RANDOM FORESTS AND COMPARISON TO COMPETING METHODS

ABSTRACT:

The provided code demonstrates a basic neural network using Python's Kera’s library for binary classification. It utilizes synthetic data, splitting it into training and test sets, configuring layers, and compiling the model. This code showcases the network's structure, training, and evaluation processes. To apply this neural network to real data, appropriate preprocessing and adjustments should be made. The abstract highlights its fundamental implementation, serving as a foundational guide for constructing neural networks in Python.

WORD COUNT:1192

TANMISHA MURALI

GROUP 14

DURHAM UNIVERSITY

1 INTRODUCTION

There is a range of methods that aim to model a function that captures the relationship between an output variable Y and a feature space x = (x1, ..., xp), such that Y = f(x) + ε, ε ∼ N (0, σ2), where p is the number of features considered in the analysis. Some competing methods to model this function included linear regression, Naive Bayes, Support Vector machines and Neural Networks. Random Forests, first introduced in Breiman (2001), is an ensemble method that use a predetermined number of decision trees. Ensemble methods/Binary Trees/ Pruning(overfitting)/ Bagging The remainder of this report contains, explanations of the fundamental concepts behind Random Forests; a description on how hyperparameters are tuned for model optimisation and a comparison with competing methods using a range of metrics on the ... dataset.

2 FUNDAMENTALS OF RANDOM FORESTS

Random forest is an ensemble learning technique that constructs multiple decision trees during training and outputs the mode of classes or the mean prediction of individual trees. It operates by creating a forest of decision trees, each trained on random subset of the data and using random features subsets. The trees prediction is then combined to give a final output, reducing overfitting, and improving accuracy

Random Forest works by:

- Randomly selecting subsets of data for each tree.
- Constructing decision trees using these subsets.
- Making predictions by aggregating results from all trees.

It is highly effective due to its ability to handle complex relationships in data, manage high-dimensional spaces, and provide feature importance

2.1 Decision Trees

Random Forests uses decision tress to partition the feature space and give an output. A decision tree consists of a root node

Random Forest is an ensemble learning method built upon Decision Trees. Decision Trees are a fundamental concept in machine learning, representing a tree-like graph where each internal node denotes a feature, each branch represents a decision rule, and each leaf node represents an outcome. They are constructed by recursively splitting the dataset based on features to maximize information gain or reduce impurity.

2.2 Bagging

Random Forest uses bagging, this technique combines multiple models to make more accurate predictions. Bagging creates multiple subsets of the original dataset by random sampling with replacement (bootstrap samples). It trains several Decision Trees on these subsets independently. In the case of Random Forest, each tree is trained on a different subset.

2.3 Pruning

Pruning is a technique used in decision trees to avoid overfitting. While individual trees in random forest tend to overfit less due to bagging, pruning can further refine them. It involves removing branches that do not significantly increase the tree's predictive accuracy on validation data. In Random Forest, pruning might not be explicitly applied to individual trees due to their diverse construction but is implicitly achieved by averaging the results of multiple trees.

Random Forest combines these elements effectively: utilizing decision trees that have been both bagged and often implicitly pruned through ensemble averaging. The ensemble nature of Random Forest reduces overfitting and variance, improves accuracy, and provides robust predictions by combining the outputs of multiple trees.

3. NEURAL NETWORK AND STRUCTURE

When it comes to optimizing the network structure for Neural Networks in contrast to Random Forest, two critical aspects often considered are hyperparameter tuning and cross-validation.

3.1 Hyperparameter Tuning:

In Neural Networks, the architecture's performance heavily relies on hyperparameters like the number of layers, neurons per layer, activation functions, learning rate, and regularization techniques. Fine-tuning these parameters is crucial to enhance the model's learning capabilities and generalization.

Hyperparameters are settings we can adjust to make the Random Forest work better for a particular task.

Two important ones are:

- n_estimators: This tells us how many decision trees are in our group. More trees can lead to better accuracy but might take longer to compute.
- max_features: It refers to the number of features (like mass, width, length, etc.) considered for each split in the decision trees.

Compared to Random Forest, Neural Networks demand more intricate tuning. Adjusting the number of layers and neurons in each layer impacts the model's capacity to capture complex patterns within the data. Selecting appropriate activation functions like ReLU, Sigmoid, or Tanh influences how the network learns and adapts. Additionally, optimizing learning rates and regularization parameters like dropout or L2 regularization helps prevent overfitting.

3.2 Neural network

A neural network is structured with sequential layers—Input, Hidden, and Output. Trained with the Adam optimizer and categorical cross-entropy loss function for multi-class classification, it's trained for 10 epochs.

For a test size of 0.2, the network exhibits promising performance:

Training Time: 4.77 seconds
Accuracy: 93.7%

This demonstrates the network's robustness in learning and accurately classifying patterns within the dataset.

4 COMPARISONS TO COMPETING METHODS

4.1 Accuracy Scores

The model demonstrates an accuracy of 93.6% with a test size of 0.2, whereas for a test size of 0.3, the accuracy slightly decreases to 91.9%. This indicates that employing a smaller test size of 0.2 yields a higher accuracy compared to the larger test size. The difference in accuracies between these test sizes suggests that a smaller proportion of the dataset for testing (0.2) better supports the model's ability to generalize, resulting in improved performance in accurately predicting outcomes.

4.2 Computational Time:

The random forest model's training time varied depending on the execution. The fastest recorded time was significantly faster at 1.351 seconds, while the slowest was 9.235 seconds. The training process took about 2.409 seconds on average. The significant variability in computational speed during model training is indicated by these time disparities. The model demonstrated a broad range of training times, including both comparatively long runs and incredibly quick calculations, illustrating the variation in random forest algorithm processing times.

4.3 Confusion matrices

Generated specifically for Random Forest models to offer a detailed view of classification performance. Heatmaps were employed to visually represent the correspondence between true and predicted labels, specifically designed for different test sizes.

5 CONCLUSIONS

In conclusion, there are clear benefits to using both Random Forest and Neural Networks in machine learning. Known for its simplicity, Random Forest works well with missing values and is reliable with big datasets, which makes it appropriate for a variety of applications. On the other hand, because of their intricate architectures, neural networks are excellent at learning complex patterns and work well for complex tasks like image recognition and natural language processing. The features of the dataset, the requirements for interpretability, and the particular problem domain all influence which of these models is best. The best model to use for a given task can be chosen by balancing interpretability, computational requirements, and accuracy.