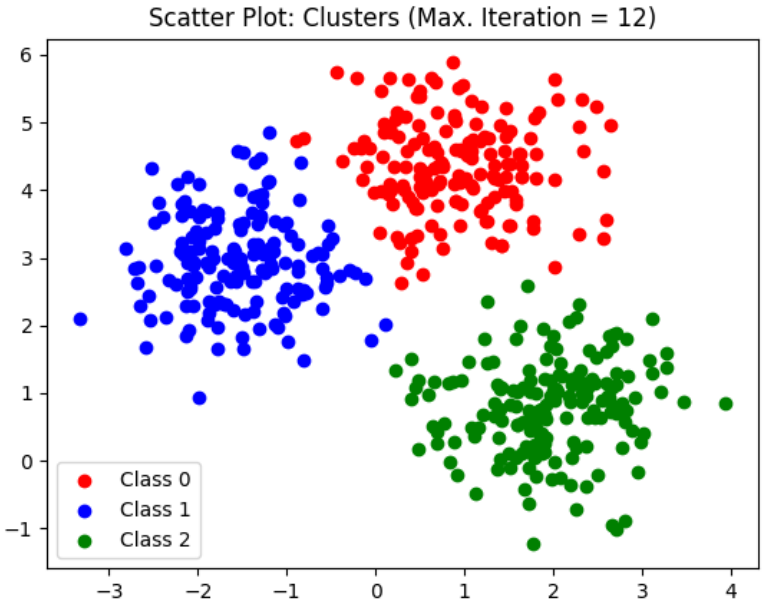

Figure 2: Clustering with N=12

2.3 K-Means Clustering

After implementing the K-Means Algorithm with a maximum number of iterations as a stopping condition, we can now test our algorithm with different number of iterations. Before testing our algorithm, we expect to see better results as the number of iterations increases until some point beyond which there could be no further improvement. This number is our convergence point. Depending on the data set and its size, the number of iterations which takes the algorithm to converge could be different. Another factor that affects the convergence point is the number of clusters formed in the algortihm.

In our case, there are 3 clusters to be formed. And our aim is to observe the behavior of our algorithm with respect to increasing number of iterations. Below, you can see a 3x3 plot for 9 executions of the algorithm with number of iterations $N = 1,2,3,...,9$. To observe the accuracy of our results, we refer to the ground truth obtained in the first part of the task. Below, you can see our observations about the results as well.

Now, we can immediately observe that the algorithm works poorly when $N=1$, which is expected since the algorithm cannot manage to divide clusters in one step. However we can see that the locations of the centroids are very close to the real ones. Also we should note that the initial centroids plays a huge role in the algorithm, so it is quite normal for the algorithm to give these results which are not so bad. When $N=2$ and $N=3$, we can observe that unlike the clusters when $N=1$, now the clusters are more accurately divided. Class 0 has roughly 80% of the data points, but now as the centroids and points that belong to the clusters of associated centroids change in each step, the clusters are very close to the ground truth.

When $N=4$, it can be seen visually that the clusters converge to their final form. Comparing it with the figure 1, there are only 2 data points that belong to the wrong cluster.

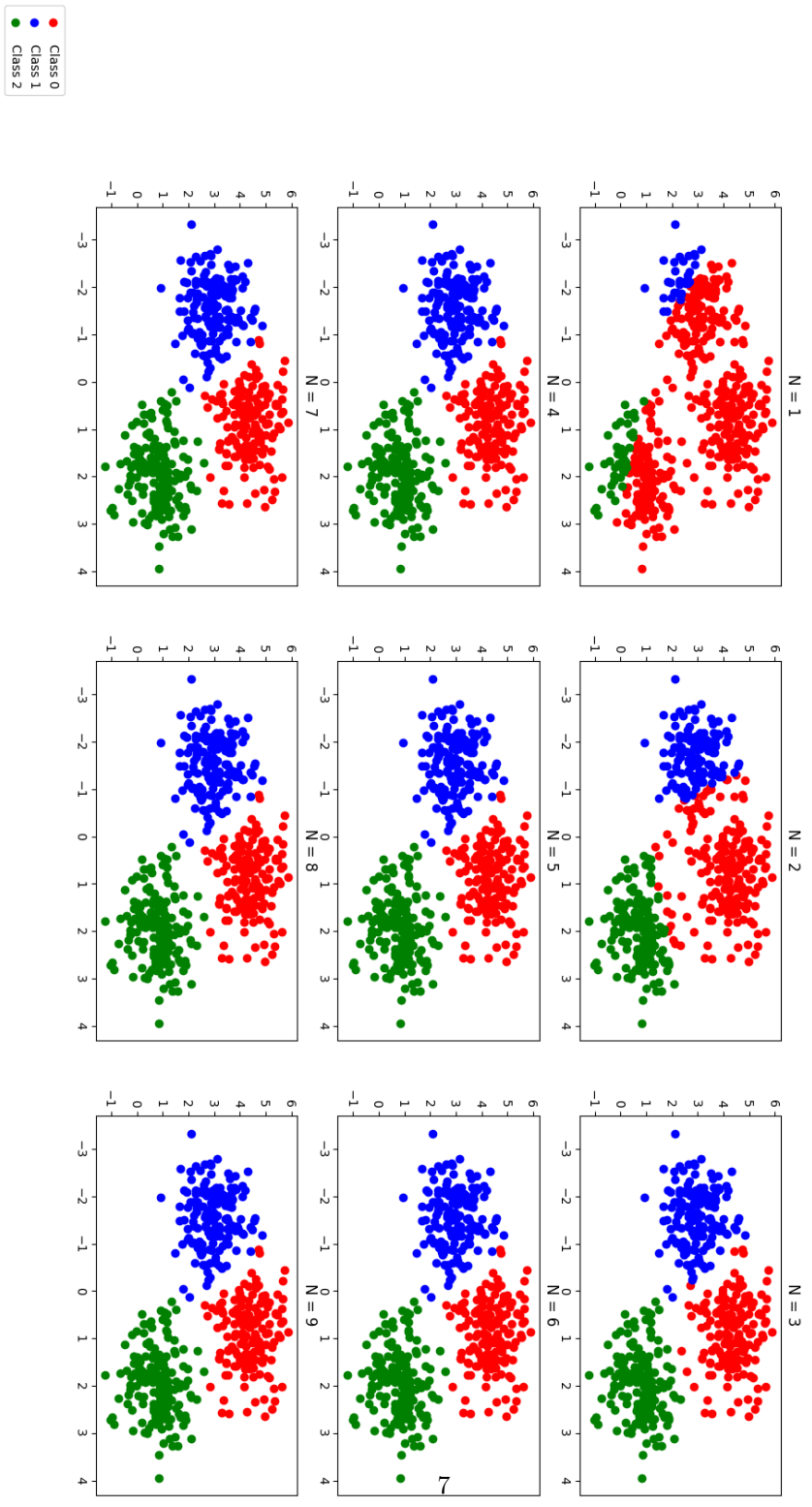
Scatter Plot: Clusters vs Max. Iterations
N = 1
N = 2
N = 3
N = 4
N = 5
N = 6
N = 7
N = 8
N = 9
Class 0
Class 1
Class 2

Figure 3

3 Task 2: Principal Component Analysis

In real-life applications, we usually deal with high-dimensional data, which brings some disadvantages. In most cases, high dimensional data requires a high-dimensional model construction, increasing the computational cost and data storage expenses. Plus, the intrinsic dimension may be much smaller, that is, the real dimensionality required to represent the data might not be the actual dimensionality. Getting rid of unnecessary attributes will also provide the removal of noise which increases the accuracy.

At this stage, dimensionality reduction helps us providing efficient algorithms to find necessary features to represent the data by applying transformation onto the data set. Let us express dimensionality reduction more explicitly. Suppose that we have a data set of p number of attributes, and we aim at finding new d many features from original features. Our goal is to learn a transformation matrix G from the data, where G $\in \mathrm{R}^{p*d}$. Taking a closer look into the G, we observe that the columns of G, $d_i \in R^p$, represents the coefficients that will construct the ith feature of a given data sample.

Dimensionality reduction can be applied by techniques including supervised and unsupervised learning approaches. We must note that the criterion for dimensionality reduction varies between learning approaches. In unsupervised feature reduction, we aim at minimizing the information loss, whereas in supervised feature reduction, we target maximizing the class discrimination. Below, you can find a number of useful dimensionality reduction techniques classified based on their learning approaches:

1. Unsupervised Learning
   (a) Latent Semantic Indexing (LSI)
   (b) Independent Component Analysis (ICA)
   (c) Principal Component Analysis (PCA)
   (d) Canonical Correlation Analysis (CCA)
2. Supervised Learning
   (a) Linear Discriminant Analysis (LDA)

We must be careful to distinguish feature selection and feature reduction. The feature reduction, which we deal with in this project, extracts the new features of lower dimensionality by applying linear or nonlinear transformations onto the original features. Whereas the feature selection is the procedure of selecting a subset of original features where no transformation is applied.

In the context of dimensionality reduction, the second unsupervised learning algorithm we focus on in the project is Principal Component Analysis which is a commonly used method. In the next section, we will cover Principal Component Analysis algorithm in detail.

3.1 PCA Algorithm

Principal Component Analysis is a dimensionality reduction algorithm which reduces the dimensionality of a data set by extracting a new set of variables, smaller than the original set of variables. Principal Component Analysis aims at minimizing the information loss caused by transformation of original features into new features by retaining the sample variance. Variance is an essential statistics implying the range of values we can encounter within the data set, therefore providing a key information about the data.

PCA algorithm results in a transformation matrix to apply onto the data to convert it into lower-dimensional version. The components constructing the transformation matrix are called Principal Components, which the algorithm is named after actually. The Principal Components are designed to be uncorrelated and in a specific order whereby they are ordered by the fraction of total information retained.

Suppose that the original data set is X, where X $\in \mathbb{R}^{N*p}$, and the transformed data set is Z, where Z $\in \mathbb{R}^{N*d}$. Let us define the first principal component, $a_1 \in R^p$, of the sample by the following transformation:

$$z_{j1} = a_1^T x_j = \sum_{i=1}^{p} a_{1i} x_{ji}$$

where x_{ji} is the value of ith feature of jth sample in the original data set for $x_j \in \mathbb{R}^p$, z_{j1} is the first feature of jth sample in the transformed data set for $z_j \in \mathbb{R}^d$.

Introducing the definition of principal component above, let us moving forward into a detailed algebraic definition. Stated previously, PCA extracts the features that preserves the sample variance. Thus, we can provide the computation of the first principal component, denoted by a_1 as the following constrained optimization problem:

$$\min_{a_1} \quad Var[z_1] \tag{1a}$$

$$\text{subject to} \quad a_1^T a_1 = 1 \tag{1b}$$

Please recall that the constraint implies the orthogonality property of transformation matrix. As we stated previously, the principal components forming the transformation matrix used in reconstruction of original data set are uncorrelated. To satisfy uncorrelation property, we design the transformation matrix in a way that the principal components composing the columns are orthogonal to each other with a unit norm.

Indeed, the computation of the variance is given by the following procedure, since we only work with samples within the data set:

$$\text{Var}[z_1] = \text{E}((z_1 - \overline{z_1})) = \frac{1}{N} \sum_{i=1}^{N} a_1^T (x_i - \bar{x})(x_i - \bar{x})^T a_1 = a_1^T S a_1$$

where,

1. $S = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T$, The Covariance Matrix

2. $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$, The Mean

After converting the constrained optimization problem into unconstrained optimization problem by using Lagrange Transformation, we obtain the following unconstrained optimization problem:

$$\min_{a_1} \qquad L = a_1^T S a_1 - \lambda(a_1 T a_1 - 1) \tag{2a}$$

Taking the derivative for a_1, we get the following equation:

$$Sa_1 = \lambda a_1$$

Repeating the procedure stated for the first principal component for the other principal components, we observe that we end up with the equation obtained above again, which enables us rewriting the generalized equation for the computation of principal components as follows:

$$Sa_i = \lambda a_i$$

The expression above presents the basis of the transformation matrix. The interpretation is as follows. The equation shows that the eigenvalues of the correlation matrix and the corresponding eigenvectors are actually our principal components. By obtaining eigenvalues, we are able to order them in terms of the information they retain in descending manner.

After we have explicitly denoted the algebraic definition of principal component analysis, we are ready to move forward into introducing the algorithm:

PCA Algorithm

1. Main Step: Extraction of Principal Components:
   (a) Standardize the data set $X \in \mathbb{R}^{N*p}$
   (b) Form the covariance matrix.
   $$S = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T = \frac{1}{N} \sum_{i=1}^{N} x_i x_i^T = \frac{1}{N} X^T X$$
   (c) Compute the eigenvectors of covariance matrix.
   (d) Use the first d eigenvectors $\{a_i\}_{i=1}^{d}$ with largest d eigenvalues.
   (e) Form the transformation matrix:
   $$\mathrm{G} = \begin{bmatrix} a_1 & ... & a_d \end{bmatrix},\ a_i \in R^p$$
2. Apply transformation onto the original data set to obtain reconstructed matrix. For each data sample:

$$x_j \in R^p \rightarrow \mathrm{z}_j = G^T x_j \in R^d$$

In the project, we are given a data set called USPS. The data set contains 3000 handwritten digits of 256 pixels. Our goal is to apply Principal Component Analysis subsequently for changing number of features in the new data set. The number of new dimensions will be 50, 100, and 200 in the new data set.

3.2 Reconstructed Matrix

Recall that the final stage of PCA Algorithm is to reconstruct the original matrix by applying transformation in order to obtain new features. This stage is stated previously in **3.1 PCA Algorihm** as follows:

$$\text{For each data sample } x_j \in R^p \rightarrow \mathrm{z}_j = G^T x_j \in R^d$$

We can also generalize the reconstruction procedure for the overall data matrix by the following equation:

$$\mathrm{Z} = (G^T X^T)^T = XG, \text{ where G is the transformation matrix} \in R^{p*d}$$

By applying the transformation, we obtain the new data matrix $\mathrm{Z} \in R^{N*d}$. Since we have obtained the reconstructed matrix which is composed of the transformed version of images, we are ready to move forward into the next section where we visualize the reconstructed images.

3.3 Visualization of Reconstructed Images

In the final stage of Principal Component Analysis, we visualize the reconstructed images as they are ready to get obtained within the rows of the reconstructed matrix. Below, you can see the plot for the reconstructed images at indices i=0, 500, 1000, 2000 for the reconstructed matrices of dimension d = 50, 100, 200 and finally for the original data set.

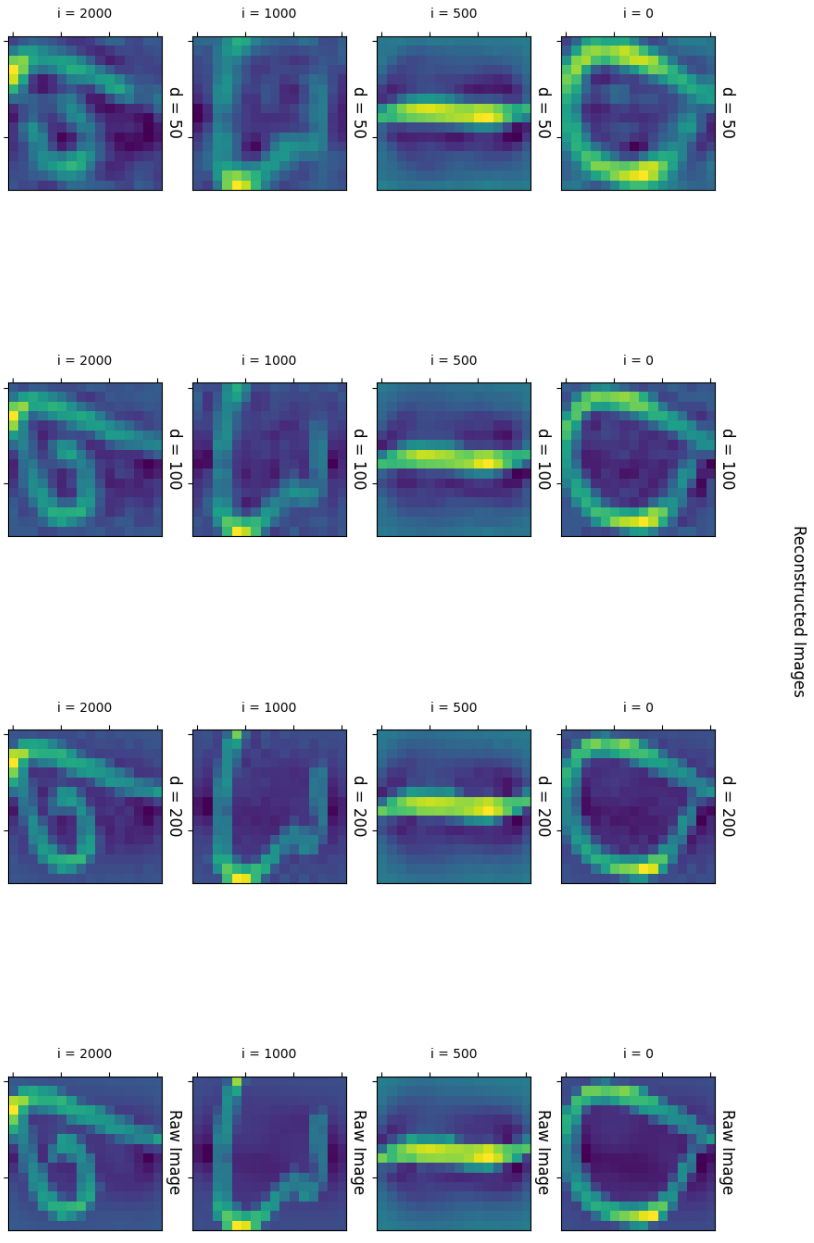
i = 0
i = 500
i = 1000
i = 2000
d = 50
d = 100
d = 200
Raw Image
Reconstructed Images

Figure 4

The reconstructed images present the information retained over the subsequent pca applications per changing new dimensions d. We anticipate that as the dimensionality of new data increases, i.e. new dimensions, the information loss must be lower. Observing the figure above, we clearly see that, for each image, as the new dimension increases, that is moving from d=50 to d=200, the blur within the image decreases and it resembles into raw image much more.

Considering the theoretical aspect, the outcome is as expected. As the number of new features decrease, we are guaranteed to lose information. The outcome does not contradict the target and functionality of feature reduction. In feature reduction, we aim at obtaining a new feature set with a lower dimensionality and retaining as much information as possible simultaneously. Hence, we can conclude that there exists a trade-off between the information retained and the performance of further applications within the new data set after PCA is applied.

4 Conclusion

In the final curricular project in CMPE 462, the assignees are challenged to present the theoretical and practical aspect of two fundamental unsupervised learning approaches: Clustering and Dimensionality Reduction.

Within the context of unsupervised learning approaches, we have overcome our tasks by implementing K-Means Clustering and Principal Component Analysis algorithms. In the project, we are challenged to not only implement successfully, but also to make observations about the expected and actual outcomes derived.

The team BetaZero profoundly believes that we have put our best effort on presenting a comprehensive work. During our study, we have tried to enlarge our perspective so that our work can also aid into the individuals who are interested in any kinds of machine learning applications as much as possible. We would be very glad, if we could have managed to achieve our goal.

Finally, our best regards to dear professor İnci Meliha Baytaş and teaching assistant Rıza Özçelik for their patience, attention, and motivation inspiring us deeply...

5 Appendix

5.1 Analysis of Information Loss

This section provides extra work presenting the information loss level per varying number of new features in the new data set, after the principal component analysis is applied. As stated in **3.3 Visualization of Reconstructed Images**, as the new dimension increases, the information loss decreases.

As we have stated theoretical aspect of principal component analysis, we can reinterpret the dimensionality reduction problem, which can be stated as follows:

$$\min_{G \in \mathbb{R}^{p*d}} ||X^T - G(G^T X^T)||_F^2 \tag{3a}$$

where the matrix G consists of d eigenvectors corresponding to the first d largest eigenvalues of the covariance matrix S, solving the problem above.

The problem above implies the return of transformation matrix G, which minimizes the information loss, computed by the formula expressed in terms of Frobenius norm.

Given the formula, we have computed the information loss as the number of new features change.

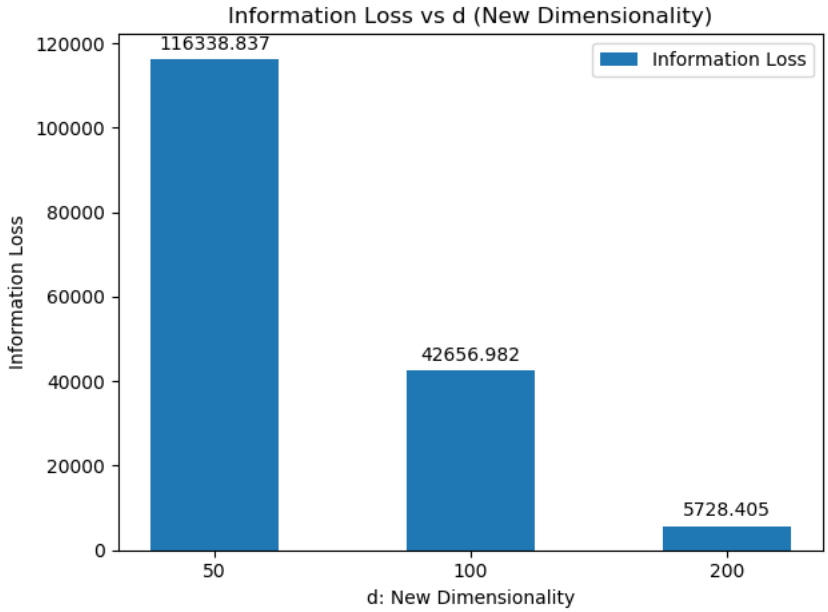

Figure 5: Information Loss vs d

The resulting plot supports our expectations, since the increase in dimensionality must guarantee lower information loss. This section can also be interpreted an auxiliary test supporting the section **3.3 Visualization of Reconstructed Images**, as both of these sections aim at analyzing the information loss caused by varying dimensionality in new data set.

6 References

- Lecture Slides
- `https://matplotlib.org/3.2.1/gallery/lines_bars_and_markers/barchart.html#sphx-glr-gallery-lines-bars-and-markers-barchart-py`