Introduction to Machine Learning

Work2

Course 2020-2021
November 16th, 2020

Students:

Victor Badenas	-	victor.badenas.crespo@estudiantat.upc.edu
Roger Marrugat	-	roger.marrugat@estudiantat.upc.edu
Ronald Rivera	-	ronald.rivera@estudiantat.upc.edu

Contents

Execution Instructions

The configs folder contains the json required to run the main.py script. The json has the following structure:

```
{
    "path": "<relative_path_to_arff_file>",
    "resultsDir": "path_to_results_folder",
    "verbose": <bool>,
    "plotClusters": <bool>,
    "parameters": <nested_objects_with_execution_parameters>
}
```

With that JSON, the `main.py` file can be called using the following command:

```
python main.py -c <path_to_json>
```

This main script will run the pca algorithm and the phases results and it will store the confusion matrices and the metrics extracted as well as the plots in the *resultsDir* directory declared in the json file. It will also show the values needed for the delivery.

Methods

PCA Implementation

The PCA algorithm implemented by us is mainly based on the one explained in the class slides. It is based on the computation of the covariance matrix of the input data and then the most relevant components are selected through a ranking of the eigenvectors and eigenvalues of that matrix.

The class implemented has three main methods:

- fit: main training method for PCA. Fit is responsible for computing the mean of the input data's features, compute the eigenvectors and eigenvalues and extract the n most important components from the array.
- predict: main inference method. It computes the projection of the data into the new vector space of n dimensions.
- fit_transform: Performs the two previously described operations and returns the transformed data.
- inverse_transform: performs the inverse operation to predict.

First, in the fit function, the mean for each feature is calculated. This is needed for both transform and inverse_transform methods. After the mean, the covariance matrix is computed using *np.cov* function from the *numpy* package. Next, the eigenvectors and eigenvalues are computed with the function *np.linalg.eig* also from the numpy package and then are sorted in decreasing order of eigenvalues. The eigenvectors matrix is composed of the eigenvectors as columns in the same order as their eigenvalues. After that, a metric of covariance ratio is computed as the square of each eigenvalue divided by the number of training examples and the sum of all covariances to obtain a percentage array containing the importance of each component of the output vector space for the algorithm.

The inference is done by a simple mean subtraction and the dot product with the eigenvectors matrix. In our case, the data matrix is of shape (n_samples, n_features) and the eigenvectors matrix is of shape (n_features, n_components), so the dot product will result in a matrix of (n_samples, n_components) that will be the data projected into the new vector space.

The reverse procedure has the inverse process to the inference. First, a dot product is performed with the data matrix of shape (n_samples, n_components) and the eigenvalues matrix of shape (n_features, n_components). Note that a transposition of the eigenvalues matrix is needed to obtain a matrix of shape (n_samples, n_features). Then the mean is added to the data.

Results

Our PCA Results

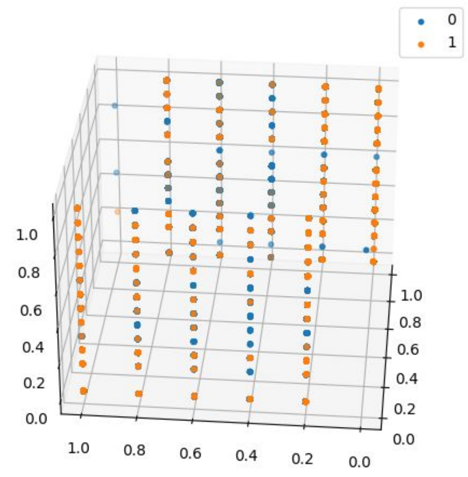

Figure 1: Adult dataset 3 dimensions with most most variance

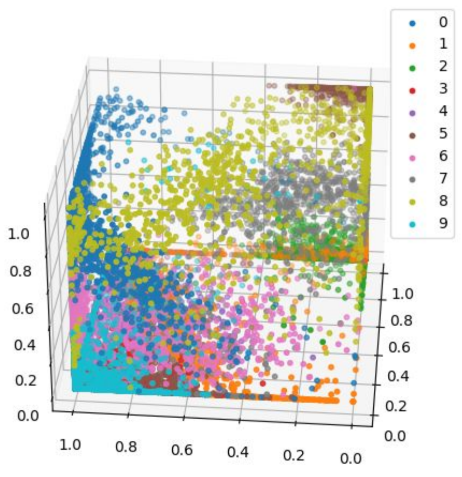

Figure 2: Pen-based dataset 3 dimensions with most variance

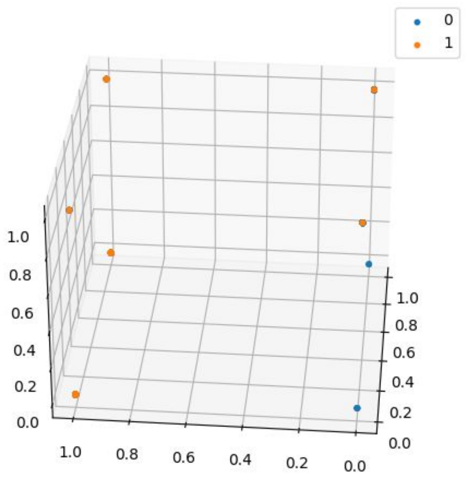

Figure 3: Vote dataset 3 dimensions with most variance

In the three figures above can be seen the 3D plots for the components with more variance of the datasets adult, pen-based and vote respectively. These figures can be seen as a baseline for the selection of the principal components in PCA.

Our PCA Results vs sklearn PCA and Incremental PCA

Vote

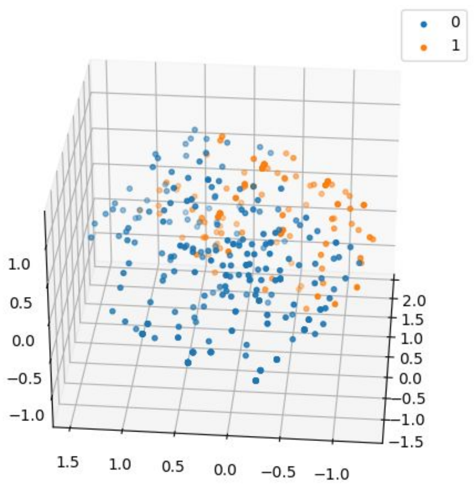

Figure 4: 3D plot of sklearn PCA result with 3 n_Components in Vote dataset

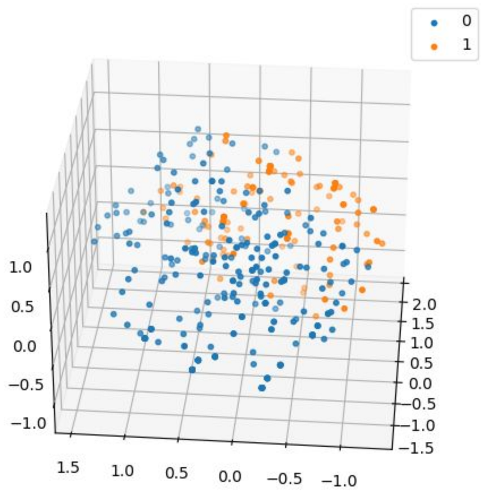

Figure 5: 3D plot of sklearn Incremental PCA result with 3 n_Components in Vote dataset

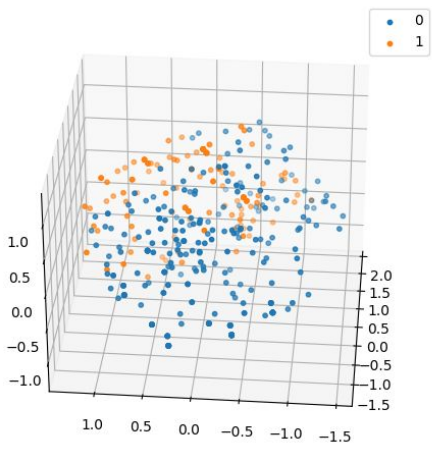

Figure 6: 3D plot of our PCA result with 3 n_Components in Vote dataset

In order to compare the performance of each PCA implementation given the *vote* dataset, we fitted and transformed the same training data with the maximum number of components allowed -- in this case 16, which is the number of features of the corpus --.

For each algorithm we extracted a particular attribute called *explained variance ratio* which provides the percentage of variance that each component is accountable for.

The results of this comparison can be observed in Table 1, where all PCA implementations display similar results, where 86% of the variance can be expressed by the nine first principal components.

Figures 4, 5 and 6 illustrate a visualization of the *vote* dataset using the three first principal components extracted by the different implementations. We can observe how all implementations exhibit the same projection -- even though our PCA provides a “mirrored” version of the other two --.

Pen-Based

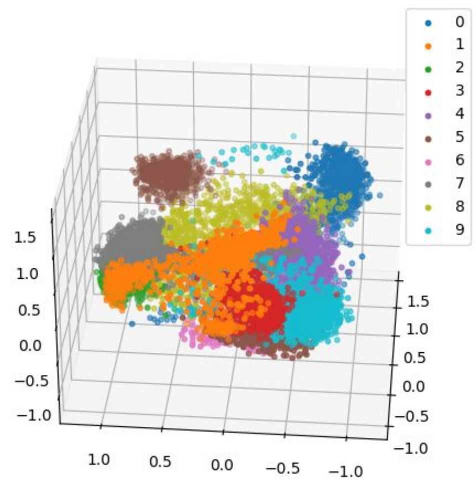

Figure 7: 3D plot of sklearn PCA result with 3 n_Components in Pen-based dataset

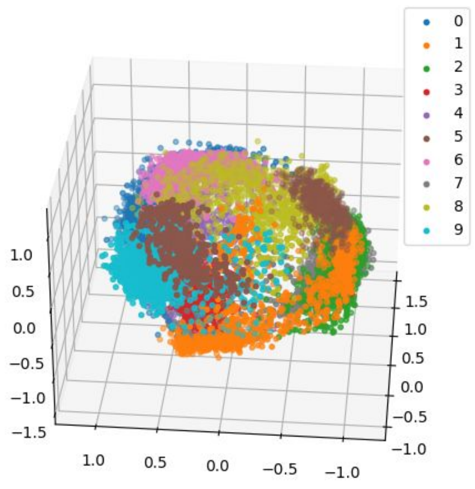

Figure 8: 3D plot of sklearn Incremental PCA result with 3 n_Components in Pen-based dataset

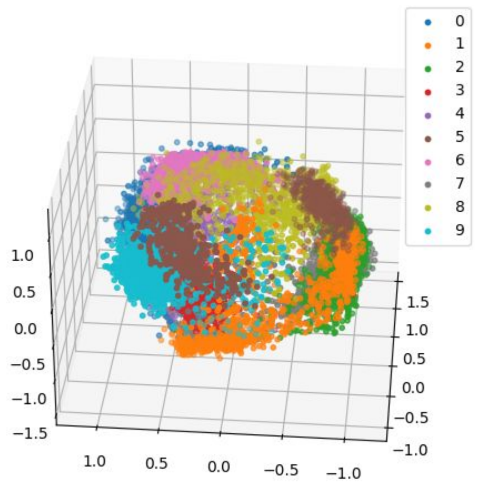

Figure 9: 3D plot of our PCA result with 3 n_Components in Pen-based dataset

Concerning the *Pen-based* dataset, the results for *explained variance ratio* can be observed in Table 2, where 88% of the variance can be expressed by the six first principal components.

Figures 7, 8 and 9 illustrate a visualization of the *Pen-based* dataset using the three first principal components extracted by the different implementations. It’s difficult to distinguish the data points even if they are displayed by class colours because of the high density distributions. However, class 1 printed in orange color has a similar projection even for sklearn's PCA, though it's reversed or "mirrored" while Incremental PCA gives exactly the same projection as our PCA.

Adult

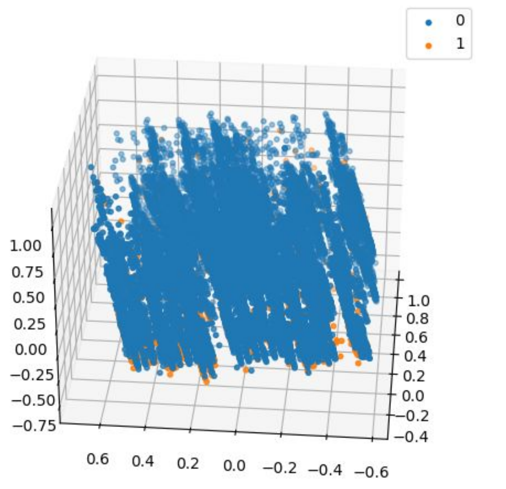

Figure 10: 3D plot of sklearn PCA result with 3 n_Components in Adult dataset

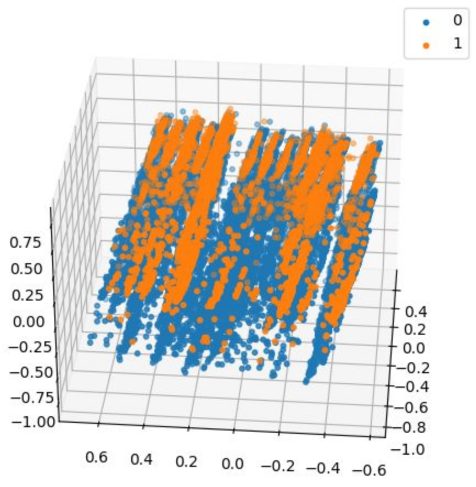

Figure 11: 3D plot of sklearn Incremental PCA result with 3 n_Components in Adult dataset

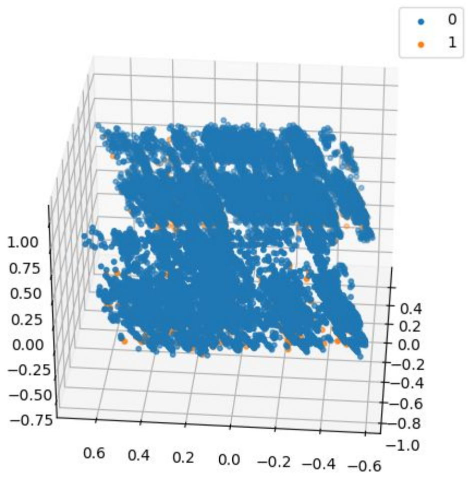

Figure 12: 3D plot of our PCA result with 3 n_Components in Adult dataset

As for the *Adult* dataset, the results for *explained variance ratio* can be observed in Table 3, where 86% of the variance can be expressed by the seven first principal components.

Figures 10, 11 and 12 illustrate a visualization of the *Adult* dataset using the three first principal components extracted by the different implementations. We can observe by the shape of the visualization -- some straight lines can be distinguished -- that the projections are the same but flipped.

In order to carry out this comparison between the three implementations of PCA, we've tried using different sets of parameters, specially concerning the *sklearn* implementations.

One of those parameters was the *whiten* parameter, which tries to remove redundancies in the input and ensure uncorrelated outputs.

Furthermore, we tried tinkering the parameters of PCA Singular Value Decomposition and it's computation, but it did not offer significant improvements.

At this point, we've analysed how our own implementation of PCA performs and which is the optimal number of components for each dataset, that we will use to reduce their dimensionality that should help in the posterior *K-Means* clustering analysis. In order to decide the number of components we determined an heuristic to be used as a threshold where we would get the *n* principal components that represented at least 85% of the total variance.

Clustering with KMeans

The goal of this section is to obtain and compare the clustering metrics obtained in the previous work with the lower dimensional data.

In the following tables we present the metrics obtained on the labels obtained from kmeans. The k means algorithm was run with the parameters that were found to be best in Work1. It also took as data input 4 different sets of data: the n dimensional original data, the reduced data from ourPCA, the reduced data from sklearn's PCA and the reduced data from sklearn's IPCA.

Adult	originalData	myPCA	sklearnPCA	incrementalPCA
davies_bouldin_score	1.6183	1.0059	0.9224	0.9333
adjusted_rand_score	-0.0055	-0.0028	-0.0025	0.0043
completeness_score	0.0478	0.0448	0.0441	0.0441
purity_score	0.7607	0.7607	0.7607	0.7607

Table 4: Results of clustering metrics for Adult dataset

In the previous Table 4 it can be appreciated that the sklearn PCA with 3 components performed way better in terms of davies_bouldin score, however IPCA performed better in terms of adjusted_rand score and the originalData performed marginally better in terms of completeness. All the data had the same purity score.

Pen-Based	originalData	myPCA	sklearnPCA	incrementalPCA
davies_bouldin_score	1.2357	0.7682	0.7680	0.7873
adjusted_rand_score	0.4919	0.4435	0.4434	0.4355
completeness_score	0.6996	0.6593	0.6592	0.6542
purity_score	0.6342	0.5849	0.5849	0.5782

Table 5: Results of clustering metrics for Pen-Based dataset

In the previous Table 5 it can be appreciated that the sklearn PCA with 3 components performed way better in terms of Davies Bouldin Index score, however the originalData performed better in all other scenarios.

Vote	originalData	myPCA	sklearnPCA	incrementalPCA
davies_bouldin_score	2.1688	1.0658	1.0031	0.9802
adjusted_rand_score	0.1863	0.1852	0.2145	0.2152
completeness_score	0.1992	0.2083	0.2220	0.2267
purity_score	0.9080	0.8920	0.9080	0.9034

Table 6: Results of clustering metrics for Vote dataset

In the previous Table 6 it can be appreciated that IPCA with 3 components was the best performer overall having better results, even if small, in everything except in purity.

TSNE and PCA Visualization

In this section we will present some examples of comparisons of t-SNE and PCA visualizations done where the original Labels will be used to show the different clusters.

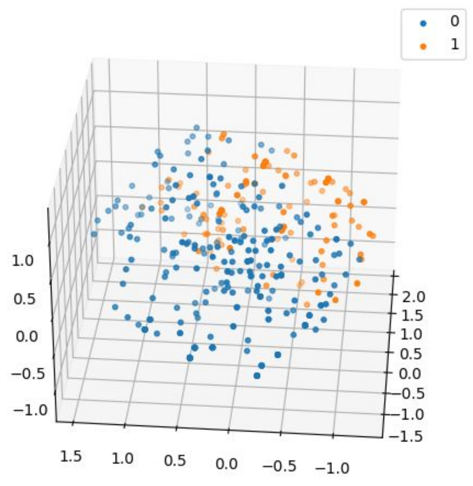

Figure 13: 3D plot of PCA visualization of Vote

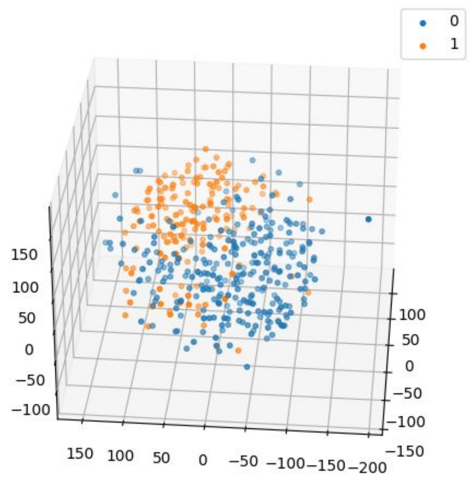

Figure 14: 3D plot of t-SNE visualization of Vote

Figures 13 and 14 represent scatter plots of the vote dataset. In figure 13 we can see the 3D scatter of PCA data and in figure 14 we can see the 3D scatter of t-SNE vectorial representation of the data. It is very apparent that t-SNE tends to perform very dense representation of the data which can be good to visualize smaller datasets like the Vote that is being shown in 13 and 14. Also seems to compact the categorical datasets (specially binary ones) a little more than PCA.

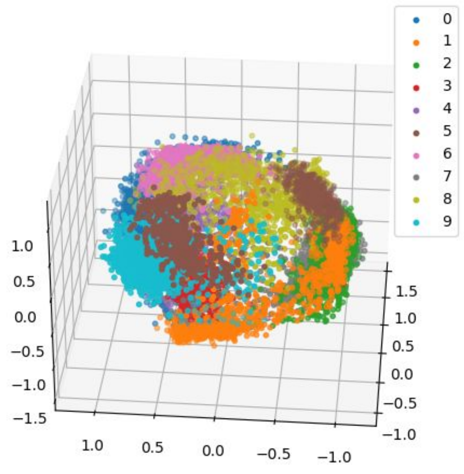

Figure 15: 3D plot of PCA visualization of Pen-Based

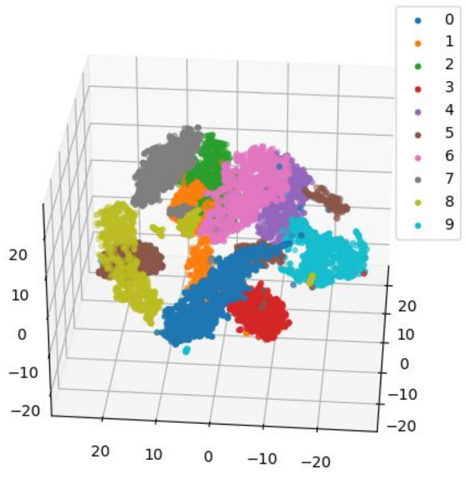

Figure 16: 3D plot of t-SNE visualization of Pen-Based

Figures 15 and 16 represent scatter data from the pen-based dataset. On those figures we can see the 3D Scatter of PCA data and the 3D Scatter of t-SNE data respectively. In the case of numerical datasets, t-SNE tends to make well defined clusters, way more easily differentiable and more separable than the PCA does.

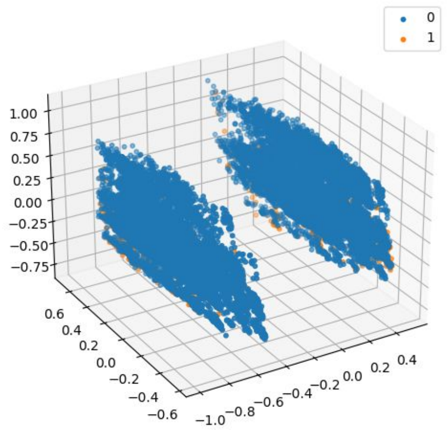

Figure 17: 3D plot of PCA visualization of Adult

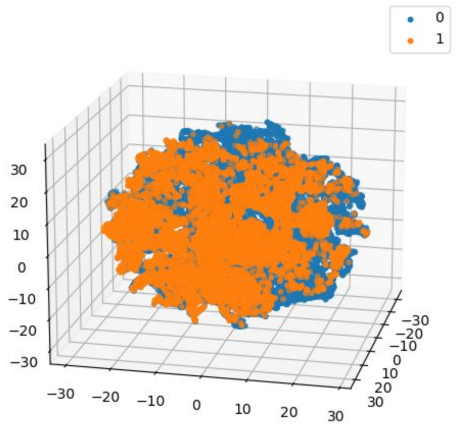

Figure 18: 3D plot of t-SNE visualization of Adult

Finally, for the data in the Adult dataset, we present the 3D scatter plot of the data. On figure 17 we can see the PCA scatter plot and on figure 18 we can see the t-SNE scatter plot. It is apparent that either one of them allows us to get an accurate visualization of the data.

Conclusions

Is PCA giving you more advice for knowing the underlying information in the dataset?

The PCA has given us an idea of the relevance of certain principal components in the dataset. These principal components can be expressed as linear combinations of the original dataset features. In order to know which components matter the most of those linear combinations the eigenvalues and the eigenvectors can be observed. The eigenvalues will provide a ranking of the principal components with highest variability while the eigenvectors will determine the weight of each original feature as well as the direction of the maximal variability in the feature space.

For example, we can observe the features for each dataset with the maximum variance by inspecting the eigenvector of the first principal component.

- *Adult*: ''sex', 'hours-per-week', 'age' and 'race' are the features of maximum variance. Given a balanced dataset (which is expected if the data was obtained from an unbiased survey), 'sex', 'age' and 'race' are expected to have a high variance. 'hours-per-week' could be explained given a reasonable number of unemployed people.

- *Pen-Based*: 'a16', 'a14', 'a12' and 'a15' are the attributes of maximum variance.

- *Vote*: 'el-salvador-aid', 'physician-fee-freeze', 'crime' and 'education-spending' are the dimensions of maximum variance. Therefore, those are the topics of maximum 'disagreement' between the voters in the dataset.

Can you reduce the dimensionality of the data set? In case of an affirmative answer, detail how you do and how many features have been reduced from the original data set.

As it has been seen in the results section of the report, PCA and IPCA are able to extract a smaller number of features that contain most of the variance of the dataset (~85%). The data could be reduced to:

- *Vote*: 9 components.
- *Pen-Based*: 6 components.
- *Adult*: 7 components.

In order to check if the dimensionality of a dataset can be reduced and to infer the number of features, we need to compute a PCA with the maximum number of components and extract the *variance ratio*, which tells us the amount of variance that each principal component of the dataset is able to capture.

Once we know this information, we select a percentage by heuristics (in our case we decided 85%) and we take up to the *nth* principal component that achieves the selected percentage.

Do you obtain the same results from your code to the code in *sklearn*? Explain the similarities and the differences among the two implementations.

Although the underlying algorithms are different, we get the same results -- respecting the difference in the projection of the data into the new space as it could be mirrored out --.

In our PCA algorithm, the dimensionality reduction is based on the principle of the eigenvalues and eigenvectors of the covariance matrix extracted from the data. On the other hand, *sklearn* PCA is based on the singular value decomposition of the centered adjusted data without computing a covariance matrix using the *np.svd* function.

The Principal components can be determined via eigen decomposition of the covariance matrix. The idea is to find a new coordinate system -- that maximizes the variance -- of the eigenvectors for the covariance matrix through rotations.

$$Covariance\ Matrix\ V = \Lambda V -> Covariance\ Matrix = V \Lambda V^{-1}$$

Therefore, our covariance matrix (*m* x *m* features) is decomposed to a matrix of eigenvectors *V* (*m* x *m*) and a diagonal matrix of *m* eigenvalues Λ. By multiplying the eigenvectors *V* of *maximum variance* by the data we can project it into the Principal Components space.

Singular Value Decomposition is another way of decomposing matrices, but in this case into the product of two unitary matrices (*U* and *V**) and a rectangular matrix of singular values (Σ).

$$Covariance\ Matrix = \frac{X^T X}{n-1} = \frac{V \Sigma U^T U \Sigma V^T}{n-1} = V \frac{\Sigma^2}{n-1} V^T = V \frac{\Sigma^2}{n-1} V^{-1}$$

We can compute the covariance matrix by multiplying the adjusted matrix *X* (mean subtracted to the original data) by itself and dividing by the number of samples. Then, we can decompose the matrix *X* with Singular Value decomposition.

$$W \Lambda W^{-1} = V \frac{\Sigma^2}{n-1} V^{-1} -> \Lambda = \frac{\Sigma^2}{n-1}$$

We can easily see the relationship between singular values and eigenvalues. As they are equivalent, we can compute either to apply PCA. In fact, SVD is more efficient, accurate and is able to handle sparse matrices.

Can you explain if you obtain similar clusters with the data set reduced to those obtained in the original data? In addition, have you observed a reduction in time?

The clusters obtained differ as the centers of the clusters (centroids) in *K-Means* are found to be quite different. As displayed in the *Clustering with K-Means* section, the quality of the new clusters have also improved, especially Davies–Bouldin index metric. Not only decreasing the number of features but also increasing the variance (or separation) of the data points should make the clustering analysis easier, as we are reducing the density of the clusters and making the distance of the datapoint to the other centroids larger.

The execution time has been lower as *K-Means* is linearly faster with respect to the number of features in the input data. As we are able to reduce the number of components to almost a half of the original, it can be seen as a huge improvement in execution time.

Which are the similarities and differences in the visualizations obtained using PCA and t-SNE?

Both methods are used for visualization, however t-SNE does not incorporate the dimensionality reduction that PCA does. In PCA, our main focus is to reduce the dimensionality of the data by evaluating the correlation between variances and maximizing the variance present in the data. This would help in the visualization of the data by creating more distinct and separate clusters.

t-SNE was mainly used to visualize data in a 3-D space from data with any number of features. This algorithm follows a nonlinear approach by calculating the probability of similarity of points on a higher dimensionality to a lower dimension. Then it would minimize the difference between the higher and lower dimension probabilities. This method would also help the visualization of the data by compacting it.

In the dataset utilized to evaluate this method we can observe that for visualization purposes, only the t-SNE is better for both the Vote (Categorial) and Pen-based (Numerical) datasets. However, this is not the case for the Adult(Mixed) dataset where we could not obtain a good visual representation with both methods. We assume this is due to the complexity of the data and the mixed categories in the dataset. It is worth mentioning that, if the goal is to maintain the data dimensionality, PCA should be used over t-SNE.

Document Summary

In this document, we present an evaluation of the PCA algorithm for both dimensionality reduction and visualization purposes. For the reduction task, a covariance based PCA algorithm and sklearn's PCA and IPCA algorithm were compared and contrasted. For visualization, the PCA algorithm was compared to TSNE.

For dimensionality reduction, the results obtained by PCA were very promising, especially when clustered afterwards with the *K-M*eans algorithm developed for the previous work of the IML subject.

For visualization, we have perceived a much better representation in categorical and numerical data in the t-SNE algorithm. However, the results were much more promising in the PCA visualization.

Overall, PCA is a very useful algorithm that, when coupled with data with the right properties, can make visualization, clustering and interpretation of the results easier to obtain and more intuitive to interpret.

References

- Scikit-learn.org. 2020. *Sklearn.Decomposition.PCA — Scikit-Learn 0.23.2 Documentation*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html> [Accessed 14 November 2020].
- Scikit-learn.org. 2020. *Sklearn.Decomposition.PCA — Scikit-Learn 0.23.2 Documentation*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html> [Accessed 14 November 2020].
- Scikit-learn.org. 2020. *Sklearn.Manifold.TSNE — Scikit-Learn 0.23.2 Documentation*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html> [Accessed 14 November 2020].
- Es.wikipedia.org. 2020. *Análisis De Componentes Principales*. [online] Available at: <https://es.wikipedia.org/wiki/An%C3%A1lisis_de_componentes_principales> [Accessed 14 November 2020].
- sitiobigdata.com. 2020. *Tsne Con Python, Una Introducción - Sitiobigdata.Com*. [online] Available at: <https://sitiobigdata.com/2019/10/27/una-introduccion-a-tsne-con-python/#> [Accessed 14 November 2020].
- Ufldl.stanford.edu. 2020. *Unsupervised Feature Learning And Deep Learning Tutorial*. [online] Available at: <http://ufldl.stanford.edu/tutorial/unsupervised/PCAWhitening/> [Accessed 14 November 2020].
- data?, W. and SX, c., 2020. *Why PCA Of Data By Means Of SVD Of The Data?*. [online] Cross Validated. Available at: <https://stats.stackexchange.com/questions/79043/why-pca-of-data-by-means-of-svd-of-the-data> [Accessed 14 November 2020].

Annex (Tables)

myPCA	sklearn PCA	IPCA
Variance Ratio		
0.46	**0.46**	**0.46**
0.09	**0.09**	**0.09**
0.07	**0.07**	**0.07**
0.06	**0.06**	**0.06**
0.05	**0.05**	**0.05**
0.04	**0.04**	**0.04**
0.03	**0.03**	**0.03**
0.03	**0.03**	**0.03**
0.03	**0.03**	**0.03**
0.03	0.03	0.03
0.02	0.02	0.02
0.02	0.02	0.02
0.02	0.02	0.02
0.02	0.02	0.02
0.01	0.01	0.01
0.01	0.01	0.01
nº of Components		
9	9	9
Captures Variance		
0.86	0.86	0.86

Table 1: Comparison between PCA implementations, vote dataset

myPCA	sklearn PCA	IPCA
Variance Ratio		
0.28	**0.28**	**0.28**
0.25	**0.25**	**0.25**
0.15	**0.15**	**0.15**
0.09	**0.09**	**0.09**
0.06	**0.06**	**0.06**
0.05	**0.05**	**0.05**
0.03	0.03	0.03
0.03	0.03	0.03
0.02	0.02	0.02
0.01	0.01	0.01
0.01	0.01	0.01
0.01	0.01	0.01
0	0	0
0	0	0
0	0	0
0	0	0
nº of Components		
6	6	6
Captures Variance		
0.88	0.88	0.88

Table 2: Comparison between PCA implementations, pen-based dataset

myPCA	sklearn PCA	IPCA
Variance Ratio		
0.37	**0.37**	**0.37**
0.13	**0.13**	**0.13**
0.1	**0.1**	**0.1**
0.09	**0.09**	**0.09**
0.07	**0.07**	**0.07**
0.06	**0.06**	**0.06**
0.04	**0.04**	**0.04**
0.03	0.03	0.03
0.03	0.03	0.03
0.03	0.03	0.03
0.02	0.02	0.02
0.01	0.01	0.01
0.01	0.01	0.01
0.01	0.01	0.01
nº of Components		
7	7	7
Captures Variance		
0.86	0.86	0.86

Table 3: Comparison between PCA implementations, adult dataset