Algebraic Machine Learning: a new program for Symbolic AI

This white paper explains, in simple terms, the main ideas behind Algebraic Machine Learning (AML) and provides insights on how and why it works. AML is a symbolic method that is good for reasoning and has the advantage of being able to learn from data. AML can use continuous input and output, can deal with uncertainty and can combine learning from both, data and formulas. These unique properties show that the main limitations of symbolic methods can be overcome and open a path to a more transparent, trustworthy and understandable AI.

Symbolic AI

Symbolic artificial intelligence refers to a variety of methods that directly operate on symbolic representations of the world. Symbolic AI has been successful in a variety of domains, including planning, scheduling, natural language processing and game playing. Symbolic AI methods, such as expert systems, remain the preferred choice in critical real-world applications where human control and transparency are essential and the consequences of errors significant.

Progress in symbolic AI has been hindered by its inability to effectively learn from data and its reliance on fixed rules, which can make it less effective at handling uncertainty and new situations. In recent years, these limitations have made it difficult for symbolic AI to develop as fast as the more adaptable and data-driven statistical learning methods such as neural networks.

While neural networks generate internal representations of data that are not easily interpretable by humans, symbolic AI uses user-defined symbols to represent concepts and relationships. This makes symbolic AI methods more easily understood and interpreted by humans, making them well-suited to tasks that require clear and interpretable models. On the other hand, the internal representations of neural networks seem to be essential for learning and adaptability.

Algebraic Machine Learning

Algebraic Machine Learning (AML) [1] is a new mathematical approach that combines user-defined symbols with self-generated symbols. Generated symbols provide internal representations that permit AML to learn from the data and adapt to the world like neural networks do.

AML is a purely symbolic approach and neither uses neurons nor is a neuro-symbolic method. Algebraic Machine Learning does not use parameters and it does not rely on fitting, regression, backtracking, constraint satisfiability, logical rules, production rules or error minimization. Instead, AML relies on various notions of abstract algebra such as semantic embeddings, algebraic freedom and subdirect irreducibility. Each of these will be explained in this text. We will start by providing a concise description of the whole AML approach and then we will explain it in more detail.

At the heart of AML is the idea that once input data is embedded in an idempotent algebra, an algebraic model is produced. It turns out that the subdirectly irreducible components of the algebraic model naturally capture the main characteristics of the data. Generalization then occurs by simply selecting a suitable subset of said subdirectly irreducible components. Each of the many possible subsets represents a generalizing model.

A detailed explanation

By an algebra we refer to an abstract algebra[2], discrete or continuous. An algebra is a set of objects and a set of operations, where each operation transforms one or many objects into other objects of the set. Numbers and their internal operations, such as addition, form an algebra. However, we can make almost everything into an algebra! For example, the ingredients in our kitchen could be our set and the various things we can do to them the opera-

Fernando Martin Maroto[1,2], Nabil Abderrahaman[2] and Gonzalo García de Polavieja[1,2]
[1]*Champalimaud Research, Champalimaud Foundation, Lisbon, Portugal,* [2]*Algebraic AI, Madrid, Spain*

tions. One operation, baking, transforms dough into bread.

For AML, the algebra should have at least one idempotent[2] operation, i.e., an operation that satisfies the axiom: a ⊙ a = a. The simplest algebra we can use for AML is a semilattice [2] that only has one idempotent commutative and associative operation.

With the word "embedding" we refer to a particular form of encoding known as "semantic embedding" [2,3], which is a representation of one mathematical structure into another. Theoretical computer scientists and mathematical logicians have used semantic embeddings to study undecidability [2], for example.

An embedding in AML [4, 5] has the form of a set of rules (formal expressions) provided by the user. The formal expressions represent both, the data and the things we know about the data, such as constraints, laws, symmetries or even the goal of the task at hand. A semantic embedding in the context of AML should not be mistaken with the term of the same name used in the field of neural networks.

The word "model" is used here with the same meaning as in model theory[3]. A model is an instance (an example) of an algebra that satisfies the rules of a semantic embedding. The word "model" is also used in the field of neural networks but this time the meaning is similar. A model, in the context of AML, is an instance of an algebra, for example a particular semilattice, in the same manner that a neural network model is a particular configuration of a neural network weights and architecture.

AML relies on a mechanism to go from the embedding formulas to the algebraic model. For semilattices, this mechanism can be based on an operation, full crossing[1, 4], and its sparse version[1], the sparse crossing, which is a stochastic method. With this mechanism we could say that AML synthetizes its own parameters, that are created automatically when needed and are born with their definitive value. We refer to such synthetized parameters as atoms [1, 4].

Atoms represent the model in a very particular manner; atoms map one-to-one to the subdirectly irreducible components of the model and that is key as irreducible components break up the data in its fundamental constituents. What are the irreducible components?

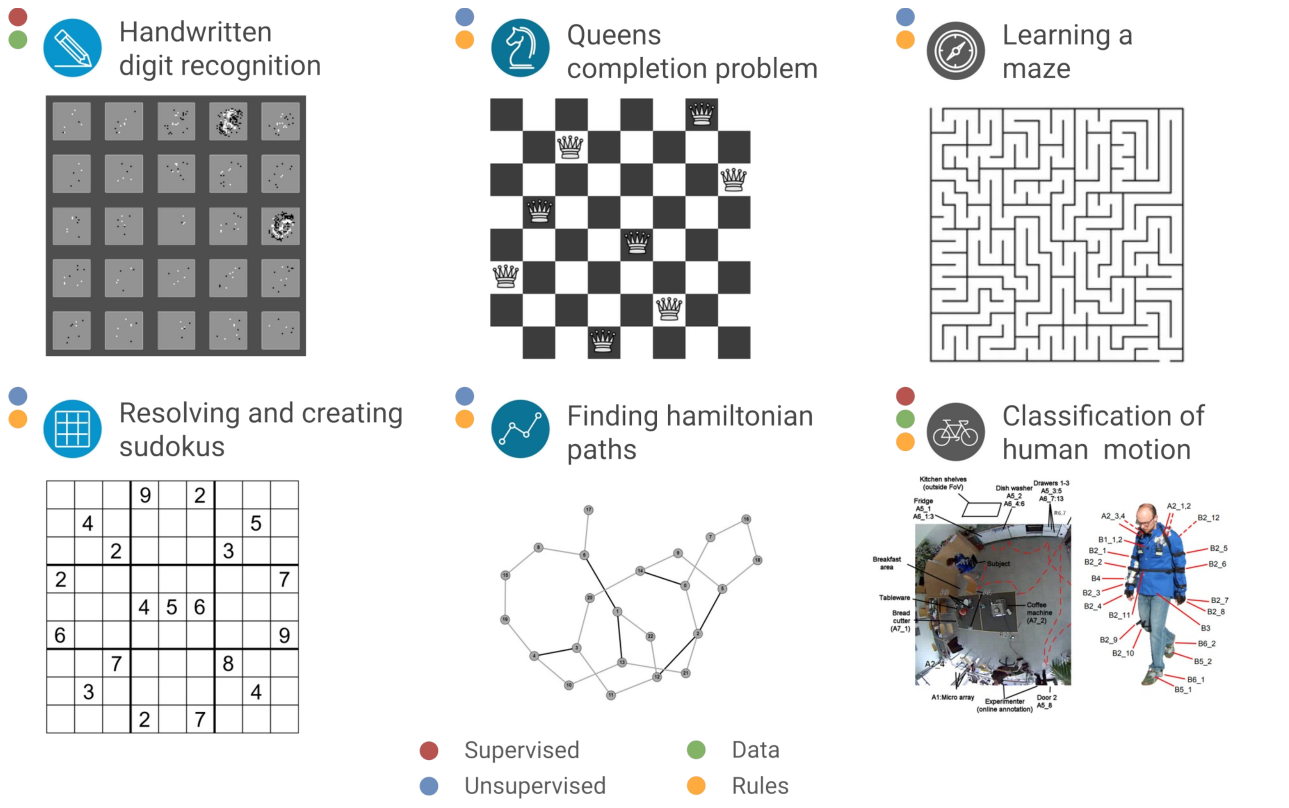
Handwritten digit recognition
Queens completion problem
Learning a maze
Resolving and creating sudokus
Finding hamiltonian paths
Classification of human motion
Supervised
Unsupervised
Data
Rules

Figure 1: Some example applications of AML. From left to right, top to bottom. 1 - MNIST digit classification [1]. 2 - N-Queen completion problem [1,5]. 3 - Maze pathfinding. 4 – Sudoku [5]. 5 - Hamiltonian paths and Hamiltonian cycles [5]. 6 - OPPORTUNITY dataset[6] for human movement recognition [7].

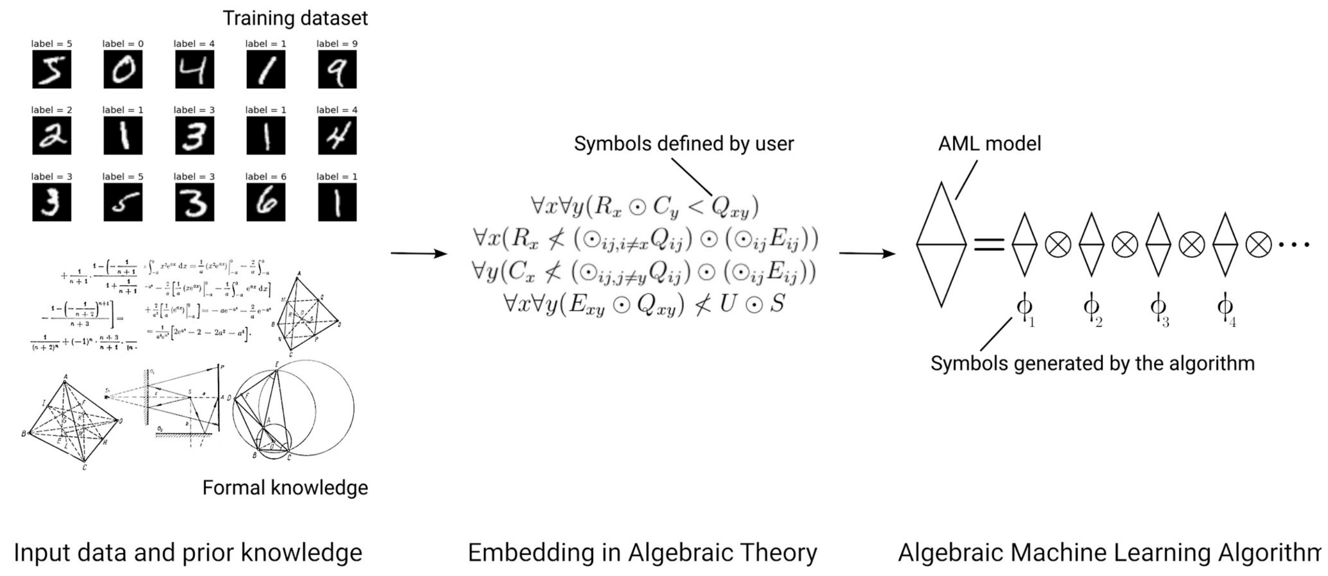
Training dataset
label = 5
label = 0
label = 4
label = 1
label = 9
label = 2
label = 1
label = 3
label = 1
label = 4
label = 3
label = 5
label = 3
label = 6
label = 1
Formal knowledge
Symbols defined by user
$\forall x \forall y (R_x \odot C_y < Q_{xy})$
$\forall x (R_x \not< (\odot_{ij, i \neq x} Q_{ij}) \odot (\odot_{ij} E_{ij}))$
$\forall y (C_x \not< (\odot_{ij, j \neq y} Q_{ij}) \odot (\odot_{ij} E_{ij}))$
$\forall x \forall y (E_{xy} \odot Q_{xy}) \not< U \odot S$
AML model
ϕ_1 ϕ_2 ϕ_3 ϕ_4
Symbols generated by the algorithm
Input data and prior knowledge
Embedding in Algebraic Theory
Algebraic Machine Learning Algorithm

Figure 2: The training dataset and formal knowledge about the problem are expressed as logic formulas. These equations use symbols defined by the user. Using sparse crossing we can embed these equations into an algebraic model. The algorithm itself generates the necessary symbols.

Subdirect irreducibility and the mechanism of AML

Imagine an algebraic model that describes a piano keyboard. Each element of the algebra is a note, a key of the piano. The algebra may have various operations compatible with playing piano, for example, one operation steps up a note one semitone, i.e., it maps one piano key to the adjacent key to the right. Our algebra can be represented as a Cartesian product of two algebraic models: one model with the twelve notes of one octave and another model with the seven octaves of the piano. Each component captures independent aspects of the model and provides an optimal decomposition of the diatonic scale.

An intelligent system listening to piano should ideally discover this decomposition. The same system reading sport news should discover that players in a world tournament are better understood as elements in a Cartesian product of team and role within the team.

However, in most cases things are more complicated as concepts cannot be decomposed so cleanly. The same occurs to algebras. In the general case, an algebraic model cannot be represented as a product but it is always possible to represent it as a subdirect product. A subdirect product is similar to a product but somehow weaker. It is the best decomposition we can do when a product is not possible. The fact that any algebraic model can be represented as a subdirect product of irreducible models is known as the Birkhoff's subdirect representation [2] theorem.

To further illustrate the concept of a subdirect product, consider now the group of the Rubik's Cube configurations. The operations of this algebra (a group) are the possible ways to rotate the faces of the cube. This group cannot be represented as a direct product but it can be represented as a subdirect product of the corner orientation, corner permutation, edge orientation and edge permutation groups. These four component groups describe how the face rotations change the cube configuration. The group of configurations is a subdirect product because it is a subgroup of the direct product of these component groups. It is not equal to the cartesian product of the component groups because not all edge permutations are compatible with each other, neither are all corner orientations compatible with each other. Only some elements of the cartesian product of the components are valid configurations of the cube. These valid elements form a subalgebra which, in the case of the cube, is a subgroup that actually permits resolving the cube.

The Rubik's Cube exemplifies well what happens in general. Our algebraic model cannot be represented as a product but it can be represented as a subalgebra of a direct product of irreducible algebraic models (components). These irreducible components capture essential properties of the data, and they are "irreducible" because they cannot be further decomposed into smaller components.

There are two things to be determined in this context: the irreducible components and the location of the subalgebra within the product of the irreducible components, i.e., the identification of the subset of valid elements within the direct product. Each atom in AML relates simultaneously to both: to one irreducible component and to the location of the model as a subalgebra.

AML understands the data by breaking it up into its irreducible components and this is possible when the data is embedded in an algebra. This mechanism provides the best possible decomposition for the data and it can be proven that if the data is generated using some hidden rule this mechanism will eventually discover the rule with a sufficiently large amount of data.

Understanding is not just the capability to predict, it is the ability to break things up into its fundamental constituents which results in the capability to predict. AML proposes to use subdirect products to achieve this goal.

Generalization

The set of all the atoms contain the same information as the input data. In fact, the set of all the atoms spawn the freest model of the embedding formulas. The freest model of the data is the model that assumes nothing more than the data itself and it is equivalent to the data.

It is usually the case that the freest model requires a very large number of atoms to be described. The number of atoms needed often exceeds the size of the input data by orders of magnitude. However, it turns out that, at least for semilattices and other closely related algebras, it is always possible (and easy) to find suitable subsets of atoms, the size of which is (a lot) smaller than the size of the input data.

The suitable subsets of atoms are those that satisfy the constraints of the embedding. Each suitable subset contains less information than the whole model and yet suffices to satisfy all the constraints, thereby having generalizing capabilities. We just need to find one among the immense number of suitable subsets to find a generalizing model.

A consequence of having many more atoms than input data is that small subsets of atoms cannot overfit as they contain a lot less information than the input data.

In AML, error is never computed. AML does not seek to explicitly reduce error rate. Reduction of error rate occurs naturally as a result of subdirect irreducibility, maximization of freedom and minimization of size.

	Symbolic AI	Neural networks	AML
Learn from data	🟠	🟢	🟢
Learn from formulas	🟢	🟠	🟢
Transparency	🟢	🟠	🟢
Generate its own representation of the data	🔴	🟢	🟢
Symbols or parameters are modified over training	🟠	🟢	🟢
Parameter or symbol count changes during training	🟠	🔴	🟢
Models can be combined	🔴	🟠	🟢

Table 1: Comparison between symbolic AI, neural networks, and algebraic machine learning

Unique capabilities of AML

AML has features that could be key to building safer and less centralized AI applications:

User-defined symbols – Algebraic Machine Learning (AML) combines the advantages of using user-defined symbols with the capability to generate internal representations and learn from data.

Set goals and constraints – User-defined symbols can be used to set goals and constraints using formulas which provide a more controllable behavior and better transparency.

Low sensitivity to data distribution – AML is not a statistical method and is less sensitive to the particular frequency distribution of the input data.

Perfect memory – AML systems do not forget one thing when taught another.

No overfit – AML models do not overfit.

Reduced number of hyperparameters – There is no need to fit parameters and the number of hyperparameters is very small, often reduced to little more than batch size.

Multi-domain – AML can resolve problems on very different domains, from supervised pattern recognition to unsupervised learning of games, to resolving sudokus from the rules of the game or even to find Hamiltonian paths (without using backtracking or even search).

Continuous and discrete symbols – AML can use continuous and discrete input and output.

Single algorithm – AML relies on a single, problem-independent algorithm, with no need for custom architectures.

Models can be combined – The subsets of atoms that satisfy the embedding have a "linearity property"[5]: the union of two or more of said subsets of atoms also spawn a model that satisfies the embedding formulas.

Scalable and parallelizable – As a consequence of the linearity, AML models can be computed in parallel, to the extent that models can be trained separately and asynchronously, only to be combined at a later stage. This is due to the fact that atoms can be calculated independently. This opens the door to the possibility of using scalable distributed systems with inexpensive hardware and network setups.

New mathematical approach – AML is very amenable to mathematical analysis and is a mostly unexplored field of research in which ideas from abstract algebra can potentially be brought to the field of symbolic AI. Ideas from adjacent fields such as Galois theory, topology or model theory could potentially be applied here. Furthermore, crossbreeding of ideas with the field of deep learning could be fructiferous to both.

ALMA

The ALMA [8] project relies on the use of Algebraic Machine Learning and, so far, has applied AML to a range of tasks, including robotics, the recognition of human motion from accelerometer data or the identification of gestures in on-screen keyboards. The ALMA project is a research collaboration that involves multiple labs across Europe. The project is part of an effort by the European Commission to explore *radically new* ideas in the field of AI.

Research in AML is led by the Champalimaud Foundation and Algebraic AI.

Bibliography

[1] Martin-Maroto, F. & de Polavieja, G. (2018). Algebraic Machine Learning. arxiv.org/abs/1803.05252.

[2] Burris, S. & Sankappanavar, H. P. (1981) A course in universal algebra. Springer-Verlag.

[3] Hodges, W. (1997). A shorter model theory. Cambridge University Press.

[4] Martin-Maroto, F. & de Polavieja, G. (2021). Finite Atomized Semilattices. arxiv.org/abs/2102.08050.

[5] Martin-Maroto, F. & de Polavieja, G. (2022). Semantic Embeddings in semilattices. arxiv.org/abs/2205.12618.

[6] Roggen, D., et all. (2010). Collecting complex activity datasets in highly rich networked sensor environments.

[7] Tschoepe, M., Fortes, V. & Lukowicz, P. (2022). alma-ai.eu/index.php/news/12-first-steps-of-aml-for-human-activity-recognition

[8] ALMA project. alma-ai.eu