Sparse Distributed Memory

Marcelo Salhab Brogliato, Alexandre Linhares

Getulio Vargas Foundation

Abstract

Ca. 100 words

Keywords: Theoretical Neuroscience, Artificial Intelligence, Machine Learning, Memory, Cognitive Science

Required Metadata

1

Current code version

Ancillary data table required for subversion of the codebase. Kindly replace examples in right column with the correct information about your current code, and leave the left column as it is.

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v.1.6
C2	Permanent link to code/repository used for this code version	$https : //github.com/msbrogli/sdm - framework/releases/tag/v1.6$
C3	Code Ocean compute capsule	????????? For example: $https : //codeocean.com/2017/07/30/neurospeech - colon - an - open - source - software - for - parkinson - apos - s - speech - analysis/code$
C4	Legal Code License	GPL-2.0
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	C, python, OpenCL, Docker
C7	Compilation requirements, operating environments	python (anaconda), , libbsd
C8	If available Link to developer documentation/manual	For example: $https : //buildmedia.readthedocs.org/media/pdf/sdm - framework/stable/sdm - framework.pdf$
C9	Support email for questions	linhares@sdm.ai

Table 1: Code metadata (mandatory)

The permanent link to code/repository or the zip archive should include the following requirements:

README.txt and LICENSE.txt.

Source code in a src/ directory, not the root of the repository.

Tag corresponding with the version of the software that is reviewed.

Documentation in the repository in a docs/ directory, and/or READMEs, as appropriate.

1. Motivation and significance

Sparse Distributed Memory (SDM) [?] (see also [? ? ? ? ? ? ? ? ? ? ?]) is a mathematical model of long-term memory that has a number of neuroscientific and psychologically plausible dynamics. This model is used in all sort of applications due to its incredible ability to closely reflect the human capacity to remember past experiences from the subtlest of clues. Applications range from call admission control [? ?], to behavior-based robotics [? ? ?], to noise filtering [?], among others. To understand the breadth of topics that SDM encompasses, consider the following questions:

1. Why are most concepts orthogonal, unrelated to each other?
2. Why is there Miller's magic number, i.e., we can't hold too many things in mind at once?
3. Why do we at times instantly recall an experience; other times we can't recall anything at all; and still other times we get into this strange tip-of-the-tongue situation... the memory is clearly 'there'... but remains innacessible.
4. How does this recall process work? What is remembering?
5. Why do neurons die and we still remember most everything?
6. What do neurons actually do? What is their primary function?

While these implementations are extremely interesting, they do not afford the flexibility to experiment that software does:

1. The original 1989 hardware implementation developed in NASA by ?];
2. a 1995 LISP implementation for the Connection Machine by ?];
3. a 1992 APL implementation by ?];
4. a 2004 FPGA implementation by ?];
5. a 2005 C++ implementation by ?] from Lancaster University, in the 'CommonSense ToolKit' (CSTK) [?] for realtime sensor data includes SDM as one of its classification algorithms;
6. a 2015 'C Binary Vector Symbols (CBVS)': includes SDM implementation as a part of vector symbolic architecture developed by ?] from EISLAB at Luleå University of Technology [1];
7. a 2013 Java implementation 'Learning Intelligent Distribution Agent' (LIDA) developed by [? ? ?] Stan Franklin's group from the University of Memphis includes implementation. [2];

Let us analyze these. The Connection Machine is obsolete. The NASA implementation is hardware-based and obsolete. APL, while reasonably influential, is not a mainstream language in science.

The FPGA implementation by ?] has yielded a fast scan of hard locations at low energy costs, provided one has access to the proper hardware. Their

[1] The code is available at http://pendicular.net/cbvs.php

[2] http://ccrg.cs.memphis.edu/framework.html; see also http://ccrg.cs.memphis.edu/projects.html where they link to a github repository.

article claims a four-fold speedup over assembly language; but it does not deal with parallel processing details. For example, it is unclear whether there was more than a single thread running on the software implementation. Note that the framework presented here is also able to reconfigure field-programmable gate arrays, through the OpenCL heterogeneous computing platform ability to interface with Hardware Description Language and hence, reconfigure FPGAs [**?** **?**] for our tasks.

Then there is LIDA — a whole cognitive architecture based on Hofstadter's Fluid Concepts, Kanerva's SDM, and other ideas [**?** **?** **?** **?**]. It is developed in Java; which makes it difficult to connect to the lowest levels of hardware; to connect to GPUs or FPGAs, and to other languages — at least in comparison to the combination Python and OpenCL proposed here[3]. It has a non-standard license, strange to the open-source community, *the LIDA Framework Software NonExclusive, Non-Commercial Use License.* We have not found any parallelism in their code [**?**], which may make simulations slow or unfeasible. Moreover, potential contributors must sign an "Agreement Regarding Contributory Code for the LIDA Framework Software"... *'before Memphis can accept it'* [4].

The closest implementations to ours, in philosophy at least, is the one in 'the common sense toolkit'. It is executed in C++, with a normal open-source license, and hosted on an open-source code repository. It is, however, strikingly dissimilar to ours on the following aspects:

1. SDM is but a part of the system; the description of the system reads that cstk is 'A toolkit for processing and visualising sensor data in real time with support for use with embedded platforms.'
2. The whole SDM code is composed of 143 *lines* of C++ in the $cstk/cstk-devonly/sdm$ folder.
3. There is no work on making the system parallel.
4. There is strong coupling between location address and location data, which makes experimentation hard.

[3]Python is sometimes called a 'glue language'. That is, in my opinion, not the best metaphor. A glue connects two things leaving an inflexible structure. Python is perhaps best described as the interstate highway system of Programming; if something is out there, there is a way to reach it with Python. In the comparison with Java, for instance, take the $popcnt(xor(b_i, b_j))$ operation, executed billions of times in SDM. How easy is it to program that for a particular GPU or FPGA with Java?

[4]What they are attempting to do with this bureaucracy remains unclear, the history of computing has not been kind to those who favored centralization [**?**]. We certainly refrain from contributing given the legal uncertainties of non-standard licenses and dubious processes — even as we would like to link these libraries

5. There are no tests or examples to be found instantly.
6. Finally, the last commit to this repository seems to have been made in 2005?
7. There are no publishable or published scientific applications or experiments available to be reproduced at installation time.
8. there is no tutorial, installation instructions, performance benchmarks, framework validation or SDM Documentation.

Note that all these criticisms apply to the implementations in both the ‘common sense toolkit’ and the ‘C Binary Vector Symbols’ [**? ? ?**]. While these implementations have around 150 lines of C++; at last count, the *documentation of our implementation* had *over 100 pages* [**?**]: they have aimed at running code, and we aim at improving a community and industry standard.

There is obviously a demand for use of SDM; but each group has been tied to their own *ad-hoc* needs, and there has not been the emergence of a community centered on a tool. It is our belief that a tool such as standard open-source framework could bring orders of magnitude more researchers and attention if they were able to use the model, at zero cost, with an easy to use high-level language such as Python, in an intuitive platform such as Juypyter notebooks. Neuroscientists interested in long-term memory storage should not have to worry about high-bandwidth vector parallel computation. This new tool would provide a ready to use system in which experiments could be executed almost as soon as designed — and provide the needed replication of studies [**?**].

The main contribution of this work is a reference implementation which yields (i) orders of magnitude gains in performance, (ii) has several backends[5] and operations, (iii) is fully validated against the mathematical model, (iv) is cross-platform[6], and (v) is easily extensible to test new research ideas — and to let others replicate the studies.

Another issue is *extensibility*: Extensions of SDM have been used in many applications. For example, **?**] extended SDM to store sequences of vectors and trees efficiently. **?**] used a modified SDM in an autonomous robot. **?**] modified SDM to clean patterns from noisy inputs. **?**] extended SDM with genetic algorithms. **?**] extended SDM creating the Rotational Sparse Distributed Memory (RSDM), which models network motifs, dynamic flexibility, and hierarchical organization — reflecting results from the neuroscience literature.

[5]CPUs, GPUs, FPGAs

[6]Unix, Linux, MacOs, Windows, Amazon Web Services, etc.

Our reference implementation may, hopefully, accelerate research into the model's dynamics and make it easier for readers to replicate any previous results and easily understand the source-code of the model. Moreover, it is compatible with Jupyter notebook and researchers may share their notebooks possibly accelerating the advances in their fields [**?**].

Other contributions have also been introduced, which include (i) a noise filtering approach, (ii) a supervised classification algorithm, (iii) and a reinforcement learning algorithm, all of them using only the original SDM proposed by Kanerva, i.e., with no additional mechanisms, algorithms, data structures, etc. Although some of these applications have already been explored in previous work [**? ? ?**], all of them have adapted SDM to fit their problems, and none of them have used just the ideas introduced by Kanerva. We have presented different approaches with no adaptations whatsoever.

Finally, I have striven to provide a visual tour of the theory and application of SDM: whenever possible, detailed figures should tell the story — or at least do the heavy lifting. In this study, we will see an anomaly in one of Kanerva's predictions, which I believe is related to SDM capacity. We will see tests of a generalized reading operation proposed by Physics Professor Paulo Murilo (personal communication). We will see what happens when neurons — and all their information — is simply and suddenly lost. We will see whether information-theory can improve some of Kanerva's ideas. From (basic) noise filtering to learning to play tic-tac-toe, we will review the entirety of Dr. Pentti Kanerva's proposal.

2. Software description

Describeddddddd the software in as much as is necessary to establish a vocabulary needed to explain its impact.

2.1. Software Architecture

Give a short overview of the overall software architecture; provide a pictorial component overview or similar (if possible). If necessary provide implementation details.

2.2. Software Functionalities

Present the major functionalities of the software.

2.3. Sample code snippets analysis (optional)

3. Illustrative Examples

Provide at least one illustrative example to demonstrate the major functions.

Optional: you may include one explanatory video that will appear next to your article, in the right hand side panel. (Please upload any video as a single supplementary file with your article. Only one MP4 formatted, with 50MB maximum size, video is possible per article. Recommended video dimensions are 640 x 480 at a maximum of 30 frames/second. Prior to submission please test and validate your .mp4 file at $http : //elsevier - apps.sciverse.com/GadgetVideoPodcastPlayerWeb/verification$. This tool will display your video exactly in the same way as it will appear on ScienceDirect.).

4. Impact

This is the main section of the article and the reviewers weight the description here appropriately

Indicate in what way new research questions can be pursued as a result of the software (if any).

Indicate in what way, and to what extent, the pursuit of existing research questions is improved (if so).

Indicate in what way the software has changed the daily practice of its users (if so).

Indicate how widespread the use of the software is within and outside the intended user group.

Indicate in what way the software is used in commercial settings and/or how it led to the creation of spin-off companies (if so).

5. Conclusions

Set out the conclusion of this original software publication.

6. Conflict of Interest

Please select the appropriate text:

Potential conflict of interest exists: We wish to draw the attention of the Editor to the following facts, which may be considered as potential conflicts of interest, and to significant financial contributions to this work. The nature of potential conflict of interest is described below: [Describe conflict of interest]

No conflict of interest exists: We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgements

Optionally thank people and institutes you need to acknowledge.

elsarticle-num

Current executable software version

Ancillary data table required for sub version of the executable software: (x.1, x.2 etc.) kindly replace examples in right column with the correct information about your executables, and leave the left column as it is.

Nr.	**(Executable) software metadata description**	**Please fill in this column**
S1	Current software version	For example 1.1, 2.4 etc.
S2	Permanent link to executables of this version	For example: $https://github.com/combogenomics/DuctApe/releases/tag/DuctApe-0.16.4$
S3	Legal Software License	List one of the approved licenses
S4	Computing platforms/Operating Systems	For example Android, BSD, iOS, Linux, OS X, Microsoft Windows, Unix-like , IBM z/OS, distributed/web based etc.
S5	Installation requirements & dependencies	
S6	If available, link to user manual - if formally published include a reference to the publication in the reference list	For example: $http://mozart.github.io/documentation/$
S7	Support email for questions	

Table 2: Software metadata (optional)