The Journal of Open Source Software

AdaptiveResonance.jl: A Julia Implementation of Adaptive Resonance Theory (ART) Algorithms

Sasha Petrenko[1] **and Donald C. Wunsch II**[1]

1 Missouri University of Science and Technology

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Adi Singh

Reviewers:

- @aaronpeikert
- @hayesall

Submitted: 28 May 2021
Published: unpublished

Summary

AdaptiveResonance.jl is a Julia package for machine learning with adaptive resonance theory (ART) algorithms, written in the numerical computing language Julia. ART is a neurocognitive theory of how competitive cellular networks can learn distributed patterns without supervision through recurrent field connections, eliciting the mechanisms of perception, expectation, and recognition (Grossberg, 1980, 2013). Engineering algorithms based upon ART are principally in the class of competitive, incremental, single-layer, neurogenesis clustering algorithms, but they have been adapted for multimodal learning in diverse applications.

Statement of Need

There exist many variations of algorithms built upon ART (Brito da Silva et al., 2019). Each variation is related by utilizing recurrent connections of fields, driven by learning through match and mismatch of distributed patterns, and though they all differ in the details of their implementations, their algorithmic and programmatic requirements are often very similar. Despite the relevance and successes of this class of algorithms in the literature, there does not exist to date a unified repository of their implementations in Julia. Therefore, the purpose of this package is to create a unified framework and repository of ART algorithms in Julia.

Target Audience

This package is principally intended as a resource for researchers in machine learning and adaptive resonance theory for testing and developing new ART algorithms. However, implementing these algorithms in the Julia language brings all of the benefits of the Julia itself, such as the speed of being implemented in a low-level language such as C while having the transparency of a high-level language such as MATLAB. Being implemented in Julia allows the package to be understood and expanded upon by research scientists while also being able to be used in resource-demanding production environments.

Comparison to Existing Implementations

There exist a myriad of open implementations of ART algorithms that are the result of reproducibility efforts in the ART literature. The Boston University Department of Cognitive and Neural Systems (CNS) Technology Laboratory software repository contains many one of the largest collections of algorithms and utilities related to ART, principally implemented in the MATLAB and C++ programming languages, from demonstrations of the learning laws of ART to implementations of ART and ARTMAP modules (Cognitive & Lab, 2009). However, this repository serves as a codebase for the reproducibility of the software associated CNS papers rather than as a single unified framework for ART implementations.

The Missouri University of Science and Technology Applied Computational Intelligence Laboratory (ACIL) hosts a myriad of individual ART algorithm implementations on its public GitHub group repository page (Science & Laboratory, 2022), chiefly implemented in the MATLAB and Python programming languages. Though these ART algorithms are designed for open use, so too do they principally serve the reproducibility of their associated ACIL papers.

The ACIL group GitHub page additionally contains the NuART-Py library, which organizes a suite of clustering and biclustering ART algorithms as a distributable package in the Python language (Elnabarawy, 2019). A similar package exists in the Java programming language in a separate repository containing only fundamental ART algorithms (Chen, 2018), and a new package in the R statistical programming language has only begun development at the time of this writing (Steinmeister & Wunsch, 2021).

Though each of these ART software projects (and the very many and disparate implementations of individual algorithms in the literature) combined may implement the majority of ART algorithms relevant to modern research and engineering, together they lack cohesion in programming language and usage. When considering ease of use and barrier to entry, many of these projects may be difficult to utilize for those less versed in the ART literature who might still significantly benefit from their use and understanding.

Lastly, many ART implementations exist in the MATLAB programming language due to its popularity amongst the research scientists that have been the theory's primary clientele, which is at the detriment to those without private MATLAB licenses in research and industry. The Julia programming language is selected for this open-source ART package implementation due to its syntactic ease of use and speed of development without comprimising computational efficiency due to the language's just-in-time compilation.

Adaptive Resonance Theory

ART is originally a theory of how competitive fields of neurons interact to form stable representations without supervision, and ART algorithms draw from this theory as a biological inspiration for their design. It is not strictly necessary to have an understanding of the theory to understand the use of the algorithms, but they share a common nomenclature that makes knowledge of the former useful for the latter.

Theory

Adaptive resonance theory is a collection of neurological study from the neuron level to the network level (Hestenes, 1987). ART begins with a set of neural field differential equations and procedurally tackles problems such as why sigmoidal neural activations are used, the conditions of stability for competitive neural networks (Cohen & Grossberg, 1983), how the mammalian visual system works (Grossberg & Huang, 2009), and the hard problem of consciousness linking resonant states to conscious experiences (Grossberg, 2017, 2021).

Algorithms

ART algorithms are generally characterized in behavior by the following:

1. They are inherently *unsupervised* learning algorithms at their core, but they have been adapted to supervised and reinforcement learning paradigms with frameworks such as ARTMAP (Carpenter et al., 1991, 1992) and FALCON (Tan et al., 2019), respectively.
2. They are *incremental* learning algorithms, adjusting their weights or creating new ones at every sample presentation.
3. They are *neurogenesis* neural networks, representing their learning by the modification of existing prototype weights or instantiating new ones entirely.

4. They belong to the class of *competitive* neural networks, which compute their outputs with more complex dynamics than feedforward activation.

Because of the breadth of the original theory and variety of possible applications, ART-based algorithms are diverse in their nomenclature and implementation details. Nevertheless, they are generally structured as follows:

1. ART models typically have two layers/fields denoted F1 and F2.
2. The F1 field is the feature representation field. Most often, it is simply the input feature sample itself (after some requisite feature preprocessing, depending on the model).
3. The F2 field is the category representation field. With some exceptions, each node in the F2 field represents its own category. This is most easily interpreted as a weight vector representing a prototype for a class or centroid of a cluster.
4. An activation function is used to find the order of categories "most activated" for a given sample in F1.
5. In order of highest activation, a match function is used to compute the agreement between the sample and the categories.
6. If the match function for a category evaluates to a value above a threshold known as the vigilance parameter (ρ), the weights of that category may be updated according to a learning rule.
7. If there is complete mismatch across all categories, then a new category is created according to an instantiation rule.

Implementation

In creating a unified framework for ART modules in Julia, the development of this package faces the challenges of organizing and categorizing the designs and objectives of many different ART algorithms, which necessitates the formalization of the distinctions between training versus inference, batch versus incremental learning, supervised versus unsupervised learning modes, and match versus mismatch.

Training vs. Inference

All modules in the package have states that are tracked and updated during learning, and so they have their own module constructors with options that are themselves also stateful. The two most simple operations available on these ART modules are `train!` and `classify` for training and inference, respectively. This package utilizes the Julia convention of appending an exclamation point to the end of functions that modify their parameters. During training, ART modules are allowed to mutate their internal parameters, wheras during inference, they report their categorization of the data without allowing parameters to change.

Batch vs. Incremental Learning

ART modules are generally incremental learning algorithms, meaning that they update their parameters or conduct inference on one data sample at a time rather than in large batches. If many samples are presented at once, batch learning is still done by incrementally learning upon all provided samples. This package, however, accommodates batch learning without the need to implement multiple methods by utilizing Julia's multiple dispatch system, correctly inferring which function to use by the dimensionality of the input samples. As done in many other machine learning methods, a single sample is denoted by a vector of features, while a set of samples is a matrix of many features.

Supervised vs. Unsupervised Learning

Though ART modules are generally multimodal machine learning algorithms in that they may be designed to learn with or without prescribed labels (i.e., supervised or unsupervised), algorithms in the ARTMAP family are expressly supervised. To accommodate this distinction, this package organizes algorithms that are by default unsupervised but that can accept optional labels as ART modules while distinguishing explicitly supervised modules as ARTMAP modules. This distinction is enforced programmatically by making labels an optional argument in `train!` declarations upon ART modules and a required positional argument in `train!` declarations upon ARTMAP modules.

Match vs. Mismatch

A match function is used in ART and ARTMAP modules to evaluate if a given sample sufficiently aligns with a particular category for the weight to be mutated during learning or for the category to be reported during inference. Mismatch occurs when this match function evaluates to below the vigilance threshold for all internal categories. Complete mismatch during learning triggers the creation of a new category. Mismatch during inference, on the other hand, results in the module reporting an unknown category signal, which is the default behavior for all modules by precedent in the literature.

It is sometimes desirable to report a next-best category in the case of complete mismatch, which is referred to as the best matching unit. All modules are equipped with an option to report the best matching unit in the case of mismatch.

Acknowledgements

This package is developed and maintained with sponsorship by the Applied Computational Intelligence Laboratory (ACIL) of the Missouri University of Science and Technology. This project is supported by grants from the Army Research Labs Night Vision Electronic Sensors Directorate (NVESD), the DARPA Lifelong Learning Machines (L2M) program, Teledyne Technologies, and the National Science Foundation. The material, findings, and conclusions here do not necessarily reflect the views of these entities.

References

Brito da Silva, L. E., Elnabarawy, I., & Wunsch, D. C. (2019). A Survey of Adaptive Resonance Theory Neural Network Models for Engineering Applications. *Neural Networks*, *120*(xxxx), 167–203. https://doi.org/10.1016/j.neunet.2019.09.012

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., & Rosen, D. B. (1992). Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps. *IEEE Transactions on Neural Networks*, *3*(5), 698–713. https://doi.org/10.1109/72.159059

Carpenter, G. A., Grossberg, S., & Reynolds, J. H. (1991). ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *IEEE Conference on Neural Networks for Ocean Engineering*, 341–342. https://doi.org/10.1016/0893-6080(91)90012-T

Chen, X. (2018). *Java-adaptive-resonance-theory*. https://github.com/chen0040/java-adaptive-resonance-theory

Cognitive, B. U. of, & Lab, N. S. T. (2009). *Cognitive and neural systems technology lab software repository*. http://techlab.bu.edu/resources/software/C51/index.html

Cohen, M. A., & Grossberg, S. (1983). Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks. *IEEE Transactions on Systems, Man and Cybernetics*, *SMC-13*(5), 815–826. https://doi.org/10.1109/TSMC.1983.6313075

Elnabarawy, I. (2019). *NuART-py: A python library of adaptive theory neural networks*. https://github.com/ACIL-Group/NuART-Py

Grossberg, S. (1980). How Does a Brain Build a Cognitive Code ? *Psychological Review*, *87*(1), 1–51. https://doi.org/10.1037/0033-295X.87.1.1

Grossberg, S. (2013). Adaptive Resonance Theory: How a brain learns to consciously attend, learn, and recognize a changing world. *Neural Networks*, *37*, 1–47. https://doi.org/10.1016/j.neunet.2012.09.017

Grossberg, S. (2017). Towards solving the hard problem of consciousness: The varieties of brain resonances and the conscious experiences that they support. *Neural Networks*, *87*, 38–95. https://doi.org/10.1016/j.neunet.2016.11.003

Grossberg, S. (2021). *Conscious mind, resonant brain: How each brain makes a mind*. OUP Premium Oxford University Press. ISBN: 978-0190070557

Grossberg, S., & Huang, T. R. (2009). ARTSCENE: A neural system for natural scene classification. *Journal of Vision*, *9*(4), 1–19. https://doi.org/10.1167/9.4.6

Hestenes, D. (1987). How the Brain Works: The Next Great Scientific Revolution. *Maximum-Entropy and Bayesian Spectral Analysis and Estimation Problems*, 173–205. https://doi.org/10.1007/978-94-009-3961-5_11

Science, M. U. of, & Laboratory, T. A. C. I. (2022). *Applied computational intelligence laboratory GitHub software repository*. https://github.com/ACIL-Group

Steinmeister, L., & Wunsch, D. C. (2021). *Scholars ' Mine FuzzyART : An R Package for ART-based Clustering FuzzyART : An R Package for ART-based Clustering . May.*

Tan, A.-H. H., Subagdja, B., Wang, D., & Meng, L. (2019). Self-organizing neural networks for universal learning and multimodal memory encoding. *Neural Networks*, *120*(xxxx), 58–73. https://doi.org/10.1016/j.neunet.2019.08.020