Support Vector Regression: SVR fits a function that balances flatness with tolerance to errors using the ε-insensitive loss. Kernel functions (e.g., radial basis function) enable nonlinear regression in the sliding-window feature space, where each row encodes the latest lags of the series. Important hyperparameters include the kernel, cost (C), epsilon (ε), and kernel-specific parameters. SVR is robust to outliers and works well with properly scaled inputs.

Objective: Demonstrate time-series forecasting with SVM (Support Vector Regression) using sliding windows, normalization, model fitting, and evaluation with metrics and plots.

# Time Series Regression - SVM  
  
# Installing the package (if needed)  
#install.packages("tspredit")

# Loading the packages  
library(daltoolbox)  
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)

# Preprocessing (global min-max normalization)  
  
preproc <- ts\_norm\_gminmax()

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

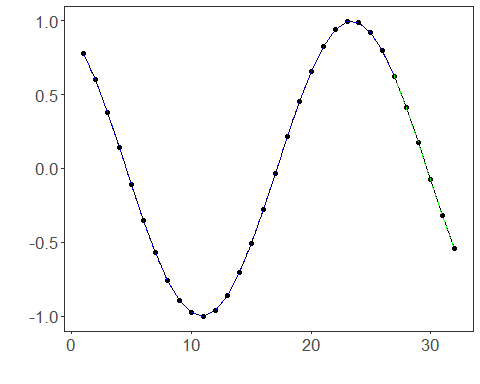
# Fit evaluation (train)  
  
adjust <- predict(model, io\_train$input)  
adjust <- as.vector(adjust)  
output <- as.vector(io\_train$output)  
ev\_adjust <- evaluate(model, output, adjust)  
ev\_adjust$mse

## [1] 5.130098e-07

# Forecast on test set (5 steps ahead)  
  
prediction <- predict(model, x=io\_test$input[1,], steps\_ahead=5)  
prediction <- as.vector(prediction)  
output <- as.vector(io\_test$output)  
ev\_test <- evaluate(model, output, prediction)  
ev\_test

## $values  
## [1] 0.41211849 0.17388949 -0.07515112 -0.31951919 -0.54402111  
##   
## $prediction  
## [1] 0.41268934 0.17333448 -0.07563299 -0.31983409 -0.54481781  
##   
## $smape  
## [1] 0.002684197  
##   
## $mse  
## [1] 3.199991e-07  
##   
## $R2  
## [1] 0.9999972  
##   
## $metrics  
## mse smape R2  
## 1 3.199991e-07 0.002684197 0.9999972

# Plot comparing actual vs fit (train) and forecast (test)  
  
yvalues <- c(io\_train$output, io\_test$output)  
plot\_ts\_pred(y=yvalues, yadj=adjust, ypre=prediction) + theme(text = element\_text(size=16))



References - A. J. Smola and B. Schölkopf (2004). A tutorial on support vector regression. Statistics and Computing, 14(3), 199–222.