Original Research Article

Exploring the Role of Dimensionality Reduction in Enhancing Machine Learning Algorithm Performance

Abstract

In this study, we delve into the pivotal role of dimension reduction techniques in influencing the performance of machine learning algorithms for heart disease prediction. Through a comprehensive exploration of a dataset encompassing crucial features such as age, sex, chest pain type, blood pressure, cholesterol levels, and more, we investigate the impact of different techniques—namely, Principal Component Analysis (PCA), Kernel Principal Component Analysis (KPCA), and Linear Discriminant Analysis (LDA) on classification algorithm effectiveness. The classification algorithms considered were Logistic Regression, Support Vector Machine (SVM), k-Nearest Neighbors (KNN), Naive Bayes, and Deep Neural Network (DNN). We used K-fold cross validation to train and validate the classification algorithms. The performance of these algorithms was assessed using a range of key metrics including accuracy, F1-score, precision, recall, and specificity. The results reveals that Linear Discriminant Analysis consistently emerged as a potent method, remarkably enhancing algorithm performance across all assessed metrics. We also identified Naive Bayes and Logistic Regression as standout algorithms, demonstrating remarkable resilience and reliability across diverse scenarios. These findings collectively shed light on the intricate interplay between dimension reduction techniques and algorithm selection, offering critical insights for crafting more accurate and robust strategies in the prediction of heart disease.

1 Introduction

Machine learning represents a dynamic field of computational methods created to simulate human intelligence through learning from the surrounding context. Machine learning techniques have demonstrated their effectiveness across a wide array of domains, encompassing pattern recognition, computer vision, aerospace engineering, finance, entertainment, computational biology, as well as applications within the realms of biomedical and medical fields [1]. In the rapidly evolving landscape of machine learning, the explosion of data availability has brought both opportunities and challenges. While vast datasets hold the potential to unveil hidden insights and patterns, they also introduce complexities that can hinder the performance of machine learning algorithms. One critical challenge is the curse of dimensionality, where high-dimensional data spaces can lead to increased computational demands, overfitting, and reduced generalization ability [2].

The "curse of dimensionality" has garnered significant attention in basic research due to its implications for increased data storage and computing costs [2]. Dimensionality reduction refers to the process of transforming high-dimensional data into a lower-dimensional representation while

preserving essential characteristics. By reducing the number of features or variables, dimensionality reduction methods aim to simplify the data, improve computational efficiency, and enhance the interpretability of models [3, 4]. These techniques encompass both feature selection, which identifies the most informative attributes, and feature extraction, which constructs new features that capture the essence of the original data [5]. This field is particularly challenging and has become a focal point for scholars due to its complexity. Finding effective ways to reduce feature dimensions while preserving essential information has become a hot and difficult area of research within these domains [6]. Over the years, various techniques have been proposed and studied to enhance the performance of machine learning algorithms by reducing the data's dimensionality while preserving essential information [7].The utilization of Machine Learning classifier models in the medical sector is steadily increasing [8]. These models have demonstrated significant utility in effectively diagnosing diverse medical and clinical datasets [9].

The authors in [10] investigated impact of dimensionality reduction techniques on machine learning models for cancer prediction using gene expression data was explored. Principal Component Analysis (PCA), PCA with a kernel, and autoencoder were employed to reduce RNA sequencing data's dimensionality. Neural network and support vector machine classifiers were trained and tested using original, dimensionally reduced, and cancer-relevant data. The results demonstrated that dimensionality reduction enhances classifier performance, with the autoencoder outperforming PCA and PCA with a kernel. This study highlights the potential of dimensionality reduction in improving machine learning models on high-dimensional data in cancer research .

The authors in [11] investigated the potential of machine learning dimensionality reduction methods, including principal component analysis (PCA), kernel PCA (KPCA) with polynomial kernel function, latent semantic analysis (LSA), Gaussian random projection (GRP), sparse random projection (SRP), multidimensional scaling (MDS), Isomap, and locally linear embedding (LLE), to enhance risk stratification models for chest pain patients in the emergency department (ED). The data of 795 patients presenting with chest pain at Singapore General Hospital between September 2010 and July 2015 were analyzed. These methods were used in combination with logistic regression to create prediction models. The multidimensional scaling algorithm demonstrated the best performance with an AUC of 0.901. While the models outperformed existing clinical scores in ROC analysis, the improvement in predicting 30-day major adverse cardiac events (MACE) was only marginal. Moreover, the black box nature of these models made them challenging to interpret in clinical practice. Further investigation is needed to explore their practical clinical implementation.

The authors in [12] focused on a study aimed to enhance the prediction of Diabetic Retinopathy, a significant cause of global vision loss, by employing machine learning techniques. It undertook a comprehensive approach by addressing data preprocessing, dimensionality reduction, and classifier selection. The researchers collected a Diabetic Retinopathy dataset from the UCI repository and initially normalized it using the Standardscalar technique. Principal Component Analysis (PCA) was then applied to extract essential features, followed by the implementation of the Firefly algorithm for further dimensionality reduction. Subsequently, a Deep Neural Network Model was utilized for disease classification. This approach sought to improve prediction accuracy, accounting for often overlooked data preprocessing and dimensionality reduction aspects. However, the study's scope might be limited by the specific dataset employed, potentially affecting generalizability, and the performance of the Firefly algorithm and the chosen classifier may vary across diverse datasets or scenarios.

Machine learning algorithms play a vital role in diverse fields by enabling predictions and pattern discovery from vast datasets. However, the curse of dimensionality presents challenges such as increased computational complexity and potential overfitting. Dimensionality reduction techniques have emerged as effective tools to mitigate these issues. This project aims to investigate the role of dimensionality reduction in enhancing the performance of machine learning algorithms using various dimension reduction methods such as PCA, Kernel PCA and LDA. This proposed project will deploy popular machine learning classification models for the original data and reduced data and compare the perform of different machine algorithms developed from each of the data. The data used will involve a discrete response variable (whether an individual has heart disease or not) for the purpose

of supervised learning and 13 features

The main objective of this research paper is to investigate and assess the impact of dimensionality reduction techniques on the performance of machine learning classification algorithms for heart disease. In pursuit of this overarching goal, the study encompasses several specific objectives. These objectives include conducting dimensionality reduction on heart disease classification data using techniques like PCA, kernel PCA, and LDA. Additionally, the study involves the training and validation of machine learning classification algorithms on both the original and reduced datasets. Furthermore, the evaluation of machine learning classification algorithm performance, using metrics such as precision, recall, F1-score, sensitivity, specificity, and accuracy, on both the original and reduced data, is a critical aspect of this research. Ultimately, the study aims to demonstrate that the employed dimensionality reduction techniques do not significantly degrade the performance of machine learning classification algorithms in the context of heart disease classification.

In this work we will delve into the pivotal role of dimensionality reduction techniques in enhancing the performance of machine learning algorithms. We will focus on prediction of heart disease from a selected dataset using various machine learning classfiers such as logistic regression, kernel support vector machines, Naive Bayes, K nearest neighbours and deep neural networks. By mitigating the curse of dimensionality, we will explore how these machine learning algorithms deployed to reduced data will contribute to more accurate predictions, faster training times, and improved generalization to new data. We will examine a variety of dimensionality reduction methods, ranging from classical linear techniques such as Principal Component Analysis (PCA), kernel PCA and linear discriminant analysis.

2 Data and Methods

2.1 Data

The dataset used consists of 14 features encompassing patient attributes such as age, sex, blood pressure, cholesterol levels, and exercise test results, including variables like chest pain type, EKG results, and thallium stress test outcomes. The response variable, "Heart Disease," indicates the presence (1) or absence (0) of heart disease. Given the binary classification nature of the problem, supervised machine learning classification algorithms were used predicting heart disease presence. Table (1) shows the description of the variables.

Table 1: Description of Variables

Variable Name	**Description**	**Type**
Age	Age of the patient	Continuous
Sex	Sex of the patient (0 = Female, 1 = Male)	Discrete
Chest pain type	Type of chest pain experienced	Discrete
BP	Blood Pressure	Continuous
Cholesterol	Serum Cholesterol levels	Continuous
FBS over 120	Fasting Blood Sugar $>$ 120 mg/dL (1 = True, 0 = False)	Discrete
EKG results	Electrocardiogram results	Discrete
Max HR	Maximum Heart Rate achieved during exercise	Continuous
Exercise angina	Exercise-induced angina (1 = Yes, 0 = No)	Discrete
ST depression	ST segment depression induced by exercise	Continuous
Slope of ST	Slope of the ST segment during exercise	Discrete
Number of vessels fluro	Number of major vessels colored by fluoroscopy	Discrete
Thallium	Thallium stress test results	Discrete
Heart Disease	Presence of heart disease (1 = Yes, 0 = No)	Discrete

2.2 Dimension Reduction Techniques

In this project we explored three dimension reduction techniques. Principal Component Analysis, Kernel Principal Component Analysis and Linear Discriminant Analysis.

2.3 Supervised Machine Learning Classification Algorithms

In this project logistic regression, K nearest neighbours, support vector machine with radial basis function, Naive bayes and deep neural network were trained on both the original and reduced data.

2.4 Performance Evaluation Metrics

We used accuracy, precision, recall, specificity, and F1-score to evaluate the performance of the classification models. These metrics are defined as follows:

1. **Accuracy**: Accuracy measures the ratio of correctly predicted instances to the total instances in the dataset.

$$\text{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{True Positive + True Negative + False Positive + False Negative}}$$

   If the class label of a record in a dataset is positive, and the classifier predicts the class label for that record as positive, then it is called a true positive. If the class label of a record in a dataset is negative, and the classifier predicts the class label for that record as negative, then it is called a true negative. If the class label of a record in a dataset is positive, but the classifier predicts the class label for that record as negative, then it is called a false negative. If the class label of a record in a dataset is negative, but the classifier predicts the class label for that record as positive, then it is called a false positive.

2. **Precision**: Precision measures the ratio of true positive predictions to the total predicted positives. It's a measure of how many of the predicted positive instances are actually positive.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

3. **Recall (Sensitivity or True Positive Rate)**: Recall measures the ratio of true positive predictions to the total actual positives. It's a measure of how many of the actual positive instances were correctly predicted.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

4. **F1 score**: The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall.

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision + Recall}}$$

5. **Specificity (True Negative Rate)**: Specificity measures the ratio of true negative predictions to the total actual negatives. It's a measure of how many of the actual negative instances were correctly predicted.

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives + False Positives}}$$

2.5 Cross Validation and Model Performance Metrics

In this project, K-fold cross-validation was used in training and validation of classification algorithms. Cross-validation is a technique crucial in machine learning for assessing a model's performance on unfamiliar data. It involves partitioning the available data into distinct subsets, or folds. One fold is reserved as a validation set, while the rest serve as training data. This cycle is repeated multiple times, with different folds acting as the validation set each time. The outcomes from these iterations

are then averaged to yield a more reliable measure of the model's performance.
The core objective of cross-validation is to counteract overfitting, where a model excels on the training data but falters on new, unseen data. By evaluating the model on multiple validation sets, cross validation provides a more realistic estimate of the model's generalization performance, i.e., its ability to perform well on new, unseen data. The model performance metrics in this project will be reported in terms of the mean validation accuracy, mean validation F1 score, mean validation recall, mean validation precision, mean validation specificity and mean validation sensitivity.

3 Results and Discussion

In this work, we have explored different dimension reduction techniques and trained various machine learning classification algorithms using both the reduced and original data. A train test split with 0.8/0.2 on the data gave 54 samples of the validation set. Due to relatively small size of the validation set which is susceptible to significance variance leading to less reliable validation scores. We adopted a k-fold cross validation with k=5 to train and validate machine learning classification algorithms for both original and reduced data.

3.0.1 Performance of classifiers with the original data

In this section, we discuss the results of experimentation with the original data using logistic regression, K nearest neighbours, support vector machines, Naive Bayes and Deep Neural Networks. The table (2) below shows the performance of the algorithms as averaged in K-fold cross validation (mean validation-MV) scores using the training and validation method. From (2). Naive Bayes

Table 2: Classifier Performance based on mean validation for original data(%)

Algorithm	Accuracy.	F1-score	Precision.	Recall.	Specificity.
Logistic Regression	84.07	81.16	84.80	78.29	88.36
SVM	65.19	53.95	65.56	46.30	80.71
KNN	62.69	50.66	61.14	44.35	78.14
Naive Bayes	85.19	82.78	84.76	81.68	87.52
DNN	76.05	64.57	67.01	63.13	75.07

achieved the highest accuracy at 85.19%, closely followed by Logistic Regression at 84.07%. These models seem to perform better in overall classification. Naive Bayes has the highest F1-score of 82.78%, followed by Logistic Regression at 81.16%. This suggests that Naive Bayes and Logistic Regression have better trade-offs between precision and recall. Naive Bayes and Logistic Regression exhibit high precision values, with Naive Bayes at 84.76% and Logistic Regression at 84.80%. Naive Bayes leads in recall with a score of 81.68%, and Logistic Regression follows closely with 78.29%. Logistic Regression has the highest specificity at 88.36%, implying that it is effective at identifying negative cases. Logistic Regression demonstrates balanced performance in terms of precision, recall, and specificity. Its accuracy and F1-score are also notably high. This algorithm is simple yet effective and can serve as a baseline model for many classification tasks. These results suggest that both Naive Bayes and Logistic Regression are strong candidates for further evaluation and potential deployment in real-world applications. However, the choice between the two depends on the specific requirements of the task and the dataset characteristics. While the SVM, KNN, and DNN algorithms show lower performance in comparison to Naive Bayes and Logistic Regression, it's worth noting that model performance can be influenced by hyperparameter tuning, dataset size, and data preprocessing techniques. Further experimentation and optimization might lead to improved results for these models.

3.0.2 Performance of classfiers with PCA reduced data with 5 features

We performed a PCA on the data and reduced the number of features to 5. Table (3) shows the performance metrics of the 5 classification algorithms. In general, the accuracy and F1-scores show relatively consistent patterns between PCA-reduced and original data for most algorithms. The Logistic Regression algorithm's performance remains stable, with only a slight drop in accuracy and F1-score when using PCA-reduced data. Similarly, Naive Bayes retains its high accuracy and F1-score. However, SVM, KNN, and DNN experience more noticeable drops in accuracy and

Table 3: Performance Metrics (mean validation) of Classification Algorithms based on PCA reduced data %

Algorithm	Accuracy.	F1-score	Precision.	Recall.	Specificity.
Logistic Regression	82.96	80.09	83.84	77.25	86.83
SVM	84.07	76.21	74.70	78.00	78.80
KNN	81.48	76.83	85.98	70.44	89.77
Naive Bayes	82.22	78.66	82.72	75.54	87.16
DNN	76.48	74.75	74.85	75.20	79.45

F1-score when using PCA-reduced data. Precision and recall values also show variations. While some algorithms like Logistic Regression and Naive Bayes maintain similar precision and recall between the two datasets, others like SVM and KNN exhibit trade-offs. SVM's precision decreases while recall increases with PCA-reduced data. On the other hand, KNN's precision improves, but recall drops significantly. This suggests that the choice of algorithm can have different impacts when using PCA-reduced data. The specificity values for most algorithms remain relatively stable between the two datasets. However, KNN notably improves in specificity when using PCA-reduced data, suggesting better performance in correctly identifying negative cases. Comparing the results from PCA-reduced and original data, it's evident that the impact of dimensionality reduction on algorithm performance varies. While some algorithms maintain their performance, others experience changes in accuracy, precision, recall, and specificity. The decision to use PCA-reduced data should be carefully considered based on the specific algorithm's behavior and the desired trade-offs in performance metrics.

3.0.3 Perfomance of the Classifiers using Kernel PCA reduced data using 5 Features

We performed Kernel PCA and reduced the data to 5 features, the five components were selected based on the eigen values of the resulting covariance matrix. We trained and validated the classification algorithms using the kernel reduced data From table (4) we observe that Kernel PCA-reduced data

Table 4: Performance Metrics (mean validation) of Classification Algorithms based on kernel PCA reduced data %

Algorithm	Accuracy.	F1-score	Precision.	Recall.	Specificity.
Logistic Regression	82.96	78.93	83.14	77.33	87.20
SVM	77.78	75.85	74.37	77.66	77.76
KNN	80.37	76.56	79.58	74.06	85.34
Naive Bayes	81.48	78.05	81.45	75.80	85.17
DNN	76.67	74.24	75.84	73.08	80.39

shows a slight drop in accuracy and F1-scores across all algorithms compared to the original data. Logistic Regression, Naive Bayes, and KNN maintain their accuracy quite well, but SVM and DNN exhibit more noticeable decreases. Precision values are generally maintained or slightly reduced with Kernel PCA-reduced data. Recall, on the other hand, is somewhat affected, with SVM and KNN experiencing drops in recall. This suggests that Kernel PCA might influence recall more than precision for certain algorithms. Specificity values mostly remain consistent between the original and Kernel PCA dataset, with a slight drop for SVM and DNN in the Kernel PCA-reduced data. Comparing the results from Kernel PCA-reduced and original data, we observe that dimensionality reduction using Kernel PCA has a varying impact on algorithm performance. While some algorithms maintain their performance well, others experience decreases in accuracy, F1-score, and recall. Precision and specificity tend to be more stable across the board. The decision to use Kernel PCA should be made based on the specific algorithm's behavior and the desired balance between performance metrics. Kernel PCA can be effective in capturing complex patterns in the data, but the trade-offs should be carefully considered.

3.0.4 Performance of classifiers with LDA reduced data with 1 feature

In this section, we present the experimental results obtained after training and validation classification algorithm using the LDA reduced data and compare the performance of the classification algorithms on both the LDA reduced data and original data. From table (5) comparing the two sets of results, we

Table 5: Performance Metrics (mean validation) of Classification Algorithms based on LDA reduced data %

Algorithm	Accuracy.	F1-score	Precision.	Recall.	Specificity.
Logistic Regression	85.56	83.58	85.74	81.73	88.25
SVM	84.81	82.96	84.49	81.73	86.92
KNN	85.93	82.85	90.19	77.47	92.61
Naive Bayes	84.81	82.49	85.32	80.06	88.26
DNN	78.09	82.52	85.70	80.17	89.40

observe that, LDA-reduced data with one feature generally outperforms the original data in terms of accuracy and F1-score for all algorithms. This indicates that the single feature extracted through LDA contains more discriminative information than the original dataset's features. Precision and recall values also show improvements with LDA-reduced data for most algorithms. This suggests that the single LDA feature better separates the classes, leading to higher precision and recall rates. LDA-reduced data maintains or improves specificity values for all algorithms, indicating a better ability to correctly identify negative cases. Comparing the results from LDA-reduced data and the original data, it's evident that the LDA transformation to one feature has significantly enhanced the performance of the classification algorithms. The single feature extracted through LDA captures meaningful discriminatory information, leading to improved accuracy, F1-score, precision, recall, and specificity across the board. The success of LDA in improving classification performance demonstrates its efficacy in feature extraction for dimensionality reduction. This result suggests that the LDA-reduced feature space is better suited for separating classes compared to the original data's feature space. In summary, the results strongly indicate the advantages of LDA in enhancing the discrimination between classes. Leveraging LDA-reduced data can lead to more accurate and reliable classification models.

4 Conclusion

In this research, we compared the performance of classification algorithms across four different scenarios: using the original data, data reduced by Principal Component Analysis (PCA), Kernel Principal Component Analysis (KPCA), and Linear Discriminant Analysis (LDA). We examined the performance of five algorithms: Logistic Regression, Support Vector Machine (SVM), k-Nearest Neighbors (KNN), Naive Bayes, and Deep Neural Network (DNN). The original data served as the baseline for comparison. It demonstrated varying performance across different algorithms. Naive Bayes showcased strong accuracy, precision, and recall, while SVM and KNN struggled with low accuracy and F1-scores. PCA-reduced data exhibited consistent patterns across most algorithms. While accuracy and F1-scores remained relatively stable for Logistic Regression and Naive Bayes, SVM, KNN, and DNN experienced decreases. PCA demonstrated a trade-off between reducing dimensionality and maintaining algorithm performance. Kernel PCA-reduced data displayed mixed results. While some algorithms maintained performance, others experienced drops in accuracy, F1-score, and recall. KPCA's impact on performance was varied and algorithm-specific. LDA-reduced data consistently outperformed the original data across all algorithms. The single LDA feature captured valuable class separation information, leading to improvements in accuracy, F1-score, precision, recall, and specificity. LDA showcased its ability to enhance discrimination between classes. Across all scenarios, Naive Bayes consistently exhibited strong overall performance. Its balanced accuracy, precision, and recall made it a dependable choice. Logistic Regression also maintained competitive performance in most cases, while SVM, KNN, and DNN faced challenges, particularly in scenarios involving dimensionality reduction. LDA-reduced data consistently yielded the best results among the dimensionality reduction techniques. Its ability to enhance class separation led to improved performance in all metrics for all algorithms. Naive Bayes and Logistic Regression stood out as the best performing algorithms with LDA-reduced data. While challenges exist, these findings provide valuable insights for selecting suitable algorithms and data transformation techniques based on the specific characteristics of the problem domain. Additionally, a combination of hyperparameter tuning, model selection, and feature engineering can further optimize algorithm performance and facilitate better decision-making in real-world applications.

References

[1] Issam El Naqa and Martin J Murphy. *What is machine learning?* Springer, 2015.

[2] S Velliangiri, SJPCS Alagumuthukrishnan, et al. A review of dimensionality reduction techniques for efficient computation. *Procedia Computer Science*, 165:104–111, 2019.

[3] Nandakishore Kambhatla and Todd K Leen. Dimension reduction by local principal component analysis. *Neural computation*, 9(7):1493–1516, 1997.

[4] Christopher JC Burges et al. Dimension reduction: A guided tour. *Foundations and Trends® in Machine Learning*, 2(4):275–365, 2010.

[5] Miguel A Carreira-Perpinán. A review of dimension reduction techniques. *Department of Computer Science. University of Sheffield. Tech. Rep. CS-96-09*, 9:1–69, 1997.

[6] Weikuan Jia, Meili Sun, Jian Lian, and Sujuan Hou. Feature dimensionality reduction: a review. *Complex & Intelligent Systems*, 8(3):2663–2693, 2022.

[7] Claudio Turchetti and Laura Falaschetti. A manifold learning approach to dimensionality reduction for modeling data. *Information Sciences*, 491:16–29, 2019.

[8] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[9] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[10] Md Faisal Kabir, Tianjie Chen, and Simone A Ludwig. A performance analysis of dimensionality reduction algorithms in machine learning models for cancer prediction. *Healthcare Analytics*, 3:100125, 2023.

[11] Nan Liu, Marcel Lucas Chee, Zhi Xiong Koh, Su Li Leow, Andrew Fu Wah Ho, Dagang Guo, and Marcus Eng Hock Ong. Utilizing machine learning dimensionality reduction for risk stratification of chest pain patients in the emergency department. *BMC medical research methodology*, 21:1–13, 2021.

[12] Thippa Reddy Gadekallu, Neelu Khare, Sweta Bhattacharya, Saurabh Singh, Praveen Kumar Reddy Maddikunta, In-Ho Ra, and Mamoun Alazab. Early detection of diabetic retinopathy using pca-firefly based deep learning model. *Electronics*, 9(2):274, 2020.

[13] Caterina Labrín and Francisco Urdinez. Principal component analysis. In *R for Political Data Science*, pages 375–393. Chapman and Hall/CRC, 2020.

[14] Quan Wang. Kernel principal component analysis and its applications in face recognition and active shape models. *arXiv preprint arXiv:1207.3538*, 2012.

[15] K Ming Leung et al. Naive bayesian classifier. *Polytechnic University Department of Computer Science/Finance and Risk Engineering*, 2007:123–156, 2007.

[16] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[17] Kevin P Murphy et al. Naive bayes classifiers. *University of British Columbia*, 18(60):1–8, 2006.

SUPPLEMENTARY MATERIALS

4.0.1 Principal Component Analysis (Linear)

Principal Component Analysis (PCA) is a dimensionality reduction technique commonly used in data analysis and machine learning. It aims to transform high-dimensional data into a lower-dimensional representation while retaining as much of the original variance as possible [13]. Below are the steps that describe PCA:

1. **Data Standardization**: If the features have different scales, it's important to standardize the data by subtracting the mean and dividing by the standard deviation for each feature.

$$\mathrm{X}_s\mathrm{X}_s\mathrm{X}_s\mathrm{X}_s = \frac{X_{ij} - \bar{X}_j}{\sigma_j}$$

2. **Compute the Covariance Matrix**: Calculate the covariance matrix of the standardized data.

$$\mathrm{Cov}(X_s) = \frac{1}{n-1}\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T$$

3. **Calculate Eigenvectors and Eigenvalues**: Compute the eigenvectors and eigenvalues of the covariance matrix. Eigenvectors represent the directions of maximum variance, and eigenvalues represent the amount of variance along each eigenvector.

$$\mathrm{Cov}(X_s)\mathbf{v} = \lambda \mathbf{v}$$

4. **Sort Eigenvectors and Select Principal Components**: Sort the eigenvectors by their corresponding eigenvalues in decreasing order. This determines the most important principal components. Choose the top k eigenvectors to retain the most important information while reducing dimensionality. These eigenvectors form the new basis $\mathbf{W}$.
5. **Project Data**: Project the original data onto the new basis formed by the selected eigenvectors to obtain the lower-dimensional representation.

$$\text{Projected Data} = X_s\mathbf{W} \tag{1}$$

4.0.2 Kernel Principal Component Analysis

Kernel Principal Component Analysis (Kernel PCA) is a technique for non-linear dimensionality reduction. It extends the traditional PCA to capture complex patterns in the data by using kernel functions. The steps of Kernel PCA are as follows:

1. **Choose a Kernel Function** Select a suitable kernel function that measures the similarity between data points. Common choices include the Gaussian (RBF) kernel, polynomial kernel, and sigmoid kernel. The kernel choice depends on the nature of the data.
2. **Kernel Matrix**:
   Compute the kernel matrix $\mathbf{K}$, where K_{ij} represents the similarity between data points x_i and x_j using the chosen kernel function:

$$K_{ij} = \phi(x_i) \cdot \phi(x_j)$$

   Here, $\phi(x)$ represents the feature map induced by the kernel.
3. **Center the Kernel Matrix**: Center the kernel matrix $\mathbf{K}$ by subtracting the mean of each row/column and adjusting the diagonal elements. The centered kernel matrix $\tilde{K}$ is computed as follows:

$$\tilde{K}_{ij} = K_{ij} - \frac{1}{n}\sum_{l=1}^{n} K_{il} - \frac{1}{n}\sum_{l=1}^{n} K_{lj} + \frac{1}{n^2}\sum_{l,m=1}^{n} K_{lm}$$

4. **Calculate Eigenvalues and Eigenvectors**:
   Compute the eigenvalues and eigenvectors of the centered kernel matrix $\tilde{K}$. The eigenvalues represent the variance captured by the corresponding eigenvectors in the transformed feature space.
5. **Sort Eigenvectors**:
   Sort the eigenvectors based on their corresponding eigenvalues in decreasing order. This determines the most significant principal components.
6. **Select Principal Components**: Choose the top k eigenvectors to retain. These eigenvectors form the projection matrix $\mathbf{A}$.
7. **Project Data**:
   Project the original data onto the new basis formed by the selected eigenvectors. The transformed data is obtained by multiplying the centered kernel matrix $\tilde{K}$ with the projection matrix $\mathbf{A}$:

$$\text{Projected Data} = \tilde{K}\mathbf{A} \tag{2}$$

   Kernel PCA provides a non-linear representation of the data that can capture intricate relationships and patterns that linear PCA might miss [14].

4.0.3 Linear Discriminant Analysis(LDA)

Linear Discriminant Analysis (LDA) is a technique for dimensionality reduction and classification. The steps of LDA are as follows:

1. **Compute Class Means**: Calculate the mean vector for each class:

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

   Where C_i represents class i and n_i is the number of samples in class i.
2. **Compute Scatter Matrices**: Compute the within-class scatter matrix (S_W) and the between-class scatter matrix (S_B):

$$S_W = \sum_{i=1}^{c} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T$$

$$S_B = \sum_{i=1}^{c} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$$

   Where c is the number of classes and $\mathbf{m}$ is the overall mean.
3. **Compute Eigenvalues and Eigenvectors**: Compute the eigenvalues and eigenvectors of $S_W^{-1} S_B$.
4. **Sort Eigenvectors**: Sort the eigenvectors by their corresponding eigenvalues in decreasing order.
5. **Select Discriminant Components**: Choose the top k eigenvectors to retain as the new feature subspace.
6. **Project Data**: Project the original data onto the new subspace formed by the selected eigenvectors. The new projected data in the LDA subspace is calculated as follows:

$$P(y|x) = \frac{P(x|y) \cdot P(y)}{P(x)}$$

Where:

output layer corresponds to the number of classes we are trying to classify.