Laboration report in Machine Learning

Computer lab 2 block 1

732A99

Simge Cinar
Duc Tran
William Wiik

Division of Statistics and Machine Learning
Department of Computer Science
Linköping University

07 december 2023

Contents

1 Assignment 3. Neural Networks

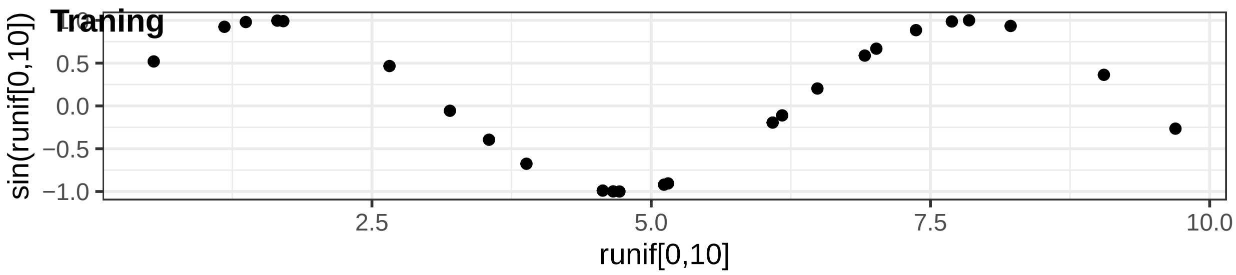
1.1 3.1

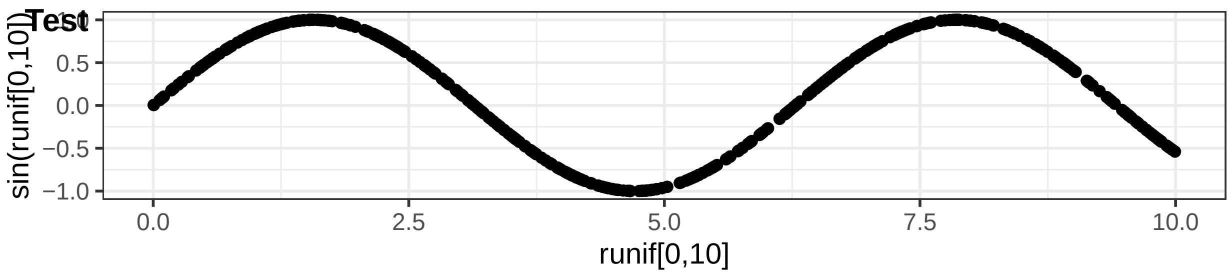
Train a neural network to learn the trigonometric sine function. To do so, sample 500 points uniformly at random in the interval [0, 101010]. Apply the sine function to each point. The resulting value pairs are the data points available to you. Use 25 of the 500 points for training and the rest for test. Use one hidden layer with 10 hidden units. You do not need to apply early stopping. Plot the training and test data, and the predictions of the learned NN on the test data. You should get good results. Comment your results.

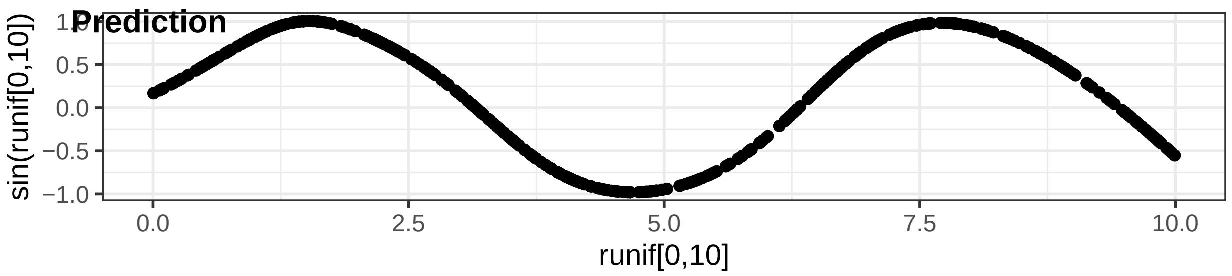
```r
# The data
set.seed(123)
data_runif <- runif(500, min = 0, max = 10)
data_sin <- sin(data_runif)
data_df <- data.frame(runif = data_runif, sin = data_sin)



# Split the data into training and test
n <- dim(data_df)[1]
id <- sample(1:n, 25) # 25 training points

tr_data <- data_df[id, ]
te_data <- data_df[-id, ]
```

```r
# Neural Network
nn_model <- neuralnet(sin ~ runif, data = tr_data, hidden = c(10))

predictions <- predict(nn_model, newdata = te_data) %>% as.vector()

# Adding the predictions to the data.frame with test data
te_data$predicted_sin <- predictions
```

```r
tr_plot <- ggplot(tr_data, aes(runif, sin)) + geom_point() + theme_bw() +
  labs(x = "runif[0,10]", y = "sin(runif[0,10])")

te_plot <- ggplot(te_data, aes(runif, sin)) + geom_point() + theme_bw() +
  labs(x = "runif[0,10]", y = "sin(runif[0,10])")


pred_plot <- ggplot(te_data, aes(runif, predicted_sin)) + geom_point() + theme_bw() +
  labs(x = "runif[0,10]", y = "sin(runif[0,10])")

plot_grid(tr_plot, te_plot,pred_plot, labels = c("Traning","Test","Prediction"),
          label_size = 12, nrow = 3)
```







The neural network have successfully learned the sin(x) pattern as the test plot and prediction plot follows each other very well.

1.2 3.2

In question (1), you used the default logistic (a.k.a. sigmoid) activation function, i.e. act.fct = "logistic". Repeat question (1) with the following custom activation functions: $h_1(x) = x, h_2(x) = max0, x$ and $h_3(x) = \ln(1 + expx)$ (a.k.a. linear, ReLU and softplus). See the help file of the neuralnet package to learn how to use custom activation functions. Plot and comment your results.

1.3 3.3

Sample 500 points uniformly at random in the interval [0, 505050], and apply the sine function to each point. Use the NN learned in question (1) to predict the sine function value for these new 500 points. You should get mixed results. Plot and comment your results.

1.4 3.4

In question (3), the predictions seem to converge to some value. Explain why this happens. To answer this question, you may need to get access to the weights of the NN learned. You can do it by running *nn* or *nn$weights* where *nn* is the NN learned.

1.5 3.5

Sample 500 points uniformly at random in the interval [0, 101010], and apply the sine func-tion to each point. Use all these points as training points for learning a NN that tries to predict x from $sin(x)$, i.e. unlike before when the goal was to predict $sin(x)$ from x. Use the learned NN to predict the training data. You should get bad results. Plot and comment your results. Help: Some people get a convergence error in this ques- tion. It can be solved by stopping the training before reaching convergence by setting threshold = 0.1.

```r
set.seed(1234567890)
Var <- runif(500, 0, 10)
mydata <- data.frame(Var, Sin = sin(Var))
tr <- mydata[1:25,] # Training
te <- mydata[26:500,] # Test

# Random initialization of the weights in the interval [-1, 1]
winit <- # Your code here
nn <- neuralnet# (Your code here)


# Plot of the training data (black), test data (blue), and predictions (red)
# plot(tr, cex = 2)
# points(te, col = "blue", cex = 1)
# points(te[,1], predict(nn, te), col = "red", cex = 1)
```