Algorithm Package Manual: Best Low Rank Tensor Approximation

Berkant Savas
Department of Mathematics, Linköping University
besav@math.liu.se

June 19, 2009

Abstract

This report is a short manual for algorithms for computing best low (multilinear) rank approximation of tensors. The algorithms are based on Newton and quasi-Newton methods operating on product of Grassmann manifolds. The package is implemented in MATLAB and uses the MATLAB Tensor Toolbox and Grassmann Classes toolbox. General tensors of arbitrary order can be approximated. For symmetric tensors order-3 and order-4 tensors can be approximated.

1 Contents of package

This package contains implementation of seven algorithms described in the references [3, 6] for the best low rank approximation of tensors. There are over 20 functions supporting the different algorithms. The algorithms are based on the MATLAB Tensor Toolbox[1] [1] and the Grassmann Classes[2] [5]. I am assuming the reader has the necessary theoretical knowledge about the best low multilinear rank approximation of tensors. Otherwise consider e.g. [3, 6] and the references therein.

File name	Algorithm description
`algNgc.m`	Newton-Grassmann method implemented in *global* coordinates.
`algNlc.m`	Newton-Grassmann method implemented in *local* coordinates.
`algQNgc.m`	Quasi-Newton-Grassmann method with BFGS updates in *global* coordinates.
`algQNlc.m`	Quasi-Newton-Grassmann method with BFGS updates in *local* coordinates.
`algLBFGS.m`	Quasi-Newton-Grassmann method with limited memory L-BFGS updates implemented using *global* coordinates.
`symalgQNgc.m`	Quasi-Newton-Grassmann method with BFGS updates in *global* coordinates for symmetric tensors.
`symalgQNlc.m`	Quasi-Newton-Grassmann method with BFGS updates in *local* coordinates for symmetric tensors.

All algorithms work with third order tensors. The two algorithms for symmetric tensor work on fourth order tensors as well. In addition `algQNlc.m` works for arbitrary order tensors within the memory limitations of a computer.

This package is available at http://www.mai.liu.se/~besav/soft.html. To test the algorithms download the zip-file, unpack it and run the `test.m` file.

[1] http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/
[2] http://www.mai.liu.se/~besav/soft.html

2 Function descriptions

In this section we will give a short description of the different files and how they can be used.

2.1 Newton-Grassmann algorithms

The first two algorithms `algNgc.m` and `algNlc.m` have the same input arguments and give the same output arguments. The inputs are (in the given order): (1) The original tensor `A` which we want to approximate; (2) An initial approximation to the solution, which is given in a cell array; (3) Dimensions of the tensor `A`, denoted `n`; (4) The multilinear rank of the approximating tensor, denoted `r`; (5) The maximal number of iterations; (6) The required relative tolerance, i.e. $\nabla f(X,Y,Z)/f(X,Y,Z)$ where

$$f(X,Y,Z) = \|\mathcal{A} \cdot (X,Y,Z)\|$$

Here is an example of how it may look.

```
>> tol = 5e-13; maxIt = 10;
>> n = [7 10 11]'; r = [3 4 5]';
>> A = tensor(randn(n'));
>> % Random initial approximations of the matrices
>> X{1} = orth(randn(n(1),r(1)));
>> X{2} = orth(randn(n(2),r(2)));
>> X{3} = orth(randn(n(3),r(3)));
>> [X1,f1,ng1] = algNgc(A,X,n,r,maxIt,tol);
>> [X2,f2,ng2] = algNlc(A,X,n,r,maxIt,tol);
```

The algorithms return the computed matrices X, Y, Z stored in the cell arrays `X1` and `X2` that (locally) maximize the objective function $f(X,Y,Z)$. The second and third output, `f1` and `ng1` (also `f2` and `ng2`), contain the function values and the norm of the gradient from each iteration.

Remark 2.1. In Newton algorithms it is important to have good initial approximation, i.e. to be sufficiently close to a local minimizer. If this is not the case the Hessian of the objective function may not be negative definite which will (most of the time) cause the algorithm to diverge. No checks are made to see or ensure that the Hessian at the initial approximation is negative definite. There are various ways to fix this, see e.g. [4]. In our experiments the one method that worked best was to flip the signs of the positive eigenvalues of the Hessian. One should be aware that this does still require to be quite close to a local maximizer and not to forget the (expensive) computation of the eigenvalues and the corresponding eigenvectors of the Hessian.

Theoretically the two algorithms are equivalent, i.e. they give the very same iterates. In practice there are differences but these differences are very small. Run the `test.m` file and compare the two Newton-Grassmann plots.

2.2 Quasi-Newton-Grassmann (BFGS) algorithms

The algorithms `algQNgc.m` and `algQNlc.m` are Quasi-Newton algorithms with BFGS updates of the Hessian approximation implemented on a product of Grassmannians. The first algorithm is implemented using global coordinates and the second algorithm is implemented using local coordinates. These two algorithms give the same outputs as the Newton algorithms in Section 2.1, i.e. a cell array with the matrices X, Y, Z, the function values from each iteration and the norm of the gradient from each iteration. The inputs and ordering to the quasi-Newton algorithms is almost the same as for the Newton algorithms. There is an additional text string input for quasi-Newton algorithms which indicates the form of the initial Hessian in the BFGS updates. Currently one may chose to initiate the Hessian approximation with the exact Hessian using the string `'exHess'` or a suitably scaled identity matrix using the string `'scldIdent'` (see [4] for details). The function input arguments specifying the initial Hessian is put last in the argument list. Assuming necessary variables are initiated we call these two algorithms with the following commands. Here we initiated with the exact Hessian

```
>> % Variable X contains the initial approximations.
>> [X3, f3,ng3] = algQNgc(A,X,n,r,maxIt,tol,'exHess');
>> [X4, f4,ng4] = algQNlc(A,X,n,r,maxIt,tol,'exHess');
```

and here with a scaled identity.

```
>> [X3,f3,ng3] = algQNgc(A,X,n,r,maxIt,tol,'scldIdent');
>> [X4,f4,ng4] = algQNlc(A,X,n,r,maxIt,tol,'scldIdent');
```

Also for quasi-Newton algorithms, when the initial Hessian is specified, the outputs of local and global coordinate implementations are practically the same.

2.3 Limited memory Quasi-Newton-Grassmann (L-BFGS) algorithms

The fifth algorithm is a quasi-Newton algorithm implemented using the limited memory L-BFGS update. The Hessian is stored in factored form and only the last m s_k and y_k vectors are used to form the Hessian approximation. The intended use of L-BFGS methods are for large problems, in which it is not suited to compute an exact Hessian[3]. For this reason the `algLBFGS.m` only uses the scaled identity as an approximation to the initial Hessian and thus there is no need to explicitly specify this with input arguments. The integer m, stating the number of vectors used in the approximation, has to be specified in the argument list. Compared to the input arguments for the Newton algorithms there is this addition to the list. Assuming necessary variables are initiated the command has this form,

```
>> m = 10;  % Number of vectors in the L-BFGS method.
>> [X5,f5,ng5] = algLBFGS(A,X,n,r,maxIt,tol,m);
```

2.4 Test the code and compare the algorithms!

The package contains a `test.m` file which runs all of these algorithms and plots the results for comparison with an Alternating Least Squares (ALS) implementation of the tensor approximation problem, [2], also know as the Higher Order Orthogonal Iteration.

References

[1] B. W. Bader and T. G. Kolda. Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. *ACM Trans. Math. Softw.*, 32(4):635–653, 2006.

[2] L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

[3] L. Eldén and B. Savas. A Newton–Grassmann method for computing the best multi-linear rank-(r_1, r_2, r_3) approximation of a tensor. *SIAM J. Matrix Anal. Appl.*, 31:248–271, 2009.

[4] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.

[5] B Savas. Algorithm package manual: Best low rank tensor approximation. Department of Mathematics, Linköping Univeristy, 2008. http://www.mai.liu.se/~besav/soft.html.

[6] B. Savas and L.-H. Lim. Quasi-Newton methods on Grassmannians and multilinear approximations of tensors. *Submitted to SIAM Journal on Optimization*, 2009.

[3] The Hessian may be large but more important issue is the it is required to be in factored form.