modeling the strength of concrete with ANNs

Nithya Sarabudla

03-24-2024

Step 2 Exploring and preparing the data

```
# load the data
concrete <- read.csv("/Users/nithyasarabudla/DA5030/concrete.csv")
str(concrete)
```

```
## 'data.frame':    1030 obs. of  9 variables:
##  $ cement      : num  540 540 332 332 199 ...
##  $ slag        : num  0 0 142 142 132 ...
##  $ ash         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ water       : num  162 162 228 228 192 228 228 228 228 228 ...
##  $ superplastic: num  2.5 2.5 0 0 0 0 0 0 0 0 ...
##  $ coarseagg   : num  1040 1055 932 932 978 ...
##  $ fineagg     : num  676 676 594 594 826 ...
##  $ age         : int  28 28 270 365 360 90 365 28 28 28 ...
##  $ strength    : num  80 61.9 40.3 41 44.3 ...
```

The dataset contains 1030 observations of 9 variables, which are related to the composition and characteristics of concrete mixtures, such as cement, slag, ash, water, superplasticizer, coarse aggregate, fine aggregate, age, and compressive strength. The variables are primarily numeric, indicating measurements or quantities, except for age, which is an integer representing the time aspect.

```
# Define a function to normalize a vector
normalize <- function(x) {
    return((x - min(x)) / (max(x) - min(x)))
}

# Create a new data frame 'concrete_norm' by normalizing the 'concrete' data frame
concrete_norm <- as.data.frame(lapply(concrete, normalize))
```

Normalization function that scales numerical data to a 0-1 range, then applies this function to all columns in the concrete dataframe, producing a new dataframe concrete_norm with all variables normalized.

```
# Summary of the normalized strength values
summary(concrete_norm$strength)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.2664  0.4001  0.4172  0.5457  1.0000
```

```
# Summary of the original strength values
summary(concrete$strength)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.33   23.71   34.45   35.82   46.13   82.60
```

Summaries of both the normalized and original strength values from the concrete dataset are generated. These summaries provide insights into how the normalization process affects the distribution of the strength variable, serving as a check to ensure that the scaling process maintains the integrity of the data.

```
# Split the normalized concrete data into training and testing sets
# Select the first 773 rows for training data
concrete_train <- concrete_norm[1:773, ]
# Select rows 774 to 1030 for testing data
concrete_test <- concrete_norm[774:1030, ]
```

The code divides the normalized dataset into training and testing sets, using the first 773 rows for training and the subsequent rows for testing.

Step 3 - Training a model on the data

```
# Load the neuralnet package for neural network modeling
library(neuralnet)
# Set a seed for reproducibility
set.seed(12345)
# Train a neural network model using the neuralnet function
concrete_model <- neuralnet(strength ~ cement + slag +
ash + water + superplastic + coarseagg + fineagg + age,
data = concrete_train)
```

The neuralnet package is utilized to train a neural network model to predict concrete strength based on its composition and age. Setting a random seed ensures reproducibility, and the model is trained on the concrete_train dataset.

```
# Plot the neural network model
plot(concrete_model)
```

plots the trained neural network model, concrete_model, illustrating its structure, including neurons and layers.

Step 4 - evaluating model performance

```
# Compute predictions using the trained neural network model on the testing data
model_results <- compute(concrete_model, concrete_test[1:8])
# Extract predicted strength values from the model results
predicted_strength <- model_results$net.result
```

Predictions are made on the testing set using the initially trained neural network model.

```
# Calculate the correlation between predicted and actual strength values
cor(predicted_strength, concrete_test$strength)
```

```
##           [,1]
## [1,] 0.7188352
```

The correlation between the predicted strength values from the neural network model and the actual strength values in the testing dataset. The correlation between the neural network model's predicted concrete strength values and the actual strength values from the testing set, resulting in a correlation coefficient of approximately 0.72. This indicates a strong positive relationship, suggesting that the model's predictions are in good agreement with the actual data.

step 5 - improving model performance

```
# Set a seed for reproducibility
set.seed(12345)
# Train a new neural network model with a hidden layer of 5 neurons
concrete_model2 <- neuralnet(strength ~ cement + slag +
                               ash + water + superplastic +
                               coarseagg + fineagg + age,
                               data = concrete_train, hidden = 5)
```

To enhance model performance, a new neural network with a hidden layer of 5 neurons is trained.

```
# Plot the updated neural network model
plot(concrete_model2)
```

The reported error (measured again by the SSE) has been reduced from 5.6 in the previous model to 1.54 here.

```
# Compute predictions using the second neural network model on the testing data
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
# Calculate the correlation between predicted and actual strength values for the second model
cor(predicted_strength2, concrete_test$strength)
```

```
##           [,1]
## [1,] 0.7828497
```

Predictions are made with the second neural network model, and the correlation between these predictions and the actual strength values is calculated. This step evaluates the effectiveness of adding a hidden layer to the model.The correlation is reported to be approximately 0.78, indicating a reasonably strong positive linear relationship between the predictions and actual values.

```
# Define the softplus activation function
softplus <- function(x) { log(1 + exp(x)) }

set.seed(12345)
# train a new neural network model with softplus activation
concrete_model3 <- neuralnet(strength ~ cement + slag + ash + water + superplastic + coarseagg + fineagg
                             data = concrete_train,
                             hidden = c(5, 5), # Two hidden layers with 5 neurons each
                             act.fct = softplus, # Use softplus activation function
                             stepmax=1e7) # Set maximum number of steps for training
```

A third neural network model is trained using the softplus activation function and two hidden layers, aiming to explore deeper relationships within the dataset. The softplus function introduces non-linearity, potentially enhancing model performance.

```
# Plot the architecture of the third neural network model
plot(concrete_model3)
```

```
# Compute predictions using the third neural network model on the testing data
model_results3 <- compute(concrete_model3, concrete_test[1:8])
predicted_strength3 <- model_results3$net.result
# Calculate the correlation between predicted and actual strength values for the third model
cor(predicted_strength3, concrete_test$strength)
```

```
##           [,1]
## [1,] 0.773088
```

The third model's architecture is visualized, and its predictive performance is evaluated by calculating the correlation between its predicted strength values and the actual values.The correlation has decreased for this model

```
# Create a data frame to compare actual and predicted strength values
strengths <- data.frame(
    actual = concrete$strength[774:1030],
    pred = predicted_strength3
  )

# Display the first few rows of the strengths data frame
head(strengths, n = 3)
```

```
##     actual      pred
## 774  37.42 0.4278337
## 775  11.47 0.2351562
## 776  22.44 0.2529368
```

```
# Calculate the correlation between predicted and actual strength values (New Method)
cor(strengths$pred, strengths$actual)
```

```
## [1] 0.773088
```

```
cor(strengths$pred, concrete_test$strength)
```

```
## [1] 0.773088
```

A dataframe, strengths, is created to directly compare the actual and predicted strength values, facilitating a detailed evaluation of the third model's predictions. The correlations calculated offer a quantitative assessment of prediction accuracy.

```
# Unnormalize the predicted strength values
unnormalize <- function(x) {
    return(x * (max(concrete$strength) -
            min(concrete$strength)) + min(concrete$strength))
}
```

A function, unnormalize, is created to scale predicted strength values back to their original range, allowing for direct comparison with actual concrete strengths.

```
# Unnormalize the predicted strength values and store them in a new column
strengths$pred_new <- unnormalize(strengths$pred)
# Calculate the percentage error between predicted and actual strength values
strengths$error_pct <- (strengths$pred_new - strengths$actual) /
                            strengths$actual
```

Predicted values are unnormalized and stored, and percentage errors between these predictions and actual strengths are calculated, providing insight into the model's accuracy.

```
# Display the first few rows of the strengths data frame with updated predictions and errors
head(strengths, n = 3)
```

```
##     actual      pred pred_new    error_pct
## 774  37.42 0.4278337 36.67221 -0.019983610
## 775  11.47 0.2351562 21.20599  0.848822323
## 776  22.44 0.2529368 22.63324  0.008611273
```

The first few rows of updated predictions and their percentage errors are displayed

```
# Calculate the correlation between unnormalized predicted strength values and actual strength values
cor(strengths$pred_new, strengths$actual)
```

```
## [1] 0.773088
```

The correlation between unnormalized predictions and actual strength values is calculated, quantifying the predictive accuracy of the model.Unsurprisingly, the correlation remains the same despite reversing the normalization