LSTM: Long Short-Term Memory (LSTM) networks are recurrent neural networks that incorporate gating mechanisms to mitigate vanishing/exploding gradients and capture long-range temporal dependencies. In a sliding-window setup, the sequence of lagged inputs is processed to output the next-step forecast. Key hyperparameters include the input window length, hidden size, and number of training epochs.

Objective: Train and evaluate an LSTM model for time-series forecasting with sliding windows, including normalization, fitting, and test evaluation.

# Time Series Regression - LSTM  
  
# Installing packages (if needed)  
  
#install.packages("tspredit")

# Loading the packages  
library(daltoolbox)  
library(daltoolboxdp)  
library(tspredit)

# Series for study and sliding windows  
  
data(tsd)  
ts <- ts\_data(tsd$y, 10)  
ts\_head(ts, 3)

## t9 t8 t7 t6 t5 t4 t3 t2 t1  
## [1,] 0.0000000 0.2474040 0.4794255 0.6816388 0.8414710 0.9489846 0.9974950 0.9839859 0.9092974  
## [2,] 0.2474040 0.4794255 0.6816388 0.8414710 0.9489846 0.9974950 0.9839859 0.9092974 0.7780732  
## [3,] 0.4794255 0.6816388 0.8414710 0.9489846 0.9974950 0.9839859 0.9092974 0.7780732 0.5984721  
## t0  
## [1,] 0.7780732  
## [2,] 0.5984721  
## [3,] 0.3816610

# Series visualization  
library(ggplot2)  
plot\_ts(x=tsd$x, y=tsd$y) + theme(text = element\_text(size=16))



# Train-test split and projection (X, y)  
  
samp <- ts\_sample(ts, test\_size = 5)  
io\_train <- ts\_projection(samp$train)  
io\_test <- ts\_projection(samp$test)

# Training the LSTM model  
  
model <- ts\_lstm(ts\_norm\_gminmax(), input\_size=4, epochs=10000)  
model <- fit(model, x=io\_train$input, y=io\_train$output)

# Fit evaluation (train)  
  
adjust <- predict(model, io\_train$input)  
adjust <- as.vector(adjust)  
output <- as.vector(io\_train$output)  
ev\_adjust <- evaluate(model, output, adjust)  
ev\_adjust$mse

## [1] 0.0001092335

# Forecast on test set  
  
steps\_ahead <- 1  
io\_test <- ts\_projection(samp$test)  
prediction <- predict(model, x=io\_test$input, steps\_ahead=steps\_ahead)  
prediction <- as.vector(prediction)  
  
output <- as.vector(io\_test$output)  
if (steps\_ahead > 1)  
 output <- output[1:steps\_ahead]  
  
print(sprintf("%.2f, %.2f", output, prediction))

## [1] "0.41, 0.42" "0.17, 0.18" "-0.08, -0.07" "-0.32, -0.32" "-0.54, -0.55"

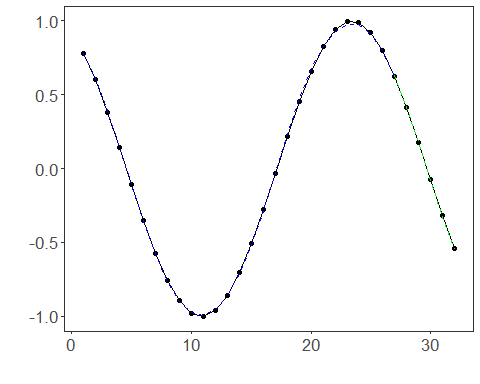
# Test evaluation  
  
ev\_test <- evaluate(model, output, prediction)  
print(head(ev\_test$metrics))

## mse smape R2  
## 1 3.086096e-05 0.01748563 0.9997335

print(sprintf("smape: %.2f", 100\*ev\_test$metrics$smape))

## [1] "smape: 1.75"

# Plot results  
  
yvalues <- c(io\_train$output, io\_test$output)  
plot\_ts\_pred(y=yvalues, yadj=adjust, ypre=prediction) + theme(text = element\_text(size=16))



References - S. Hochreiter and J. Schmidhuber (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.