Package: TRMF (via r-universe)

September 12, 2024

Type Package

Title Temporally Regularized Matrix Factorization

Version 0.2.1

Author Chad Hammerquist [aut, cre], Scentsy Inc [cph]

Maintainer Chad Hammerquist <chammerquist@scentsy.com>

Description Functions to estimate temporally regularized matrix factorizations (TRMF) for forecasting and imputing values in short but high-dimensional time series. Uses regularized alternating least squares to compute the factorization, allows for several types of constraints on matrix factors and natively handles weighted and missing data.

License GPL-3

Encoding UTF-8

Imports Matrix(>= 1.3-3),limSolve,generics

Suggests magrittr, knitr,rmarkdown

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2024-06-13 20:30:01 UTC

Contents

`coef.TRMF` *Extract TRMF Coefficients (Fm)*

Description

Returns the Fm (transposed) matrix from the matrix factorization Xm*Z*Fm.

Usage

```
## S3 method for class 'TRMF'
coef(object, ...)
```

Arguments

`object`	a trained TRMF object.
`...`	other arguments.

Value

the coefficient matrix, t(Fm)

Author(s)

Chad Hammerquist

See Also

`create_TRMF`, `TRMF_columns`, `TRMF_trend`

Examples

```
xm = poly(x = (-10:10)/10,degree=4)
fm = matrix(runif(40),4,10)
Am = xm%*%fm+rnorm(210,0,.2)

# create model
obj = create_TRMF(Am)
out = train(obj)
coef(out)
```

`components.TRMF` *Access TRMF factors*

Description

This function returns the factors (Xm, Fm,Z) from a trained TRMF object

Usage

```
## S3 method for class 'TRMF'
components(object, XorF = c("Xm","Fm","Z","Fm_each"), ...)
```

Arguments

`object`	trained TRMF object
`XorF`	which factor to return
`...`	ignored

Details

Returns the matrix factors. If matrix normalization was used in `create_TRMF`, `Xm%*%diag(Z)%*%Fm` could look much different than the input data matrix. If external regressor were included in the model and `XorF = "F_each"` then a returns a list with Fm split up by parts.

Value

A matrix or vector, or a possibly a list if `XorF = "F_each"`.

Author(s)

Chad Hammerquist

See Also

`create_TRMF`, `TRMF_columns`, `TRMF_trend`

Examples

```
# create test data
xm = poly(x = (-10:10)/10,degree=4)
fm = matrix(rnorm(40),4,10)
Am = xm%*%fm+rnorm(210,0,.2)

# create model
obj = create_TRMF(Am)
out = train(obj)
plot(out)
components(out,"Xm")
```

`create_TRMF` *Create a TRMF object*

Description

Creates a TRMF object from a data matrix. This function is always needed to initialize a TRMF model.

Usage

```
create_TRMF(dataM, weight = 1,
      normalize = c("none", "standard", "robust", "range"),
      normalize.type = c("global", "columnwise", "rowwise"),
      na.action = c("impute", "fail"),
      scaleXm = c("no","project","track"))
```

Arguments

`dataM`	The data matrix, each column represents a time series.
`weight`	An optional matrix of weights to be used in the fitting process. If used, `sum(w^2*e^2)` is minimized.
`normalize`	Type of scaling/centering for the data. Recommended to reduce bias when using regularization. `none` does nothing, `standard` centers with mean, and scales by `sd()`, `robust` centers with the median and scales by `mad(,constant=1)`, `range` maps to `[0-1]` interval
`normalize.type`	how should normalization be applied. `global` scales and centers matrix by one value. `columnwise` and `rowwise` normalize each column or row separately.
`na.action`	what action to take when data contains NAs
`scaleXm`	Should the columns of Xm be rescaled at each iteration. See details

Details

This function doesn't do any computation, it is the entry point for creating a TRMF model. To train the model or add additional details, see examples. Normalization is recommended in general. Regularization biases the factorization toward zero a little bit, centering changes that to bias towards the mean. Scaling makes the choosing of regularization parameters easier. If the factorization is to be used for forward forecasting, rowwise normalization is not recommended as it could remove some temporal information. If `scaleXm = 'project'` then the columns of Xm will be rescaled to have sum squared value = 1. If set to 'track' then the scaling factors will be stored and updated and used scale Fm before fitting Xm at each iteration.These options were added to allow for more stable behavior of temporal regularization.

Value

`create_TRMF` returns an object of `class` "TRMF" to be passed to other TRMF functions.

Author(s)

Chad Hammerquist

References

Yu, Hsiang-Fu, Nikhil Rao, and Inderjit S. Dhillon. "High-dimensional time series prediction with missing values." arXiv preprint arXiv:1509.08333 (2015).

See Also

`train.TRMF`, `TRMF_columns`, `TRMF_trend`

Examples

```
# create test data
xm = poly(x = (-10:10)/10,degree=4)
fm = matrix(runif(40),4,10)
Am = xm%*%fm+rnorm(210,0,.2)

# create model
obj = create_TRMF(Am)
obj = TRMF_columns(obj,reg_type ="interval")
obj = TRMF_trend(obj,numTS=4,order=2)
out = train(obj)
plot(out)
```

`fitted.TRMF` *Extract TRMF fitted values.*

Description

A function to extract fitted values from a trained TRMF object.

Usage

```
## S3 method for class 'TRMF'
fitted(object,impute = FALSE,...)
```

Arguments

`object`	a trained TRMF object.
`impute`	logical, should imputed values be returned?
`...`	other arguments.

Value

Fitted values extracted from object. If `impute` is TRUE then entire fitted (unscaled and uncentered) matrix is returned, otherwise there are NAs in the same locations as the time series matrix.

Author(s)

Chad Hammerquist

See Also

`create_TRMF`, `TRMF_columns`, `TRMF_trend`

Examples

```
xm = poly(x = (-10:10)/10,degree=4)
fm = matrix(runif(40),4,10)
Am = xm%*%fm+rnorm(210,0,.2)

# create model
obj = create_TRMF(Am)
out = train(obj)
fitted(out)
```

impute_TRMF *Impute missing values in a matrix*

Description

Impute missing values in matrix from a pre-trained TRMF object.

Usage

```
impute_TRMF(obj)
```

Arguments

obj	a trained TRMF object

Details

Essentially an accessor function. Replaces the missing values in data matrix with values from the fitted TRMF object.

Value

data matrix with missing values imputed

Author(s)

Chad Hammerquist

References

Yu, Hsiang-Fu, Nikhil Rao, and Inderjit S. Dhillon. "High-dimensional time series prediction with missing values." arXiv preprint arXiv:1509.08333 (2015).

See Also

train.TRMF, create_TRMF, TRMF_trend

Examples

```
# create test data
xm = poly(x = (-10:10)/10,degree=4)
fm = matrix(rnorm(40),4,10)
Am = xm%*%fm+rnorm(210,0,.2)
Am[sample.int(210,20)] = NA

# create model
obj = create_TRMF(Am)
obj = TRMF_trend(obj,numTS=4,order=2)
```

```
out = train(obj)
impute_TRMF(out)
```

NormalizeMatrix *Matrix Scaling*

Description

A function for normalizing (scaling and centering) a matrix.

Usage

```
NormalizeMatrix(X, method = c("standard", "robust", "range", "none"),
            type = c("global", "rowwise", "columnwise"), na.rm = TRUE)
```

Arguments

X	a numeric matrix(like object)
method	type of scaling to perform, standard centers with mean, and scales by sd(), robust centers with the median and scales by mad(,constant=1), range maps to [0-1] interval
type	how should normalization be applied. global scales and centers matrix by one value. columnwise and rowwise normalize each column or row separately.
na.rm	logical value, ignore NA values or not.

Details

Scaling and centering quantities are stored as attributes.

Value

The possibly centered and scaled matrix. Scaling and centering quantities are stored as attributes.

Author(s)

Chad Hammerquist

Examples

```
x = matrix(1:10, ncol = 2)
NormalizeMatrix(x)
```

plot.TRMF *Plot Latent Time Series for a TRMF Object*

Description

Plots all the time series in Xm from a trained TRMF object.

Usage

```
## S3 method for class 'TRMF'
plot(x, ...)
```

Arguments

x	a trained TRMF object.
...	ignored.

Value

No return value, called for side effects

Author(s)

Chad Hammerquist

See Also

create_TRMF, TRMF_columns, TRMF_trend

Examples

```
xm = poly(x = (-10:10)/10,degree=4)
fm = matrix(runif(40),4,10)
Am = xm%*%fm+rnorm(210,0,.2)

# create model
obj = create_TRMF(Am)
out = train(obj)
plot(out)
```

predict.TRMF *Predict method for TRMF model fit*

Description

Predict values based on the TRMF fit

Usage

```
## S3 method for class 'TRMF'
predict(object,newdata=NULL, ...)
```

Arguments

object	A trained TRMF object
newdata	A list with slot Xm and possibly with slots cXreg and gXreg
...	other arguments, ignored.

Details

If newdata is NULL, returns fitted model. If newdata doesn't have the term Xm or if it has a different number of columns than the number of latent time series, it will throw an error. If the object also contains a global regression, gXreg must be present and appropriately sized. If the object also contains a column-wise regression, cXreg must be present and appropriately sized.

Value

Returns a matrix of predictions.

Author(s)

Chad Hammerquist

See Also

create_TRMF, TRMF_columns, TRMF_trend,train.TRMF

Examples

```
xm = poly(x = (-10:10)/10,degree=4)
fm = matrix(runif(40),4,10)
Am = xm%*%fm+rnorm(210,0,.2)

# create model
obj = create_TRMF(Am)
out = train(obj)
fitted(out)
newXm = 1:5
```

```
predict(out,newdata=list(Xm=newXm))
```

`residuals.TRMF` *Extract TRMF residuals*

Description

A function to extract residuals from a trained TRMF object.

Usage

```
## S3 method for class 'TRMF'
residuals(object, ...)
```

Arguments

`object`	a trained TRMF object.
`...`	ignored

Value

residuals extracted from TRMF object

Author(s)

Chad Hammerquist

See Also

`create_TRMF`, `TRMF_columns`, `TRMF_trend`

Examples

```
xm = poly(x = (-10:10)/10,degree=4)
fm = matrix(runif(40),4,10)
Am = xm%*%fm+rnorm(210,0,.2)

# create model
obj = create_TRMF(Am)
out = train(obj)
resid(out)
```

`retrain` *Retrain TRMF objects.*

Description

Continue training on a pre-trained TRMF object.

Usage

```
retrain(obj, numit, fit_xm_first = TRUE,Xm=NULL,Fm=NULL,Z=NULL)
```

Arguments

`obj`	Pre-trained TRMF object
`numit`	Number of training iterations
`fit_xm_first`	Fit the Xm factor first? This could be useful it modifications are made to one of the factors that we don't want to be overwritten.
`Xm`	Optional update to Xm matrix. See details
`Fm`	Optional update to Fm matrix. See details
`Z`	Optional update to Z vector. See details

Details

This is basically the same function as `train()` but it doesn't create any of the constraint matrices and doesn't do any initialization. train() must be called on obj before this function. If external regressors are in the model and Fm is provided, the number of rows of columns must equal number of regressors plus columns of Xm. The format of Fm in this case is: [column_xreg_parameters,global_xreg_parameters,Fm_para

Value

A trained TRMF object.

See Also

`train.TRMF`

Examples

```
# create test data
tm = 3*poly(x = (-20:20)/10,degree=3)
sm = diffinv(rnorm(29,0,.1),lag=12,xi=(-5:6)/6)
xm = cbind(sm,tm)
fm = matrix(runif(40),4,10)
Am = xm%*%fm+rnorm(410,0,.1)

# create model
obj = create_TRMF(Am)
```

```
obj = TRMF_columns(obj,reg_type ="interval")
obj = TRMF_trend(obj,numTS=3,order=2)
obj = TRMF_seasonal(obj,numTS=1,freq=12,lambdaD=5)

# train
out = train(obj,numit=0) # intialize
plot(out)
new_out = retrain(out,numit=10)
plot(new_out)
```

---

summary.TRMF *Summarize TRMF*

Description

summary method for class "TRMF"

Usage

```
## S3 method for class 'TRMF'
summary(object, ...)
```

Arguments

object	TRMF object.
...	other arguments.

Value

NULL

Author(s)

Chad Hammerquist

See Also

create_TRMF, TRMF_columns, TRMF_trend

Examples

```
xm = poly(x = (-10:10)/10,degree=4)
fm = matrix(runif(40),4,10)
Am = xm%*%fm+rnorm(210,0,.2)

# create model
obj = create_TRMF(Am)
out = train(obj)
summary(obj)
summary(out)
```

`train.TRMF` *Train a TRMF model*

Description

This function is the "engine" of the TRMF package. It takes a previously created TRMF object, sets up the internal objects and fits it to the data using an alternating least squares (ALS) algorithm.

Usage

```
## S3 method for class 'TRMF'
train(x,numit=10,Xm=NULL,Fm=NULL,Z=NULL,...)
```

Arguments

`x`	A TRMF object to be fit.
`numit`	Number of alternating least squares iterations
`Xm`	Optional initial matrix. See details
`Fm`	Optional initial matrix. See details
`Z`	Optional initial vector. See details
`...`	ignored

Details

If a coefficient model is not present in `object`, it adds a L2 regularization model. If no time series models have been added to `object`, it adds a simple model using `TRMF_simple`. If `Xm, Fm` are both NULL, then initial estimates are provided. If both are provided, the ALS algorithm is started with `Xm` (with a warning) and `Fm` is overwritten with the next ALS step. If external regressors are in the model and Fm is provided, the number of rows of columns must equal number of regressors plus columns of Xm. The format of Fm in this case is: [column_xreg_parameters,global_xreg_parameters,Fm_parameters]^T. If `Z` is NULL, then a vector of 1's is used as initial estimate.

Value

`train` returns a fitted object of `class` "`TRMF`" that contains the data, all added models, matrix factorization and fitted model. The matrix factors Xm, Fm are stored in `object$Factors$Xm` and `object$Factors$Fm`, `object$Factors$Z` respectively. Use `fitted` to get fitted model, use `resid` to get residuals, use `coef` to get coefficients (Fm matrix) and `components` to get `Xm` or `Fm`.

Author(s)

Chad Hammerquist

References

Yu, Hsiang-Fu, Nikhil Rao, and Inderjit S. Dhillon. "High-dimensional time series prediction with missing values." arXiv preprint arXiv:1509.08333 (2015).

See Also

create_TRMF, TRMF_columns, TRMF_trend

Examples

```
# create test data
xm = poly(x = (-10:10)/10,degree=4)
fm = matrix(rnorm(40),4,10)
Am = xm%*%fm+rnorm(210,0,.2)

# create model
obj = create_TRMF(Am)
out = train(obj)
plot(out)
```

TRMF_ar *Add an Auto-Regressive Regularization Model to a TRMF Object.*

Description

Creates a regularization scheme that constrains latent time-series based on auto-regressive parameters and adds it to a TRMF object. In matrix optimization form, it adds the following term to the TRMF cost function: $R(x) = lambdaD^2||w(DX_s)||^2 + lambdaA^2||X_s||^2$ where X_s is sub-set of the Xm matrix controlled by this model and D is a matrix that corresponds to an auto-regressive model.

Usage

```
TRMF_ar(obj,numTS = 1,AR,lambdaD=1,lambdaA=0.0001,weight=1)
```

Arguments

obj	A TRMF object
numTS	number of latent time series in this model
lambdaD	regularization parameter for temporal constraint matrix
lambdaA	regularization parameter to apply simple L2 regularization to this time series model
weight	optional vector of weights to weight constraints, i.e. R(x) = lambdaD^2*||w*(D%*%X)||^2
AR	vector of autoregressive parameters. No checks are performed

Details

Setting AR = c(1) gives a random walk model, same as TRMF_trend(..., order=1).

Value

Returns an updated object of class TRMF.

Note

Unlike `TRMF_columns()`, these lambdas are squared in the code.

Author(s)

Chad Hammerquist

References

Yu, Hsiang-Fu, Nikhil Rao, and Inderjit S. Dhillon. "High-dimensional time series prediction with missing values." arXiv preprint arXiv:1509.08333 (2015).

See Also

`create_TRMF`, `TRMF_columns`, `TRMF_trend`

Examples

```
# create test data
xm = matrix(rnorm(80),20,4)
fm = matrix(rnorm(40),4,10)+1
Am = xm%*%fm+rnorm(200,0,.1)

# create model
obj = create_TRMF(Am)
obj = TRMF_columns(obj,reg_type ="interval")
obj = TRMF_ar(obj,numTS=2,AR=c(0.5),lambdaD=4)
out = train(obj)
plot(out)
```

TRMF_columns *Add a column regularization model to TRMF object*

Description

Adds a regularization model to TRMF object created by `create_TRMF()` to constrain the fitting process of the coefficient matrix.

Usage

```
TRMF_columns(obj,
   reg_type = c("l2", "nnls", "constrain", "interval", "none"), lambda = 0.0001,mu0=NULL)
```

Arguments

obj	TRMF object created by `create_TRMF()`
reg_type	regularization type to apply when fitting TRMF model. `l2` regularizes using an L2 function (see details), `nnls` forces coefficients to be non-negative. `constrain` constrains coefficients to be non-negative and to sum to 1. `interval` constrains coefficients to the interval [0-1]
lambda	L2 regularization parameter used for all regularization types. Can be a single value, a vector, or a matrix. If not a scalar, the dimension must match the number of rows of `Fm`. In a Bayesian framework, this is the inverse prior covariance for matrix coefficients. Note this value is not squared in the code.
mu0	The prior value for matrix coefficient. Can be a single value or vector. If not a scalar, the dimension must match the number of rows of `Fm`. If NULL it is set to zero. If external regressors are used these are included in Fm, so if lambda and mu0 are not scalars, they need to be arranged as: [column_xreg_parameters,global_xreg_parameters,Fm_l

Details

This function doesn't do any computations, it just sets up regularization parameters for the coefficient matrix. This function should only be called once on a TRMF object. If called twice, it will overwrite previous model with a warning. In addition to the possible constraints, a possible L2 regularization term is added to the fit. The regularization term for each column `i` is $(Fm_i - \mu_0)^T P (Fm_i - \mu_0)$ where $P = \lambda I$ or $P = diag(\lambda)$ or $P = \lambda$ depending on the size of provided lambda. A nonzero value for `lambda` is recommended to ensure stability of fit.

Value

Returns an updated object of class TRMF.

Author(s)

Chad Hammerquist

References

Yu, Hsiang-Fu, Nikhil Rao, and Inderjit S. Dhillon. "High-dimensional time series prediction with missing values." arXiv preprint arXiv:1509.08333 (2015).

See Also

`train.TRMF`, `create_TRMF`, `TRMF_trend`

Examples

```
# create test data
xm = poly(x = (-10:10)/10,degree=4)
fm = matrix(abs(rnorm(40)),4,10)
Am = xm%*%fm+rnorm(210,0,.2)

# create model
```

```
obj = create_TRMF(Am)
obj = TRMF_columns(obj,reg_type ="nnls")
out = train(obj)
plot(out)
```

`TRMF_regression` *Add external regressors to TRMF object*

Description

A function to add external regressors to a TRMF object.

Usage

```
TRMF_regression(obj, Xreg, type = c("global", "columnwise"))
```

Arguments

`obj`	TRMF object created by `create_TRMF()`
`Xreg`	Vector or matrix of external regressors. If `type = "columnwise"`, `Xreg` can be a matrix or array, but the first two dimensions must match those of the data matrix.
`type`	how are the regressors added to the model. If `type = "global"` the matrix factorization includes all the regressors. If `type = "columnwise"` each column in the data matrix is regressed of the corresponding column of `Xreg`.

Details

The coefficients model for the regressors are subject to the same regularization as the rest of the matrix factorization. Only one columnwise and one global model should be used in the same model. Both types can be include in the same model though.

Value

Returns an updated object of class TRMF.

Author(s)

Chad Hammerquist

See Also

`create_TRMF`, `TRMF_columns`, `TRMF_trend`

Examples

```
# ~ Global regression example ~
# create test data
bb = (-10:10)/10
xReg = 10*cos(bb*10)
xm = poly(x = bb,degree=3)
fm = matrix(rnorm(40),4,10)
Am = cbind(xReg,xm)%*%fm+rnorm(210,0,.2)

# creat model and fit
obj = create_TRMF(Am)
obj = TRMF_trend(obj,numTS=3,order=2)
obj = TRMF_regression(obj,Xreg=xReg,type="global")
out = train(obj)
plot(out)

# ~ columnwise regression example ~
# create test data
bb = (-10:10)/10
xm = poly(x = bb,degree=4)
fm = matrix(rnorm(84),4,21)
Am = xm%*%fm+rnorm(441,0,.2)

layers = array(0,dim=c(21,21,2))
layers[,,1] = 2*cos(2*bb)%o%sin(4*bb)
layers[,,2] = 2*sqrt(abs(bb%o%bb))
nAm = Am+layers[,,1]+layers[,,2]

# creat model and fit
obj = create_TRMF(nAm)
obj = TRMF_trend(obj,numTS=4,order=2)
obj = TRMF_regression(obj,Xreg=layers,type="columnwise")
out = train(obj)
plot(out)
```

`TRMF_seasonal` *Add seasonal regularization model to a TRMF object*

Description

Creates a regularization scheme that favors seasonally varying solutions and adds it to a TRMF object. In matrix optimization form, it adds the following term to the TRMF cost function: $R(x) = lambdaD^2||w(DX_s)||^2 + lambdaA^2||X_s||^2$ where X_s is sub-set of the Xm matrix controlled by this model and D is a (with a lag of freq) finite difference matrix (or rolling sum matrix, see details).

Usage

```
TRMF_seasonal(obj,numTS = 1,freq = 12,sumFirst=FALSE,lambdaD=1,lambdaA=0.0001,weight=1)
```

Arguments

obj	A TRMF object
numTS	number of latent time series in this model
lambdaD	regularization parameter for temporal constraint matrix
lambdaA	regularization parameter to apply simple L2 regularization to this time series model
weight	optional vector of weights to weight constraints, i.e. R(x) = lambdaD^2*‖w*(D%*%X)‖^2
freq	The frequency of the seasonal time series model.
sumFirst	Minimize the rolling sum of length freq instead of a lagged finite difference.

Details

TRMF_seasonal(freq=N) fits a lag N random walk. For monthly data, use freq=12, for quarterly data, freq=4. If sumFirst = TRUE, (called this for legacy reasons), a different regularization matrix is used which minimizes a rolling sum of length freq in the latent time series. This can be useful to prevent drift and force the seasonal component to vary around a zero mean.

Value

Returns an updated object of class TRMF.

Note

Unlike TRMF_columns(), these lambdas are squared in the code.

Author(s)

Chad Hammerquist

References

Yu, Hsiang-Fu, Nikhil Rao, and Inderjit S. Dhillon. "High-dimensional time series prediction with missing values." arXiv preprint arXiv:1509.08333 (2015).

See Also

create_TRMF, TRMF_columns,TRMF_simple, TRMF_trend

Examples

```
# create test data
tm = 3*poly(x = (-20:20)/10,degree=3)
sm = diffinv(rnorm(29,0,.1),lag=12,xi=(-5:6)/6)
xm = cbind(sm,tm)
fm = matrix(runif(40),4,10)
Am = xm%*%fm+rnorm(410,0,.1)

# create model
```

```
obj = create_TRMF(Am)
obj = TRMF_columns(obj,reg_type ="interval")
obj = TRMF_trend(obj,numTS=3,order=2)
obj = TRMF_seasonal(obj,numTS=1,freq=12,lambdaD=5)
out = train(obj)
plot(out)
```

`TRMF_simple` *Add L2 regularization model to a TRMF object*

Description

Creates an L2 regularization and adds it to a TRMF object. In matrix optimization form, it adds the following term to the TRMF cost function: $R(x) = lambdaA^2||w(X_s)||^2$ where X_s is sub-set of the Xm matrix controlled by this model.

Usage

```
TRMF_simple(obj,numTS = 1,lambdaA=0.0001,weight=1)
```

Arguments

`obj`	A TRMF object
`numTS`	number of latent time series in this model
`lambdaA`	regularization parameter to apply simple L2 regularization to this time series model
`weight`	optional vector of weights to weight constraints, i.e. R(x) = lambdaA^2*‖w*X‖^2

Details

This is called by `train_TRMF` if the TRMF object doesn't have any time series models.

Value

Returns an updated object of class TRMF.

Author(s)

Chad Hammerquist

References

Yu, Hsiang-Fu, Nikhil Rao, and Inderjit S. Dhillon. "High-dimensional time series prediction with missing values." arXiv preprint arXiv:1509.08333 (2015).

See Also

create_TRMF, TRMF_columns,TRMF_seasonal, TRMF_trend

Examples

```
# create test data
xm = matrix(rnorm(160),40,4)
fm = matrix(runif(40),4,10)
Am = xm%*%fm+rnorm(400,0,.1)

# create model
obj = create_TRMF(Am)
obj = TRMF_simple(obj,numTS=4,lambdaA=0.1)
out = train(obj)
plot(out)
```

TRMF_trend *Add Trend Model to a TRMF Object*

Description

Creates a regularization scheme that favors trend-like solutions and adds it to a TRMF object. In matrix optimization form, it adds the following term to the TRMF cost function: $R(x) = lambdaD^2||w(DX_s)||^2 + lambdaA^2||X_s||^2$ where X_s is sub-set of the Xm matrix controlled by this model and D is a finite difference matrix.

Usage

```
TRMF_trend(obj,numTS = 1,order = 1,lambdaD=1,lambdaA=0.0001,weight=1)
```

Arguments

obj	A TRMF object
numTS	number of latent time series in this model
order	The order of derivative for finite difference constraint matrix. Fractionally and negative values allowed.
lambdaD	regularization parameter for temporal constraint matrix
lambdaA	regularization parameter to apply simple L2 regularization to this time series model
weight	optional vector of weights to weight constraints, i.e. R(x) = lambdaD^2*‖w*(D%*%X)‖^2

Details

An arbitrary number of time series models can be added. `TRMF_trend(order = 1)` fits a random walk. `TRMF_trend(order = 2)` fits a cubic smoothing spline. For a single time series, `TRMF_trend(order = 2)` is basically equivalent to the Hodge-Prescot filter. A fractional value for order minimizes a squared fractional derivative. A negative value minimizes a (possibly fractional order) squared integral of time-series. Using a fractional or negative order for `TRMF_trend` could drastically reduce the sparsity of constraint matrix and slow down training. Fractional or negative order has only been lightly tested, so use with care.

Value

Returns an updated object of class TRMF.

Note

Unlike `TRMF_columns()`, these lambdas are squared in the code.

Author(s)

Chad Hammerquist

References

Yu, Hsiang-Fu, Nikhil Rao, and Inderjit S. Dhillon. "High-dimensional time series prediction with missing values." arXiv preprint arXiv:1509.08333 (2015).

See Also

`create_TRMF`, `TRMF_columns`, `TRMF_simple`, `TRMF_seasonal`

Examples

```
# create test data
xm = poly(x = (-10:10)/10,degree=4)
fm = matrix(runif(40),4,10)
Am = xm%*%fm+rnorm(210,0,.1)

# create model
obj = create_TRMF(Am)
obj = TRMF_columns(obj,reg_type ="interval")
obj = TRMF_trend(obj,numTS=4,order=2,lambdaD=2)
out = train(obj)
plot(out)

# more complex model
require(magrittr) # for pipes

obj = create_TRMF(Am)%>%
    TRMF_columns(reg_type ="interval")%>%
    TRMF_trend(numTS=2,order=1,lambdaD=4)%>%
    TRMF_trend(numTS=2,order=2,lambdaD=4)%>%
```

```
    TRMF_trend(numTS=1,order=1.5)

out = train(obj)
plot(out)
```

Index