## Time Series Encoding and Reconstruction (encode-decode)

Time series windows of size p are encoded into k-dimensional latent vectors and then decoded back to p dimensions. Training minimizes reconstruction loss so that latent codes capture the essential structure of windows, enabling quality assessment via reconstruction error.

This example shows how to transform a time series into windows (p) and train an autoencoder to encode (p -> k) and reconstruct (k -> p) these windows, allowing evaluation of reconstruction quality.

Prerequisites - R packages: daltoolbox, ggplot2 - Python with PyTorch accessible via reticulate (backend called internally)

# Loading required packages  
library(daltoolbox)

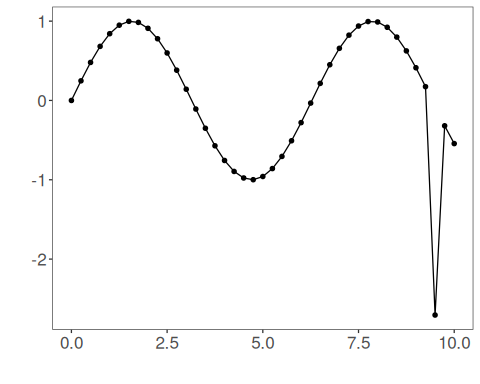
Series for study

data(tsd)  
tsd$y[39] <- tsd$y[39] \* 6 # inject a synthetic outlier for illustration

sw\_size <- 5 # sliding window size (p)  
ts <- ts\_data(tsd$y, sw\_size) # series -> windows with p columns  
ts\_head(ts, 3)

## t4 t3 t2 t1 t0  
## [1,] 0.0000000 0.2474040 0.4794255 0.6816388 0.8414710  
## [2,] 0.2474040 0.4794255 0.6816388 0.8414710 0.9489846  
## [3,] 0.4794255 0.6816388 0.8414710 0.9489846 0.9974950

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



Data sampling

samp <- ts\_sample(ts, test\_size = 5)  
train <- as.data.frame(samp$train)  
test <- as.data.frame(samp$test)

Train the model (encode-decode)

auto <- autoenc\_ed(5, 3) # 5 -> 3 -> 5 dimensions  
auto <- fit(auto, train)

Reconstruction evaluation (train)

print(head(train)) # original windows (p columns)

## t4 t3 t2 t1 t0  
## 1 0.0000000 0.2474040 0.4794255 0.6816388 0.8414710  
## 2 0.2474040 0.4794255 0.6816388 0.8414710 0.9489846  
## 3 0.4794255 0.6816388 0.8414710 0.9489846 0.9974950  
## 4 0.6816388 0.8414710 0.9489846 0.9974950 0.9839859  
## 5 0.8414710 0.9489846 0.9974950 0.9839859 0.9092974  
## 6 0.9489846 0.9974950 0.9839859 0.9092974 0.7780732

result <- transform(auto, train) # reconstructed windows (p columns)  
print(head(result))

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0.009684362 0.2480524 0.4798980 0.6901832 0.8448679  
## [2,] 0.252034873 0.4790744 0.6818609 0.8459865 0.9527500  
## [3,] 0.477562696 0.6824882 0.8416348 0.9459279 0.9943677  
## [4,] 0.675233424 0.8440890 0.9486990 0.9923663 0.9829662  
## [5,] 0.844732940 0.9475346 0.9979154 0.9820423 0.9095937  
## [6,] 0.950951040 0.9950162 0.9845353 0.9110637 0.7810502

Reconstruction of the test set

print(head(test))

## t4 t3 t2 t1 t0  
## 1 0.9893582 0.9226042 0.7984871 0.6247240 0.4121185  
## 2 0.9226042 0.7984871 0.6247240 0.4121185 0.1738895  
## 3 0.7984871 0.6247240 0.4121185 0.1738895 -2.7054403  
## 4 0.6247240 0.4121185 0.1738895 -2.7054403 -0.3195192  
## 5 0.4121185 0.1738895 -2.7054403 -0.3195192 -0.5440211

result <- transform(auto, test)  
print(head(result))

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0.9888958 0.9224395 0.7964306 0.6299026 0.4069477  
## [2,] 0.9287465 0.8036430 0.6261265 0.4086141 0.1670102  
## [3,] 0.8167595 0.2774367 -0.2963374 -0.8056232 -1.2503191  
## [4,] 0.1617254 -0.1050199 -0.4034734 -0.6320236 -0.8370287  
## [5,] -0.2455083 -0.5472478 -0.8183674 -1.0323898 -1.1752121

References - Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. (Chapter on Autoencoders)