Sparse Array Toolbox

Andrew J. Milne*

The MARCS Institute for Brain, Behaviour and Development, Western Sydney University, NSW 2751, Australia

1. Sparse Array Toolbox

For sparse data, sparse formats can significantly increase speed and reduce memory requirements. However, MATLAB's built-in sparse format represents only vectors and matrices, not N-dimensional arrays (tensors). This toolbox contains a new *sparse array structure* and associated operations – outer, entrywise, and inner products, addition, summation, convolution, permutation, (circular) shifts, and distance measures – that can be applied to sparse representations of N-dimensional arrays. All functions have been carefully designed to optimize speed.

A sparse array structure has the following fields:

- `Size`, which is a row vector of the sizes of each dimension of the full array – for example, a column vector with N entries has a size of N; a row vector with N entries has a size of $(1, N)$; a matrix with N entries has a size (J, K), where $JK = N$; a three-way array with N entries has a size (J, K, L), where $JKL = N$;
- `Ind`, which is a column vector of linear indices of nonzero values in the full array that they represent;
- `Val`, which is a column vector of the values at those indices.

1.1. *array2spArray*

`spA = array2spArray(A)`: Convert a full array into a sparse array structure. All singleton dimensions are removed.

1.2. *spArray2Array*

`A = spArray2Array(spA)`: Convert a sparse array structure into a full array.

1.3. *spInd2spSub*

`subsA = spInd2spSub(spA)`: Convert a sparse array's linear index into a matrix of subscripts.

*Corresponding author. Email: a.milne@westernsydney.edu.au

1.4. *spSub2spInd*

`indC = spSub2spInd(siz,subsA)`: Convert a matrix of subscripts into a vector of the linear indices for an array of size `siz`.

1.5. *spOuter*

`spC = spOuter(varargin)`: The outer (tensor) product of full arrays, each represented as a sparse array structure (spA). The sparse array structures can be entered as a comma separated list or as a members of a cell. The output is a sparse array structure. Calculations are performed from left to right in the list; that is, `spOuter(spA,spB,spC,spD)` corresponds to $(((A \otimes B) \otimes C) \otimes D)$. All singleton dimensions are collapsed: if A is a row vector of size $(1, M)$ and B is a matrix of size (N, P), the resulting tensor has size (M, N, P), not size $(1, M, N, P)$. The output is a sparse array structure.

1.6. *spTimes*

`spC = spTimes(varargin)`: Entrywise (Hadamard) product of full arrays and/or scalars, the former represented as sparse array structures. The arrays and scalars can be entered as a comma separated list or as a members of a cell. The output is a sparse array structure.

1.7. *spInner*

`c = spInner(varargin)`: Inner (scalar) product of two full arrays represented by sparse array structures. There are alternative definitions of 'inner product' for tensors/arrays. Here, it is their scalar product – the sum of entries resulting from their entrywise (Hadamard) product. The sparse arrays can entered as a comma separated list or as a members of a cell.

1.8. *spPlus*

`spC = spPlus(varargin)`: The sum of identically sized full arrays, each represented as a sparse array structure. The sparse arrays can entered as a comma separated list or as a members of a cell. The output is a sparse array structure.

1.9. *spSum*

`spC = spSum(spA,dim)`: Sum a full array, represented as a sparse array structure, over the dimension specified as a scalar in `dim`. The output is a sparse array structure.

1.10. *spConv*

`spC = spConv(spA,spB,shape)`: The N-dimensional convolution of two N-dimensional arrays represented as sparse array structures. The output is a sparse array structure.

`shape == 'full'`: full convolution (default). Its size is the sum of the sizes of its arguments.

`shape == 'same'`: central part of the convolution, same size as spA.

`shape == 'circ'`: circular convolution over the size of spA.

1.11. *spPerm*

`spC = spPerm(spA,order)`: Permute the dimensions of the full array represented as a sparse array structure. The second argument is the vector of permutations. The output is a sparse array structure.

1.12. *spShift*

`spC = spShift(spA,shifts,isPer,isProg,collapse)`: Shift each dimension of the full array represented as a sparse array structure by the amounts specified in the integer row vector or matrix `shifts`. The output is a sparse array structure.

When `shifts` is a row vector, all entries of the array are shifted by the amounts specified in `shifts`: the nth entry of `shifts` is the shift for the nth dimension of the N-dimensional array. When `shifts` is a matrix, its mth row gives the N-dimensional shift for the mth nonzero element of the full array (i.e., the mth entry of sparse array structure's `Ind` and `Val` fields).

`isPer == 1`: the shifts are circular (default).

`isProg == 1`: the shifts, as specified by the shift vector, for all dimensions except the last are multiplied by the index of the last dimension. For example, if the shift value for the first dimension is m, the shift of that dimension when the final dimension's index is n is mn. Default is `isProg = 0`.

`collapse == 1`: the array is summed over this last dimension. Default is `collapse = 0`.

Setting `isProg = 1` and `collapse = 1` is useful for converting absolute r-ad expectation tensors into relative r-ad expectation tensors (Milne et al. 2011).

1.13. *spCosSim*

`s = spCosSim(spA,spB)`: Cosine similarity of two vectorized full arrays, represented as sparse array structures, with same numbers of entries. The output is a scalar.

1.14. *spPDist*

`d = spPDist(spA,spB,p)`: The p-norm distance between two vectorized full arrays, represented as sparse array structures, with same numbers of entries. When there are only two arguments, `p = 2` is the default, which gives the Euclidean distance between the two vectors; when `p == 1`, this function gives the taxicab distance; when `p == inf`, it gives the maximum difference. The output is a scalar.

Acknowledgements

Dr Andrew Milne is the recipient of an Australian Research Council Discovery Early Career Award (project number DE170100353) funded by the Australian Government.

References

Milne, A. J., Sethares, W. A., Laney, R., and Sharp, D. B. (2011). Modelling the similarity of pitch collections with expectation tensors. *Journal of Mathematics and Music*, 5(1):1–20.