Package 'BoolFilter'

January 8, 2017

Type Package

Title Optimal Estimation of Partially Observed Boolean Dynamical Systems

Version 1.0

Date 2017-01-08

Author Levi McClenny, Mahdi Imani, Ulisses Braga-Neto

Maintainer Levi McClenny <levimcclenny@tamu.edu>

Depends R (>= 2.15.0), Rlab, BoolNet

BuildManual yes

Description Implementation of optimal estimation of Partially-Observed Boolean Dynamical Systems. The optimal solution is the Boolean Kalman Filtering algorithm, as well as modifications of the same.

License Artistic-2.0

NeedsCompilation no

R topics documented:

`BoolFilter-package` *Optimal Estimation of Partially-Observed Boolean Dynamical Systems*

Description

Implementation of optimal estimation of Partially-Observed Boolean Dynamical Systems. The optimal solution is the Boolean Kalman Filtering algorithm, as well as modifications of the same.

Details

Package:	BoolFilter
Type:	Package
Version:	1.0
Date:	2017-01-08
License:	Artistic-2.0

Estimation of Partially-Observed Boolean Dynamical Systems (POBDS) is handled by various algorithms, revolving primarily around the Boolean Kalman Filter (*BKF*). Other similar algorithms have been developed and documented in the BoolFilter package, including *BKS* for batch datasets and the *SIR_BKF* for large POBDS.

These algorithms can be run on data with various types of noise including:

- Bernoulli
- Gaussian
- Poisson
- Negative-Binomial

These types of noise models are included to handle different potential observation noise that could be encountered in real applications.

These observation noise models can be simulated in the *simulateNetwork* function, which can generate both state and observation trajectories of user defined length and observation model preference.

Additionally, the *plotTrajectory* function can be implemented to visualize the trajectories of state variables.

Author(s)

Levi McClenny, Mahdi Imani, Ulisses Braga-Neto

Maintainer: Levi McClenny <levimcclenny@tamu.edu>

References

Braga-Neto U. Optimal state estimation for Boolean dynamical systems. In 2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR) 2011 Nov 6 (pp. 1050-1054). IEEE.

Imani, M., & Braga-Neto, U. Maximum-likelihood adaptive filter for partially-observed boolean dynamical systems. IEEE transaction on Signal Processing, 65:359-371, 2017.

Further references are listed in the corresponding reference sections

Examples

```
data(p53net_DNAdsb0)

#Simulate data from a Bernoulli observation model
data <- simulateNetwork(p53net_DNAdsb0, n.data = 100, p = 0.02,
                        obsModel = list(type = "Bernoulli",
                                        p = 0.02))

#Derive an estimate of the network using a BKF approach
Results <- BKF(data$Y, p53net_DNAdsb0, .02,
                        obsModel = list(type = "Bernoulli",
                                        p = 0.02))

#View network approximation vs. correct trajectory
plotTrajectory(Results$Xhat,
                labels = p53net_DNAdsb0$genes,
                dataset2 = data$X,
                compare = TRUE)
```

BKF *Boolean Kalman Filter*

Description

Implements the Boolean Kalman Filter to derive the optimal MMSE estimate of state of a Partially Observed Boolean Dynamical System

Usage

```
BKF(Y, net, p, obsModel)
```

Arguments

Y	Time series of noisy observations of the Boolean regulatory network. Each row and column correspond to a specific Boolean variable and time point respectively.
net	A Boolean Network object (specified in BoolNet vernacular) that the time series of observations presented in Y is based on
p	Intensity of Bernoulli process noise
obsModel	Parameters for the chosen observation model.

Details

A novel state-space signal model has been proposed for stochastic Boolean dynamical systems observed through noisy measurements, of which the Boolean Kalman Filter is the optimal MMSE estimator. State evolution is governed by Boolean functions (i.e. logic gates) and noise. The current system state, in which transitions between Boolean states can be perturbed by some process noise, creating an ergodic state transition matrix, is observed through an arbitrary noisy measurement. The optimal recursive MMSE estimator for this model is called the Boolean Kalman Filter (BKF), and an efficient algorithm has been proposed for its exact computation. This algorithm is presented here.

The Boolean Kalman Filtering algorithm can handle various observation models, including Bernoulli, Gaussian, Poisson, and Negative-Binomial, based on the input to the `obsModel` parameters.

- Bernoulli observation model requires only one parameter, aside from declaring the type, e.g.

  ```
  obsModel = list(type = 'Bernoulli', q = 0.05)
  ```

- Gaussian observation model requires a vector of the observation parameters, which include the mean and standard deviation of Boolean variables in inactivated and activated states. This will be defined as a vector, e.g.

  ```
  mu0 = 1
  sigma0 = 2
  mu1 = 5
  sigma1 = 2
  obsModel = list(type = 'Gaussian', model = c(mu0, sigma0, mu1, sigma1))
  ```

- The Poisson observation model requires a list of parameters. This list will have 3 entries in addition to the type definition, for a total of 4 entries:
  - Sequencing depth `s`
  - Baseline espression in inactivated state, referred to as `mu`
  - The differential expression, referred to as `delta`, which must be input as a vector of the same length as the number of Boolean variables in the network.

  In this way, the user can define the exact observation parameter for each individual gene. For a 4-gene network, a potential `obsModel` parameter for a Poisson distribution could be defined as:

  ```
  obsModel = list(type = 'Poisson', s = 10.875, mu = 0.01, delta = c(2, 2, 2, 2))
  ```

- Negative-Binomial observation model also requires a list of parameters. This list will have 4 entries in addition to the type definition, for a total of 5 entries:
  - Sequencing depth `s`
  - Baseline espression in inactivated state, referred to as `mu`
  - Differential expression, referred to as `delta`, which must be input as a vector of the same length as the number of Boolean variables in the network.
  - Inverse Dispersion, referred to as `phi`, which must also be input as a vector of the same length as the number of Boolean variables in the network.

  For a 4-gene network, a potential `obsModel` parameter for a Negative-Binomial observation model could be defined as:

  ```
  delta = c(2, 2, 2, 2)
  ```

```
phi = c(3, 3, 3, 3)
obsModel = list(type = 'NB', s = 10.875, mu = 0.01, delta, phi)
```

Value

Xhat	Estimate the state of of the Partially-Observed Boolean Dynamical System based on the recursive BKF algorithm
MSE	Mean Squared Error of the estimate returned by the BKF algorithm for each time instance
Beta	Normalized PDV for each time instance

Source

Braga-Neto, U. (2011, November). Optimal state estimation for Boolean dynamical systems. In Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on (pp. 1050-1054). IEEE. Imani, M., & Braga-Neto, U. Maximum-likelihood adaptive filter for partially-observed boolean dynamical systems. IEEE transaction on Signal Processing, 65:359-371, 2017.

Examples

```
data(p53net_DNAdsb0)

obsModel = list(type = 'NB',
                  s = 10.875,
                 mu = 0.01,
              delta = c(2, 2, 2, 2),
                phi = c(3, 3, 3, 3))

#Simulate a network with Negative-Binomial observation model
data <- simulateNetwork(p53net_DNAdsb0, n.data = 100, p = 0.02, obsModel)

#Derive the optimal estimate of the network using a BKF approach
Results <- BKF(data$Y, p53net_DNAdsb0, p = 0.02, obsModel)
```

BKS *Boolean Kalman Smoother*

Description

Genereates the optimal MMSE estimate of the state of a Partially-Observed Boolean Dynamical System by implementing a Boolean Kalman Smoother to batch data

Usage

```
BKS(Y, net, p, obsModel)
```

Arguments

Y	Time series of noisy observations of the Boolean regulatory network. Each row and column correspond to a specific Boolean variable and time point respectively.
net	A Boolean Network object (specified in BoolNet vernacular) that the time series of observations presented in Y is based on
p	Intensity of Bernoulli process noise
obsModel	Parameters for the chosen observation model.

Details

In the event that a sequence of measurements is available offline, the BKS can be used for computation of the optimal MMSE of smoothed trajectory.

The Boolean Kalman Smoother algorithm can handle various observation models, including Bernoulli, Gaussian, Poisson, and Negative-Binomial, based on the input to the obsModel parameter.

The obsModel parameter is defined the same as the Boolean Kalman Filter and simulateNetwork functions, reference the documentation for *BKF* or *simulateNetwork* for details.

Value

Xhat	Estimate of the sate of the Partially-Observed Boolean Dynamical System based on the BKS algorithm
MSE	Mean Squared Error of the estimate returned by the BKS algorithm for each time instance

Source

Imani, M., & Braga-Neto, U. (2015, December). Optimal state estimation for boolean dynamical systems using a boolean Kalman smoother. In 2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP) (pp. 972-976). IEEE.

Examples

```
data(p53net_DNAdsb0)

obsModel = list(type = 'Bernoulli', q = 0.02)

#Simulate a network with Bernoulli observation noise
data <- simulateNetwork(p53net_DNAdsb0, n.data = 100, p = 0.02, obsModel)

#Derive the optimal estimate of state of the network using the BKS algorithm
Results <- BKS(data$Y, p53net_DNAdsb0, p = 0.02, obsModel)
```

melanoma *Melanoma Regulatory Network*

Description

Melanoma Boolean regulatory network as described by E.R. Dougherty et al.

Usage

```
data("melanoma")
```

Details

The data consists of activities of 7 genes in melanoma regulatory network. The 7 genes are as follows: WNT5A, pirin, S100P, RET1, MART1, HADHB and STC2.

Source

E. R. Dougherty, R. Pal, X. Qian, M. L. Bittner, and A. Datta, "Stationary and structural control in gene regulatory networks: basic concepts," International Journal of Systems Science, vol. 41, no. 1, pp. 5-16, 2010.

Examples

```
data(melanoma)

data <- simulateNetwork(melanoma, n.data = 100, p = .02,
                        obsModel = list(type = 'Bernoulli',
                                        q = 0.05))
```

MMAE *Multiple Model Adaptive Estimation*

Description

This function implements the Multiple Model Adaptive Estimation (MMAE) algorithm, which implements a bank of Boolean Kalman Filters running in parallel on a dataset in order to estimate model parameters.

Usage

```
MMAE(data, net, p, threshold, Prior = NA, obsModel = NA)
```

Arguments

`data`	Data from which the MMAE algorithm should attempt to estimate parameters
`net`	A vector of potential networks, defined in the BoolNet `loadNetwork` vernacular, from which the MMAE algorithm should choose the most probable network for the given data in `Y`
`p`	A vector containing different possible intensities of Bernoulli process noise, from which the MMAE algorithm should choose the most likely
`threshold`	A posterior probability threshold value (between 0 and 1) for which the MMAE should stop running
`Prior`	A vector of the prior probabilities of the possible combinations of the various networks and process noise parameters input to the algorithm. The vector must be of size `length(net)*length(p)`. The entries into the vector will follow the pattern: net1, p1 net1, p2 ... netm, pn where m is the total number of potential networks and n is the total number of the possible `p` values. The default prior ovse models is uniform.
`obsModel`	Parameters for the chosen observation model.

Value

The MMAE algorithm will return which network and process noise value is most likely to have generated the given dataset supplied in `Y`.

If the posterior probability for any of the combinations of networks and process noise parameters fails to surpass the value defined in the `threshold` variable, the algorithm will output that no decision could be made using the given data.

Source

Imani, M., & Braga-Neto, U. (2015, November). Optimal gene regulatory network inference using the boolean kalman filter and multiple model adaptive estimation. In 2015 49th Asilomar Conference on Signals, Systems and Computers (pp. 423-427). IEEE.

Examples

```
#load potential networks
data(p53net_DNAdsb0)
data(p53net_DNAdsb1)

net1 <- p53net_DNAdsb0
net2 <- p53net_DNAdsb1

#define observation model
observation = list(type = 'NB', s = 10.875, mu = 0.01, delta = c(2, 2, 2, 2), phi = c(3, 3, 3, 3))
```

```
#simulate data using one of the networks and a given 'p'
data <- simulateNetwork(net1, n.data = 100, p = 0.02, obsModel = observation)

#run MMAE to determine model selection and parameter estimation
MMAE(data, net=c("net1","net2"), p=c(0.02,0.1,0.15), threshold=0.8, obsModel = observation)
```

---

p53net_DNAdsb0 *p53 Negative-Feedback Gene Regulatory Boolean Network*

Description

The data consists of activities of 4 genes in the well-known p53-MDM2 negative-feedback gene regulatory network when the external input (called dna_dsb) to this system is ON or OFF. The p53 gene codes for the tumor suppressor protein p53 in humans, and its activation plays a critical role in cellular responses to various stress signals that might cause genome instability. The four genes are as follows: ATM, p53, Wip1, and MDM2. The input 'dna_dsb' indicates the presence of DNA double strand breaks. In this dataset, **dna_dsb is 0**, indicating no external input.

Usage

```
data("p53net_DNAdsb0")
```

Details

The p53-MDM2 negative-feedback gene regulatory network described by E. Batchelor et al.

Source

E. Batchelor, A. Loewer, and G. Lahav, "The ups and downs of p53: Understanding protein dynamics in single cells," Nature Rev. Cancer, vol. 9, no. 5, pp. 371-377, 2009.

Examples

```
data(p53net_DNAdsb0)

data <- simulateNetwork(p53net_DNAdsb0, n.data = 100, p = .02,
                        obsModel = list(type = 'Bernoulli',
                                        q = 0.05))
```

---

p53net_DNAdsb1 *p53 Negative-Feedback Gene Regulatory Boolean Network*

Description

The data consists of activities of 4 genes in the well-known p53-MDM2 negative-feedback gene regulatory network when the external input (called dna_dsb) to this system is ON or OFF. The p53 gene codes for the tumor suppressor protein p53 in humans, and its activation plays a critical role in cellular responses to various stress signals that might cause genome instability. The four genes are as follows: ATM, p53, Wip1, and MDM2. The input 'dna_dsb' indicates the presence of DNA double strand breaks. In this dataset, **dna_dsb is 1**, indicating no external input.

Usage

```
data("p53net_DNAdsb1")
```

Details

The p53-MDM2 negative-feedback gene regulatory network described by E. Batchelor et al.

Source

E. Batchelor, A. Loewer, and G. Lahav, "The ups and downs of p53: Understanding protein dynamics in single cells," Nature Rev. Cancer, vol. 9, no. 5, pp. 371-377, 2009.

Examples

```
data(p53net_DNAdsb1)

data <- simulateNetwork(p53net_DNAdsb1, n.data = 100, p = .02,
                        obsModel = list(type = 'Bernoulli',
                                        q = 0.05))
```

---

plotTrajectory *Plot state variables of Boolean Regulatory Systems*

Description

Allows for visualization of Boolean state variables. If `compare = TRUE`, the trajectory in **BLACK** is dataset1 and the trajectory in **RED** is dataset2.

Usage

```
plotTrajectory(dataset1,
               labels = NA,
               dataset2 = NA,
               compare = FALSE,
               byrow = TRUE)
```

Arguments

`dataset1`	Trajectory to be viewed. Input is limited to 4 variables to allow for concise viewing. Shown in **BLACK**.
`labels`	Vector of labels to apply to plots
`dataset2`	If `compare = TRUE`, another dataset to be overlayed on dataset1. This is, again, limited to 4 variables. Shown in **DASHED RED** if `compare = TRUE`.
`compare`	Set `compare = TRUE` if an overlay is desired
`byrow`	Set `byrow = FALSE` if the input data is not in the format of a single state variable corresponding to a row of the dataset.

Examples

```
data(p53net_DNAdsb1)

data <- simulateNetwork(p53net_DNAdsb1, n.data = 100, p = 0.02,
                        obsModel = list(type = 'Bernoulli',
                                        q = 0.05))


plotTrajectory(data$X,
               labels = p53net_DNAdsb1$genes)


#View both (original state trajectory and observation) datasets overlayed
plotTrajectory(data$X,
               labels = p53net_DNAdsb1$genes,
               dataset2 = data$Y,
               compare = TRUE)
```

`simulateNetwork` *Simulate Boolean Network*

Description

Simulates a Boolean network and generates noise of user-defined model preference and paramters.

Usage

```
simulateNetwork(net, n.data, p, obsModel)
```

Arguments

`net`	A boolean Network object (specified in BoolNet vernacular) to be simulated
`n.data`	Length of time-series to simulate
`p`	Process noise to build into the simulation. Should be 0<p<0.5.
`obsModel`	Parameters for the chosen observation model.

Details

The `simulateNetwork` function can simulate many observation models to create observational data. An overvoew of how to use the various types of observation models present in `BoolFilter` is given below:

- Bernoulli observation model requires only one parameter, aside from declaring the type, e.g.

  ```
  obsModel = list(type = 'Bernoulli', q = 0.05)
  ```

- Gaussian observation model requires a vector of the observation parameters, which include the mean and standard deviation of Boolean variables in inactivated and activated states. This will be defined as a vector, e.g.

```
mu0 = 1
sigma0 = 2
mu1 = 5
sigma1 = 2
obsModel = list(type = 'Gaussian', model = c(mu0, sigma0, mu1, sigma1))
```

- Poisson observation model requires a list of parameters. This list will have 3 entries in addition to the type definition, for a total of 4 entries:
  - Sequencing depth `s`
  - Baseline espression in inactivated state, referred to as `mu`
  - The differential expression, referred to as `delta`, which must be input as a vector of the same length as the number of genes in the network.

  In this way, the user can define the exact observation parameter for each individual gene. For a 4-gene network, a potential `obsModel` parameter for a Poisson distribution could be defined as:

  ```
  obsModel = list(type = 'Poisson', s = 10.875, mu = 0.01, delta = c(2, 2, 2, 2))
  ```

- Negative-Binomial observation models also require a list of parameters. This list will have 4 entries in addition to the type definition, for a total of 5 entries:
  - Sequencing depth `s`
  - Baseline espression in inactivated state, referred to as `mu`
  - Differential expression, referred to as `delta`, which must be input as a vector of the same length as the number of genes in the network.
  - Inverse Dispersion, referred to as `phi`, which must also be input as a vector of the same length as the number of genes in the network.

  For a 4-gene network, a potential `obsModel` parameter for a Negative-Binomial observation model could be defined as:

  ```
  delta = c(2, 2, 2, 2)
  phi = c(3, 3, 3, 3)
  obsModel = list(type = 'NB', s = 10.875, mu = 0.01, delta, phi)
  ```

Value

`X`	Original Boolean state trajectory, **without** observation noise
`Y`	Observation trajectory

Examples

```
data(p53net_DNAdsb1)

#generate data from poisson observation model
dataPoisson <- simulateNetwork(p53net_DNAdsb1, n.data = 100, p = 0.02,
                          obsModel = list(type = 'Poisson',
                                          s = 10.875,
                                          mu = 0.01,
                                          delta = c(2,2,2,2)))

#generate data from Bernoulli observation model
dataBernoulli <- simulateNetwork(p53net_DNAdsb1, n.data = 100, p = 0.02,
```

```
obsModel = list(type = 'Bernoulli',
                q = 0.05))
```

SIR_BKF *Particle Filter*

Description

Performs the SIR-BKF algorithm approach in order to approximate the optimal MMSE estimate of state variables of a Partially-Observed Boolean Dynamical System.

Usage

```
SIR_BKF(Y, N, alpha, net, p, obsModel)
```

Arguments

Y	Time series of noisy observations of the boolean regulatory network. Each row and column correspond to a specific Boolean variable and time point respectively.
N	Number of particles used in approximation
alpha	Parameter denoting the cutoff threshold for resampling
net	A Boolean Network object (specified in BoolNet vernacular) that the time series of observations presented in Y is based on
p	Intensity of Bernoulli process noise
obsModel	Parameters for the chosen observation model.

Details

Exact optimal state estimation for partially-observed Boolean dynamical systems may become impractical computationally if system dimensionality is large. This opens the floor for an approximation algorithm to handle cases of high-dimensional systems. The SIR-BKF Particle Filtering algorithm contained in BoolFilter handles situations in which the Boolean Kalman Filter proves too computationally inefficient.

The Particle Filtering approximation, like the Boolean Kalman Filter, can handle various observation models, including Bernoulli, Gaussian, Poisson, and Negative-Binomial, based on the input to the obsModel paramter.

The obsModel parameter is defined the same as the Boolean Kalman Filter and simulateNetwork functions, reference the documentation for *BKF* or *simulateNetwork* for details.

Source

Braga-Neto U. Particle filtering approach to state estimation in Boolean dynamical systems. In Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE 2013 Dec 3 (pp. 81-84). IEEE.

Examples

```
data(cellcycle)

obsModel = list(type = 'Gaussian',
                model = c(mu0 = 1, sigma0 = 2, mu1 = 5, sigma1 = 2))

#generate data from Negative Binomial observation model for the
#10-gene Mammalian Cell Cycle Network
data <- simulateNetwork(cellcycle, n.data = 100, p = 0.02, obsModel)

#perform SIR-BKF algorithm
approx <- SIR_BKF(data$Y, N = 1000, alpha = 0.95, cellcycle, p = 0.02, obsModel)
```

Index