Learning Gradual Argumentation Frameworks using Meta-heuristics

Nico Potyka[1], Mohamad Bazo[2], Jonathan Spieler[2] and Steffen Staab[2,3]

[1]Imperial College London, UK
[2]University of Stuttgart, Germany
[3]University of Southampton, UK

Abstract

Gradual argumentation frameworks represent arguments and their relationships in a weighted graph. Their mechanics are closely related to neural networks, which are considered as black-box models due to their dense structure with millions of neurons. Recent work tried making them human-understandable by trying to learn parameters that can be well approximated by decision trees. However, the tree remains just an approximation, which leaves the question how faithful it really captures the actual mechanics of the neural network. To circumvent the problem, structure learning ideas can be tailored to the discrete structure of gradual argumentation frameworks to learn sparse neural networks that can be directly interpreted as gradual argumentation frameworks. We discuss the learning problem, sketch a genetic and a particle swarm optimization algorithm to solve the problem and show first results on data sets from the UCI machine learning repository.

Keywords

Abstract Argumentation, Gradual Argumentation, Interpretable Machine Learning, Explainable AI

1. Introduction

As black-box neural networks are increasingly applied in intelligent systems, questions about their fairness, reliability and safety become louder. Various proposals have been made to make their mechanics more transparent. This includes model-agnostic approaches [1, 2] as well as approaches tailored to the structure of neural networks [3, 4]. However, it remains somewhat unclear how faithful these explanation approaches capture the actual mechanics of the network and how robust and reliable the explanations methods themselves are [5]. An interesting idea to facilitate faithful explanations is to regularize the training procedure of neural networks such that it can be well approximated by interpretable machine learning models [6]. It has been demonstrated that this can be done for decision trees without significant performance loss of the network [6]. However, as the decision tree remains an approximation of the network, there remain some doubts about its faithfulness. Furthermore, decision trees themselves can quickly become complex and difficult to interpret due to a large number of branches.

As it turns out, there is a family of argumentation frameworks that can faithfully capture the mechanics of multilayer perceptrons (MLPs). This the family of gradual abstract argumentation

1st International Workshop on Argumentation & Machine Learning
n.potyka@imperial.ac.uk (N. Potyka); wahedbazo@outlook.com (M. Bazo); st169996@stud.uni-stuttgart.de (J. Spieler); steffen.staab@ipvs.uni-stuttgart.de (S. Staab)

frameworks (GAFs) that has been studied extensively in recent years [7, 8, 9, 10, 11, 12, 13]. In fact, every MLP corresponds to a GAF under a particular semantics, and conversely, every acyclic GAF under this semantics corresponds to a MLP [14]. Therefore, a neural network can obviously be represented faithfully by a GAF. However, since a gradual argumentation framework with millions of arguments is not easier to interpret than a MLP with millions of arguments, this connection is not immediately helpful. In order to use GAFs for interpretable machine learning, we have to learn a sparse network structure. We explicitly highlight two questions that have been raised before about this idea.

1. How does learning GAFs differ from learning neural networks? The short answer is that learning GAFs is a special case of learning neural networks. However, this does not mean that we could simply apply existing algorithms. This is because we have to take account of the special structure that characterizes the special case. We explain the main differences in more detail in the next three paragraphs. In short, when learning GAFs, we can apply standard algorithms for learning the parameters of a fixed network structure, but need specialized algorithms for structure learning (exploring the space of GAF structures).

Let us first note that the mainstream research in learning neural networks does not consider structure learning at all because of its computational complexity. The major architectures are based on a fixed and often densely connected graphical structure [15]. The user can typically define hyperparameters like the number of nodes per layer, but the nodes are always connected in a uniform way (e.g., fully connected dense layers or convolutional layers defined by kernels).

Learning sparse neural networks has become a more active area in recent years, but existing work does not focus on learning an interpretable network, but on decreasing the risk for overfitting, the memory and runtime complexity and the associated power consumption. Ideas include regularization to encourage neurons with weight 0 that can be deleted [16], pruning of edges [17], compression [18] and low rank approximation [19]. The idea is usually to make the network as small as possible (secondary goal), while keeping the performance high (primary goal). For us, the priorities are reversed: we want to achieve good learning performance (secondary goal) while keeping the network structure comprehensible (primary goal). We therefore focus on the structure learning problem, where we try to find the best performing structure in a space of interpretable network structures.

The argumentation view results in another model bias that differs from learning neural networks. The input for neural networks is typically numerical and discrete features have to be vectorized by one-hot-encoding or embedding functions. For us, the picture is again reversed. Our natural inputs are discrete arguments. While discrete features can naturally be represented by arguments, numerical features have to be discretized as we explain in more detail later.

2. Are the learnt GAFs really argumentation frameworks?

As GAFs have been studied extensively in the community, we assume that the reader accepts that they are argumentation frameworks. However, as we focus on learning acyclic GAF structures, one may criticize that acyclic graphical structures are not particularly interesting from an argumentation perspective. While this is true from a reasoning perspective, our position is that there is no need to consider more complicated structures if the acyclic structure already solves the problem. In particular, this simplicity should not be seen as a bug, but as a feature because it allows evaluating the argumentation graph in linear time, which is highly desirable when applying the learnt GAFs. Let us note that many applications of argumentation

like the analysis of online discussions or online reviews naturally result in acyclic graphs [20, 21, 22]. Artificially introducing complexity will make the solution less comprehensible and computationally more demanding. We acknowledge that more structure may be necessary to solve more complicated problems, but we demonstrate that the structure that we consider here already suffices to solve many interesting problems.

Overview: We explain the necessary basics of gradual argumentation frameworks in the next section. In Section 3, we discuss the learning problem in more detail and sketch a genetic and a particle swarm optimization algorithm for structure learning. Section 4 shows first experimental results and some examples of learnt GAFs.

2. Background

We consider gradual argumentation frameworks (GAFs) represented as graphs. Nodes represent arguments and edges represent attack and support relationships. Here, an argument is just an abstract entity that can be accepted to a certain degree based on the state of its attackers and supporters. Every argument is associated with a base score that can be seen as its apriori strength when ignoring its attackers and supporters. We consider edge-weighted GAFs similar to [12].

Definition 1 (Gradual Argumentation Framework (GAF))**.** A GAF is a tuple $(\mathcal{A}, E, \beta, w)$ that consists of

- a set of arguments $\mathcal{A}$ and a set of edges $E \subseteq \mathcal{A} \times \mathcal{A}$ between the arguments,
- a function $\beta : \mathcal{A} \to [0, 1]$ that assigns a *base score* to every argument and
- a function $w : E \to [-1, 1]$ that assigns a weight to every edge.

Edges with negative weights are called *attack* and edges with positive weights are called *support* edges and denoted by Att and Sup, respectively.

Figure 1 shows on the left the graphical structure of a GAF that formalizes a simple decision problem. We assume that we want to decide if we should buy or sell stocks of a company and that we consider three arguments put forward by different experts. Attack relationships are denoted by solid and support relationships by dashed edges. We can define the semantics of GAFs by interpretations that assign a strength value to every argument.

Definition 2 (GAF interpretation)**.** Let $G = (\mathcal{A}, E, \beta, w)$ be a GAF. An interpretation of G is a function $\sigma : \mathcal{A} \to [0, 1] \cup \{\bot\}$ and $\sigma(a)$ is called the strength of a for all $a \in \mathcal{A}$. If $\sigma(a) = \bot$ for some $a \in \mathcal{A}$, σ is called *partial*. Otherwise, it is called *fully defined*.

Interpretations are often defined by an iterative procedure that initializes the strength values of arguments with their base scores and repeatedly updates the values based on the strength of their attackers and supporters. Interpretations are partial when the procedure fails to converge, which can happen in cyclic graphs [12]. However, we will only consider acyclic graphs here and all interpretations are guaranteed to be fully defined [13]. The strength update of an argument can often be divided into two main steps [12]: first, an *aggregation function* α aggregates the strength values of attackers and supporters. Then, an *influence function* ι adapts the base

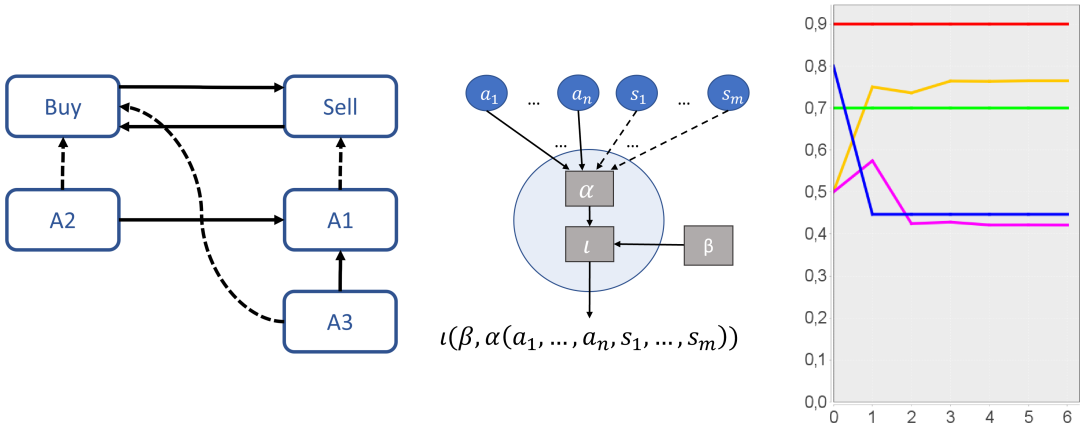

Figure 1: Example of a GAF (left), illustration of update mechanics under modular semantics (middle) and evolution of strength values under MLP-based semantics (right), where the x-axis shows the number of iterations and the y-axis the current strength value of Buy (yellow), Sell (violet), A1 (blue), A2 (green) and A3(red).

score based on the aggregate as illustrated in Figure 1 in the middle. Examples of aggregation functions include product [8, 9], addition [23, 11] and maximum [12] and the influence function is defined accordingly to guarantee that strength values fall in the desired range.

We will only look at the MLP-based semantics from [14] here. Under this semantics, layered acyclic GAFs correspond to multilayer perceptrons (MLPs) with logistic activation functions and vice versa. This connection can be preserved for other activation functions of the MLP by changing the influence function of the GAF. However, bounded activation functions like the logistic function are most intuitive because they always result in finite strength values. The strength values of arguments are computed by the following iterative procedure: For every argument $a \in \mathcal{A}$, we let $s_a^{(0)} := \beta(a)$ be the initial strength value. The strength values are then updated by doing the following two steps repeatedly for all $a \in \mathcal{A}$ (we use an auxiliary variable α_a^i that carries the aggregate at iteration i):

Aggregation: We let $\alpha_a^{(i+1)} := \sum_{(b,a)\in E} w(b,a) \cdot s_b^{(i)}$.

Influence: We let $s_a^{(i+1)} := \varphi_l\big(\ln(\frac{\beta(a)}{1-\beta(a)}) + \alpha_a^{(i+1)}\big)$, where $\varphi_l(z) = \frac{1}{1+\exp(-z)}$ is the logistic function.

Figure 1 shows on the right how the MLP-based strength values of our running example evolve. The weight of all attack (support) edges was set to -1 (1) and the base scores correspond to the strength values (y-axis) at iteration 0 (x-axis).

As shown in [14], the MLP-based semantics satisfies almost all semantical properties from the literature perfectly. A comparison to the earlier semantics DF-QuAD [9], Euler-based semantics [23] and the quadratic energy model [11] are shown in Figure 2. Only the *Open-Mindedness* property can be violated, and this can only happen when arguments have base scores 0 or 1. This case can actually not occur in our setting because base scores 0 or 1 correspond to infinite weights of the corresponding MLP that cannot be taken in practice. Roughly speaking,

Property	DfQ	Euler	QEM	MLP	Property	DfQ	Euler	QEM	MLP
Anonymity	✓	✓	✓	✓	Reinforcement	(✓)	✓	✓	✓
Independence	✓	✓	✓	✓	Resilience	(✓)	✓	✓	✓
Directionality	✓	✓	✓	✓	Franklin	✓	✓	✓	✓
Equivalence	✓	✓	✓	✓	Weakening	(✓)	✓	✓	✓
Stability	✓	✓	✓	✓	Strengthening	(✓)	✓	✓	✓
Neutrality	✓	✓	✓	✓	Duality	✓	×	✓	✓
Monotony	(✓)	✓	✓	✓	Open-Mindedness	×	×	✓	(✓)

Figure 2: Desirable properties for gradual argumentation semantics from [14]. ✓ indicates full satisfaction, (✓) satisfaction if base scores 0 and 1 are excluded and *x* indicates that the property can be violated even when excluding base scores 0 or 1.

Open-Mindedness demands that continuously adding attackers/supporters to an argument will eventually bring the strength to 0/1 [24]. For readers familiar with some, but not all of the properties, let us note that *Open-Mindedness* is similar to some "monotonicity"-properties that demand that an attacker/supporter must decrease/increase the strength of an argument. However, these properties are not sufficient to guarantee that the strength converges to 0/1 in the limit. For example, the Euler-based semantics from [23] satisfies these monotonicity properties in most cases, but the lower limit is not 0, but $\beta(a)^2$ (base score squared) [24]. The different semantics shown in Table 2 can be compared experimentally with the Java library Attractor[1] [25].

3. Meta-Heuristics for Learning GAFs

Our goal is to learn an interpretable MLP/GAF that can solve classification problems with sufficient accuracy. The abstract goal of classification is to map inputs $\mathbf{x}$ to outputs y. A typical example is classifying a customer as credit-worthy or not (output) based on personal data like age and income (input). The inputs are feature tuples $\mathbf{x} = (x_1, \ldots, x_k)$, where the i-th value is taken from some domain D_i. The output y is taken from a finite set of class labels L. A classification problem $P = ((D_1, \ldots, D_k), L, E)$ consists of the domains, the class labels and a set of training examples $E = \{(\mathbf{x}_i, y_i) \mid 1 \leq i \leq N\}$.

A numerical classifier is a function $c : \left(\bigtimes_{i=1}^{k} D_i\right) \times L \to \mathbb{R}$ that assigns to every pair $(\mathbf{x}, y)$ a numerical value. An important special case is a probabilistic classifier $p : \left(\bigtimes_{i=1}^{k} D_i\right) \times L \to [0, 1]$ where $\sum_{j=1}^{|L|} p(\mathbf{x}, y_i) = 1$. Then $p(\mathbf{x}, y) \in [0, 1]$ can be understood as the confidence of the classifier that an example with features $\mathbf{x}$ belongs to the class y. Note that every numerical classifier c can be turned into a probabilistic classifier p_c by normalizing the label outputs by a softmax function. That is, $p_c(\mathbf{x}, y) = \frac{\exp(c(\mathbf{x}, y_i))}{\sum_{j=1}^{|L|} \exp(c(\mathbf{x}, y_j))}$.

[1] https://sourceforge.net/projects/attractorproject/

As we explained in the previous section, MLPs can be seen as special cases of GAFs [14]. Hence, when we train an MLP, the resulting network can be understood as a classification GAF that is defined as follows.

Definition 3 (Classification GAF [26])**.** A *Classification GAF with k layers* for a classification problem $P = ((D_1, \ldots, D_k), L, E)$ is a GAF $(\mathcal{A}, E, \beta, w)$ such that $\mathcal{A} = \bigcup_{i=0}^{k+1} \mathcal{A}_{\langle i \rangle}$ consists of the input arguments $\mathcal{A}_{\langle 0 \rangle} = \mathcal{A}_{\text{in}}$ and output arguments $\mathcal{A}_{\langle k+1 \rangle} = \mathcal{A}_{\text{out}}$ for P and additional layers of arguments $\mathcal{A}_{\langle i \rangle}$ such that $\mathcal{A}_{\langle i \rangle} \cap \mathcal{A}_{\langle j \rangle} = \emptyset$ for $i \neq j$. Furthermore, $E \subseteq \bigcup_{i=0}^{k} \bigcup_{j=i+1}^{k+1} \left(\mathcal{A}_{\langle i \rangle} \times \mathcal{A}_{\langle j \rangle}\right)$, that is, edges can only be directed towards deeper layers.

Classification GAFs under MLP-based semantics are always MLPs. However, while MLPs are usually dense and use continuous inputs, we want to learn sparse and well interpretable network structures with binarized inputs that can be understood as (propositonal) abstract arguments.

3.1. The Structure Learning Problem

Given the graphical structure of a *classification GAF*, we can train the weights using the usual backpropagation procedure for neural networks. However, we also need a way to identify good candidates in the space of all structures. As we are interested in learning sparse networks, we will restrict the depth (number of layers), width (number of arguments per layer) and outdegree (number of outgoing edges from an argument) of the network.

Our *hypothesis space* $\mathcal{H}$ is a set of graphical candidate structures. Here, the candidates are classification GAFs and we define the space by fixing a set of arguments $\mathcal{A} = \bigcup_{i=0}^{k+1} \mathcal{A}_{\langle i \rangle}$ that can occur in GAFs. The hypothesis space $\mathcal{H}(\mathcal{A}, n_e)$ then consists of all possible subgraphs of the fully connected GAF over $\mathcal{A}$ with at most n_e outgoing edges per argument, where subgraphs are defined by removing nodes from $\mathcal{A} \setminus \mathcal{A}_{\langle k+1 \rangle}$ (only the output arguments are fixed) and removing arbitrary edges. Even though the depth k, the size of the layers $\mathcal{A}_{\langle i \rangle}$ and the outdgree n_e can be chosen arbitrarily, they should be kept small in order to find an easily interpretable GAF.

The candidate input arguments $\mathcal{A}_{\langle 0 \rangle}$ can be created automatically from a given dataset $\mathcal{D}$. For every boolean feature B, we introduce a corresponding argument A_B. For every categorical feature X with domain $\{v_1, \ldots, v_l\}$, we introduce l arguments $A_{X=v_1}, \ldots, A_{X=v_l}$. For every continuous feature X with domain $[l, u]$ we can partition $[l, u]$ into bins $b_1, \ldots, b_l$ and introduce l input arguments $A_{X=b_1}, \ldots, A_{X=b_l}$. The bins can be created using statistics like quantiles as bin boundaries or using values that actually occur in $\mathcal{D}$.

In order to evaluate candidates, we consider a *scoring function* $s : \mathcal{H} \to [0, 1]$ that assigns to every candidate a score between 0 and 1. In the following, we will consider the scoring function

$$s_\lambda(C) = (1 - \lambda) \cdot \text{Accuracy}(C, \mathcal{D}_{\text{train}}) + \lambda \cdot \frac{n_{max} - n_C}{n_{max}}, \tag{1}$$

where $C \in \mathcal{H}$ is a candidate network structure. $\text{Accuracy}(C, \mathcal{D}_{\text{train}})$ is computed by training C on the training set $\mathcal{D}_{\text{train}}$ using standard neural network parameter learning algorithms and computing and returning the accuracy (the percentage of correctly classified instances). n_c is the number of edges in C and n_{max} the number of edges in the fully connected GAF corresponding

GeneticAlgorithm :

pop ← initialize(N)

do

  mating_pool ← select(pop)

  offspring ← recombine(mating_pool)

  offspring ← mutate(offspring)

  population ← replace(pop, offspring)

until *termination condition reached*

return best(pop)

ParticleSwarmOptimization :

pop ← initialize(N)

do

  evaluate(pop)

  move_particle(pop)

until *termination condition reached*

return best(pop)

Figure 3: Genetic algorithm and particle swarm optimization templates.

to C (that is, the GAF that contains the same arguments like C in the same layers and connects every argument to every argument in the next layer). The hyperparameter $\lambda \in [0, 1]$ indicates the relative importance of an individual's sparsity in relation to its accuracy on the training data. Since both accuracy and the regularization term yield values between 0 and 1, we have $s(C) \in [0, 1]$ for every choice of λ.

Formally, our structure learning problem corresponds to the following optimization problem: find a candidate $C^* \in \mathcal{H}$ such that

$$C^* = \arg\max_{C \in \mathcal{H}} s_\lambda(C). \tag{2}$$

Let us note that $\mathcal{H}$ is exponentially large even if we fix the nodes in the network. To see this, assume that we consider n nodes and allow only 2 edges per node. Then every node can have 1 or 0 outgoing edges, which results in 2^n possible structures. Therefore, we will not attempt to find an optimal solution C^*, but consider meta-heuristics in order to find a good solution.

3.2. Genetic Algorithm Implementation

The first meta-heuristic that we consider is a genetic algorithm. Genetic algorithms maintain a set P of candidate solutions from $\mathcal{H}$ during the search process. The candidate solutions $c \in P$ are called *chromosomes* and the set P the *population*. Given the hypothesis space $\mathcal{H}(\mathcal{A}, n_e)$ over $\mathcal{A} = \bigcup_{i=0}^{k+1} \mathcal{A}_{\langle i \rangle}$, a chromosome is given by a vector of length $\sum_{i=0}^{k} |\mathcal{A}_{\langle i \rangle}| \cdot |\mathcal{A}_{\langle i+1 \rangle}|$. The entries can be 0 or 1 and indicate which neuron in layer i is connected to which neuron in layer $i + 1$. When creating the GAF from the chromosome representation, we only consider neurons that are either output neurons or have at least one outgoing edge.

Figure 3 shows on the left the template of our genetic algorithm. To begin with, a subset of the population (mating pool) is selected for reproduction based on their fitness defined by our scoring function s. The offspring is created by applying a *recombination function* that creates a new chromosome by combining features of the parent chromosomes. The reproduction step is followed by a *mutation step* that is supposed to move to interesting new regions of the search

space, e.g., by randomly perturbing chromosomes. Finally, a *replacement function* replaces part of the current population with promising offspring. The algorithm continues until a termination criterion is met. For details about the individual steps, we refer to the technical report [27]. The implementation is available at

https://github.com/jspieler/QBAF-Learning.

3.3. Particle Swarm Optimization

The second meta-heuristic that we consider is particle swarm optimization. Figure 3 shows on the right the template of our particle swarm optimization algorithm. Similar to genetic algorithms, we maintain a population of individuals (the swarm). Individuals are represented by vectors and we apply the same encoding like in our genetic algorithm. In every iteration, the algorithm evaluates all individuals and stores the globally best found solution and, for every individual, the individual best found solution. Each particle is then moved in a randomized direction that depends on the previous direction (initialized randomly), the location of the globally best solution found and the individual best solution found. For details about the individual steps, we refer to the thesis [28]. The implementation is available at

https://github.com/bazomd/sparse-mlp-structure-learning.

4. Experiments

To illustrate our approach, we show some examples of learnt GAFs and compare them to other interpretable models.

Data sets: We used three different data sets from the UCI Machine Learning Repository[2], namely the Iris (4 numerical features, 150 instances), the Mushroom (22 categorical features, 8,124 instances) and the Adult Income (14 mixed features, 48,842 instances) data sets.

Baselines: We compared our two meta-heuristics for learning GAFs to two other interpretable classification models, namely logistic regression and decision trees with different depths.

Hypothesis space: The input arguments were generated using default discretization methods from the scikit-learn library. We allowed one hidden layer and experimented with different numbers of arguments and outgoing edges. We kept both numbers below 20 to find an easily interpretable GAF.

Results: We split the datasets into training and test set (80/20 %). To take account of randomness during training, we averaged the test results over 10 runs. For our baselines, decision trees worked generally better than logistic regression. Overall, the performance of GAFs is similar to decision trees. The interpretability is similar as well. However, we feel that the flat GAFs (one hidden layer) that we learnt can be easier to interpret than deeper decision trees as we explain later. We show some learning statistics in Table 1. More details about the experiments can be found in the technical report [27] (baselines and genetic algorithm) and the thesis [28] (particle swarm optimization).

[2] https://archive.ics.uci.edu

Dataset	Algorithm	Test Accuracy (mean/std)	Size (mean/std)
Iris	DTree	95 (3.73)	Depth 2-5
	Genetic	97.34 (2)	8.2 (1.72) edges
	PSO	94.33 (3)	6.4 (2.1) edges
Mushroom	DTree	99.98 (0.05)	Depth 7
	Genetic	97.12 (1.63)	16.5 (5.02) edges
	PSO	97.61 (0.59)	6.4 (1.27) edges
Adult	DTree	82.44 (0.23)	Depth 2
	Genetic	81.72 (0.29)	13.6 (4.22) edges
	PSO	81.29 (0.7)	9.1 (2.82) edges

Table 1
Performance and size of best baselines and GAFs learnt by the genetic and particle swarm optimization (PSO) algorithm averaged over 10 learning trials.

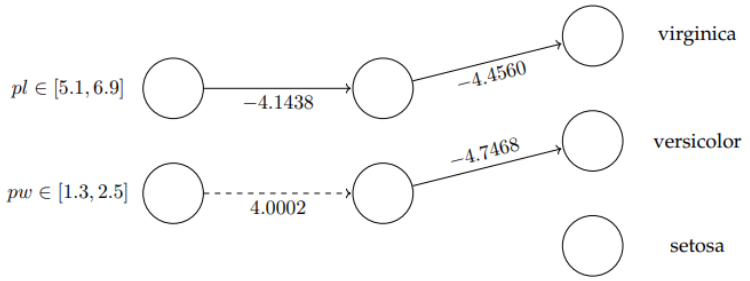

Figure 4: Sparse GAF for Iris dataset with 93.33 % test accuracy.

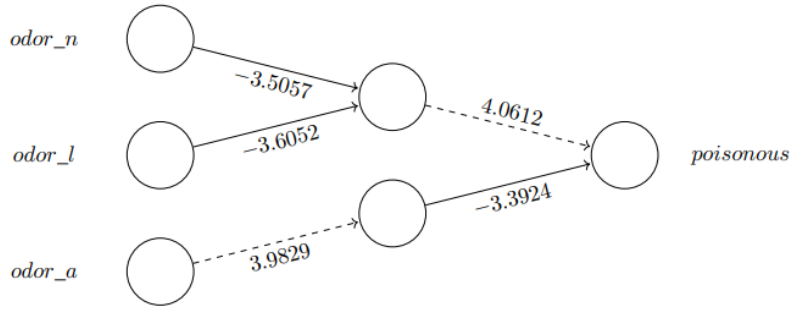

Figure 5: Sparse GAF for Mushroom dataset with 98.33 % test accuracy.

Examples: We close the section with three hand-picked examples of sparse GAFs that have been created in the experiments in Table 1. More examples can be found in the technical report [27] (genetic algorithm) and the thesis [28] (particle swarm optimization or can be generated with the corresponding code. Figures 4, 5 and 6 show one of the 10 GAFs learnt for the Iris, Mushroom and Adult dataset, respectively. The picked examples are sparser than the mean

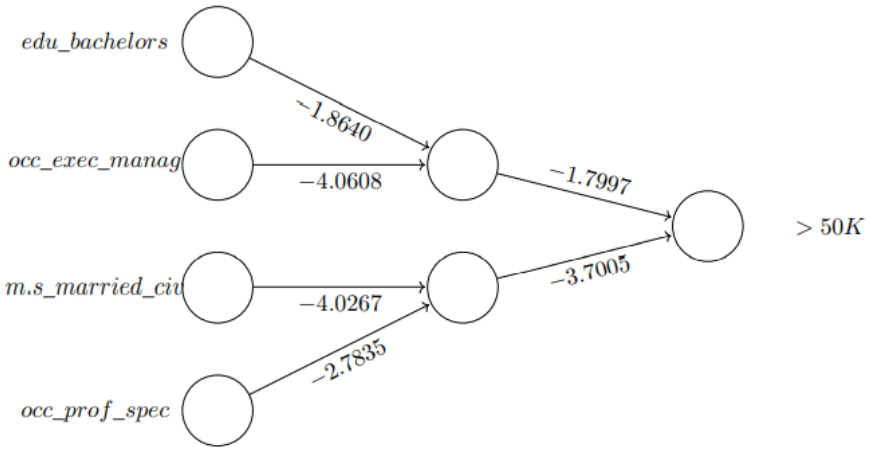

Figure 6: Sparse GAF for Adult dataset with 82.67 % test accuracy.

in Table 1. Notably, the examples for the Mushroom and Adult dataset even have a better test accuracy than the mean in Table 1. This can be explained by the fact that sparser models are less prone to overfit the data because they have fewer parameters.

When interpreting multiclass classification problems like the Iris dataset, one has to keep in mind that a softmax function is applied on top. That is, when the strength values for the classes virginica, versicolor and setosa are s_{vi}, s_{ve}, s_{se}, they are normalized to $\frac{s_{vi}}{N}, \frac{s_{ve}}{N}, \frac{s_{se}}{N}$, where $N = s_{vi} + s_{ve} + s_{se}$. Hence, when s_{vi} is large, it will make s_{ve} and s_{se} smaller. Hence, the class arguments can actually be seen as mutual attackers. In future versions, we may take account of this in the plotted graphs by printing special attack edges between them.

The GAFs should be interpreted recursively starting from the classes. For example, when interpreting the GAF for the Mushroom dataset in Figure 5, poisonous is supported by the upper hidden node. If it did not have any predecessors, the strength of its support would basically be given by its base score. However, it is rather strongly attacked by both odor_n and odor_l. Since both of them attack a supporter, they can be seen as indirect attackers of poisonous. However, the intermediate hidden node is necessary to capture their joint effect (remember that their effects are combined by the non-linear logistic function). The path from odor_a to poisonous can be interpreted similarly. In this case, the path can actually be simplified in a post-processing step to replace the supported attack with a simple attack. This is a simple, but useful improvement for the next version.

5. Conclusions and Future Work

We discussed the problem of learning classification GAFs from data. While the parameter learning problem can be solved using standard methods for neural networks, we focus on the structure learning problem to learn easily interpretable GAFs. As opposed to regularization and pruning techniques that are commonly used for learning sparse neural networks, the structure learning problem allows restricting the space of possible structures, e.g., by defining upper

bounds for the maximum number of outgoing edges per argument.

Our meta-heuristics learnt classification GAFs that are competitive with other interpretable models like logistic regression and decision trees. The main shortcoming of logistic regression is that it can only distinguish linearly separable classes. In contrast, as our GAFs are MLPs, the universal approximation theorem for MLPs implies that classification GAFs can approximate every function with a single hidden layer that is sufficiently large. Decision trees are both expressive and well interpretable. However, as the trees grow, it becomes increasingly difficult to understand the various case differentiations. The quantitative semantics of GAFs sometimes allows capturing the relationships in the data more succinctly. For example, while a decision tree with depth 7 achieved almost perfect test accuracy in our experiments, such a deep tree is already difficult to comprehend. The GAF in Figure 5 performed slightly worse, but is significantly easier to understand.

We are planning several improvements. We will add a postprocessing step that replaces paths between single arguments like in Figure 4 and the bottom path in Figure 5 with a single attack/support edge by merging the base scores and edge weights along the path appropriately. An interesting structural improvement is to replace interval arguments as $pl \in [5.1, 6.9]$ in Figure 4 with fuzzy arguments like $pl \sim 6$ that are not simply 0 or 1, but can increase/decrease gradually based on the degree to which they are satisfied. Another interesting structural improvement is to consider joint attacks/supports that could capture joint effects of arguments as in the upper path in Figure 5 without using hidden layers to make the GAF more interpretable.

Finally, let us note that [29] recently considered a similar problem of learning classical classification argumentation frameworks. As the structure learning problems in both settings are very similar, we will look at possible cross-fertilization in future work.

References

[1] M. T. Ribeiro, S. Singh, C. Guestrin, " why should i trust you?" explaining the predictions of any classifier, in: KDD, 2016, pp. 1135–1144.

[2] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Int. conference on neural information processing systems, 2017, pp. 4768–4777.

[3] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, PloS one 10 (2015) e0130140.

[4] L. M. Zintgraf, T. S. Cohen, T. Adel, M. Welling, Visualizing deep neural network decisions: Prediction difference analysis, in: Int. Conference on Learning Representations ICLR, OpenReview.net, 2017, p. Virtual.

[5] J. Heo, S. Joo, T. Moon, Fooling neural network interpretations via adversarial model manipulation, Advances in Neural Information Processing Systems 32 (2019) 2925–2936.

[6] M. Wu, S. Parbhoo, M. C. Hughes, V. Roth, F. Doshi-Velez, Optimizing for interpretability in deep neural networks with tree regularization, JAIR 72 (2021) 1–37.

[7] L. Amgoud, C. Cayrol, M.-C. Lagasquie-Schiex, P. Livet, On bipolarity in argumentation frameworks, International Journal of Intelligent Systems 23 (2008) 1062–1093.

[8] P. Baroni, M. Romano, F. Toni, M. Aurisicchio, G. Bertanza, Automatic evaluation of design alternatives with quantitative argumentation, Argument & Computation 6 (2015) 24–49.
[9] A. Rago, F. Toni, M. Aurisicchio, P. Baroni, Discontinuity-free decision support with quantitative argumentation debates., in: KR, 2016, pp. 63–73.
[10] L. Amgoud, J. Ben-Naim, D. Doder, S. Vesic, Acceptability semantics for weighted argumentation frameworks, in: IJCAI, 2017, pp. 56–62.
[11] N. Potyka, Continuous dynamical systems for weighted bipolar argumentation, in: KR, 2018, pp. 148–157.
[12] T. Mossakowski, F. Neuhaus, Modular semantics and characteristics for bipolar weighted argumentation graphs, arXiv preprint arXiv:1807.06685 (2018).
[13] N. Potyka, Extending modular semantics for bipolar weighted argumentation, in: Int. Conference on Autonomous Agents and MultiAgent Systems (AAMAS), 2019, pp. 1722–1730.
[14] N. Potyka, Interpreting neural networks as gradual argumentation frameworks, in: AAAI Conference on Artificial Intelligence (AAAI), 2021, pp. 6463–6470.
[15] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (2015) 436–444.
[16] C. Louizos, M. Welling, D. P. Kingma, Learning sparse neural networks through l_0 regularization, in: Int. Conference on Learning Representations, ICLR, OpenReview.net, 2018, p. Virtual.
[17] S. Han, H. Mao, W. J. Dally, Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding, in: Int. Conference on Learning Representations (ICLR), 2016, p. Virtual.
[18] C. Louizos, K. Ullrich, M. Welling, Bayesian compression for deep learning, in: Int. Conference on Neural Information Processing Systems (NIPS), Curran Associates Inc., Red Hook, NY, USA, 2017, p. 3290–3300.
[19] C. Tai, T. Xiao, X. Wang, E. Weinan, Convolutional neural networks with low-rank regularization, in: Int. Conference on Learning Representations (ICLR), 2016, p. Virtual.
[20] A. Rago, O. Cocarascu, F. Toni, Argumentation-based recommendations: Fantastic explanations and how to find them, in: Int. Joint Conference on Artificial Intelligence (IJCAI), 2018, pp. 1949–1955.
[21] O. Cocarascu, A. Rago, F. Toni, Extracting dialogical explanations for review aggregations with argumentative dialogical agents, in: Int. Conference on Autonomous Agents and MultiAgent Systems (AAMAS), 2019, pp. 1261–1269.
[22] N. Kotonya, F. Toni, Gradual argumentation evaluation for stance aggregation in automated fake news detection, in: Workshop on Argument Mining, 2019, pp. 156–166.
[23] L. Amgoud, J. Ben-Naim, Evaluation of arguments in weighted bipolar graphs, in: European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU), Springer, 2017, pp. 25–35.
[24] N. Potyka, Open-mindedness of gradual argumentation semantics, in: Scalable Uncertainty Management (SUM), volume 11940 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 236–249.
[25] N. Potyka, A tutorial for weighted bipolar argumentation with continuous dynamical systems and the java library attractor, Int. Workshop on Non-Monotonic Reasoning (NMR), 2018.

[26] N. Potyka, Foundations for solving classification problems with quantitative abstract argumentation, in: Explainable and Interpretable Machine Learning (XI-ML), CEUR-WS.org, 2020, p. Virtual.
[27] J. Spieler, N. Potyka, S. Staab, Learning gradual argumentation frameworks using genetic algorithms, arXiv preprint arXiv:2106.13585 (2021).
[28] M. W. Bazo, Learning quantitative argumentation frameworks using sparse neural networks and swarm intelligence algorithms, 2021.
[29] P. Dondio, Towards argumentative decision graphs: Learning argumentation graphs from data, in: M. D'Agostino, F. A. D'Asaro, C. Larese (Eds.), Advances in Argumentation in Artificial Intelligence, CEUR-WS.org, 2021, p. Virtual.