skprocrustes Documentation

Release 0.1

Melissa Weber Mendonça

Jul 06, 2017

CONTENTS

Collection of solvers for the (Weighted) Orthogonal Procrustes Problem.

$$\min \|AXC - B\|_F^2 \qquad s.t. \quad X^T X = I$$

where $A_{m\times n}, B_{m\times q}, C_{p\times q}, X_{n\times p}$. Usually n >> p, which means we can solve unbalanced problems.

CHAPTER

ONE

AVAILABLE SOLVERS

- *spg* Nonmonotone Spectral Projected Gradient Method for the (unbalanced) WOPP, as described in[1].
- *gkb* Nonmonotone Spectral Projected Gradient Method using incomplete Lanczos (Golub-Kahan) Bidiagonalization, as described in[2].
- *eb* Expansion-Balance method, as described in[3].
- *gpi* Generalized Power Iteration for the WOPP, as described in[5].

[1] J.B. Francisco, F.S. Viloche Bazán, Nonmonotone algorithm for minimization on closed sets with applications to minimization on Stiefel manifolds, Journal of Computational and Applied Mathematics, 2012, 236(10): 2717–2727 <http://dx.doi.org/10.1016/j.cam.2012.01.014>

[2] J.B. Francisco, F.S. Viloche Bazán and M. Weber Mendonça, Non-monotone algorithm for minimization on arbitrary domains with applications to large-scale orthogonal Procrustes problem, Applied Numerical Mathematics, 2017, 112: 51–64 <https://doi.org/10.1016/j.apnum.2016.09.018>

[3] J.M.F. ten Berge and D.L. Knol, Orthogonal rotations to maximal agreement for two or more matrices of different column orders, Psychometrika 1984, 49: 49–55 <https://doi:10.1007/BF02294205>

[5]

6. Nie, R. Zhang, X. Li, A generalized power iteration method for solving quadratic problem on the Stiefel manifold, Sci. China Inf. Sci., 2017, 60: 112101:1–112101:10. <http://dx.doi.org/10.1007/s11432-016-9021-9>

CHAPTER

TWO

USAGE

To use the package to solve a given problem with predefined matrices A, B and C using the SPG solver, for example, use

```
>>> import skprocrustes as skp
>>> problem = skp.ProcrustesProblem((m,n,p,q),      # tuple
                                    matrices=[A, B, C])
>>> mysolver = skp.SPGSolver(**kwargs)
>>> result = mysolver.solve(problem)
```

where ***kwargs* are the selected solver's options (see the Module Reference for more details).

To use the package to solve one of the three predefined problems (as described in[4]), using the GKB solver, for example, use

```
>>> import skprocrustes as skp
>>> problem = skp.ProcrustesProblem((m,n,p,q),      # tuple
                                    problemnumber=1)
>>> mysolver = skp.GKBSolver(**kwargs)
>>> result = mysolver.solve(problem)
```

4

26. Zhang, K. Du, Successive projection method for solving the unbalanced Procrustes problem, Sci. China Ser. A, 2006, 49: 971–986.

CHAPTER

THREE

REFERENCES

CHAPTER

FOUR

INSTALLATION

4.1 Quick Installation

In the root directory of the package, just do:

```
python setup.py install
```

4.2 Latest Software

The latest software can be downloaded from GitHub

4.3 Installation Dependencies

`scikit-procrustes` requires the following software packages to be installed:

- Python 3.6.1 or later.
- NumPy 1.13.0 or later.
- SciPy 0.19.0 or later.
- Matplotlib 2.0.2 or later.

CHAPTER

FIVE

CONTENTS

5.1 skprocrustes package

5.1.1 Module contents

class skprocrustes.**ProcrustesProblem**(*sizes*, *problemnumber=None*, *matrices=[]*)

Bases: object

The problem we want to solve.

Usage example (default problem):

```
>>> import skprocrustes as skp
>>> problem = skp.ProcrustesProblem((10,10,2,2), problemnumber=1)
```

Usage example (user defined problem):

```
>>> import skprocrustes as skp
>>> import numpy as np
>>> A = ... # given by the user
>>> B = ... # given by the user
>>> C = ... # given by the user
>>> X = ... # given by the user (optional)
>>> problem = skp.ProcrustesProblem((m,n,p,q), matrices=(A,B,C,X))
```

Input Parameters:

sizes: tuple (m,n,p,q), where $A_{m\times n}, B_{m\times q}, C_{p\times q}$ and $X_{n\times p}$.

***(optional)* problemnumber: int** Can be 1, 2 or 3, and selects one of the predefined problems as described in reference *[4]*. (for more details, see the documentation for _setproblem)

***(optional)* matrices: list of ndarrays** If present, must contain a list of three or four matrices corresponding to A, B, C, and optionally X (known solution) with adequate sizes.

Note: Currently, m must be equal do n, and p must be equal do q. This is the case for all three solvers. (However, n can be greater than p)

Note: If matrices is not given by the user, problemnumber (1, 2 or 3) must be selected so that one of the default problems is built.

Attributes:

The problem matrices (generated or given) are accessible via