Quick Guide to the 'slants' Package

Jiaqi Liu, Jie Ding

4/26/2019

This vignette serves as a quick guide on how to use the 'slants' R package to perform nonlinear modeling and prediction for online streaming data. This user guide is still under construction. A reference to this package is here

Synthetic Data Experiment 1

The purpose of this experiment is to show the performance of SLANTS in stationary environment where the data generating model is fixed over time. We will generate two-dimensional time series data and treat X_2 as our y. We want to detect the true relationship between the two dimensional time series and y. The following function is used to creating lagged version of a time series vector.

```
library(slants)
utils.lag <- function(ts, lag = 1, pad = NA) {
  # return the lagged version of a time series vector
  return(c(rep(pad, lag), ts[1:(length(ts)-lag)]))
}
```

Here we generate two-dimensional time series with lag order $L = 8$, standard Gaussian noise and 2000 observations. D here stands for dimension and L here stands for lag. The true data generating model is $X_{t,2} = 2X_{t-1,1}^2 - 0.8 * X_{t-7,1} + e_{t,1}, X_{t,1} = e_{t,2}$, where $e_{t,1} \sim \mathcal{N}(0, 0.2^2), e_{t,2} \sim \mathcal{N}(0, 1)$.

```
Ex1 <- (function(N=2000, D=2, L=8){
  err <- matrix(rnorm(D*(N+L)), N+L, D)
  X <- err[ ,1]
  X <- cbind(X, 1 * utils.lag(X,1)^2 - 0.8 * utils.lag(X,7) + 0.2 * err[,2])
  list(N=N, D=D, L=L, X=X[-(1:L),], y=X[-(1:L),2])
})()
```

We set parameters for data preprocessing and model training. *Lambda parameter* we used here is group LASSO penalty coefficient which allows to control the level or shrinkage, *SrhinkStepSize* is the shrink step size, *Order* is the order for B spline, *nBspline* is the number of B splines to fit the nonlinear model, *SpaTol_gamma* is the tolerance of three gamma channels used for automatic tuning, *MoveSize* is the ratio between the channels which controls the distance between channels, *gamma_init* is the initial value for three channels. For robustness, users need to choose a good initial value (which is usually small). Finally, *alpha2_init* is the initial value for innovation parameter which helps to decompose the problem in EM

algorithm. Details on the choices of tuning parameters are elaborated in the reference paper. All the tuning parameters have default values in the package.

```
Ex1_algo <- (function(experiment_config){
  ec <- experiment_config # short name
  lambda <- 1/c(1:ec$N) # same as batch
  shrinkStepSize <- 1/c(1:ec$N)
  list(
    order = 3,
    nBspline = 10,
    lambda = lambda,
    shrinkStepSize = shrinkStepSize,
    # if the performance does not exceed this ratio, the sparser (larger gamma) is preferred
    spaTol_gamma = 1.1,
    moveSize = 10^(0.4), # multipler to move among channels
    safeShrink_gamma = 10^(0.1), # to adjust gamma before they get too crazy
    gamma_init = 0.0001, #lambda in the paper,three channel
    alpha2_init = 0.05 #tao in the paper
  )
})(Ex1)
```

'getPreprocess' converts two-dimensional data into $N \times (2 \cdot 8 \cdot D \cdot L)$ matrix and also assign knots for B splines.

```
Ex1_algo <- c(Ex1_algo, do.call(getPreprocess, c(Ex1, Ex1_algo)))
```

getSequentialNonlinearModel uses EM algorithm and B splines to get the fitted model.
The historical values of automatically tuned parameters and estimated coefficients are all stored in EX1_result.

```
Ex1_result <- do.call(getSequentialNonlinearModel, c(list(ifPrint=TRUE, testSize=50),Ex1,
Ex1_algo))
```

Users can visualize the evolution of various tuning patameters.

```
plot(Ex1_result$gamma_opt,type = "l")
plot(Ex1_result$alpha_opt,type = "l",ylab = "Tao2")
```

Users can also visualize the marginal relation between the response y and each active predictor X.

```
plotcoeff(Ex1_result$beta_opt,Ex1_algo$knots,Ex1_algo$nBspline)
```