nearest_corr_mat_pkg

thibault.bourgeron@protonmail.com

Abstract

This package is intended for a fast computation of the nearest correlation matrix, in the Euclidian sense.

1 Problem and its dual

A correlation matrix is a real symmetric positive semidefinite matrix with unit diagonal. The problem is to find the closest correlation matrix to a given positive semidefinite matrix, for the Frobenius norm.

With symbols, the problem is:

$$\text{min. } \frac{1}{2}\|G - X\|^2 \quad \text{s.t. } \text{diag}(X) = 1, X \in S_n^+,$$

where S_n^+ (resp. S_n) is the cone (resp. space) of real positive definite matrices (resp. real symmetric matrices), G is a given positive semidefinite matrix, $\|G\|$ is the Frobenius norm, $\text{diag} : S_n \to \mathbb{R}^n$ is the linear operator returning the diagonal vector of a matrix, 1 is the vector of all ones in $\mathbb{R}^n$.

The functional is convex and smooth. The constrained problem has a unique solution. The affine space $\{X \in S_n \mid \text{diag}(X) = 1\}$ and the cone S_n^+ are convex sets of S_n.

The dual formulation considered here is:

$$\text{min. } \theta(y) := \frac{1}{2}\|(G + \text{Diag}(y))_+\| - 1^T y,$$

where $\text{Diag} : \mathbb{R}^n \to S_n$ is the adjoint of diag, $+ : S_n \to S_n^+$ is the projection onto S_n^+. Its gradient:

$$\nabla\theta(y) := \text{diag}(G + \text{Diag}(y)_+) - 1,$$

is the most important function in this problem, as it is enough to find a zero of the gradient, say y^*, to solve the primal problem, thanks to the formula:

$$x^* := G + \text{Diag}(y^*)_+.$$

The dual problem is convex, unconstrained, and $\nabla\theta$ is strongly semismooth (but not differentiable).

The more use of the regularity of these functionals is performed, the more efficient the algorithm, theoretically.

Although the algorithms below were not published recently, I am not aware of any freely accessible implementation, apart from the Python module Statsmodels which implements the simplest algorithm.

2 Four algorithms implemented

As the problem is constrained in the intersection of two convex sets, Von Neumann alternative projection can be used, *cf.* `performances.ipynb`. This algorithm may converge to a sub optimal solution (as S_n^+ is not a vector space). Dykstra's projection does not converge at all, *cf.* `performances.ipynb`.

Four methods, are, either indirectly or directly, based on the dual problem, and are implemented in the main function `nearest_corr`.

1. The algorithm 'grad' is based on Higham, 2002. It is a modified Dykstra's projection algorithm, adapted to find a point in the intersection of an affine subspace and convex subset. As noted in Malick, 2004, in a general way, this algorithm is exactly a gradient descent for the dual problem.
2. The algorithm 'bfgs' is the BFGS algorithm applied to the convex, differentiable, unconstrained dual problem, as suggested in Malick. The implementations are based on BGLS textbook.
3. The algorithm 'l_bfgs' is the limited BFGS algorithm applied to the dual problem, as suggested in Malick. The implementations are based on the same book.
4. The algorithm 'newton' is a Newton's method for the dual problem. It uses the strong semismoothness of $\nabla\theta$ and an explicit construction of elements of its Clarke Jacobian. It has been developed in Qi, Sun, 2006.

3 Performances

On random 100×100 matrices (*cf.* `test.py`), for $err\_max = 10^{-6}$, on a given machine, the algorithms have the following performances.

algo	time	n_iterations
grad	20 s	1500
bfgs	2 s	100
l-bfgs	2 s	100
newton	60 s	100

The (complicated, double-loop) construction of the Newton's matrix for 'newton' algorithm may be responsible for this algorithm to be slowest, for this set of parameters. For higher dimension problems (1000×1000), the method 'l_bfgs' seems to be the fastest (when 'memory' remains unchanged).

For none of the method a line search is performed because a Wolfe (or a even simpler) line search considerably slows down all the methods and setting the step to 1 is a simple and good enough (this is consistent with a remark in Qi, Sun and with the initial value for quasi-Newton methods; experimentaly, Wolfe condition is often satisfied for value 1). For 'newton', for the same speed reason, the conjugate gradient is not used; Numpy's solve is used instead, and the direction checking is not performed.

For all the methods, for $err\_max = 10^{-6}$, the convergence seem to be linear. This is expected for the modified Dysktra's projection. Quasi-Newton methods converge superlinearly when the functional is smooth and a Wolfe line search is performed, for instance. The algorithms 'bfgs', 'l_bfgs' seem to converge only linearly, maybe because the gradient of the functional is only semismooth. It is surprising for the Newton's method to have only a linear convergence, because it is proved to be quadratically convergent. The quadratic convergence may be observed for other ranges of parameters, as noted in Qi, Sun.