Perceptron Rule, Logistic Regression, Kernal PCA: Methods for Classification

Student Number: 10219553

0.1 Abstract:

PURPOSE: Evaluate the effectiveness of the Perceptron Rule against Logistic Regression to determine if a US College is Public or Private. Addiotionally, Investigate the effectiveness of Kernal PCA and K-means to classify Fisher's Iris Dataset.

METHODS: The first dataset was ran through the perceptron rule algorithm using the built-in regression function, and the Kernal PCA was applied to the second dataset through a Gram matrix and a Gaussian Kernel.

RESULTS: The accuracy of the hyperplane estimation using Logistic Regression was more accurate and had a larger AUC than the Perceptron Rule. Additionally, clustering the Iris dataset using K-means combined with Kernal PCA resulted in a perfect score.

CONCLUSIONS: Logistic Regression and Kernal PCA met the metrics evaluation criteria specified in the scientific question for their respective datasets, with both accuracies exceeding 90%, while the Perceptron Rule fell short. These methods are effective for their use-cases.

Word Count: 141

0.2 Introduction:

The objectives of this study were twofold. The first was to compare the accuracy of the Perceptron Rule to Logistic Regression to determine the sector of an American College. The second was to investigate the effectiveness of Kernal PCA and K-means to cluster Fisher's Iris Dataset.

The Perceptron Rule is a binary classifier that operates by finding a decision boundary that separates the two classes in the feature space. To apply the Perceptron Rule, one initializes the weights (including the bias) to small random values. For each instance in the dataset, the weighted sum of the inputs is calculated, and an activation function is applied to this sum to make a prediction. If the prediction is incorrect, the weights are updated by modifying the input vector, scaled by a learning rate, depending on whether the instance was misclassified as positive or negative. This process is repeated for a fixed number of iterations or until the algorithm converges (i.e., no errors). The result is a set of weights that define a hyperplane which can classify new instances by applying the weighted sum and activation function [2].

Logistic Regression is a linear model for binary classification. It is performed by using the logistic (sigmoid) function to model the probability that an entity belongs to a particular class. The process involves initializing the weights, then iteratively adjusting these weights based on the difference between the predicted probabilities and the actual class labels. This adjustment is done through gradient descent, by minimizing the cost function. Each weight is updated by subtracting the derivative of the cost function with respect to that weight, scaled by a learning rate. Unlike the Perceptron, which updates weights based on misclassified examples only, Logistic Regression updates weights based on the degree of error in the prediction for all instances. The outcome is a set of weights that can be used to compute the probability of class membership [4].

A kernel function is a mathematical algorithm that allows us to find patterns or structures in data by transforming it into a higher-dimensional space without actually having to perform the computation to move it into that space.

Principal Component Analysis (PCA) is a method of reducing the dimensionality of a dataset. To perform PCA, the first two right singular vectors from the V matrix are taken from the Singular Value Decomposition (SVD), and multiplied with the Zero-mean data matrix. In our case, this will reduce the dimensionality of the dataset to 2 Dimensions.

Kernel Principal Component Analysis (Kernel PCA) extends the dimensionality reduction capabilities of PCA to handle nonlinear data structures. To perform Kernel PCA, one initially applies a kernel function to map the original data into a higher-dimensional feature space

where the data can potentially be linearly separable. This process does not require explicit computation in the higher-dimensional space thanks to the kernel trick. After this mapping, the equivalent of Singular Value Decomposition (SVD) is performed in this feature space to find the principal components. Specifically, one computes the eigenvalues and eigenvectors of the centered kernel matrix, which corresponds to the covariance matrix in the feature space. The dataset is then projected onto the leading eigenvectors associated with the largest eigenvalues to reduce its dimensionality [6].

Receiver Operating Characteristic (ROC) Curves are graphical plots that illustrate the ability of a binary classification system as its discrimination threshold changes. The curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR).

Area Under the Curve (AUC) evaluations measure the 2D area under the ROC curve, which exists between (0,0) and (1,1). It provides a single scalar value that summarizes the performance of a binary classification value, ranging from 0 to 1. An AUC of 0.5 implies that the classification is just as effective as picking a class at random every time, while higher values indicate better predictions.

The scientific question to be looked at is, can Kernal PCA, the Perceptron Rule, and Logistic Regression be applied to their respective datasets such that the accuracy of predictions are above 90%.

0.3 Methods:

The analytical framework of this study is structured into two distinct sections, each answering questions posed in the report outline, with numerous supporting functions. The preliminary requirements are to access the datasets and standardize the data. For the College dataset, PCA is also performed. The data and labels are then passed to their respective functions.

The instructor provided two supporting, non-modifiable functions, the first of which is logreg, which computes the logistic regression for a data matrix given its label vector. The second is plotline, which adds a separating hyperplane to a 2D plot.

The first part of the code applies the perceptron rule and logistic regression using helper functions, and computes their predicted classifications and accuracies. The original Xmat for the data is augmented by appending the 1s vector.

The sepbinary helper function performs the perceptron rule. It iterates to adjust weight vectors based on the perceptron learning rule until convergence or a maximum iteration count is reached. The estimate of the weight vector v_est is scored for each data point using the Heaviside function. Then, the residual error vector, rvec, is calculated by subtracting the quantization, q, from the original labels. The weight vector is then modified using rvec and eta. The looping continues if the data is mis-classified, but it stops if the estimate converges. We then use v_est to set v_final as the final weight vector. As previously mentioned, the logistic regression is computed using the provided logreg function, which estimates the hyperplane and does not need to be modified.

After the estimates are calculated by both the logistic regression and perceptron rule, the originial data is scored using hyperplane augmented vectors. The ROC curves and AUC values are computed in the perfcurve function. The accuracy of the hyperplane estimates is calculated by comparing the number of correct label predictions to the labels in Xaug, which is the augmented data matrix. The ROC curves are plotted, alongside the scatter plots of the data along with hyperplane lines of separation.

The second section of the code focuses on Kernal PCA and K-means clustering on the Fisher's Iris dataset. This segment begins with An anonymous function Gmat is defined to generate a centering matrix of a specified size, crucial for the Gram matrix's centering process in Kernel PCA.

This segment begins with the definition of a Gaussian kernel function through an anonymous function that calculates the centered matrix of a given size. The process continues with the computation of the Gram matrix using a Gaussian kernel, which serves as a preliminary step for Kernel PCA. Then, the first two variables from are selected Xmat as the default pro-

jection (Mgram). It sets a Gaussian kernel's variance (sigma2) based on the dataset size, which influences the kernel's sensitivity to distances between data points.

The gramgauss function calculates the gram matrix using the Gaussian Kernel. This matrix is then centered using Gmat. Subsequently, the eigenvalues and eigenvectors are calculated, and the first two components are selected to reduce the data to 2D. THe original data is projected onto this space by multiplying the Centered matrix by the selected eigenvectors.

The reduced dataset is then subjected to K-means clustering to partition the data into clusters based on similarity. This clustering aims to categorize the data into two distinct groups, corresponding to the binary labels within the Iris dataset. The output from the K-means algorithm - indices representing cluster membership - is adjusted to align with the original binary labeling of the dataset by subtracting one from each cluster index.

To evaluate the effectiveness of the Kernel PCA in conjunction with K-means clustering, the script visualizes the results through two separate plots. The first plot displays the original labels of the dataset, allowing for a visual assessment of the data's inherent grouping. The second plot shows the clusters as identified by the K-means algorithm post Kernel PCA transformation.

0.4 Results:

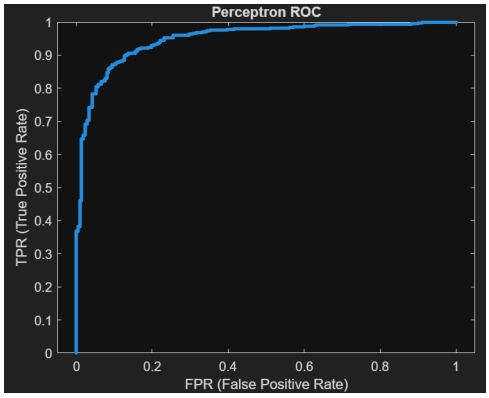

Figure 1: The ROC curve for the Perceptron Rule.

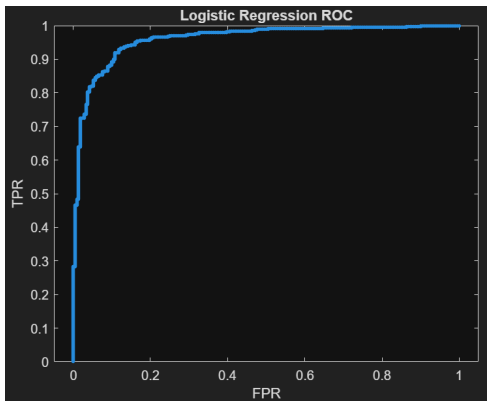

Figure 2: The ROC curve for Logistic Regression.

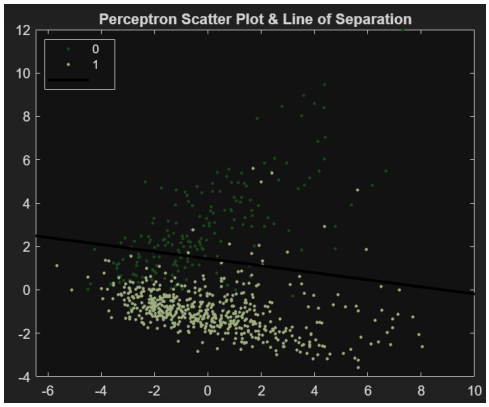

Figure 3: The Scatter Plot and Line of Separation for the Perceptron Rule.

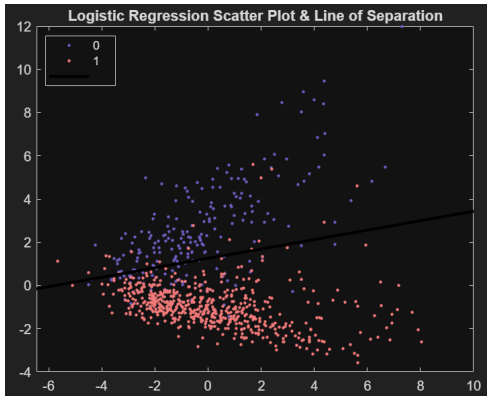

Figure 4: The Scatter Plot and Line of Separation for Logistic Regression.

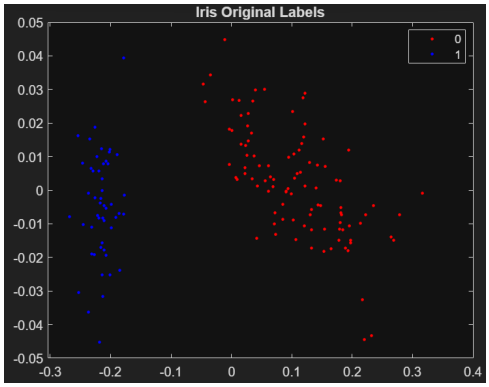

Figure 5: The original labels for the Iris data.

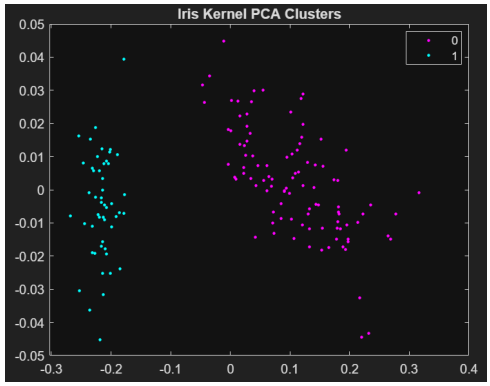

Figure 6: The Kernal PCA clusters for the Iris Data.

Table 1: The AUC values, Prediction Accuracy, and Optimal Threshold for the Perceptron Rule and Logistic Regression.

	Perceptron Rule	**Logistic Regression**
AUC	0.9495	0.9600
Accuracy	0.8764	0.9112
Optimal Threshold	0.4456	0.1457

0.5 Discussion:

The findings of this experiment are mostly contained within the figures and table in the results section. Figures 1 and 2 are plots pertaining to the ROC curves of the perceptron rule and logistic regression, with AUC values of 0.9495 and 0.9600 respectively. Figures 3 and 4 show the hyperplane seperability achieved by the perceptron rule and logistic regression, with the perceptron rule sporting an accuracy of 87.64%, while logistic regression achieved 91.12%. These results are good, but can be improved upon significantly, especially for the perceptron rule. Table 1 shows these AUC and accuracy values, as well as the optimal threshold for both methods. Figure 5 is a plot of the original data labels from the dataset, while figure 6 shows the clustering achieved by Kernel PCA with K-means. The two graphs are the same, indicating that this method had a 100% accuracy.

Furthermore, the precision of the classifications is reflected through the ROC curves and AUC metrics. The AUC metric for the perceptron rule stands at 0.9495, significantly surpassing the benchmark value of 0.5 indicative of random guessing, yet it falls short of the AUC score achieved by logistic regression, which is 0.9600. Given that the AUC represents the area beneath the curve, the logistic regression demonstrates a slightly superior curve elevation relative to that of the perceptron rule. This elevation suggests enhanced classification accuracy for logistic regression, attributed to its higher ratio of true positive rates over false positive rates.

When comparing logistic regression to the perceptron rule, it's essential to acknowledge the inherent complexity and sophistication of logistic regression. Unlike the perceptron rule, which employs a straightforward linear combination of input features for classification, logistic regression leverages a sigmoid function to model the intricate relationships between input variables and the resultant classes. This fundamental difference in approach grants logistic regression a higher degree of flexibility and nuanced observation capability. Consequently, this distinction provides a clear rationale for logistic regression's superior performance over the perceptron rule in our analysis.

Examining figures 5 and 6, which display the labels derived from the dataset and the clusters identified through K-means clustering applied on the Kernel PCA. The graphs appear the same to the eye, indicating that the method was extremely effective. This implies that Kernel PCA is highly effective for the visualization of high-dimensional data in lower-dimensional space, and is especially useful in cases where data may have non-linear characteristics. IT can be helpful for manipulating and exploring complex data, and can also be used to detect anomalies and outliers in data that should be uniform. Kernel PCA is very computationally complex, and often struggles with extremely large datasets, where significant dimensionality

reduction is detrimental to the accuracy of the data. As a result, the right machine learning technique may not always be the most accurate one, and choosing the correct method requires thorough analysis of the dataset at hand.

The perceptron rule, a foundational concept in the field of artificial intelligence and neural networks, was developed by Frank Rosenblatt in 1957 at Cornell University. Rosenblatt's work on the perceptron was significant because it laid the groundwork for the development of neural networks by demonstrating how machines could learn from data [1]. The perceptron was initially designed as a machine (the Mark I Perceptron) and was one of the first computer models to simulate the thought processes of the human brain, using a form of neural network that could learn new skills through trial and error. It is critical to keep this info in mind, as it this method was a very early and simple prototype. While it was a key moment in the history of machine learning, it has significant limitations, and in the modern age, it is no longer very effective. The history of the perceptron is incredibly interesting, as it led to a period of time in which there were few advancements in the field of AI [7].

This ended when the back propagation algorithm was designed in the 1980s. This algorithm allowed for efficient training of multi-layer neural networks, as it computes gradients for each neuron from the output layer back to the input layer, iteratively adjusting the weights to minimize errors. The development of back propagation, along with improvements in computational power and the availability of large datasets, has been crucial in overcoming the initial limitations faced by perceptrons. Additionally, neural networks started to implement hidden layers of neurons that were not input or output facing, and this improved performance further. An implementation using these more advanced networks will help improve the classification of the dataset [8].

In conclusion, Logistic Regression and Kernel PCA both met the evaluation criteria proposed in the scientific question, with accuracy values above 90%, indicating that the classifiers had strong performance. Meanwhile, the accuracy of the Perceptron Rule fell below the criteria, with an accuracy of about 88%. Further exploration into the applications of other methods such as neural networks with back propagation and hidden neuron layers may prove more fruitful.

0.6 References:

1. The perceptron: A probabilistic model for information storage and organization in the brain. Available from:
   https://psycnet.apa.org/record/1959-09865-001

2. Classification – Single Artificial Neuron. Available from:
   https://research.cs.queensu.ca/home/cisc271/pdf/Class27.pdf

3. Supervised Learning – Perceptron Rule. Available from:
   https://research.cs.queensu.ca/home/cisc271/pdf/Class30.pdf

4. Classification – Logistic Regression. Available from:
   https://research.cs.queensu.ca/home/cisc271/pdf/Class31.pdf

5. Nonlinear Separation – Embeddings and Gram Matrix. Available from:
   https://research.cs.queensu.ca/home/cisc271/pdf/Class32.pdf

6. Nonlinear Separation – Kernel PCA. Available from:
   https://research.cs.queensu.ca/home/cisc271/pdf/Class33.pdf

7. A Brief History of Neural Networks. Available from:
   https://www.dataversity.net/a-brief-history-of-neural-networks/

8. History and Evolution of Artificial Neural Networks (ANN). Available from:
   https://www.unrepo.com/ann/history-and-evolution-of-artificial-neural-networks-ann