Package ‘learner’

July 22, 2025

Type Package

Title Latent Space-Based Transfer Learning

Version 1.0.0

Maintainer Sean McGrath <sean.mcgrath514@gmail.com>

Description Implements transfer learning methods for low-rank matrix estimation. These methods leverage similarity in the latent row and column spaces between the source and target populations to improve estimation in the target population. The methods include the LatEnt spAce-based tRaNsfer lEaRning (LEARNER) method and the direct projection LEARNER (D-LEARNER) method described by McGrath et al. (2024) <doi:10.48550/arXiv.2412.20605>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

URL https://github.com/stmcg/learner

BugReports https://github.com/stmcg/learner/issues

Imports ScreeNOT, Rcpp (>= 1.0.11), RcppEigen

LinkingTo Rcpp, RcppEigen

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 2.10)

NeedsCompilation yes

Author Sean McGrath [aut, cre] (ORCID: <https://orcid.org/0000-0002-7281-3516>),
Ryan ODea [aut],
Cenhao Zhu [aut],
Rui Duan [aut]

Repository CRAN

Date/Publication 2025-03-03 10:00:02 UTC

Contents

`cv.learner` *Cross-validation for LEARNER*

Description

This function performs k-fold cross-validation to select the nuisance parameters (λ_1, λ_2) for `learner`.

Usage

```
cv.learner(
  Y_source,
  Y_target,
  r,
  lambda_1_all,
  lambda_2_all,
  step_size,
  n_folds = 4,
  n_cores = 1,
  control = list()
)
```

Arguments

`Y_source`	matrix containing the source population data, as in `learner`
`Y_target`	matrix containing the target population data, as in `learner`
`r`	(optional) integer specifying the rank of the knowledge graphs, as in `learner`
`lambda_1_all`	vector of numerics specifying the candidate values of λ_1 (see Details)
`lambda_2_all`	vector of numerics specifying the candidate values of λ_2 (see Details)
`step_size`	numeric scalar specifying the step size for the Newton steps in the numerical optimization algorithm, as in `learner`
`n_folds`	an integer specify the number of cross-validation folds. The default is 4.
`n_cores`	an integer specifying the number of CPU cores in OpenMP parallelization. Parallelization is performed across the different candidate (λ_1, λ_2) pairs. The default is 1, i.e., no parallelization.
`control`	a list of parameters for controlling the stopping criteria for the numerical optimization algorithm, as in `learner`.

Details

Given sets of candidate values of λ_1 and λ_2, this function performs k-fold cross-validation to select the pair (λ_1, λ_2) with the smallest held out error. This function randomly partitions the entries of `Y_target` into k (approximately) equally sized subsamples. The training data sets are obtained by removing one of the k subsamples and the corresponding test data sets are based on the held out subsamples. The `learner` function is applied to each training data set. The held out error is computed by the mean squared error comparing the entries in the test data sets with those imputed from the LEARNER estimates. See McGrath et al. (2024) for further details.

Value

A list with the following elements:

`lambda_1_min`	value of λ_1 with the smallest MSE
`lambda_2_min`	value of λ_2 with the smallest MSE
`mse_all`	matrix containing MSE value for each (λ_1, λ_2) pair. The rows correspond to the λ_1 values, and the columns correspond to the λ_2 values.
`r`	rank value used.

References

McGrath, S., Zhu, C,. Guo, M. and Duan, R. (2024). *LEARNER: A transfer learning method for low-rank matrix estimation*. arXiv preprint arXiv:2412.20605.

Examples

```
res <- cv.learner(Y_source = dat_highsim$Y_source,
                  Y_target = dat_highsim$Y_target,
                  lambda_1_all = c(1, 10, 100),
                  lambda_2_all = c(1, 10, 100),
                  step_size = 0.003)
```

`dat_highsim` *Simulated data set: High similarity in the latent spaces*

Description

This data set contains simulated data in the source and target populations where there is a high degree of similarity in the underlying latent spaces between these populations.

Usage

```
dat_highsim
```

Format

A list containing the observed and true matrices in the source and target populations. The list contains the following components:

`Y_source` A matrix of size 100×50 representing the observed source population matrix.

`Y_target` A matrix of size 100×50 representing the observed target population matrix.

`Theta_source` A matrix of size 100×50 (rank 3) representing the true source population matrix.

`Theta_target` A matrix of size 100×50 (rank 3) representing the true target population matrix.

Details

In this data set, there is a high degree of similarity in the latent spaces between the source and target populations. Specifically, the true source population matrix was obtained by reversing the order of the singular values of the true target population matrix. The observed target population matrix was obtained by adding independent and identically distributed noise to the entries of the true source population matrix. The noise was generated from a normal distribution with mean 0 and standard deviation of 1. The observed source population matrix was generated analogously, where the noise had a standard deviation of 0.5.

See Also

`dat_modsim`

`dat_modsim` *Simulated data set: Moderate similarity in the latent spaces*

Description

This data set contains simulated data in the source and target populations where there is a moderate degree of similarity in the underlying latent spaces between these populations.

Usage

```
dat_modsim
```

Format

A list containing the observed and true matrices in the source and target populations. The list contains the following components:

`Y_source` A matrix of size 100×50 representing the observed source population matrix.

`Y_target` A matrix of size 100×50 representing the observed target population matrix.

`Theta_source` A matrix of size 100×50 (rank 3) representing the true source population matrix.

`Theta_target` A matrix of size 100×50 (rank 3) representing the true target population matrix.

Details

In this data set, there is a moderate degree of similarity in the latent spaces between the source and target populations. Specifically, the true source population matrix was obtained by (i) reversing the order of the singular values of the true target population matrix and (ii) adding perturbations to the left and right singular vectors of the true target population matrix. The observed target population matrix was obtained by adding independent and identically distributed noise to the entries of the true source population matrix. The noise was generated from a normal distribution with mean 0 and standard deviation of 1. The observed source population matrix was generated analogously, where the noise had a standard deviation of 0.5.

See Also

dat_modsim

`dlearner` *Latent space-based transfer learning*

Description

This function applies the Direct project LatEnt spAce-based tRaNsfer lEaRning (D-LEARNER) method (McGrath et al. 2024) to leverage data from a source population to improve estimation of a low rank matrix in an underrepresented target population.

Usage

```
dlearner(Y_source, Y_target, r)
```

Arguments

`Y_source`	matrix containing the source population data
`Y_target`	matrix containing the target population data
`r`	(optional) integer specifying the rank of the knowledge graphs. By default, ScreeNOT (Donoho et al. 2023) is applied to the source population knowledge graph to select the rank.

Details

Data and notation:

The data consists of a matrix in the target population $Y_0 \in \mathbb{R}^{p \times q}$ and the source population $Y_1 \in \mathbb{R}^{p \times q}$. Let $\hat{U}_k \hat{\Lambda}_k \hat{V}_k^\top$ denote the truncated singular value decomposition (SVD) of Y_k, $k = 0, 1$.

For $k = 0, 1$, one can view Y_k as a noisy version of Θ_k, referred to as the knowledge graph. The target of inference is the target population knowledge graph, Θ_0.

Estimation:

This method estimates Θ_0 by $\hat{U}_1^\top \hat{U}_1 Y_0 \hat{V}_1^\top \hat{V}_1$.

Value

A list with the following components:

`dlearner_estimate`
: matrix containing the D-LEARNER estimate of the target population knowledge graph.

`r`
: rank value used.

References

Donoho, D., Gavish, M. and Romanov, E. (2023). *ScreeNOT: Exact MSE-optimal singular value thresholding in correlated noise*. The Annals of Statistics, 51(1), pp.122-148.

Examples

```
res <- dlearner(Y_source = dat_highsim$Y_source,
                Y_target = dat_highsim$Y_target)
```

`learner` *Latent space-based transfer learning*

Description

This function applies the LatEnt spAce-based tRaNsfer lEaRning (LEARNER) method (McGrath et al. 2024) to leverage data from a source population to improve estimation of a low rank matrix in an underrepresented target population.

Usage

```
learner(Y_source, Y_target, r, lambda_1, lambda_2, step_size, control = list())
```

Arguments

`Y_source`	matrix containing the source population data
`Y_target`	matrix containing the target population data
`r`	(optional) integer specifying the rank of the knowledge graphs. By default, ScreeNOT (Donoho et al. 2023) is applied to the source population knowledge graph to select the rank.
`lambda_1`	numeric scalar specifying the value of λ_1 (see Details)
`lambda_2`	numeric scalar specifying the value of λ_2 (see Details)
`step_size`	numeric scalar specifying the step size for the Newton steps in the numerical optimization algorithm

`control` a list of parameters for controlling the stopping criteria for the numerical optimization algorithm. The list may include the following components:

`max_iter`	integer specifying the maximum number of iterations
`threshold`	numeric scalar specifying a convergence threshold. The algorithm converges when $\lvert\epsilon_t - \epsilon_{t-1}\rvert <$`threshold`, wh
`max_value`	numeric scalar used to specify the maximum value of the objective function allowed before terminating the algo

Details

Data and notation:

The data consists of a matrix in the target population $Y_0 \in \mathbb{R}^{p\times q}$ and the source population $Y_1 \in \mathbb{R}^{p\times q}$. Let $\hat{U}_k\hat{\Lambda}_k\hat{V}_k^\top$ denote the truncated singular value decomposition (SVD) of Y_k, $k = 0, 1$.

For $k = 0, 1$, one can view Y_k as a noisy version of Θ_k, referred to as the knowledge graph. The target of inference is the target population knowledge graph, Θ_0.

Estimation:

This method estimates Θ_0 by $\tilde{U}\tilde{V}^\top$, where $(\tilde{U}, \tilde{V})$ is the solution to the following optimization problem

$\arg\min_{U\in\mathbb{R}^{p\times r}, V\in\mathbb{R}^{q\times r}} \big\{ \|UV^\top - Y_0\|_F^2 + \lambda_1\|\mathcal{P}_\perp(\hat{U}_1)U\|_F^2 + \lambda_1\|\mathcal{P}_\perp(\hat{V}_1)V\|_F^2 + \lambda_2\|U^\top U - V^\top V\|_F^2 \big\}$

where $\mathcal{P}_\perp(\hat{U}_1) = I - \hat{U}_1^\top\hat{U}_1$ and $\mathcal{P}_\perp(\hat{V}_1) = I - \hat{V}_1^\top\hat{V}_1$.

This function uses an alternating minimization strategy to solve the optimization problem. That is, this approach updates U by minimizing the objective function (via a gradient descent step) treating V as fixed. Then, V is updated treating U as fixed. These updates of U and V are repeated until convergence.

Value

A list with the following elements:

`learner_estimate`
matrix containing the LEARNER estimate of the target population knowledge graph

`objective_values`
numeric vector containing the values of the objective function at each iteration

`convergence_criterion`
integer specifying the criterion that was satisfied for terminating the numerical optimization algorithm. A value of 1 indicates the convergence threshold was satisfied; A value of 2 indicates that the maximum number of iterations was satisfied; A value of 3 indicates that the maximum value of the objective function was satisfied.

`r` rank value used.

References

McGrath, S., Zhu, C,. Guo, M. and Duan, R. (2024). *LEARNER: A transfer learning method for low-rank matrix estimation*. arXiv preprint arXiv:2412.20605.

Donoho, D., Gavish, M. and Romanov, E. (2023). *ScreeNOT: Exact MSE-optimal singular value thresholding in correlated noise*. The Annals of Statistics, 51(1), pp.122-148.

Examples

```
res <- learner(Y_source = dat_highsim$Y_source,
               Y_target = dat_highsim$Y_target,
               lambda_1 = 1, lambda_2 = 1,
               step_size = 0.003)
```

Index