Package ‘ProbReco’

September 24, 2020

Type Package

Title Score Optimal Probabilistic Forecast Reconciliation

Version 0.1.0.1

Description Training of reconciliation weights for probabilistic forecasts to optimise total energy (or variogram) score using Stochastic Gradient Descent with automatically differentiated gradients. See Panagiotelis, Gamakumara, Athanasopoulos and Hyndman, (2020) <https://www.monash.edu/business/ebs/research/publications/ebs/wp26-2020.pdf> for a description of the methods.

License GPL-3

URL https://github.com/anastasiospanagiotelis/ProbReco

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.2), purrr(>= 0.3.2), mvtnorm, Rdpack

Suggests knitr, rmarkdown, fable, dplyr,tidyr, magrittr, stringi

LinkingTo Rcpp, RcppEigen, StanHeaders (>= 2.19.1), BH

RdMacros Rdpack

RoxygenNote 7.1.1

LazyData true

VignetteBuilder knitr

Encoding UTF-8

BugReports https://github.com/anastasiospanagiotelis/ProbReco/issues

NeedsCompilation yes

Author Anastasios Panagiotelis [aut, cre] (<https://orcid.org/0000-0001-8678-7294>)

Maintainer Anastasios Panagiotelis <anastasios.panagiotelis@sydney.edu.au>

Repository CRAN

Date/Publication 2020-09-24 08:10:06 UTC

R topics documented:

`checkinputs` *Check inputs to function.*

Description

This function checks that the inputs for `scoreopt` and `total_score` are correctly setup. It is called at the start of `scoreopt`.

Usage

```
checkinputs(data, prob, S, G, score = list(score = "energy", alpha = 1))
```

Arguments

`data`	Past data realisations as vectors in a list. Each list element corresponds to a period of training data.
`prob`	List of functions to simulate from probabilistic forecasts. Each list element corresponds to a period of training data. The output of each function should be a matrix.
`S`	Matrix encoding linear constraints.
`G`	Values of reconciliation parameters d and G where $\tilde{y} = S(d + G\hat{y})$. The first m elements correspond to translation vector d, while the remaining elements correspond to the matrix G where the elements are filled in column-major order.
`score`	Score to be used. This must be a list with two elements: score for the scoring rule (currently only energy supported) and alpha, an additional parameter used in the score (e.g. power in energy score, default is 1).

`inscoreopt` *In-Sample Score Optimisation by Stochastic Gradient Descent*

Description

Function to find a reconciliation matrix that optimises total score using training data. Stochastic gradient descent is used for optimisation with gradients found using automatic differentiation. This function differs from `scoreopt` in two main ways. First, formulation of base probabilistic forecasts is carried out from one of four options depending on whether dependence and/or Gaussianity is assumed. Second, the optimistation is based on in-sample predictions rather than a rolling window of out-of sample forecasts. For more flexibility use `scoreopt`.

Usage

```
inscoreopt(
  y,
  yhat,
  S,
  Ginit = c(rep(0, ncol(S)), as.vector(solve(t(S) %*% S, t(S)))),
  control = list(),
  basedep = "joint",
  basedist = "gaussian",
  Q = 500,
  score = list(score = "energy", alpha = 1),
  trace = FALSE
)
```

Arguments

`y`	Matrix of data, each column responds to an observation, each row corresponds to a variable.
`yhat`	Matrix of predicted values, each column responds to an observation, each row corresponds to a variable.
`S`	Matrix encoding linear constraints.
`Ginit`	Initial values of reconciliation parameters d and G where $\tilde{y} = S(d + G\hat{y})$. The first m elements correspond to translation vector d, while the remaining elements correspond to the matrix G where the elements are filled in column-major order. Default is least squares.
`control`	Tuning parameters for SGD. See `scoreopt.control` for more details
`basedep`	Should base distributions be assumed to be dependent (joint) or independent? Default is "joint", set to "independent" for independence.
`basedist`	Should base distributions be assumed to be Gaussian or bootstrapped? Default is "gaussian" set to "bootstrap" for bootstrapping.
`Q`	Number of Monte Carlo iterations used to estimate score

`score`	Score to be used. This must be a list with two elements: score for the scoring rule (currently only energy supported) and alpha, an additional parameter used in the score (e.g. power in energy score, default is 1).
`trace`	Flag to keep details of SGD. Default is FALSE

Value

Optimised reconciliation parameters.

`d`	Translation vector for reconciliation.
`G`	Reconciliation matrix (G).
`val`	The estimated optimal total score.
`Gvec_store`	A matrix of Gvec (d and G vectorised) where each column corresponds to an iterate of SGD (only produced when trace=TRUE).
`val_store`	A vector where each element gives the value of the objective function for each iterate of SGD (only produced when trace=TRUE).

See Also

Other ProbReco functions: `scoreopt.control()`, `scoreopt()`, `total_score()`

Examples

```
#Define S matrix
S<-matrix(c(1,1,1,0,0,1),3,2, byrow = TRUE)
#Set data (only 10 training observations used for speed)
y<-S%*%(matrix(rnorm(20),2,10)+1)
#Set point forecasts (chosen randomly from (0,1))
yhat<-matrix(runif(nrow(y)*ncol(y)),nrow(y),ncol(y))
#Find weights by SGD (Q set to 20 so that example runs quickly)
out<-inscoreopt(y,yhat,S,Q=20)
```

`scoreopt` *Score optimisation by Stochastic Gradient Descent*

Description

Function to find a reconciliation matrix that optimises total score using training data. Stochastic gradient descent is used for optimisation with gradients found using automatic differentiation.

Usage

```
scoreopt(
  data,
  prob,
  S,
  Ginit = c(rep(0, ncol(S)), as.vector(solve(t(S) %*% S, t(S)))),
  control = list(),
  score = list(score = "energy", alpha = 1),
  trace = FALSE,
  matches = FALSE
)
```

Arguments

`data`	Past data realisations as vectors in a list. Each list element corresponds to a period of training data.
`prob`	List of functions to simulate from probabilistic forecasts. Each list element corresponds to a period of training data. The output of each function should be a matrix.
`S`	Matrix encoding linear constraints.
`Ginit`	Initial values of reconciliation parameters d and G where $\tilde{y} = S(d + G\hat{y})$. The first m elements correspond to a translation vector d, while the remaining elements correspond to the matrix G where the elements are filled in column-major order. Default is least squares.
`control`	Tuning parameters for SGD. See `scoreopt.control` for more details
`score`	Score to be used. This must be a list with two elements: score for the scoring rule (currently only energy score and variogram score supported) and alpha, an additional parameter used in the score (e.g. power in energy score, default is 1).
`trace`	Flag to keep details of SGD. Default is FALSE
`matches`	A flag that checks for exact matches between samples from reconciled distribution. This causes NaNs in the automatic differentiation. For approaches that rely on bootstrapping set to TRUE. Otherwise set to FALSE (the default) to speed up code.

Value

Optimised reconciliation parameters.

`d`	Translation vector for reconciliation.
`G`	Reconciliation matrix.
`val`	The estimated optimal total score.
`Gvec_store`	A matrix of Gvec (d and G vectorised) where each column corresponds to an iterate of SGD (only produced when trace=TRUE).
`val_store`	A vector where each element gives the value of the objective function for each iterate of SGD (only produced when trace=TRUE).

See Also

Other ProbReco functions: `inscoreopt()`, `scoreopt.control()`, `total_score()`

Examples

```
#Use purr library to setup
library(purrr)
#Define S matrix
S<-matrix(c(1,1,1,0,0,1),3,2, byrow = TRUE)
#Set data (only 10 training observations used for speed)
data<-map(1:10,function(i){S%*%(c(1,1)+rnorm(2))})
#Set list of functions to generate 50 iterates from probabilistic forecast
prob<-map(1:10,function(i){f<-function(){matrix(rnorm(3*50),3,50)}})
#Find weights by SGD (will take a few seconds)
out<-scoreopt(data,prob,S)
```

`scoreopt.control` *Tuning parameters for score optimisation by Stochastic Gradient Descent*

Description

Function to set tuning parameters for stochastic gradient descent used to find a reconciliation matrix that optimises total score. The defaults are those of Kingma and Ba (2014) and more details on the tuning parameters can be found therein.

Usage

```
scoreopt.control(
  eta = 0.001,
  beta1 = 0.9,
  beta2 = 0.999,
  maxIter = 500,
  tol = 1e-04,
  epsilon = 1e-08
)
```

Arguments

`eta`	Learning rate. Deafult is 0.001
`beta1`	Forgetting rate for mean. Default is 0.9.
`beta2`	Forgetting rate for variance. Default is 0.999.
`maxIter`	Maximum number of iterations. Default is 500
`tol`	Tolerance for stopping criterion. Algorithm stops when the change in all parameter values is less than this amount. Default is 0.0001.
`epsilon`	Small constant added to denominator of step size. Default is 1e-8

References

Kingma DP, Ba J (2014). “Adam: A method for stochastic optimization.” *arXiv preprint*. https://arxiv.org/abs/1412.6980.

See Also

Other ProbReco functions: `inscoreopt()`, `scoreopt()`, `total_score()`

Examples

```
#Change Maximum Iterations to 1000
scoreopt.control(maxIter=1000)
```

`sim_hierarchy` *Synthetic hierarchical data from stationary Gaussian ARMA models.*

Description

A synthetic 7-variable hierachy. The series AA and AB aggregate to A, the series BA and BB aggregate to B, the series A and B aggregate to Tot. All bottom level series are simulated from ARMA models. There are 1506 observations generated.

Usage

```
sim_hierarchy
```

Format

A tibble with a time index Time and one column for each of the seven variables in the hierarchy

---

`total_score` *Total score (and gradient) for reconciled forecast*

Description

Function to find an estimate of the total energy score for a linearly reconciled probabilistic forecast. Also finds the gradient by automatic differentiation.

Usage

```
total_score(data, prob, S, Gvec, scorecode = 1, alpha = 1, matches = FALSE)
```

Arguments

`data`	Past data realisations as vectors in a list. Each list element corresponds to a period of training data.
`prob`	List of functions to simulate from probabilistic forecasts. Each list element corresponds to a period of training data. The output of each function should be a matrix.
`S`	Matrix encoding linear constraints.
`Gvec`	Reconciliation parameters d and G where $\tilde{y} = S(d + G\hat{y})$. The first m elements correspond to translation vector d, while the remaining elements correspond to the matrix G where the elements are filled in column-major order.
`scorecode`	Code that indicates score to be used. This is set to 1 for the energy score and 2 for the variogram score. Default is 1 (energy score)
`alpha`	An additional parameter used for scoring rule. Default is 1 (power used in energy score).
`matches`	A flag that indicates whether to check for exact matches between samples from reconciled distribution. This causes NaNs in the automatic differentiation. For approaches that rely on bootstrapping set to T. Otherwise set to F (the default) to speed up code.

Value

Total score and gradient w.r.t (d,G).

`grad`	The estimate of the gradient.
`value`	The estimated total score.

See Also

Other ProbReco functions: `inscoreopt()`, `scoreopt.control()`, `scoreopt()`

Examples

```
#Use purr library to setup
library(purrr)
#Define S matrix
S<-matrix(c(1,1,1,0,0,1),3,2, byrow = TRUE)
#Randomly set a value of reconciliation parameters
Gvec<-as.matrix(runif(8))
#Set data (only 10 training observations used for speed)
data<-map(1:10,function(i){S%*%(c(1,1)+rnorm(2))})
#Set list of functions generating from probabilistic forecast
prob<-map(1:10,function(i){f<-function(){matrix(rnorm(3*50),3,50)}})
#Compute total score
out<-total_score(data,prob,S,Gvec)
```

Index