Neural Network Performance Comparison

2024-08-01

Jodick Ndayisenga

ID: 666225

This report presents the training, evaluation, and comparison of neural network models with different configurations of hidden layers using the iris dataset. The objective is to understand how varying the number of hidden layers and neurons affects the model's performance and training time.

The iris dataset is a classic dataset used in machine learning, which includes measurements for 150 iris flowers from three different species. Each species is represented by 50 flowers, and each flower is characterized by four features: sepal length, sepal width, petal length, and petal width.

First, we load the iris dataset and prepare it for training and testing by converting character columns to factors and splitting the data into training and testing sets with an 80-20 split.

```
# Load the iris dataset and convert character columns to factors
iris <- iris %>% mutate_if(is.character, as.factor)

# Train and test split
set.seed(254)
train_indices <- sample(1:nrow(iris), 0.8 * nrow(iris))
train_data <- iris[train_indices, ]
test_data <- iris[-train_indices, ]
```

We will train three neural network models with different configurations of hidden layers: 1. A model with 2 hidden layers, each containing 10 neurons. 2. A model with 3 hidden layers, each containing 20 neurons. 3. A model with 4 hidden layers, each containing 25 neurons.

To measure the time taken to train each model, we define a function `measure_time` that captures the start and end time of the training process and calculates the duration.

```
# Function to measure training time
measure_time <- function(expr) {
  start_time <- Sys.time()
  result <- eval(expr)
  end_time <- Sys.time()
  duration <- end_time - start_time
  return(list(result = result, time = duration))
}
```

To evaluate the performance of each trained model, we define a function `evaluate_model` that predicts the test data and calculates the accuracy of the predictions.

```r
# Function to evaluate model performance
evaluate_model <- function(model, test_data) {
  # Predict using the trained model
  pred <- compute(model, test_data[, -5])$net.result

  # Convert predictions to class labels
  prediction_label <- apply(pred, 1, function(x) names(which.max(x)))

  # Calculate accuracy
  correct_predictions <- sum(test_data$Species == prediction_label)
  accuracy <- (correct_predictions / nrow(test_data)) * 100

  return(list(prediction_label = prediction_label, accuracy = accuracy))
}
```

Now we train the three neural network models with the specified configurations of hidden layers and measure the training time for each model.

```r
# Model with 2 hidden layers (10 neurons each)
time_10_10 <- measure_time({
  neuralnet(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
            data = train_data, hidden = c(10, 10), linear.output = FALSE)
})
result_10_10 <- evaluate_model(time_10_10$result, test_data)
```

Model with 2 Hidden Layers (10 Neurons Each)

```r
# Model with 3 hidden layers (20 neurons each)
time_20_20_20 <- measure_time({
  neuralnet(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
            data = train_data, hidden = c(20, 20, 20), linear.output = FALSE)
})
result_20_20_20 <- evaluate_model(time_20_20_20$result, test_data)
```

Model with 3 Hidden Layers (20 Neurons Each)

```r
# Model with 4 hidden layers (25 neurons each)
time_25_25_25_25 <- measure_time({
  neuralnet(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
            data = train_data, hidden = c(25, 25, 25, 25), linear.output = FALSE)
})
result_25_25_25_25 <- evaluate_model(time_25_25_25_25$result, test_data)
```

Model with 4 Hidden Layers (25 Neurons Each) The table below summarizes the accuracy and training time for each model.

```r
# Create a summary table
summary_table <- data.frame(
  Model = c("10, 10 Hidden Neurons", "20, 20, 20 Hidden Neurons", "25, 25, 25, 25 Hidden Neurons"),
  Accuracy = c(result_10_10$accuracy, result_20_20_20$accuracy, result_25_25_25_25$accuracy),
  Training_Time = c(time_10_10$time, time_20_20_20$time, time_25_25_25_25$time)
)

print(summary_table)
```

```
##                          Model Accuracy  Training_Time
## 1         10, 10 Hidden Neurons        0 0.7549131 secs
## 2     20, 20, 20 Hidden Neurons        0 0.2629550 secs
## 3 25, 25, 25, 25 Hidden Neurons        0 0.3187208 secs
```

Observations

1. **Model Accuracy**: All models showed an accuracy of 0%, indicating that the models failed to generalize the iris dataset. This could be due to incorrect prediction handling or the neural network not being properly trained.
2. **Training Time**: The training times varied significantly among the models. The model with 2 hidden layers (10 neurons each) took the longest to train, while the models with 3 and 4 hidden layers took significantly less time.

Detailed Analysis

The following table provides a detailed comparison of the training times and accuracies for the different models:

Model Configuration	Training Time (secs)	Accuracy
10, 10 Hidden Neurons	0.754913091659546	0
20, 20, 20 Hidden Neurons	0.262954950332642	0
25, 25, 25, 25 Hidden Neurons	0.318720817565918	0

Conclusion

The models did not perform well in terms of accuracy, suggesting that further tuning and validation are needed. The training times suggest that more hidden layers with more neurons do not necessarily lead to better performance and can sometimes reduce training time due to better convergence properties.

Future work should focus on: - Validating the prediction process. - Hyperparameter tuning to improve model performance. - Exploring other neural network architectures and activation functions.