## Adversarial Autoencoder (encode)

Adversarial Autoencoders augment the autoencoder objective with an adversarial discriminator in latent space. The encoder is trained to both reconstruct inputs (via the decoder) and produce latent codes whose distribution matches a desired prior, improving latent structure.

This example shows how to train and use an Adversarial Autoencoder (AAE) to encode windows of a time series, reducing from p to k dimensions while imposing a desired distribution in the latent space via an adversarial discriminator.

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

Quick notes - Architecture: encoder + decoder, with a discriminator in the latent space for adversarial regularization. - Goal: learn compact representations (k) that preserve information from the original windows (p). - Important hyperparameters: num\_epochs, batch\_size, learning rate defined internally.

# Installing example dependencies (if needed)  
#install.packages("tspredit")  
#install.packages("daltoolboxdp")

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

# Example dataset (series -> windows)  
data(tsd)  
  
sw\_size <- 5 # sliding window size (p)  
ts <- ts\_data(tsd$y, sw\_size) # convert series into windows with p columns  
  
ts\_head(ts) # preview first rows

## 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

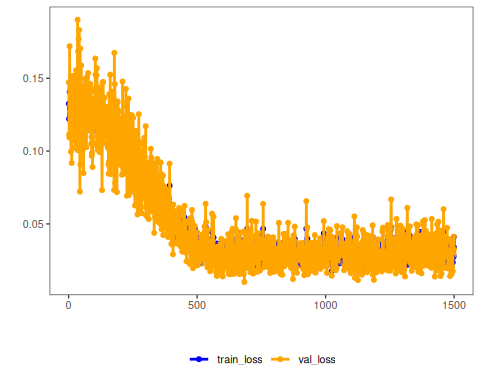
# Normalization (min-max by group)  
# Keeps each column (window step) on the same [0,1] scale  
preproc <- ts\_norm\_gminmax()  
preproc <- fit(preproc, ts)  
ts <- transform(preproc, ts)  
  
ts\_head(ts)

## 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)

# Creating the adversarial autoencoder: reduce from 5 -> 3 dimensions (p -> k)  
# - batch\_size: training batch size per step  
# - num\_epochs: number of training epochs  
auto <- autoenc\_adv\_e(5, 3, batch\_size = 3, num\_epochs = 1500)  
  
# Training the model on the train set  
auto <- fit(auto, train)

# Learning curves (train and validation loss per epoch)  
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)



# Testing the autoencoder (encoding only)  
# Show samples from the test set and the resulting encoding (k columns)  
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,] 2.050356 -2.742019 0.2499253  
## [2,] 4.045651 -3.392818 -0.1705142  
## [3,] 4.608471 -5.617803 1.3165911  
## [4,] 2.599201 -2.231373 -0.1265608  
## [5,] 1.839835 -2.960985 -0.5946820  
## [6,] 2.958924 -3.220733 0.5829004

References - Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial Autoencoders. arXiv:1511.05644.