Package ‘dynamo’

February 23, 2018

Type Package

Title Fit a Stochastic Dynamical Array Model to Array Data

Version 1.0

Date 2018-02-23

Author Adam Lund

Maintainer Adam Lund <adam.lund@math.ku.dk>

Description An implementation of the method proposed in Lund and Hansen (2018) for fitting 3-dimensional dynamical array models. The implementation is based on the glamlasso package, see Lund et al. (2017) <doi:10.1080/10618600.2017.1279548>, for efficient design matrix free lasso regularized estimation in a generalized linear array model. The implementation uses a block relaxation scheme to fit each individual component in the model using functions from the glamlasso package.

License GPL (>= 2)

Depends glamlasso (>= 3.0), abind, MortalitySmooth

Imports Rcpp (>= 0.12.12)

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-02-23 17:28:37 UTC

R topics documented:

fitmodel *Sparse Estimation of Dynamical Array Models*

Description

An implementation of the method proposed in *Lund and Hansen, 2018* for fitting 3-dimensional dynamical array models. The implementation is based on the glamlasso package, see *Lund et. al, 2017*, for efficient design matrix free lasso regularized estimation in a generalized linear array model. The implementation uses a block relaxation scheme to fit each individual component in the model using functions from the glamlasso package.

Usage

```
fitmodel(V,
         phix, phiy, phil, phit,
         penaltyfactor,
         nlambda = 15,
         lambdaminratio = 0.0001,
         reltolinner = 10^-4,
         reltola = 10^-4,
         maxalt = 10)
```

Arguments

`V`	is the $N_x \times Ny \times Nt \times G$ array containing the data
`phix`	is a $N_x \times px$ matrix containing spatial basis functions
`phiy`	is a $N_y \times py$ matrix containing spatial basis functions
`phil`	is a $Lp1 \times pl$ matrix containing temporal basis functions
`phit`	is a $Nt \times pt$ matrix containing temporal basis functions
`penaltyfactor`	An array of size $p_1 \times \cdots \times p_d$. Is multiplied with each element in `lambda` to allow differential shrinkage on the coefficients.
`nlambda`	the number of `lambda` values to fit
`lambdaminratio`	the smallest value for `lambda`, given as a fraction of
`reltolinner`	the convergence tolerance used with glamlasso
`reltola`	the convergence tolerance used for the alternation loop
`maxalt`	maximum nuber of alternations

Details

This package contains an implementation of the method proposed in *Lund and Hansen, 2018* for fitting a (partial) 3-dimensional 3-component dynamical array model to a $N_x \times N_y \times N_t \times G$ data array V, where $N_x, N_y, N_t, G \in \{1, 2, \ldots\}$. Note that N_x, N_y, N_t gives the number of observations in the two spatial dimensions and the temporal dimension respectively and G gives the number of trials. Let L be positive integer giving the length of the modelled delay, $M := N_t - L - 1$ and ϕ^i be a $N_i \times p_i$ matrix for $i \in \{x, y, l\}$. Then define a $M \times p_x p_y p_l$ matrix

$$\Phi_g^{xyt} = \left(\begin{array}{c} vec(\Phi_{1,g}^{xyl}) \\ \vdots \\ vec(\Phi_{M,g}^{xyl}) \end{array} \right),$$

for each $g \in \{1, \ldots, G\}$. Here using the so called rotated H-transform ρ from *Currie et al., 2006*, $\Phi_{k,g}^{xyt}$ is a $p_x \times p_y \times p_l$ array that, for each $k \in \{1, \ldots, M\}$, can be computed as

$$\Phi_{k,g}^{xyl} := \rho((\phi^l)^\top, \rho((\phi^y)^\top, \rho((\phi^x)^\top, V_{,,(k-L):(k-1),g}))).$$

Then we can write the model equation as for each trial g as

$$V_{,,(L+1):N_t,g} = vec(\rho(\phi^t, \rho(\phi^y, \rho(\phi^x, A))) + \rho(\Phi^{xyt}, \rho(\phi^y, \rho(\phi^x, B))) + V_{,,-1:(M-1),g} \odot C + E$$

where A and B are resp. $p_x \times p_y \times p_t$ and $p_x \times p_y \times p_x p_y p_l$ coefficient arrays that are estimated and C is a $N_x \times N_y \times M$ array containing M copies of the $N_x \times N_y$ matrix $\rho(\phi^y, \rho(\phi^x, \Gamma))$, where Γ is a $p_x \times p_y$ coefficient matrix that is estimated. Finally E is a $N_x \times N_y \times M$ array with Gaussian noise. See *Lund and Hansen, 2018* for more details.

Value

An object with S3 Class "dynamo".

`Out`	A list where the first G items are the individual fits to the trials containing:
`Out$V`	The $N_x \times N_y \times N_t$ data array for trial g.
`Out$Phixyl`	a $M \times p_x p_y p_l$ matrix, the convolution tensor.
`Out$BetaS`	$p_x p_y p_t \times$`nlambda` matrix containing the estimated parameters for the stimulus component for each λ value
`Out$BetaF`	$p_x p_y p_x p_y p_l \times$`nlambda` matrix containing the estimated parameters for the filter component for each λ value
`Out$BetaH`	$p_x p_y \times$`nlambda` matrix containing the estimated parameters for the instantaneous memory component for each λ value
`Out$lambda`	vector of length `nlambda` containing the penalty parameters.
`Out$Obj`	`maxalt`$\times$`nlambda` matrix containing the obective values for each alternation and each λ
`phix`	is a $N_x \times p_x$ matrix containing the basis functions over the spatial x domain.
`phiy`	is a $N_y \times p_y$ matrix containing the basis functions over the spatial y domain.
`phil`	is a $L + 1 \times p_l$ matrix containing the basis functions over the temporal domain for the delay.
`phit`	is a $M \times p_t$ matrix containing the basis functions over the temporal domain for the stimulus.

Author(s)

Adam Lund

Maintainer: Adam Lund, `<adam.lund@math.ku.dk>`

References

Lund, A. and N. R. Hansen (2018). Sparse Network Estimation for Dynamical Spatio-temporal Array Models. *In preparation*, url = https://.

Lund, A., M. Vincent, and N. R. Hansen (2017). Penalized estimation in large-scale generalized linear array models. *Journal of Computational and Graphical Statistics*, 26, 3, 709-724. url = https://doi.org/10.1080/10618600.2017.1279548.

Currie, I. D., M. Durban, and P. H. C. Eilers (2006). Generalized linear array models with applications to multidimensional smoothing. *Journal of the Royal Statistical Society. Series B*. 68, 259-280. url = http://dx.doi.org/10.1111/j.1467-9868.2006.00543.x.

Examples

```
## Not run:
# Example showcasing the application from Lund and Hansen (2018).
################################ data
data(V)
################################ constants
Nx <- dim(V)[1]
Ny <- dim(V)[2]
Nt <- dim(V)[3]
L <- 50          #lag length in steps
Lp1 <- L + 1     #number of lag time points (= initial points)
t0 <- 0
M <- Nt - Lp1      #number of modelled time points
sl <- floor(200 / 0.6136) - 0 + 1   #stim start counted from -tau
sr <- sl + floor(250 / 0.6136)  #stim end counted from -tau
##no. of basis func.
px <- 8
py <- 8
pl <- max(ceiling(Lp1 / 5), 4)
pt <- max(ceiling((Nt - sl) / 25), 4)
degx <- 2
degy <- 2
degl <- 3
degt <- 3
################################ basis functions
library(MortalitySmooth)
phix <- round(MortSmooth_bbase(x = 1:Nx, xl = 1, xr = Nx, ndx = px - degx, deg = degx), 10)
phiy <- round(MortSmooth_bbase(x = 1:Ny, xl = 1, xr = Ny, ndx = py - degy, deg = degy), 10)
phil <- round(MortSmooth_bbase(x = -tau:(t0 - 1), xl = -tau, xr = (t0 - 1),
ndx = pl - degl, deg = degl), 10)
phit <- round(MortSmooth_bbase(x = sl:Nt, xl = sl, xr = Nt, ndx = pt - degt, deg = degt), 10)
phit <- rbind(matrix(0,  (sl - 1) - Lp1, dim(phit)[2]), phit)
################################ penalty weights
wt <- c(1, 1, 2, 2, 3, 3, 3, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 3, 3, 3, 3, 2, 1, 1)
penSlist <- list(matrix(1,  px,  py), matrix(1 / wt, dim(phit)[2], 1))
penF <- array(1, c(px, py, px * py * pl))
penH <- matrix(1, px, py)
penaltyfactor <- list(penSlist, penF, penH)
################################ run algorithm
system.time({Fit <- fitmodel(V,
```

```
                              phix, phiy, phil, phit,
                              penaltyfactor,
                              nlambda = 10,
                              lambdaminratio = 10^-1,
                              reltolinner = 10^-4,
                              reltola = 10^-4,
                              maxalt = 10)})

################################  get one fit
modelno <- 6
fit <- Fit[[1]]
A <- array(fit$BetaS[, modelno], c(px, py, pt))
B <- array(fit$BetaF[, modelno], c(px, py, px * py * pl))
C <- array(fit$BetaH[, modelno], c(px, py))
shat <- RH(phit, RH(phiy, RH(phix, A)))
beta <- array(B, c(px, py, px, py, pl))
what <- RH(phil, RH(phiy, RH(phix, RH(phiy, RH(phix, beta)))))
################################ compute summary network quantities
wbar <- apply(abs(what), c(1, 2, 3, 4), sum)
win <- apply(wbar, c(1, 2), sum)
wout <- apply(wbar, c(3, 4), sum)
indeg <- apply((what != 0), c(1, 2), sum)
outdeg <- apply((what != 0), c(3, 4), sum)
winnorm <- ifelse(indeg > 0, win / indeg, win)
woutnorm <- ifelse(outdeg > 0, wout / outdeg, wout)
################################ plot summary network quantities
par(mfrow = c(2, 2), oma = c(0, 0 ,1, 0), mar = c(0, 0, 1, 0))
image(winnorm, main = paste("Time aggregated in effects"), axes = FALSE)
image(woutnorm, main = paste("Time aggregated out effects"), axes = FALSE)
timepoint <- which(shat[9, 9, ] == min(shat[9, 9, ]))
image(shat[,, timepoint], axes = FALSE, main = "Stimulus function")
plot(shat[1, 1, ], ylim = range(shat), type="l", main = "Stimulus function")
for(i in 1:Nx){for(j in 1:Ny){lines(shat[i, j, ])}}
abline(v = sl - Lp1, lty = 2)
abline(v = sr - Lp1, lty = 2)

## End(Not run)
```

V *VSD imaging of a live ferret brain.*

Description

This data set contains one trial of processed voltage sentisitve dye reocordings from animal 308. The original raw data set contains integer values indicitating the flourescence. See *Roland et. al, 2006*

Format

A three dimensional $25 \times 25 \times 977$ array.

References

Per E. Roland, Akitoshi Hanazawa, Calle Undeman, David Eriksson, Tamas Tompa, Hiroyuki Nakamura, Sonata Valentiniene and Bashir Ahmed (2006). Cortical feedback depolarization waves: A mechanism of top-down influence on early visual areas. *PNAS*, 103, 33, 12586-12591. url = https://doi.org/10.1073/pnas.0604925103.

Index