Data Science 3 Project: Homework 4

Group 7: Aindrila, Suchandra, Chirag, Paritosh, Vanshika

7 April, 2024

1 Classifiers

We will discuss about 4 classifiers as Depth-based Classifier, SVM, K-NN (with an optimal choice of k determined through techniques like cross-validation), and kernel density function-based methods.

1.1 Depth-based Classifier

Depth-based classifiers are a type of machine learning algorithm that utilizes the concept of depth functions to make decisions about the classification of data points. Half-space depth is one such depth function, which measures the depth of a point relative to a given half-space.

In this approach, each data point is assigned a depth value based on its distance from a given half-space. A half-space is defined by a hyperplane in the feature space, and points on one side of the hyperplane are considered to have positive depth, while points on the other side have negative depth.

1.1.1 Classification Decision:

To classify a new data point, the depth of the point relative to multiple half-spaces is computed. The decision is based on the aggregate depth values obtained from these half-spaces. Points with higher depths are considered to be more central to the data distribution and are more likely to belong to the same class. Mathematically, we need to calculate the data depth for each class including the future observation, say x. We will assign x to the class 1 or say $\mathcal{X}$ if

$$D_{\mathcal{X}}(x) > D_{\mathcal{Y}}(x)$$

otherwise x will be assigned to the class 2 or say $\mathcal{Y}$ We will do this procedure after standardizing the data.

If any future observations are out of the convex hull, then it is inconclusive and is treated like an outlier kind of.

1.2 SVM-based Classifier

Support Vector Machines (SVMs) seek to find the hyperplane that best separates the classes in the feature space. SVMs maximize the margin between classes, which helps generalize well to unseen data. In cases where classes are not linearly separable, SVMs use kernel functions to map the data into a higher-dimensional space where separation is possible. SVMs are effective in high-dimensional spaces and are less affected by the curse of dimensionality. They work by finding the optimal hyperplane that separates data points of different classes in a high-dimensional space. The primary goal of SVM is to maximize the margin between the classes, which helps in achieving better generalization and robustness to unseen data.

1.2.1 Methodology:

- **Training Data:** We have a set of training samples, $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, where x_i represents the input features and y_i represents the corresponding class labels. y_i can be either -1 or +1 for binary classification. Each input sample x_i is represented as a feature vector in a high-dimensional space.

- **Hyperplane:** SVM tries to find a decision function that maps input vectors to one of the two classes: +1 or -1. In SVM, the decision boundary is represented by a hyperplane defined as:

$$w^T x + b = 0$$

  Where w is the weight vector perpendicular to the hyperplane, $\hat{w}_n = \sum_{i=1}^{n} y_i \alpha_i x_i$, where $\alpha_i = h(< x_i, x_j >)$, $i = 1, \ldots, n, j = 1, \ldots, n$ and b is the bias term.

  The equation $w^T x + b$ gives us the signed distance of point x from the hyperplane. If the result is positive, the point lies on one side of the hyperplane, and if it's negative, it lies on the other side. To summarize, $w^T x + b > 0$ for $y_i = +1$ and $w^T x + b < 0$ for $y_i = -1$.

- **Margin:** The margin is the distance between the hyperplane and the nearest data point from either class. The goal of SVM is to maximize this margin. Mathematically, the margin can be computed as:

$$\frac{2}{\|w\|}$$

  Where $\|w\|$ is the Euclidean norm of the weight vector w.

- **Optimization Objective:** SVM aims to maximize the margin while minimizing the classification error. This can be formulated as an optimization problem:

$$\text{Minimize } \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

  Subject to the constraints:

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \text{for } i = 1, 2, ..., n$$

$$\xi_i \geq 0, \quad \text{for } i = 1, 2, ..., n$$

  Where ξ_i are slack variables representing the classification error, C is the regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification error. Higher values of C impose stricter penalties on misclassifications.

Its mathematical formulation involves defining the decision function, maximizing the margin, and minimizing classification errors through optimization. Additionally, the kernel trick extends SVM's applicability to non-linear datasets.

1.3 K-NN-based Classifier

K-Nearest Neighbors (K-NN) is a simple instance-based learning algorithm where classification is based on the majority vote of the K nearest neighbors to a query point in the feature space. K-NN does not make any assumptions about the underlying data distribution, making it non-parametric and flexible. It can handle complex decision boundaries and is robust to noisy data. However, K-NN can be sensitive to the choice of k and may suffer from the curse of dimensionality, especially with high-dimensional data.

1.3.1 Methodology

- **Step 1:** First, we will choose our k appropriately. (Complete discussion is in data analysis part).
- **Step 2:** Given a new, unlabeled data point, calculate its distance to all other data points in the dataset. Here, we compute the k-nearest neighbor of the future observation, say 'x' from the combined data cloud. The most commonly used distance metric is Euclidean distance, but other metrics like Manhattan distance or cosine similarity can also be used.
- **Step 3:** Out of k-nearest neighbor, compute the number of observations from both classes. Select the k nearest data points based on the calculated distances. sayk is a predefined hyperparameter that determines the number of neighbors to consider.
- **Step 4:** For a classification task, assign the class label that appears most frequently among the k nearest neighbors to the new data point. This is often done by majority voting. Simply saying, No of data points in class 1 > No of data points in class 2 in K-NN, then assign "x" to class 1 otherwise to class 2.

1.4 Kernel Density Function-based Classifier

Kernel Density Estimation (KDE) is a technique for estimating the probability density function of a random variable based on the observed data points. In classification, KDE-based classifiers assign class labels to query points based on the estimated density of training samples in the feature space. KDE classifiers are non-parametric and can capture complex data distributions without making strong assumptions. However, they can be computationally intensive, especially with large datasets, and the choice of kernel bandwidth can significantly affect performance. leveraging PCA for dimensionality reduction, we allow efficient representation of the dataset. Through the use of KDE, the algorithm assesses the likelihood of each observation belonging to a particular class, facilitating accurate classification based on density comparisons. Overall, this method integrates PCA and KDE seamlessly to enable robust classification of the dataset into its respective classes, enhancing the interpretability and performance of the classification model.

Each classifier has its trade-offs in terms of interpretability, computational complexity, and performance. The choice of classifier depends on factors such as the characteristics of the dataset, the desired interpretability of the model, and the computational resources available. Experimentation and validation on representative datasets are often necessary to determine the most suitable classifier for a given task.

2 Analysis on Datasets

To address the task, first obtain the Heart Disease and Breast Cancer Wisconsin datasets from the UCI Machine Learning Repository. Train these classifiers on the training data and fine-tune hyperparameters where applicable. Subsequently, on the given datasets we will evaluate the models on the test data to compute empirical misclassification probabilities, comparing predicted labels with actual labels. Repeat this process multiple times with different data splits to ensure the robustness of results, then analyze and document the performance of each classifier along with the chosen methodology and results obtained for reproducibility.

In this analysis, we examined the impact of different train-test ratios (80:20 and 70:30) on the performance of our classifier. After dividing the dataset into respective train and test sets, we iteratively fit the training data and predicted labels for the test data using the chosen classifier. This process was repeated 100 times to ensure statistical reliability. Subsequently, we computed

the mean and standard deviation of the misclassification probabilities across these runs. The mean misclassification probability provides an average measure of the classifier's performance across various train-test splits, while the standard deviation offers insights into the variability or consistency of these results. In the tables 1 and 2 mean of the misclassification probability is given with the sd that is in the bracket.

- For the depth-based method, we have executed the procedure described in section 1.1. Using "ddalpha" package we have calculated the half-space depth on the standardized data.
- For SVM classifier, we use the function "svm" from package "e1071". For the prediction of a new observation, we have used "predict" function.
- In R, we can choose the value of "k" in KNN using techniques like cross-validation to find the optimal value using the "caret" package. We define a grid of values for k from 1 to 20 to search over 12-fold cross-validation. The script trains KNN models for each value of k in the specified range and selects the best value based on cross-validation performance. Getting the best value of k, this can then be used for making predictions on new data.
- In the classification process employing PCA, the dataset is initially divided into training and test sets. Following this, each class within the dataset is separated for further analysis. Utilizing kernel density estimation (KDE) through the KS library, density values are calculated at evaluation points corresponding to the test data. By comparing the densities obtained for each class, the observation is then classified into the class with the higher density value.

2.1 Heart Disease Data

The Heart Disease dataset, originating from 1988, comprises four distinct databases: Cleveland, Hungary, Switzerland, and Long Beach V. It encompasses a total of 76 attributes, encompassing various patient health indicators. However, researchers commonly focus on a subset of 14 attributes for their analyses. The pivotal attribute of interest, labeled as "target," indicates the presence of heart disease in the patient. This target attribute is represented as an integer, where 0 signifies the absence of disease and 1 indicates its presence. This dataset serves as a fundamental resource for exploring predictive models aimed at diagnosing heart disease.

Classifier	Depth Based	SVM	K-NN	Kernel Density Based
train:test=80:20	0.0345 (0.0181)	0.1604 (0.0222)	0.3525 (0.0145)	0.4633 (0.0244)
train:test=70:30	0.0447 (0.0174)	0.1614 (0.0183)	0.3536 (0.0123)	0.4548 (0.0269)

Table 1: Comparision of the mean of misclassification probability on different train test ration on Heart Disease Data.

The results show better performance when the training-to-test ratio is 80:20 compared to 70:30, likely due to the larger proportion of data used for training, which enables the model to learn more effectively from the available information.

Prioritizing accurate classification and minimizing misclassification errors are paramount in robust classification tasks. The "Depth Based" algorithm emerges as a favorable choice, emphasizing precise classification to mitigate the risk of erroneous class assignments. Additionally, SVM demonstrates commendable performance across varying train-test ratios. Leveraging misclassification probabilities, these algorithms aim not only for correct classifications but also to curtail potential erroneous predictions. K-NN exhibits notable capability in correctly classifying data, while kernel density estimate performs suboptimally, presenting roughly a 50-50 chance of accurate classification.

2.2 Breast Cancer Wisconsin (Diagnostic) Data

The dataset contains information regarding cell nuclei characteristics, particularly for diagnosing breast cancer. Each entry is associated with an ID number and a diagnosis indicating whether the cell is malignant (M) or benign (B). The features extracted for each cell nucleus include radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension. These features provide insights into various aspects of the cell's structure and behavior, such as its size, shape, and texture. These quantitative measurements serve as crucial indicators for the diagnosis of breast cancer, aiding medical professionals in making informed decisions regarding patient treatment and management.

Classifier	Depth Based	SVM	K-NN	Kernel Density Based
train:test=80:20	0.1913 (0.0414)	0.0296 (0.0155)	0.0606 (0.0043)	0.2832 (0.0404)
train:test=70:30	0.2131 (0.0294)	0.0296 (0.0115)	0.0703 (0.0028)	0.2906 (0.0283)

Table 2: Comparision of the mean of misclassification probability on different train test ration on Breast Cancer Wisconsin (Diagnostic) Data.

Similarly, here also the results show better performance when the training-to-test ratio is 80:20 compared to 70:30.

In the realm of data classification, SVM emerges as a standout performer, boasting strong accuracy and adeptness at deciphering intricate patterns within the dataset. KNN closely follows suit, offering simplicity and intuition, albeit with potential slowdowns when confronted with extensive datasets. Depth-based methods showcase moderate effectiveness, although they may encounter challenges in handling intricate data structures. Conversely, kernel density estimation falls short in delivering robust classification outcomes, relying heavily on probabilities and struggling with the complexity of the dataset.

Overall for both the datasets and both the train test ratio, SVM and KNN are primary contenders, while depth-based methods hold promise, and kernel density estimation trails behind in terms of performance.

By comparing these metrics between different train-test ratios, we gained an understanding of how the classifier's performance varied with varying dataset sizes. This analysis aids in assessing the robustness and generalizability of the classifier under different conditions, facilitating informed decisions regarding the choice of train-test split ratio for future evaluations.