## Denoising Autoencoder (encode)

This example demonstrates how to use a denoising autoencoder to learn a robust encoding of time-series windows. During training, noise is added to the input.

Prerequisites - Python with PyTorch accessible via reticulate - R packages: daltoolbox, tspredit, daltoolboxdp, ggplot2

Quick notes - Idea: add noise to the input and train the model to recover the clean signal. - Benefit: latent representations that are more robust to perturbations.

# Denoising Autoencoder transformation (encode)  
  
# Considering a dataset with $p$ numerical attributes.   
  
# The goal of the autoencoder is to reduce the dimension of $p$ to $k$, such that these $k$ attributes are enough to recompose the original $p$ attributes.   
  
# Installing packages  
#install.packages("tspredit")  
#install.packages("daltoolboxdp")

# Loading packages  
library(daltoolbox)  
library(tspredit)  
library(daltoolboxdp)  
library(ggplot2)

# Example dataset (series -> windows)   
data(tsd)  
  
sw\_size <- 5  
ts <- ts\_data(tsd$y, sw\_size)  
  
ts\_head(ts)

## 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

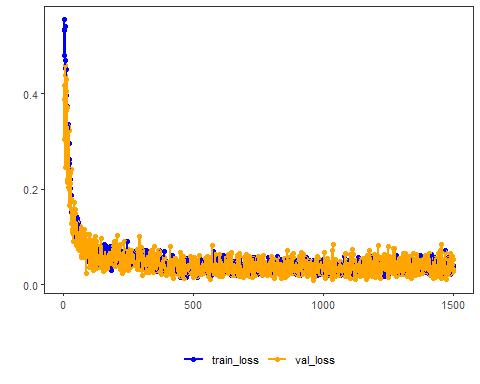
# Normalization (min-max by group)  
preproc <- ts\_norm\_gminmax()  
preproc <- fit(preproc, ts)  
ts <- transform(preproc, ts)  
  
ts\_head(ts)

## t4 t3 t2 t1 t0  
## [1,] 0.5004502 0.6243512 0.7405486 0.8418178 0.9218625  
## [2,] 0.6243512 0.7405486 0.8418178 0.9218625 0.9757058  
## [3,] 0.7405486 0.8418178 0.9218625 0.9757058 1.0000000  
## [4,] 0.8418178 0.9218625 0.9757058 1.0000000 0.9932346  
## [5,] 0.9218625 0.9757058 1.0000000 0.9932346 0.9558303  
## [6,] 0.9757058 1.0000000 0.9932346 0.9558303 0.8901126

# Train/test split  
samp <- ts\_sample(ts, test\_size = 10)  
train <- as.data.frame(samp$train)  
test <- as.data.frame(samp$test)

# Training autoencoder (reduce 5 -> 3)  
auto <- autoenc\_denoise\_e(5, 3, num\_epochs=1500)  
auto <- fit(auto, train)

fit\_loss <- data.frame(x=1:length(auto$train\_loss), train\_loss=auto$train\_loss,val\_loss=auto$val\_loss)  
  
grf <- plot\_series(fit\_loss, colors=c('Blue','Orange'))  
plot(grf)



# Convergence should be stable; too much noise can hinder fitting.

# Testing the autoencoder  
# Show test samples and display encoding  
print(head(test))

## t4 t3 t2 t1 t0  
## 1 0.7258342 0.8294719 0.9126527 0.9702046 0.9985496  
## 2 0.8294719 0.9126527 0.9702046 0.9985496 0.9959251  
## 3 0.9126527 0.9702046 0.9985496 0.9959251 0.9624944  
## 4 0.9702046 0.9985496 0.9959251 0.9624944 0.9003360  
## 5 0.9985496 0.9959251 0.9624944 0.9003360 0.8133146  
## 6 0.9959251 0.9624944 0.9003360 0.8133146 0.7068409

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

## [,1] [,2] [,3]  
## [1,] 0.349251926 1.731498 -1.331826  
## [2,] 0.271563470 1.785194 -1.406869  
## [3,] 0.180959493 1.795828 -1.438733  
## [4,] 0.085452512 1.760640 -1.429571  
## [5,] -0.009019494 1.681816 -1.379952  
## [6,] -0.084728986 1.567757 -1.294433