Explanatory predictions with artificial neural networks and argumentation

Oana Cocarascu, Kristijonas Čyras, Francesca Toni
Imperial College London, UK, {oc511, k.cyras, f.toni}@imperial.ac.uk

Abstract

Data-centric AI has proven successful in several domains, but its outputs are often hard to explain. We present an architecture combining Artificial Neural Networks (ANNs) for feature selection and an instance of Abstract Argumentation (AA) for reasoning to provide effective predictions, explainable both dialectically and logically. In particular, we train an autoencoder to rank features in input examples, and select highest-ranked features to generate an AA framework that can be used for making and explaining predictions as well as mapped onto logical rules, which can equivalently be used for making predictions and for explaining. We show empirically that our method significantly outperforms ANNs and a decision-tree-based method from which logical rules can also be extracted.

1 INTRODUCTION

Data-centric AI has recently received a great deal of attention and shown success in several domains (e.g. see [LeCun *et al.*, 2015] for Artificial Neural Networks (ANNs)), but its predictions are notoriously hard to explain (e.g. see [Andrews *et al.*, 1995], again for ANNs). Argumentation has been increasingly studied in AI in the last two decades, mostly to deal with incomplete and conflicting information (see [Atkinson *et al.*, 2017] for a recent survey) but also to provide dialectical explanations for reasoning outputs of various kinds (e.g. see [Amgoud and Serrurier, 2007; Fan and Toni, 2015]). These explanations are dialectical in that they can be seen as debates between two opposing parties. We combine ANNs (of the 'autoencoder' variety [Hinton and Salakhutdinov, 2006]) and argumentation (of the 'Abstract Argumentation' (AA) variety [Dung, 1995]) to provide effective predictions that can be explained dialectically as well as logically, from automatically extracted rules. We call our methodology ANNA (for ANNs with argumentation).

Within ANNA, during training an autoencoder is used to rank features in input examples, as more or less representative. These training examples are labelled as belonging to one of two given classes (*outcomes*). The ranking is then used to select a subset of (highest-ranked) features so that the restriction of the training examples to these features is coherent (namely, with no two restrictions having the same features but different outcomes). The resulting restriction of the training examples is then mapped onto an AA framework (AAF), using the approach of [Čyras *et al.*, 2016a], and in turn this AAF is mapped onto a set of logical rules (of the 'logic programming' variety). In the AAF, the set of arguments consists of the (restrictions of) examples as well as a *default* argument (giving the default outcome in the absence of any information), and an argument attacks another if they have different outcomes and the former is a 'concise' superset of the latter. In the logic program, the default outcome is provable by default, and rules admit exceptions, obtained from examples.

The AAF and the logic program can be equivalently used (during testing) for prediction of the outcome for unlabelled examples. For a given example M, with restriction N to the set of features selected during training, the prediction is made as follows: (i) by adding to the AAF a new argument for N, attacking every argument in the AAF with features irrelevant to N, and determining whether the default argument is dialectically acceptable in the extended AAF, using the grounded semantics [Dung, 1995]; or, equivalently (ii) by adding to the logic program a fact (rule with an empty body) for every feature in N and determining whether the default outcome being provable is a logical consequence of the extended logic program, using semantics for logic programming [Apt and Bol, 1994]. The extended AAF and logic program can be used to explain the prediction, dialctically or logically, respectively.

We show empirically, using the dataset of [Dheeru and Karra Taniskidou, 2017], that ANNA significantly outperforms ANNs, while being less sensitive to the size of the training dataset, as well as a decision-tree-based method from which logical rules can also be extracted.

2 PRELIMINARIES

In this section we give essential background on Artificial Neural Networks (ANNs), argumentation and logic programming, of the kinds used within our method.

ANNs have been widely applied both in classification tasks and in dimensionality reduction, e.g. as in [Verikas and Bacauskiene, 2002]. Typically, dimensionality reduction refers to selecting a subset of the original feature set (selection) or mapping the initial data in M-dimensional space onto a K-dimensional space with $K < M$ (extraction). ANN-based

feature selection methods use multilayer perceptrons to determine which features are redundant [Gasca *et al.*, 2006] as well as autoencoders [Hinton and Salakhutdinov, 2006; Wang *et al.*, 2017; Han *et al.*, 2017]. These are unsupervised learning models based on ANNs which take a set of features as input and aim, through training, to reconstruct the inputs [Hinton and Salakhutdinov, 2006; Erhan *et al.*, 2010].

In this paper we use autoencoders for feature selection and generic ANNs for experimental comparison.

AA-CBR [Čyras *et al.*, 2016a; Čyras *et al.*, 2016b] is an Abstract Argumentation (AA)-driven and Case-Based Reasoning (CBR)-inspired model for reasoning with (past) cases (where a case is a set of features together with an outcome) to predict the outcomes of new cases.

Consider a fixed but otherwise arbitrary (possibly infinite) set $\mathbb{F}$ of *features*, and a set $\mathbb{O} = \{\delta, \overline{\delta}\}$ of two (distinct) *outcomes*, with δ called the *default outcome*. Then:

- a *case* is a pair (X, o) with $X \subseteq \mathbb{F}$ and $o \in \mathbb{O}$;
- a *case base* is a finite set $CB \subseteq \wp(\mathbb{F}) \times \mathbb{O}$ that is *coherent*, i.e. for $(X, o_X), (Y, o_Y) \in CB$, if $X = Y$ then $o_X = o_Y$;
- a *new case* is a pair $(N, ?)$ with $N \subseteq \mathbb{F}$ and ? indicating that the outcome is yet unknown.

To illustrate, we use the following example throughout.

Example 2.1. Let $\mathbb{F} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$, $CB = \{(\{\mathsf{a}\}, \overline{\delta}), (\{\mathsf{b}\}, \overline{\delta}), (\{\mathsf{a}, \mathsf{c}\}, \delta), (\{\mathsf{b}, \mathsf{d}\}, \overline{\delta})\}$ and $(N, ?) = (\{\mathsf{a}, \mathsf{d}\}, ?)$. It is easy to see that CB is coherent.

AA-CBR maps the problem of determining the outcome for a new case into a membership problem within the grounded extension of an AA framework (AAF) [Dung, 1995] obtained from the case base CB, the new case $(N, ?)$ and the default outcome δ. In general, an *AAF* is a pair $(Args, \rightsquigarrow)$, where $Args$ is a set (of *arguments*) and $\rightsquigarrow$ is a binary relation on $Args$ (where, for $\mathsf{a}, \mathsf{b} \in Args$, if $\mathsf{a} \rightsquigarrow \mathsf{b}$, then we say that a *attacks* b and that a is an *attacker of* b). For a set of arguments $E \subseteq Args$ and an argument $\mathsf{a} \in Args$, E *defends* a if for all $\mathsf{b} \rightsquigarrow \mathsf{a}$ there exists $\mathsf{c} \in E$ such that $\mathsf{c} \rightsquigarrow \mathsf{b}$. Then, the *grounded extension* of $(Args, \rightsquigarrow)$ can be constructed as $\mathbb{G} = \bigcup_{i \geqslant 0} G_i$, where G_0 is the set of all unattacked arguments, and $\forall i \geqslant 0$, G_{i+1} is the set of arguments that G_i defends. For any $(Args, \rightsquigarrow)$, the grounded extension $\mathbb{G}$ always exists and is unique.

AA-CBR encompasses the following two modules, respectively (1) extracting an AAF from a coherent case base and a default outcome, and (2) determining the outcome of a new case from the grounded extension of the AAF augmented to take into account the new case.

Module 1 – $aaf(CB, \delta)$ gives $(Args, \rightsquigarrow)$ with:

- $Args = CB \cup \{(\{\}, \delta)\}$;
- for $(X, o_X), (Y, o_Y) \in CB \cup \{(\{\}, \delta)\}$, it holds that $(X, o_X) \rightsquigarrow (Y, o_Y)$ iff
  1. $o_X \neq o_Y$, and (different outcomes)
  2. $Y \subsetneq X$, and (specificity)
  3. $\nexists (Z, o_X) \in CB$ with $Y \subsetneq Z \subsetneq X$. (concision)

The *default argument* $(\{\}, \delta)$ represents the outcome obtained in the absence of any information (i.e. features).

Module 2 – $outcome((Args, \rightsquigarrow), N)$ gives *the AA-CBR outcome* of $(N, ?)$ defined as follows:

i. let $(Args_N, \rightsquigarrow_N)$ be the AAF with
   - $Args_N = Args \cup \{(N, ?)\}$;
   - $\rightsquigarrow_N = \rightsquigarrow \cup \{((N, ?), (Y, o_Y)) : (Y, o_Y) \in Args, Y \not\subseteq N\}$, i.e. $(Args_N, \rightsquigarrow_N)$ extends $(Args, \rightsquigarrow)$ with the new case argument $(N, ?)$ that attacks all arguments with 'irrelevant' features, i.e. features not in N;

ii. let $\mathbb{G}$ be the grounded extension of $(Args_N, \rightsquigarrow_N)$; $\mathbb{G}$ is non-empty, as $(N, ?)$ is unattacked;

iii. then the *AA-CBR outcome* of $(N, ?)$ is
   - δ, if $(\{\}, \delta) \in \mathbb{G}$;
   - $\overline{\delta}$, otherwise, if $(\{\}, \delta) \notin \mathbb{G}$.

Example 2.2. Given CB, δ and $(N, ?)$ from Example 2.1, $outcome(aaf(CB, \delta), N)$ gives $\overline{\delta}$. Indeed $(Args, \rightsquigarrow) = aaf(CB, \delta)$ is the AAF depicted in Figure 1 when the polygon node and all arrows from it are ignored, and $(Args_N, \rightsquigarrow_N)$ from step (i) in Module 2 is the full AAF in the figure.[1] The grounded extension of $(Args_N, \rightsquigarrow_N)$ is

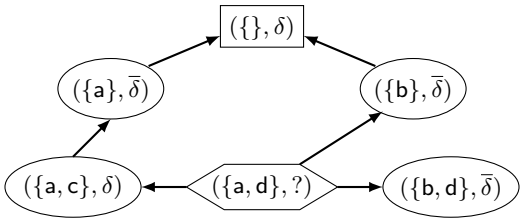

Figure 1: $(Args_N, \rightsquigarrow_N)$ for Example 2.2.

$\mathbb{G} = \{(\{\mathsf{a}, \mathsf{d}\}, ?), (\{\mathsf{a}\}, \overline{\delta})\}$, and $(\{\}, \delta) \notin \mathbb{G}$. Note that, in $(Args_N, \rightsquigarrow_N)$, $(\{\mathsf{b}, \mathsf{d}\}, \overline{\delta}) \not\rightsquigarrow (\{\}, \delta)$ because of the concision requirement: $(\{\mathsf{b}\}, \overline{\delta})$ is a more concise attacker of $(\{\}, \delta)$. Note also that $(N, ?)$ attacks all cases that have features not present in N.

For our purposes, a **logic program** is a set $\mathcal{P}$ of ground rules of the form $h \leftarrow p_1, \ldots, p_s, \mathsf{not}\ p_{s+1}, \ldots, \mathsf{not}\ p_{s+t}$. where $s, t \geq 0$, h and each p_i are atoms, and not is negation as failure. h is called the head, and $p_1, \ldots, p_s, \mathsf{not}\ p_{s+1}, \ldots, \mathsf{not}\ p_{s+t}$ the body of such a rule. A fact is a rule $h \leftarrow .$ with an empty body. All logic programs in this paper are stratified, and can be ascribed a meaning using the perfect model semantics[Apt and Bol, 1994]. We say that an atom a is *provable* (in $\mathcal{P}$) if $a \in model(\mathcal{P})$.

3 ANNA METHODOLOGY – PART I

In this section we describe our Artificial Neural Networks with Argumentation methodology for prediction (ANNA in short), summarised in Figure 2, but ignoring rule extraction and rule-based prediction (part II) until Section 5.

3.1 AUTOENCODER + FEATURE SELECTION

ANNA assumes the availability of a *training dataset* consisting of a (possibly large) set of examples, (referred to as $\mathcal{L}$ in the remainder), each amounting to a set of *features* from a

[1] Here, as conventional, the AAF is depicted as a graph with nodes holding arguments and arrows indicating attacks. We use different shapes for nodes for readability: the default argument $(\{\}, \delta)$ and the new case $(\{\mathsf{a}, \mathsf{d}\}, ?)$ argument are enclosed in rectangle and polygon respectively, while other arguments are enclosed in ellipses.

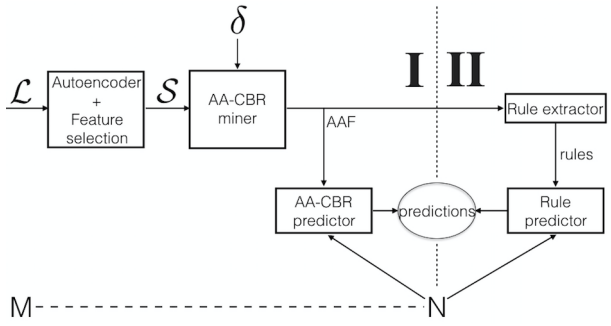

Figure 2: A bird's eye view of ANNA. **Part I**: during training with dataset $\mathcal{L}$ a trimmed dataset $\mathcal{S}$ is obtained, that, together with default outcome δ, gives rise to an AAF used for predicting the outcome for (any) M via its trimmed version N. **Part II**: the AAF is mapped onto rules for making equivalent predictions for M via N.

given set $\mathbb{F}_{\mathcal{L}}$ and an *outcome* from a set $\mathbb{O}$. We assume that there are exactly two outcomes in $\mathbb{O}$, i.e. prediction is a binary classification problem. The set $\mathbb{F}_{\mathcal{L}}$ may be large, especially if features are assignments of alternative values to attributes (e.g. 'colour' may take many different values). ANNA relies upon an autoencoder for capturing the most salient features from the training dataset, resulting in a *trimmed dataset* (referred to as $\mathcal{S}$ in the remainder), which may be much smaller than the original $\mathcal{L}$ and consisting of examples with much fewer features (from a trimmed set $\mathbb{F} \subseteq \mathbb{F}_{\mathcal{L}}$). Formally, $\mathcal{S} = \{(Y, o) : (X, o) \in \mathcal{L}, Y = X \cap \mathbb{F}\}$. ANNA enforces that feature selection, leading to $\mathbb{F}$, is such that $\mathcal{S}$ is *coherent* and thus representative of the original $\mathcal{L}$ (if coherent to start with) and devoid of noise (if $\mathcal{L}$ was not coherent). Thus feature selection guarantees that $\mathcal{S}$ is "rational".

Example 3.1. Let examples in $\mathcal{L}$ be characterised by 5 attributes $a_1, \ldots, a_5$ and an outcome amongst o_1, o_2. Suppose each attribute may take one of 4 distinct, discrete values, say $v^1_{a_i}, \ldots, v^4_{a_i}$ for a_i. Then $\mathbb{F}_{\mathcal{L}} = \{a_1 = v^1_{a_1}, \ldots, a_5 = v^4_{a_5}\}$ consists of 20 features (binary attribute-value pairs). Suppose that ANNA is realised so as to select 4 features, namely $\mathbb{F}$ consists of 4 elements, e.g. $\mathbb{F} = \{a_1 = v^1_{a_1}, a_2 = v^4_{a_2}, a_3 = v^2_{a_3}, a_4 = v^4_{a_4}\} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$: if $\mathcal{L} = \{(\{\mathsf{a}, a_1 = v^2_{a_1}, a_5 = v^1_{a_5}\}, o_1), (\{\mathsf{a}, a_1 = v^3_{a_1}\}, o_1)\}$ then $\mathcal{S} = \{(\{\mathsf{a}\}, o_1)\}$. Note that if $\mathcal{L}$ had also included $(\{\mathsf{a}, a_5 = v^1_{a_5}\}, o_2)$, then ANNA would not have selected $\mathbb{F}$, as this would have resulted in an incoherent $\mathcal{S} = \{(\{\mathsf{a}\}, o_1), (\{\mathsf{a}\}, o_2)\}$.

In our realisation of the ANNA methodology in this paper (and in particular in the experiments we present in Section 4) we use a simple autoencoder with one hidden layer as shown in Figure 3, with (for $X \subseteq \mathbb{F}_{\mathcal{L}}$): (i) an encoder function $f(X) = \sigma(XW^{(1)})$, and (ii) a decoder function $\sigma(f(X)W^{(2)})$, where $W^{(1)}, W^{(2)}$ are the weight parameters in the encoder and decoder, respectively.

To select $\mathbb{F}$, we average the weights in $W^{(1)}$ for each input and select the top F factors. F can be chosen in many alternative ways, e.g. iteratively (starting from a small number of features, until a coherent S is obtained) or empirically (as in the experiments in Section 4, where F=22, with $|\mathcal{L}| = 126$).

In the remainder we will refer to elements of $\mathcal{L}$ as *original examples*, and to elements of $\mathcal{S}$ simply as *examples*.

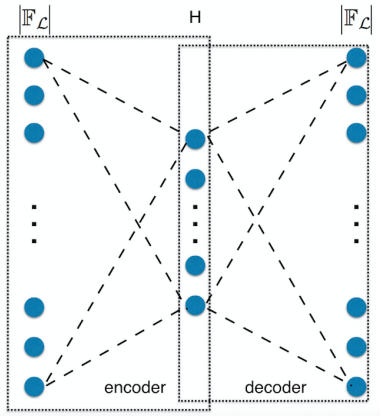

Figure 3: Autoencoder used in ANNA: $|\mathbb{F}_{\mathcal{L}}|$ binary features are used to train the autoencoder to obtain a **code** (hidden layer **h** of size $H < |\mathbb{F}_{\mathcal{L}}|$) that best captures the input $|\mathbb{F}_{\mathcal{L}}|$ features.

3.2 AA-CBR MINER

Note that $\mathcal{S}$ is a coherent case base, in the sense of Section 2, and examples in $\mathcal{S}$ can be seen as cases. ANNA assumes that a default δ can be naturally identified in $\mathbb{O}$, as the prediction that can be legitimately made in the absence of information about features. Then, the AA-CBR miner returns $aaf(\mathcal{S}, \delta)$ (see Module 1 in Section 2). Thus, if $\mathcal{S} = CB$ in Example 2.1, the AAF given by the AA-CBR miner is in Example 2.2.

Note that the AAF returned by the AA-CBR miner can be seen as a model of the trimmed dataset, and, if the feature selection is rational, also of the original training dataset. This model identified conflicts between (original) examples, that need to be resolved every time that a prediction is to be made.

3.3 AA-CBR PREDICTOR

The AA-CBR predictor takes a set M of features in $\mathbb{F}_{\mathcal{L}}$, characterising an unseen example with unknown outcome, and predicts an outcome from $\mathbb{O}$ as follows. Let the *trimming function* $\tau : \wp(\mathbb{F}_{\mathcal{L}}) \mapsto \wp(\mathbb{F})$ be $\tau(X) = X \cap \mathbb{F}$, for any $X \subseteq \mathbb{F}_{\mathcal{L}}$. Note that trimming different sets of features may result into the same set, e.g. in Example 3.1 $\tau(\{\mathsf{a}, a_1 = v^2_{a_1}, a_5 = v^1_{a_5}\}) = \tau(\{\mathsf{a}, a_1 = v^3_{a_1}\}) = \{\mathsf{a}\}$. The AA-CBR predictor takes $N = \tau(M)$ in input, alongside $(Args, \rightsquigarrow)$ given by the AA-CBR miner, and computes $outcome((Args, \rightsquigarrow), N)$ (see Module 2 in Section 2), as in Example 2.2.

4 EMPIRICAL EVALUATION

In this section we conduct experiments with a publicly available dataset, showing that ANNA is an effective method when compared to end-to-end ANNs and to a method based on decision trees. The latter is chosen for comparison as decision trees also produce logical rules for justifying predictions, as ANNA-Part II (see Section 5).

The dataset consists of 8124 examples of gilled mushrooms classified as edible or poisonous. Each example is characterised by 22 (categorical) attributes that can take a number of different values, leading to 126 (binary) features. In our experiments we choose several subsets of this dataset as $\mathcal{L}$, but, for all choices, $|\mathbb{F}_{\mathcal{L}}| = 126$.

In our experiments, $H \in \{22, 30\}$ (where H is the size of the hidden layer, see Figure 3), $|\mathbb{F}| = 22$, for $\mathbb{F}$ the trimmed set of features, $\mathbb{O} = \{\text{edible, poisonous}\}$ and $\delta = \text{edible}$. This means that in the absence of any information, i.e. features, about a mushroom, it can be deemed edible, as represented by $(\{\}, \delta)$. However, as soon as a mushroom with features is encountered, it being edible has to be justified by countering all the relevant examples of poisonous mushrooms.

We use sigmoid as activation function in the autoencoder and binary cross entropy as loss function. We have experimented with $|\mathbb{F}| \in \{22, 30, 50\}$, with tanh and ReLU as activations functions and with various optimizers, but report results for the best performing combinations of parameters.

Table 1: 5-fold cross validation results

Hidden layer size 22	**Precision**	**Recall**	$\mathbf{F_1}$
ANNA	**0.97**	**0.96**	**0.958**
Autoencoder + ANN	0.938	0.894	0.878
ANN	0.934	0.888	0.87
Hidden layer size 30			
ANNA	**0.97**	**0.962**	**0.962**
Autoencoder + ANN	0.932	0.886	0.86
ANN	0.936	0.896	0.88

In Table 1 we report 5-fold cross validation results, using weighted averages for each metric, for a stand-alone ANN, for a combination of Autoencoder+ANN, and for ANNA. The chosen ANN has one hidden layer and uses sigmoid as activation function and softmax to make predictions. The hyper-parameters were optimised using the Adam method [Kingma and Ba, 2014] with learning rate 0.001. For Autoencoder+ANN, we use the learnt weights $W^{(1)}$ from the encoder, which we do not optimise during training, and softmax for classification. In both cases, we trained for 50 epochs or until the performance on the development set stopped improving. In the case of ANNA, we use the learnt weights from the encoder to select the top 22 features. As shown in Table 1, ANNA performs better than the two ANN approaches with differences in F_1 up to 8% and 10% when using a size 22 and 30 hidden layer, respectively.

We also conducted experiments to test whether ANNA can better cope with smaller datasets than the Autoencoder+ANN method (arguably the better performing of the two end-to-end ANN methods). Hence we run experiments on 6000 randomly drawn examples and 5000 randomly drawn examples, respectively, and tested on the remaining examples in the starting dataset. We repeated the experiments 5 times and report the average performances in Table 2. Here as well we use the learnt weights $W^{(1)}$ from the encoder in Autoencoder+ANN and softmax for prediction. In this set of experiments, we also used for comparison a method, referred to simply as Decision Tree (DT) in Table 2, that also uses an autoencoder for feature selection. For ANNA and DT alike, we use the learnt weights from the encoder to select the top 22 features as $\mathbb{F}$ and give $\mathcal{S}$ as discussed in Section 3.1. We used information gain for DT and selected 1 as the minimum number of samples required to be at a leaf node to mimic ANNA treating each example as an argument.

Table 2: Average of 5 runs of training on a reduced dataset and testing on the remaining examples.

Training set size: 6000, Testing set size: 2124			
Hidden layer size 22	**Precision**	**Recall**	$\mathbf{F_1}$
ANNA	**0.978**	**0.976**	**0.976**
Autoencoder + ANN	0.802	0.642	0.61
Decision Tree (DT)	0.858	0.774	0.762
Hidden layer size 30			
ANNA	**0.966**	**0.964**	**0.966**
Autoencoder + ANN	0.802	0.638	0.604
Decision Tree (DT)	0.852	0.772	0.766
Training set size: 5000, Testing set size: 3124			
Hidden layer size 22	**Precision**	**Recall**	$\mathbf{F_1}$
ANNA	**0.954**	**0.954**	**0.954**
Autoencoder + ANN	0.84	0.76	0.75
Decision Tree (DT)	0.876	0.828	0.826
Hidden layer size 30			
ANNA	**0.97**	**0.97**	**0.97**
Autoencoder + ANN	0.84	0.756	0.748
Decision Tree (DT)	0.886	0.844	0.844

The experiments on reduced datasets show that Autoencoder+ANN is less performing than ANNA and DT. ANNA performs better than DT throughout all experiments, with improvements in F_1 up to 20% and 12% with training set size of 6000 and 5000 examples, respectively.

5 ANNA METHODOLOGY – PART II

In this section we describe how ANNA generates logic programs that capture reasoning with examples in $\mathcal{S}$. These rules allow to make exactly the same predictions for new examples as the AA-CBR predictor. At the same time, these rules summarise the structure of $\mathcal{S}$ and, by extension, that of $\mathcal{L}$. In particular, they concisely describe in logical terms what the prediction should be and which features should influence it. The rules are generic for $\mathcal{S}$ but when a prediction for a new example is made its features are added as facts to the rules.

We give ANNA Rule extractor/predictor (Figure 2) below.

5.1 RULE EXTRACTOR

Unless stated otherwise, let $(Args, \rightsquigarrow) = aaf(\mathcal{S}, \delta)$ be the AAF returned by the AA-CBR miner (see Section 3.2). Informally, rule extraction works as follows.

- Take $(\{\}, \delta) \in Args$ and its attackers (if any).
  - Create a rule stating that $(\{\}, \delta)$ is accepted unless any of its attackers are accepted.
  - For each attacker $(\{\mathsf{f}_1, \ldots, \mathsf{f}_m\}, \overline{\delta})$ of $(\{\}, \delta)$, create a rule stating that it is accepted if the features $\mathsf{f}_1, \ldots, \mathsf{f}_m$ hold, unless any of its attackers are accepted.
  - If there are no attackers, create a rule stating that the argument is accepted if all its features hold.
- Repeat for each attacker and its attackers in turn.

Formally, assume a naming function $name$ that assigns a unique name to every argument in $Args$: we write $C : \mathsf{a}$ to indicate that C names argument a, and we name $(\{\}, \delta)$ by def. Define the procedure $extract(C, (Args, \rightsquigarrow), \mathcal{R})$, taking the name C of an argument in $Args$ and a rule set $\mathcal{R}$, thus:

1. Given $C : (\{\mathsf{f}_1, \ldots, \mathsf{f}_m\}, o)$, let $\{C_1, \ldots, C_k\}$ be the set of the names of all the attackers of C in $(Args, \rightsquigarrow)$.
2. Add the following rule to $\mathcal{R}$, to obtain $\mathcal{R}'$: $acc(C) \leftarrow \mathsf{f}_1, \ldots, \mathsf{f}_m, \mathsf{not}\ acc(C_1), \ldots, \mathsf{not}\ acc(C_k)$.
3. If $\{C_1, \ldots, C_k\} = \{\}$ then return $\mathcal{R}'$.
4. For every $C_i \in \{C_1, \ldots, C_k\}$, compute $extract(C_i, (Args, \rightsquigarrow), \mathcal{R}')$, to obtain $\mathcal{R}'_1, \ldots, \mathcal{R}'_k$.
5. Return $\mathcal{R}'_1 \cup \ldots \cup \mathcal{R}'_k$.

The Rule extractor computes $extract(def, (Args, \rightsquigarrow), \{\})$, whose output is a logic program, henceforth called $\mathcal{P}$.

Example 5.1. Consider $(Args, \rightsquigarrow)$ from Example 2.1, depicted as in Figure 1 but without the polygon and attacks from it. Let $def, C_\mathsf{a}, C_\mathsf{b}, C_\mathsf{c}, C_\mathsf{d}$ be the names of arguments $(\{\}, \delta), (\{\mathsf{a}\}, \overline{\delta}), (\{\mathsf{b}\}, \overline{\delta}), (\{\mathsf{a}, \mathsf{c}\}, \delta), (\{\mathsf{b}, \mathsf{d}\}, \overline{\delta})$, respectively. Then $\mathcal{P}$ consists of the following rules:

$$acc(def) \leftarrow \mathsf{not}\ acc(C_\mathsf{a}), \mathsf{not}\ acc(C_\mathsf{b}).$$
$$acc(C_\mathsf{a}) \leftarrow \mathsf{a}, \mathsf{not}\ acc(C_\mathsf{c}).$$
$$acc(C_\mathsf{c}) \leftarrow \mathsf{a}, \mathsf{c}. \qquad acc(C_\mathsf{b}) \leftarrow \mathsf{b}.$$

These rules describe how $acc(def)$ can be proved. Note that there is no rule for acceptance of C_d, because C_d does not attack any other argument. In particular, there is no (directed) path from it to $(\{\}, \delta)$ in $(Args, \rightsquigarrow)$. Hence, C_d is never reached within $extract(def, (Args, \rightsquigarrow), \{\})$. This illustrates that the feature d is not important in proving $acc(def)$.

Note that by construction $\mathcal{P}$ is stratified as $(Args, \rightsquigarrow)$ has no cycles and every $\mathsf{not}\ acc(C')$ in the body of rules with head $acc(C)$ refers to some child (attacker) C' of C.

5.2 RULE PREDICTOR

The generic rules forming $\mathcal{P}$ can be used to predict the outcome of new example $N = \tau(M)$ by adding each feature from N as a fact to give $program(\mathcal{P}, N) = \mathcal{P} \cup \{\mathsf{f} \leftarrow . : \mathsf{f} \in N\}$. Henceforth, $\mathcal{P}_N$ denotes the logic program $program(\mathcal{P}, N)$.

Example 5.2. In Example 5.1, $\mathcal{P}_N$ consists of rules:

$$acc(def) \leftarrow \mathsf{not}\ acc(C_\mathsf{a}), \mathsf{not}\ acc(C_\mathsf{b}).$$
$$acc(C_\mathsf{a}) \leftarrow \mathsf{a}, \mathsf{not}\ acc(C_\mathsf{c}).$$
$$acc(C_\mathsf{c}) \leftarrow \mathsf{a}, \mathsf{c}. \qquad acc(C_\mathsf{b}) \leftarrow \mathsf{b}.$$
$$\mathsf{a} \leftarrow . \qquad \mathsf{d} \leftarrow .$$

Note: as $\mathcal{P}$ is stratified, $\mathcal{P}_N$ is stratified. The Rule predictor takes $(N, ?)$ and $\mathcal{P}_N$ in input, and predicts δ if $acc(def)$ is provable, and $\overline{\delta}$ otherwise. With $\mathcal{P}_N$ from Example 5.2, $model(\mathcal{P}_N) = \{\mathsf{a}, \mathsf{d}, acc(C_\mathsf{a})\}$, so $acc(def) \notin model(\mathcal{P}_N)$, i.e. $acc(def)$ is not provable, and the prediction is $\overline{\delta}$.

The Rule predictor and the AA-CBR predictor are equivalent in the following sense:

Theorem 5.1. *Let $\mathcal{S}$, δ and N be given. Let $(Args, \rightsquigarrow) = aaf(\mathcal{S}, \delta)$, $\mathcal{P} = extract(def, (Args, \rightsquigarrow), \{\})$, and $\mathcal{P}_N = program(\mathcal{P}, N)$. Then the AA-CBR outcome of $(N, ?)$ is δ iff $acc(def) \in model(\mathcal{P}_N)$.*

Proof. First note that by construction $\mathcal{P}_N$ can be stratified while traversing $(Args, \rightsquigarrow)$ from $(\{\}, \delta)$ to the leaves, by putting the rule for acceptance of $(\{\}, \delta)$ on the top stratum, and the rule for acceptance of any other argument on the stratum lower by the length of the longest (directed) path from that argument to $(\{\}, \delta)$. The facts from N can then be put on the lowest stratum.

Given such a partition, the iterated fixpoint construction effectively mimics the acceptance of $(\{\}, \delta)$ in the grounded extension of $(Args_N, \rightsquigarrow_N)$ given by $outcome((Args, \rightsquigarrow), N)$ (see Module 2 in Section 2). First, for any rule $acc(A) \leftarrow \ldots, p, \ldots.$ with $p \notin N$, $acc(A)$ is not provable. Accordingly, $A : (X, o)$ is attacked by $(N, ?)$, so $(X, o) \notin \mathbb{G}$. Then, for any $acc(C) \leftarrow p_0, \ldots, p_s.$, $acc(C)$ is provable. Accordingly, $acc(C)$ names a leaf in $(Args_N, \rightsquigarrow_N)$ and so is in $\mathbb{G}$. Any argument A attacked by C is thus not in $\mathbb{G}$. Accordingly, any rule with head $acc(A)$ has in its body some $\mathsf{not}\ p$ with p provable, so $acc(A)$ is not provable.

This iteration over rules finishes with the rule $acc(def) \leftarrow \mathsf{not}\ acc(C_1), \ldots, \mathsf{not}\ acc(C_k)$, whence $acc(def)$ is provable only if none of $acc(C_i)$ is. By the reasoning above, this amounts to $\mathbb{G}$ defending $(\{\}, \delta)$. In particular, $(\{\}, \delta) \in \mathbb{G}$ iff $acc(def) \in model(\mathcal{P}_N)$. □

This result shows that in ANNA, both AA-CBR predictor and Rule predictor can be equivalently used to make predictions for new examples. Thus, the two predictors have the same predictive performances (see Section 4).

6 ILLUSTRATING EXPLANATIONS

In this section we illustrate the explanatory power of ANNA with the mushroom dataset used in Section 4.

6.1 LOGICAL EXPLANATIONS

In what follows, we use the logic program $\mathcal{P}$ obtained for $\mathcal{S}$ from one of the datasets from one of the experiments (with $|\mathcal{L}| = 5000$) in Section 4. This $\mathcal{S}$ consists of 306 examples, and $\mathcal{P}$ consists of 13 rules. The rule in $\mathcal{P}$ concerning the acceptance of $(\{\}, \delta)$ is

$$acc(def) \leftarrow \mathsf{not}\ acc(131^{def}), \ldots, \mathsf{not}\ acc(34^{def}). \quad (1)$$

with 7 elements in the body (of which we spell out 2).[2] This rule says that the outcome is δ—namely, a mushroom being edible—unless at least one of the seven exceptions applies. Let us look at one particular exception, induced by example 131. The rule concerning acceptance of 131 is

$$acc(131^{def}) \leftarrow \mathsf{f}_1, \ldots, \mathsf{f}_{10}, \mathsf{not}\ acc(2504^{131}). \quad (2)$$

with 11 elements in the body, including one exception. This rule says that given features $\mathsf{f}_1, \ldots, \mathsf{f}_{10}$, example 131 is an exception to the default outcome, but that it also admits an exception—namely example 2504.

The exception to $acc(131^{def})$ is given by the rule

$$acc(2504^{131}) \leftarrow \mathsf{gill-size\_broad}. \quad (3)$$

This rule says that an exception to example 131 is obtained as soon as the feature $\mathsf{gill-size\_broad}$ is present. Note that example 2504 has no exceptions.

Suppose that $N = \{\mathsf{f}_1, \ldots, \mathsf{f}_{10}, \mathsf{gill-size\_broad}\}$ and that only exception 131 to rule (1) applies in $\mathcal{P}_N$. Then rules (1), (2), (3) and the facts obtained from N form the basis for a logical explanation for why δ is predicted for N.

[2] Examples in $\mathcal{S}$ are named by integer numbers.

6.2 DIALECTICAL EXPLANATIONS

The prediction for the outcome of a given N in the context of some $\mathcal{S}$ can be explained *dialectically* too, by using $(Args, \rightsquigarrow) = aaf(\mathcal{S}, \delta)$ and $(Args_N, \rightsquigarrow_N)$ given by $outcome((Args, \rightsquigarrow), N)$ (see Section 3). To this end, imagine a debate between a proponent, P, seeking to establish that a given mushroom is edible, and an opponent, O, seeking to establish that it is poisonous. This debate can be visualised via $(Args_N, \rightsquigarrow_N)$, with arguments labelled as P and O. The debate would unfold as follows.

Given some particular mushroom represented by $(N, ?)$, P moves argument $(\{\}, \delta)$, representing that by default a mushroom is edible. Note that $(\{\}, \delta)$ has no features, i.e. does not describe any mushroom in particular, but rather represents a generic mushroom. So accepting $(\{\}, \delta)$ means that in the absence of any information about a given mushroom, it should be edible. O then argues for the mushroom in question to be poisonous by putting forward examples of poisonous mushrooms. To this end, the opponent puts forward arguments with the non-default outcome $\overline{\delta}$. For instance, suppose that, mirroring Section 6.1, O uses the argument named 131.

P then has to counter-argue O's argument(s). P has two ways to do that. i) Either the example put forward by O is of a poisonous mushroom that has features different from N. For instance, it may be that $\mathsf{f}_{10} \notin N$. If this happens, P argues that the example is 'irrelevant', whence $(N, ?)$ attacks argument 131. Generally, putting forward 'irrelevant' examples is useless. ii) Otherwise, P can find a more specific example of an edible mushroom that has all the features of the given example and, in addition, some more features that are still present in N. For instance, if $\mathsf{gill-size\_broad} \in N$, then P can use argument 2504, which attacks argument 131.

The process continues in this way whereby O puts forward counter-examples and P tries to defend against those. If in the end O has no arguments to put forward and all of O's arguments are attacked by P's arguments, i.e. all leaves are labelled P, then P has established the default outcome, namely, that the mushroom in question is edible. Otherwise, if there is an applicable counter-example put forward by O that P cannot argue against, the default outcome cannot be upheld, and so the mushroom is deemed poisonous. For instance, suppose in the above that $(N, ?)$ attacks all the arguments attacking $(\{\}, \delta)$ but does not attack the one named 131, and suppose $\{\mathsf{f}_1, \ldots, \mathsf{f}_{10}, \mathsf{gill-size\_broad}\} \subseteq N$. Then P's last argument 2504, which is a leaf and hence unattacked, together with $(N, ?)$ defends $(\{\}, \delta)$, and so P wins the debate and establishes that the mushroom represented by N is edible.

7 RELATED WORK

Several works have integrated argumentation and machine learning (see e.g. [Cocarascu and Toni, 2016]). Perhaps the most relevant work to ours is that of [Amgoud and Serrurier, 2007] where argumentation and concept learning are used to reason about classifications. Arguments are examples and hypotheses. Preferences (assumed to be given) over arguments are employed to resolve conflicts stemming from an inconsistent training set, using the grounded extension to capture the version space model, while also supporting generation of argumentative explanations for classifications. In contrast, we advance a novel model using only examples. Our training sets are coherent (by construction) and the definition of attack incorporates a form of preference over arguments. Also, we provide dialectical and logical explanations, in terms of arguments and rules, respectively.

[d'Avila Garcez *et al.*, 2005] propose a Neural Argumentation Algorithm that translates argumentation frameworks to ANNs and affords semantic correspondence between the two. [Makiguchi and Sawamura, 2007] further this research to provide symbolic dialogues from ANNs. These works are not concerned with predictions or subsequent explanation of predictions. In [Thimm and Kersting, 2017] a proposal is formulated to extract (possibly inconsistent) rules from a dataset using machine learning, and then to use those rules in structured argumentation formalisms (see e.g. [Besnard *et al.*, 2014]) to classify new examples. We instead use ANNs to mine argumentation frameworks for reasoning and rule extraction.

In [Kontschieder *et al.*, 2016; Zhou and Feng, 2017; Balestriero, 2017] decision trees and ANNs are combined to give Neural Decision Trees. Such approaches can potentially be interpretable as trees are easier to analyse than ANNs and can cope with small datasets [Zhou and Feng, 2017]. By contrast, ANNA's data models are logic-based rather than probabilistic, and can be compactly represented via both (argumentation) graphs and (logic programming) rules.

Several approaches to explanation in AI have been proposed, e.g. see [Ribeiro *et al.*, 2018]. Rather than explaining existing methods, we have defined a novel method that is explainable, in two alternative ways (logically and dialectically), and in such a way that it can explain both predictions and underlying learnt model.

8 CONCLUSION AND FUTURE WORK

Our main contributions in this paper are as follows.

- Combining (a form of) Artificial Neural Networks (ANNs) and (a form of) Argumentation for solving binary classification problems, whereby ANNs perform feature selection and Argumentation performs classification.
- The resulting methodology, ANNA, provides argumentation-based predictions (classification) as well as logical rules that give equivalent predictions.
- ANNA performs well in our experiments.
- ANNA's predictions are explainable both dialectically and logically, and the explanations form argumentation-based and rule-based models of data.

In the future we plan to do (at least) the following: (i) further experiment with ANNA, considering other datasets and other forms of ANNs for feature selection or engineering; (ii) relax the notion of coherent case bases to deal with noise; (iii) incorporate probabilities over features; (iv) explore the application of logic programming transformation (e.g. see [Proietti and Pettorossi, 1995]) to simplify rule-based predictions; (v) compare our rule extraction to other approaches that mine rules from data, such as Inductive Logic Programming (e.g. see [Muggleton, 1991]); (vi) compare the forms of explanation we generate with those of other methods, e.g. that of [Ribeiro *et al.*, 2018].

References

[Amgoud and Serrurier, 2007] Leila Amgoud and Mathieu Serrurier. Agents that argue and explain classifications. *Autonomous Agents and Multi-Agent Systems*, 16(2):187–209, 2007.

[Andrews *et al.*, 1995] Robert Andrews, Joachim Diederich, and Alan B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Know.-Based Syst.*, 8(6):373–389, December 1995.

[Apt and Bol, 1994] Krzysztof R Apt and Roland N Bol. Logic programming and negation: A survey. *The Journal of Logic Programming*, 19/20:9–71, 1994.

[Atkinson *et al.*, 2017] Katie Atkinson, Pietro Baroni, Massimiliano Giacomin, Anthony Hunter, Henry Prakken, Chris Reed, Guillermo Ricardo Simari, Matthias Thimm, and Serena Villata. Towards artificial argumentation. *AI Magazine*, 38(3):25–36, 2017.

[Balestriero, 2017] Randall Balestriero. Neural decision trees. *CoRR*, abs/1702.07360, 2017.

[Besnard *et al.*, 2014] Philippe Besnard, Alejandro Javier García, Anthony Hunter, Sanjay Modgil, Henry Prakken, Guillermo Ricardo Simari, and Francesca Toni. Introduction to Structured Argumentation. *Argument & Computation*, 5(1):1–4, 2014.

[Cocarascu and Toni, 2016] Oana Cocarascu and Francesca Toni. Argumentation for machine learning: A survey. In *6th International Conference on Computational Models of Argument*, pages 219–230. IOS Press, 2016.

[Čyras *et al.*, 2016a] Kristijonas Čyras, Ken Satoh, and Francesca Toni. Abstract Argumentation for Case-Based Reasoning. In *Principles of Knowledge Representation and Reasoning, 15th International Conference*, pages 549–552. AAAI Press, 2016.

[Čyras *et al.*, 2016b] Kristijonas Čyras, Ken Satoh, and Francesca Toni. Explanation for Case-Based Reasoning via Abstract Argumentation. In *6th International Conference on Computational Models of Argument*, pages 243–254. IOS Press, 2016.

[d'Avila Garcez *et al.*, 2005] Artur S. d'Avila Garcez, Dov M. Gabbay, and Luís C. Lamb. Value-based argumentation frameworks as neural-symbolic learning systems. *Journal of Logic and Computation*, 15(6):1041–1058, 2005.

[Dheeru and Karra Taniskidou, 2017] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[Dung, 1995] Phan Minh Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-person Games. *Artificial Intelligence*, 77:321–357, 1995.

[Erhan *et al.*, 2010] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660, 2010.

[Fan and Toni, 2015] Xiuyi Fan and Francesca Toni. On computing explanations in argumentation. In *Proceedings of AAAI*, pages 1496–1502, 2015.

[Gasca *et al.*, 2006] Eduardo Gasca, José Salvador Sánchez, and R. Alonso. Eliminating redundancy and irrelevance using a new MLP-based feature selection method. *Pattern Recognition*, 39(2):313–315, 2006.

[Han *et al.*, 2017] Kai Han, Chao Li, and Xin Shi. Autoencoder feature selector. *CoRR*, abs/1710.08310, 2017.

[Hinton and Salakhutdinov, 2006] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[Kingma and Ba, 2014] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[Kontschieder *et al.*, 2016] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulò. Deep neural decision forests. In *Proceedings of IJCAI*, pages 4190–4194, 2016.

[LeCun *et al.*, 2015] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[Makiguchi and Sawamura, 2007] Wataru Makiguchi and Hajime Sawamura. A hybrid argumentation of symbolic and neural net argumentation (part II). In *Argumentation in Multi-Agent Systems, 4th International Workshop*, pages 216–233, 2007.

[Muggleton, 1991] Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[Proietti and Pettorossi, 1995] Maurizio Proietti and Alberto Pettorossi. Unfolding - definition - folding, in this order, for avaoiding unnecessary variables in logic programs. *Theoretical Journal of Computer Science*, 142(1):89–124, 1995.

[Ribeiro *et al.*, 2018] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Thimm and Kersting, 2017] Matthias Thimm and Kristian Kersting. Towards argumentation-based classification. In *Logical Foundations of Uncertainty and Machine Learning, Workshop at IJCAI'17*, 2017.

[Verikas and Bacauskiene, 2002] A. Verikas and M. Bacauskiene. Feature selection with neural networks. *Pattern Recogn. Lett.*, 23(11):1323–1335, 2002.

[Wang *et al.*, 2017] Shuyang Wang, Zhengming Ding, and Yun Fu. Feature selection guided auto-encoder. In *Proceedings of AAAI*, pages 2725–2731, 2017.

[Zhou and Feng, 2017] Zhi-Hua Zhou and Ji Feng. Deep forest: Towards an alternative to deep neural networks. In *Proceedings of IJCAI*, pages 3553–3559, 2017.