Analysis of Dimensionality Reduction Methods on ML Models

Group-16: Sreenidhi, Roshan Reddy, Jayendra, Ranjit

CSE 512 - Stony Brook University

Abstract

It gets very cumbersome to deal with huge datasets to train ML models. Datasets may have a huge number of important features with high collinearity. In such cases, it is usually suggested to compress the data such that it contains all the features in much lesser variables. We aim to do an exploratory data analysis of a dataset and observe the trends in the data, like the covariance between the different attributes of the data. We will then employ some dimensionality reduction techniques and study their effectiveness on different ML models.

1 Introduction

Since the advent of the field of machine learning, we have been able to develop a wide range of intelligent models. There are many powerful machine learning models out there which can be used for different tasks. However, as powerful as these models are. It is still vital to preprocess the data very well to ensure the best possible outcome for these models. The preprocess technique we employ will totally depend on the kind of data. Data usually comes with many issues, like missing data values, dimensionality issues, and the covariance between variables, all of which may impact the performance of the models in many ways. It's not only an issue with data, but can be an issue with the model as well. Different models may require being fed with a different method of preprocessing. Overall the best performance is got with the right preprocessing for the right model for a particular dataset. In this project, we want to address how the issue of the high dimensionality of data can impact a model's performance. We would like to preprocess the data with different dimensionality reduction techniques, specifically, Principal Component Analysis(PCA)[1], Linear Discriminant Analysis(LDA)[2], Kernal PCA[3], and Uniform Manifold Approximation and Projection(UMAP)[4]. We would like to access the performance of some widely used ML models and study their performance after employing these different dimensionality reduction techniques using some standard performance metrics. We are using the 20 news groups dataset, for this purpose.

2 Literature Survey

We survey widely used dimensionality reduction techniques in this section. Many linear and non-linear methods were explored to address the problem of high dimensions.

By employing the CTG's pathological categorization, the authors of [5] suggest a novel technique for choosing a subset of optimum functions that will boost accuracy. Improved computing speed and accuracy have been achieved by doing feature selection with PCA as a pre-processing stage in machine learning. We can observe that PCA has ruled over the past in dimension reduction space but there are limitations to it such as feature interpretability, and assumption of the linear relationship between variables. Whereas, LDA works based on supervised learning which maximizes group separation between categories which would be a great property for feature representation.

We have looked at linear methods so far. There are also non-linear methods like Kernel PCA and UMAP that address the dimensionality reduction problem. The authors in [4] have thoroughly analyzed the performance of the UMAP technique specifically over clustering algorithms. They have used multiple standard datasets over some widely used clustering algorithms like KMeans and GMM and compared their performance with and without using the UMAP as a preprocessing technique. We can clearly see the improvements in both the accuracy and execution time of these clustering methods using UMAP as a preprocessing step. For instance, a major improvement is seen over the MNIST dataset using HDBSCAN clustering with runtime reducing to 5 seconds from 26 minutes and accuracy improving to 77.65% from 27.65%.

Overall we aim to study the performance and accuracy along with other metrics by comparing these widely used dimensionality reduction techniques. We also want to highlight key differences in using linear and non-linear techniques. We want to extend upon the key ideas presented in the referred papers, by including more metrics to tell about how well the data reduction was done by the algorithms, instead of simply using the reduced data to train models and comparing accuracies. With respect to [5], we want to extend the work done by studying performance over some standard classification models.

3 Methodology

In this section, we will explore both linear and non-linear dimensionality reduction methods we have investigated linear methods like Principal Component Analysis and Linear Discriminant Analysis and non-linear methods like Kernel PCA and Uniform Manifold Approximation and Projection (UMAP) in this project. We also mention the experimental setup and dataset used for the project.

3.1 Principal Component Analysis(PCA)

PCA[1] is a dimensionality-reduction method that is used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one while preserving as much information as possible. As smaller data sets are easier to analyze and visualize faster for ML algorithms. In this case decrease in a small amount of accuracy can still be an added advantage when compared to the time required to analyze the large data sets.

Algorithm

- Standardize the dataset to analyze the contribution of each variable equally and transform the variables using the following formula
  $z = \frac{value - mean}{standarddeviation}$

- Compute the covariance matrix to know which pair of variables are correlated with each other to categorize them easily.

$$covariancematrix = \begin{bmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{bmatrix}$$

- Compute the eigenvector components and define eigenvalues in descending order to get the list of principal components which represent the direction of data. Finally, these principal components form a line of new axes for easier data evaluation and the differences between the observations can be easily monitored.

- Recast the data along the principal component axes. This step aims at the reorientation of data from their original axes to the ones calculated from the Principal components.

3.2 Linear Discriminant Analysis(LDA)

LDA [2] is another dimensionality reduction technique for the pre-processing stage in data mining and machine learning applications that is also a reliable classification method in linear discriminant analysis. The initial number of features is reduced by LDA to C—1 features, where C is the number of classes. This is accomplished by minimizing the variance and maximizing the distances between the means of all classes.
Algorithm

- Initially, a d-dimensional mean vector is created for all C classes in the dataset.

- Scatter matrices are computed and then their eigenvectors $(E_1, E_2, ...E_C)$ and their eigenvalues $(\psi_1, \psi_2, ...\psi_C)$ are computed.

- Then we sort the eigenvectors in descending order of their eigenvalues, among which we choose k eigenvectors that have maximum eigenvalues to create a $C * i$ matrix.

- This matrix is used to transform the input data in the new subspace which contains $C - 1$ dimensions.

3.3 Kernel PCA

Kernel PCA [3] is an extension of PCA used to separate nonlinear data by making use of kernels. The main goal is to project the inseparable data onto higher dimensional space where it becomes linearly separable. This is done by constructing a kernel matrix of the dataset. Solving to get the eigenvectors of the matrix we extract the principal components of data points and computer the projections onto the eigenvectors.
Algorithm

- Choose Kernel function k(x,y) and compute kernel matrix K with $K_{ij} = k(x_i, x_j)$

- Center the kernel matrix $K_{centered} = (I - 1_n)K(I - 1_n)$ where n is number of data points.

- Find eigenvectors and eigenvalues of the centered kernel matrix. Multiply each eigenvector by the square root of the respective eigenvalue and thus principal components are computed.

3.4 Uniform Manifold Approximation and Projection (UMAP)

Uniform manifold approximation and projection[4] is a dimension reduction method that may also be used for generic non-linear dimension reduction. UMAP relies on the Riemannian manifold. The embedding manifold is found by searching for a fuzzy topological structure of low-dimensional projection of the data.
Algorithm

- To construct the fuzzy topological structure UMAP represents the data in high dimensional. The high-dimensional graph is a weighted graph with the edge representing the likelihood that they are connected. UMAP uses exponential probability distribution to compute the similarity between points.
$$p_{i|j} = exp(-\frac{d(x_i,x_j)-\rho_i}{\sigma_i})$$
where $d(x_i, x_j)$ = distance between i^{th} and j^{th} points. ρ is the distance between i^{th} data point and its first nearest neighbor.

- In cases where the weight of the graph between i and j nodes is not equal to the weight between j and i nodes. UMAP uses a symmetrization of the high-dimensional probability:
$$p_{ij} = p_{i|j} + p_{j|i} - p_{i|j}p_{j|i}$$

- As the constructed graph is a likelihood graph, and UMAP needs to specify k the number of nearest neighbors:
$$k = 2^{\Sigma_i p_{ij}}$$

- UMAP constructs and optimizes the layout of a low-dimensional analogue to be as similar as possible. For modeling distance in low dimensions, UMAP uses a probability measure similar to Student t-distribution:
$$q_{ij} = (1 + a(y_i - y_j)^{2b})^{-1}$$
where a ≈ 1.93 and b ≈ 0.79 for default UMAP.

- UMAP uses binary cross-entropy (CE) as a cost function due to its capability of capturing the global data structure:
$$CE(P,Q) = \sum_i \sum_j [p_{ij}log(\frac{p_{ij}}{q_{ij}}) + (1 - p_{ij})log(\frac{1-p_{ij}}{1-q_{ij}})]$$
Where P is the probabilistic similarity of the high dimensional data points, and Q is for the low dimensional data points.

The derivative of the cross-entropy is used to update the coordination of the low-dimensional data points to optimize the projection space until the convergence.

3.5 Dataset

The 20 newsgroups dataset is a collection of around 18000 news posts divided into 20 newsgroups or categories. It has been widely used for textual analysis tasks. Many machine-learning techniques have been experimented using this dataset for tasks like classification and clustering. The dataset is structured as 20 different text files, one file for each newsgroup. Each of these files contains all the corresponding messages belonging to that newsgroup. Each message has an ID attached to it. The list.csv file comprises of all the mappings from ID to its newsgroup. Electronics, politics, sports, and medicine are some examples of the 20 newsgroups.

3.6 Experimental Setup

This project examines how dimensionality reduction methods affect how well Machine Learning models perform on the 20 newsgroups dataset. This is done in the following steps:

- We perform feature engineering techniques such as normalization, and conversion of categorical data to numeric data. To normalize the input dataset, the min-max standard scaler normalization method is used. After the vectorization, a frequency threshold of 0.1% is applied to retain only the highly frequent words.
- We experiment with the normalized dataset over different ML algorithms like SVM, KMean Clustering, and Random Forest in this project.
- We perform linear dimensionality reduction techniques like PCA and LDA, on the normalized dataset and repeat the experimentation over the ML algorithms and check the performance of the models over the preprocessing technique.
- We also perform non-linear dimensionality reduction techniques like Kernel PCA, Uniform Manifold Approximation, and Projection (UMAP) on the normalized dataset. The resultant dataset is then experimented on with the above-mentioned ML algorithms.
- We analyze the results obtained by the ML algorithms without dimensionality reduction and the results obtained by PCA, LDA, Kernel PCA, and UMAP on ML algorithms. The effect of dimensionality reduction on the performance of ML algorithms is investigated.
- The performance of these classifiers is then evaluated on the metrics like Precision, Recall, Accuracy, Purity Score, Homogeneity Score, and Completeness Score.

4 Results and Discussions

After the initial preprocessing, the dataset is employed with linear dimensionality reduction techniques namely PCA and LDA. For PCA, reduced components of 2, 512, 1024, and 2048 with explained variance ratios of 0.4%, 21%, 34%, and 54.6% respectively have been considered and tested. As expected, the highest accuracy is achieved with 512 components. It is known, LDA can only reduce the number of features to at most $n-1$ features, where n is the number of classes. So, the feature dimensions of 2, 15, 17, and 19 components have been considered for LDA. The average results computed after 5 trail runs and tested over Random Forest, SVM, KMeans are presented in Table 1 and Table 2 for PCA and LDA respectively.

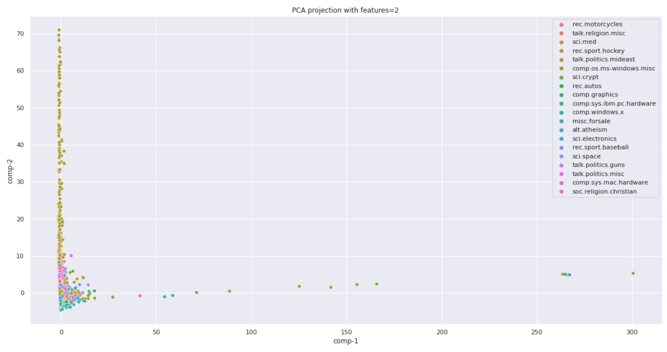

Figure 1: PCA

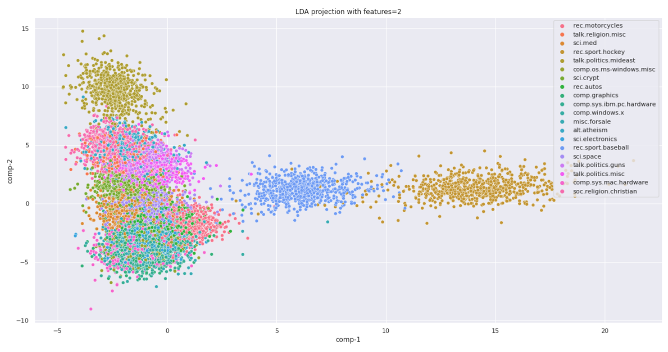

Figure 2: LDA

Table 1: 20 newsgroups with PCA

Random Forest				
Features	Accuracy	Precision	Recall	Computation Time
9K	68.6%	0.682	0.686	102.42s
2	12.2% ±0.277%	0.120	0.122	3.46s
512	**68.8% ±0.245%**	**0.688**	**0.688**	43.68s
1024	67.7%±0.135%	0.680	0.677	62.92s
2048	68.0%±0.732%	0.670	0.680	92.81s
SVM				
Features	Accuracy	Precision	Recall	Computation Time
9K	72.6%	0.747	0.726	225.67s
2	15.2%±0.044%	0.134	0.152	10.86s
512	74.3%±0.162%	0.749	0.743	49.90s
1024	75.1%±0.073%	0.759	0.751	217.51s
2048	**75.6% ±0.196%**	**0.766**	**0.756**	349.28s
KMeans				
Features	Purity Score	Homogeneity Score	Completeness Score	Computation Time
9K	**0.775**	0.188	0.502	94.92s
2	0.154±0.001	0.148	0.227	1.76s
512	0.152±0.014	0.218	0.499	13.16s
1024	0.174±0.008	**0.227**	**0.510**	27.26s
2048	0.215±0.039	0.106	0.446	50.37s

Table 2: 20 newsgroups with LDA

Random Forest				
Features	Accuracy	Precision	Recall	Computation Time
9K	**68.6%**	**0.682**	**0.686**	102.42s
2	17.8% ±0.5%	0.195	0.180	2.74s
15	57.0% ±0.0%	0.552	0.558	5.05s
17	57.6% ±0.9%	0.564	0.564	7.47s
19	57.2% ±0.45%	0.558	0.558	5.83s
SVM				
Features	Accuracy	Precision	Recall	Computation Time
9K	**72.6%**	**0.747**	**0.726**	225.67s
2	16.8%	0.198	0.173	3.50s
15	60.0%	0.590	0.594	0.54s
17	61.6%	0.606	0.606	0.52s
19	64.0%	0.630	0.630	0.46s
KMeans				
Features	Purity Score	Homogeneity Score	Completeness Score	Computation Time
9K	**0.775**	0.188	0.502	94.92s
2	0.199	0.165	0.258	0.96s
15	0.320	**0.282**	0.484	0.98s
17	0.319	0.279	0.496	1.13s
19	0.312	**0.282**	**0.510**	0.79s

We further analyzed the nonlinear dimensionality reduction techniques namely Kernel PCA and UMAP. Kernel PCA has been tested with 2, 512, 1024, and 2048 components, while UMap was tested with 2, 32, 64, and 128 components. The components are different for UMap, as we were getting a peak around 32 features, and for dimensions greater or lesser, the accuracy is reduced. The kernel PCA has shown a stable and good performance over UMAP. UMAP results varied at each run with a very high standard deviation. However, UMAP gave promising results over kernel PCA in visualizing the dataset, as well as for KMeans clustering. The average results of the reduced features dataset (using Kernel PCA and UMAP) over 5 trial runs tested over Random Forest, SVM, and KMeans are presented in Table 3 and Table 4 for Kernel PCA and UMap respectively.

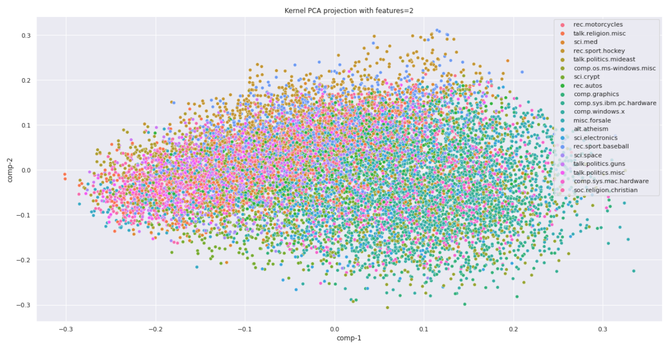

Figure 3: Kernel PCA

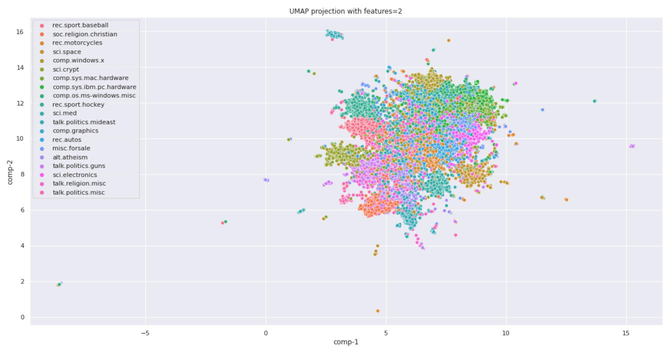

Figure 4: UMAP

The reduced feature set using PCA, LDA, Kernel PCA, and UMAP with 2 components has been visualized in Figures(1, 2, 3, 4). In Kernel PCA, we have used both *sigmoid*, *rbf* and *polynomial* as kernel functions and compared the results after testing the reduced feature dataset over RandomForest, SVM, and Kmeans. From the results, we observed that the sigmoid kernel function has better results when compared to the other two kernel functions. From the observations, UMAP has shown a good separation and clear visibility of clusters of categories. Evidently, it has shown a good performance with KMeans clustering, though it had lesser scores with SVMs, when compared with other reduction algorithms.

Results of different dimensionality reduction techniques have been presented and analyzed over the tradeoff between accuracy vs computation time. It has been observed over 20 newsgroups

Table 3: 20 newsgroups with Kernal PCA

Random Forest				
Features	**Accuracy**	**Precision**	**Recall**	**Computation Time**
9K	68.6%	0.682	0.686	102.42s
2	14.2±0.31%	0.142	0.142	0.040s
512	**68.8 ±0.43**%	**0.696**	**0.688**	0.732s
1024	68.2±0.48%	0.688	0.682	1.184s
2084	66.3±0.39%	0.674	0.663	1.596s
SVM				
Features	**Accuracy**	**Precision**	**Recall**	**Computation Time**
9K	72.6%	0.747	0.726	225.67s
2	18.9%±0	0.192	0.189	0.176s
512	75.2%±0	0.76027	0.751	1.389s
1024	**76.4**%±0	**0.7755**	**0.764**	4.802s
2084	76.1%±0	0.774	0.760	9.257s
KMeans				
Features	**Purity Score**	**Homogeneity Score**	**Completeness Score**	**Computation Time**
9K	**0.775**	0.188	0.502	94.92s
2	0.184±0.012	0.183	0.224	0.026s
512	0.203±0.025	0.225	0.508	0.251s
1024	0.306±0.0489	0.314	0.530	0.544s
2048	0.334±0.04	**0.332**	**0.542**	1.129s

Table 4: 20 newsgroups with UMAP

Random Forest				
Features	**Accuracy**	**Precision**	**Recall**	**Computation Time**
9K	**68.6**%	**0.682**	**0.686**	102.42s
2	52.52±1.08%	0.524	0.525	3.144s
32	60.64±1.49%	0.622	0.606	10.763s
64	57.58±0.97%	0.607	0.576	17.178s
128	43.15±2.72%	0.534	0.432	27.175s
SVM				
Features	**Accuracy**	**Precision**	**Recall**	**Computation Time**
9K	**72.6**%	**0.747**	**0.726**	225.67s
2	55.03±1.25%	0.558	0.551	10.585s
32	60.99±0.61%	0.618	0.61	10.83s
64	56.41±1.84%	0.589	0.564	14.898s
128	46.08±1.85%	0.571	0.461	21.597s
KMeans				
Features	**Purity Score**	**Homogeneity Score**	**Completeness Score**	**Computation Time**
9K	**0.775**	0.188	0.502	94.92s
2	0.539±0.0056	0.419	0.459	1.65s
32	**0.619±0.013**	0.444	0.502	2.391s
64	0.607±0.0147	**0.445**	**0.523**	3.253s
128	0.55±0.0418	0.424	0.508	4.762s

dataset. This dataset contains over 100K features, which have been further reduced based on the frequency threshold to 9K features. After the preprocessing, the feature set is reduced using linear methods (PCA and LDA) and non-linear methods (Kernel PCA and UMAP). These are further trained over ML models(Random Forest, SVM, and KMeans). From the results, it is observed that between the linear methods, PCA performed better than LDA. Among the nonlinear methods, Kernel PCA performed better than UMAP with Random forest and SVM, but UMap outperformed Kernal PCA with KMeans. UMAP is also better utilized to project the data on to lower dimensions. UMAP works better for the visualization of the vast featured datasets.

It is usually expected to get a lower accuracy upon dimension reduction, as information is lost. However, it can help reduce overfitting data. For the Random Forest model, we can see, how reducing the features is helping reduce the computation time while maintaining the accuracy, when PCA and Kernal PCA are used. In the case of SVM, the PCA and Kernal PCA algorithms have not only reduced computation time but helped outperform SVM without feature reduction by a large margin. UMap though not close to the actual purity, has clearly outperformed the other algorithms while clustering with decent purity. It is also ensuring better homogeneity than the normal KMeans model. From the scatter plots, we can expect UMap to perform well with clustering models. Finally, though LDA seems to have low accuracy than the other algorithms with all the models, we need to consider this performance based on the number of features trained on. It has to be noticed that for the number of features LDA is reducing the data to, it is giving a great performance compared to other algorithms with comparable features.

5 Conclusion and Future Work

The dimensionality reduction techniques show an improvement in the computation of building a model by a large fold. This can be utilized well in cases where time is precedence and small error is admissible, for instance, improving the recommendation time in large user base websites like Netflix, Amazon, etc. But, faster computation is not always the case. Predominant fields like banking and medical sciences depend largely on accuracy over computation time. In those cases, even a drop of 0.1% cannot be compromised. In the future, the effectiveness of these dimensionality reduction techniques can be approved over an insignificant change of accuracy over unseen data. Additionally, these techniques can be further analyzed over complex algorithms like Deep Neural

Networks, Convolutional Neural Networks, Recurrent Neural Networks, etc. Moreover, in the case of UMap, we could not consider large final features sizes like PCA or Kernal PCA, since UMap requires more computational power. We anticipate UMap to give better results with KMeans clustering with higher number of features. UMap can be further experimented with higher output dimensions.

References

[1] Jolliffe Ian T. and Cadima Jorge. "Principal component analysis: a review and recent developments". In: *The Royal Society* 374 (2016). DOI: https://doi.org/10.1098/rsta.2015.0202.

[2] Alaa Tharwat et al. "Linear discriminant analysis: A detailed tutorial". In: *AI Commun.* 30 (2017), pp. 169–190. DOI: 10.3233/AIC-170729.

[3] Bernhard Schölkopf, Alexander Smola, and Klaus Robert Müller. "Kernel principal component analysis". English. In: Publisher Copyright: © Springer-Verlag Berlin Heidelberg 1997.; 7th International Conference on Artificial Neural Networks, ICANN 1997 ; Conference date: 08-10-1997 Through 10-10-1997. 1997, pp. 583–588. ISBN: 3540636315. DOI: 10.1007/bfb0020217.

[4] Mebarka Allaoui, Mohammed Lamine Kherfi, and Abdelhakim Cheriet. "Considerably Improving Clustering Algorithms Using UMAP Dimensionality Reduction Technique: A Comparative Study". In: *Image and Signal Processing*. Ed. by Abderrahim El Moataz et al. Cham: Springer International Publishing, 2020, pp. 317–325. ISBN: 978-3-030-51935-3. DOI: 10.1007/978-3-030-51935-3_34.

[5] G. Thippa Reddy et al. "Analysis of Dimensionality Reduction Techniques on Big Data". In: *IEEE Access* 8 (2020), pp. 54776–54788. DOI: 10.1109/ACCESS.2020.2980942.