

Package: TensorMCMC (via r-universe)

May 12, 2026

Type Package

Title Tensor Regression with Stochastic Low-Rank Updates

Version 0.1.0

Date 2026-01-08

Maintainer Ritwick Mondal <ritwick12@tamu.edu>

Description Provides methods for low-rank tensor regression with tensor-valued predictors and scalar covariates. Model estimation is performed using stochastic optimization with random-walk updates for low-rank factor matrices. Computationally intensive components for coefficient estimation and prediction are implemented in C++ via 'Rcpp'. The package also includes tools for cross-validation and prediction error assessment.

Imports Rcpp (>= 1.0.10), glmnet, stats

LinkingTo Rcpp

Encoding UTF-8

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.3.3

License MIT + file LICENSE

VignetteBuilder knitr

NeedsCompilation yes

Author Ritwick Mondal [aut, cre]

Repository <https://cran.r-universe.dev>

Date/Publication 2026-01-12 19:30:06 UTC

RemoteUrl <https://github.com/cran/TensorMCMC>

RemoteRef HEAD

RemoteSha 03dae74c2a43f05a503817baa30108b387e2ff34

Contents

cv.tensor.reg	2
getmean	3
getmean_cpp	3
predict.tensor.reg	4
predict_tensor_cpp	4
predict_tensor_reg	5
rigamma	5
rmse	6
tensor.reg	6
update_beta_cpp	7
Index	8

cv.tensor.reg	<i>Simple rank comparison via in-sample RMSE</i>
---------------	--

Description

Simple rank comparison via in-sample RMSE

Usage

```
cv.tensor.reg(x.train, z.train, y.train, ranks = 1:3, nsweep = 50)
```

Arguments

x.train	Training tensor X data.
z.train	Training scalar covariates.
y.train	Training response vector.
ranks	Vector of rank values to test.
nsweep	Number of stochastic update iterations.

Value

A data.frame with ranks and corresponding in-sample RMSE.

getmean	<i>posterior mean for tensor regression</i>
---------	---

Description

posterior mean for tensor regression

Usage

```
getmean(X, beta, rank, rank.exclude = NULL)
```

Arguments

X	Input tensor.
beta	Estimated coefficient tensor.
rank	Rank used in tensor decomposition.
rank.exclude	Optional rank values to exclude.

Value

Mean tensor.

getmean_cpp	<i>Posterior Mean Using C++</i>
-------------	---------------------------------

Description

This function calls ‘getmean_cpp’ to compute the posterior mean from tensor X and beta list.

Usage

```
getmean_cpp(X_vec, beta, n, p, d, rank)
```

Arguments

X_vec	Flattened tensor (numeric vector of length $n \cdot p \cdot d$)
beta	List of $p \times d$ matrices (length = rank)
n	Number of observations
p	Number of rows in each beta matrix
d	Number of columns in each beta matrix
rank	Rank of tensor decomposition

Value

Numeric vector of length n

predict.tensor.reg *Prediction from tensor regression (S3 method)*

Description

Prediction from tensor regression (S3 method)

Usage

```
## S3 method for class 'tensor.reg'
predict(object, x.new, z.new, scale = TRUE, ...)
```

Arguments

object	tensor.reg object
x.new	new tensor predictors (n x p x d)
z.new	new scalar covariates (n x pgamma)
scale	Logical; whether to scale predictors
...	additional arguments

Value

Predicted response vector

predict_tensor_cpp *Predict Response Using Tensor Regression C++*

Description

This function calls the underlying C++ function ‘predict_tensor_cpp’ to compute predicted responses given a flattened tensor, low-rank coefficient matrices, and scalar covariate coefficients.

Usage

```
predict_tensor_cpp(X_vec, beta, gam, n, p, d, rank)
```

Arguments

X_vec	Flattened tensor (numeric vector of length n*p*d)
beta	List of pxd matrices representing tensor coefficients
gam	Numeric vector of scalar coefficients
n	Number of observations
p	Number of rows in each beta matrix
d	Number of columns in each beta matrix
rank	Rank of tensor decomposition

Value

Numeric vector of length n

predict_tensor_reg *Predict tensor regression (wrapper)*

Description

Predict tensor regression (wrapper)

Usage

```
predict_tensor_reg(fit, x.new, z.new, scale = TRUE)
```

Arguments

fit	tensor.reg object
x.new	new tensor predictors
z.new	new scalar covariates
scale	Logical; whether to scale predictors

Value

Predicted response vector

rigamma *Inverse-gamma random number generator*

Description

Inverse-gamma random number generator

Usage

```
rigamma(n, shape, rate)
```

Arguments

n	Number of samples.
shape	Shape parameter of the gamma distribution.
rate	Rate parameter of the gamma distribution.

Value

A numeric vector of inverse-gamma samples.

rmse *root-mean-square error (RMSE)*

Description

root-mean-square error (RMSE)

Usage

```
rmse(a, b)
```

Arguments

a Predicted values.
b True observed values.

Value

RMSE value.

tensor.reg *Tensor Regression using Rcpp*

Description

Low-rank tensor regression with stochastic updates

Usage

```
tensor.reg(  
  z.train,  
  x.train,  
  y.train,  
  nsweep = 50,  
  rank = 2,  
  scale = TRUE,  
  alpha.lasso = 1  
)
```

Arguments

z.train	Matrix of scalar covariates (n x pgamma)
x.train	3D array of tensor predictors (n x p x d)
y.train	Response vector (length n)
nsweep	Number of stochastic update iterations (default 50)
rank	Rank of tensor decomposition (default 2)
scale	whether to scale predictors and response (default TRUE)
alpha.lasso	LASSO tuning parameter for initial estimate (default 1)

Value

A list with beta.store, gam.store, rank, p, d, and scaling info

update_beta_cpp	<i>Update Beta Matrices Using C++ Random Walk</i>
-----------------	---

Description

This function calls the C++ function ‘update_beta_cpp’ to perturb beta matrices.

Usage

```
update_beta_cpp(beta, p, d, rank, sigma)
```

Arguments

beta	List of pxd matrices (length = rank)
p	Number of rows in each beta matrix
d	Number of columns in each beta matrix
rank	Rank of tensor decomposition
sigma	Standard deviation of Gaussian noise

Value

Updated list of beta matrices

Index

`cv.tensor.reg`, 2

`getmean`, 3

`getmean_cpp`, 3

`predict.tensor.reg`, 4

`predict_tensor_cpp`, 4

`predict_tensor_reg`, 5

`rigamma`, 5

`rmse`, 6

`tensor.reg`, 6

`update_beta_cpp`, 7