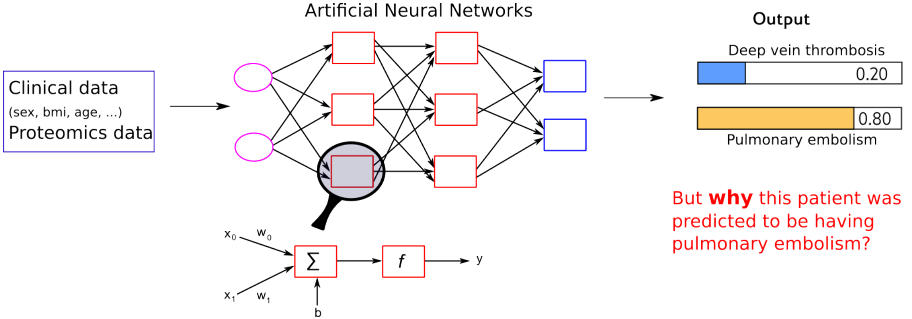
Explaining artificial neural networks using answer set programming

Roland Kaminski, Torsten Schaub, Javier Romero, Misbah Razzaq

INRAe, CNRS, Univeristy of Tours, France
University of Potsdam, Germany

September 9, 2024

Motivation



Why?

- Understanding predictions.
- Model debugging and validation.
- Discovering new biological insights.

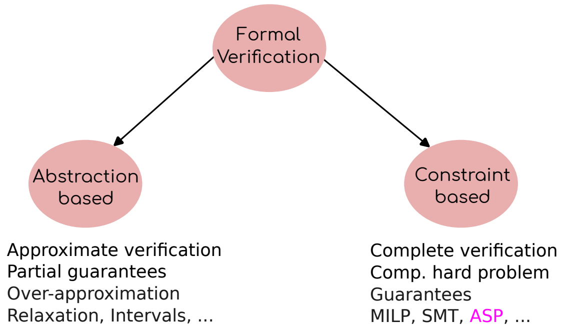
MOTIVATION

Two main approaches

- Non-formal verification methods.
  - LIME, Deeplift, SHAP, etc.
- Formal verification methods.

Some limitations!

- Susceptible to adversarial attacks.
- Quality of explanations.
- Incorrectness or incompleteness of explanations.



Narodytska et al. 2019; Al-Shedivat, Dubey, and Xing 2018

Background

Classifier

A classifier is a tuple $(X, \mathcal{D}, C, \kappa)$:

- X is a set of distinct features $x_1, \ldots, x_n$ taking values from $D_1, \ldots, D_n$ for $n > 0$.
- $\mathbb{F}$ is the feature space $D_1 \times D_2 \times \cdots \times D_n$.
- C is a set of distinct classes $c_1, \ldots, c_m$ for $m > 0$,
- κ is a classification function mapping $\mathbb{F}$ to C.

An *instance* is a pair (v, c) where $v \in \mathbb{F}$, $c \in C$, and $c = \kappa(v)$.

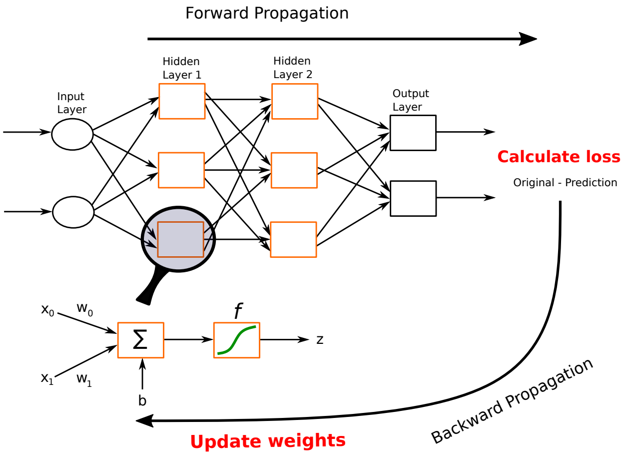
Background

Artificial Neural network

- Each neuron perform this action:

$$y = f(\sum_{i=0}^{n} w_i x_i + b) \tag{1}$$

where $x_0, x_1, \ldots, x_n$ are the inputs, $w_0, w_1, \ldots, w_n$ are the weights associated with the respective inputs, b represents the bias, y is the output of the neuron, and f is an activation function.



Definition (Abductive explanations)

An abductive explanation is a subset $E \subseteq X$ whose values are fixed according to v such that the prediction is c no matter the values of the remaining features:

$$\forall x \in \mathbb{F}\ s.t.\ [\bigwedge_{x_i \in E} (x_i = v_i)] \implies (\kappa(x) = c) \tag{2}$$

Note that we can equivalently write (2) as

$$\neg \exists x \in \mathbb{F}\ \text{s.t.}\ [\bigwedge_{x_i \in E} (x_i = v_i)] \wedge (\kappa(x) \neq c) \tag{3}$$

Note that this is weak explanation but if it is also subset minimal then it is an abductive explanation.

.

Marques-Silva 2022

Approach

Three step process

- Build the neural network for a given problem.
- Represent the classifier in the form of a logic program.
- We use a deletion based algorithm to obtain explanations.
  - Start with the initial set of features X which is an abductive explanation.
  - Iteratively drop features from X while it is still an explanation.

Input : Classifier $(X, \mathcal{D}, C, \kappa)$ and instance (v, c)
Output: Subset minimal abductive explanation E

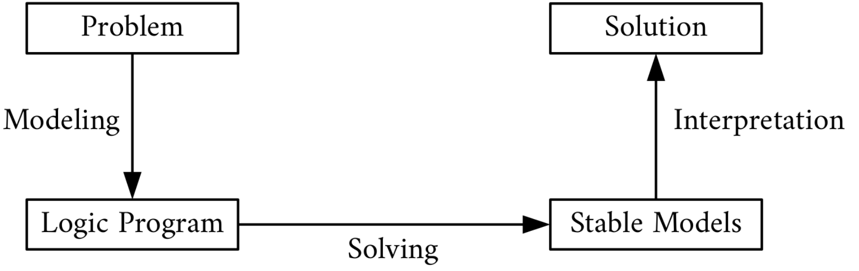
```
E ← X;
for x_i ∈ X do
    if E \ {x_i} is an abductive explanation then
        E ← E \ {x_i};
return E;
```

Algorithm 1: Algorithm to subset minimize an abductive explanation

Approach

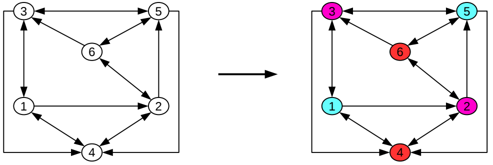
Answer set programming

- Declarative problem solving approach.
- What is the problem? versus How to solve the problem?
- Intelligence lies within the solver.
- Easy to understand.



Approach

Graph coloring problem



Facts:

$$
\begin{array}{lllll}
node(1..6) \leftarrow & & & & \\
edge(1,2) \leftarrow & edge(1,3) \leftarrow & edge(1,4) \leftarrow & edge(2,4) \leftarrow & \\
edge(2,5) \leftarrow & edge(2,6) \leftarrow & edge(3,1) \leftarrow & edge(3,4) \leftarrow & \\
edge(3,5) \leftarrow & edge(4,1) \leftarrow & edge(4,2) \leftarrow & edge(5,3) \leftarrow & \\
edge(5,4) \leftarrow & edge(5,6) \leftarrow & edge(6,2) \leftarrow & edge(6,3) \leftarrow & edge(6,5) \leftarrow
\end{array}
$$

Rules and objective function:

$$\{color(C)\} \leftarrow node(C) \tag{4}$$

$$1\ \{assign(U,C) : color(C)\}\ 1 \leftarrow node(U) \tag{5}$$

$$\leftarrow edge(U,V) \wedge assign(U,C) \wedge assign(V,C) \tag{6}$$

$$minimize\{1, C : color(C)\} \tag{7}$$

Solution:

$$\{assign(1,1), assign(2,3), assign(3,3), assign(4,2), assign(5,1), assign(6,2)\}$$

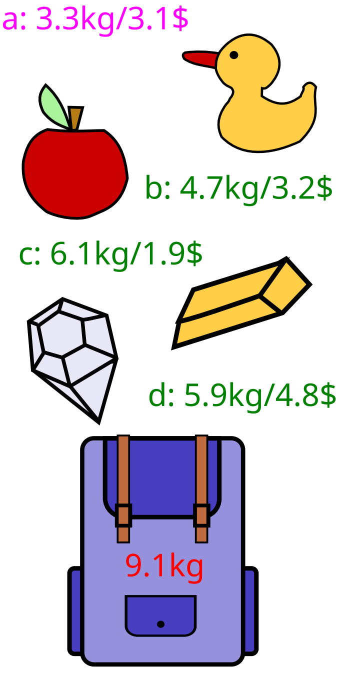
clingo[LP]

- Hybridized(ASP)
- Idea extend *clingo* with linear constraints over integers and reals (Janhunen et al. 2017).
- Features
  - Linear constraints &sum.
  - Objective function &minimize.

Example

0/1 Knapsack

```
item(a;b;c;d).
weight(a,"3.3";b,"4.7";c,"6.1";d,"5.9").
value(a,"3.1";b,"3.2";c,"1.9";d,"4.8").
load("9.1").
```

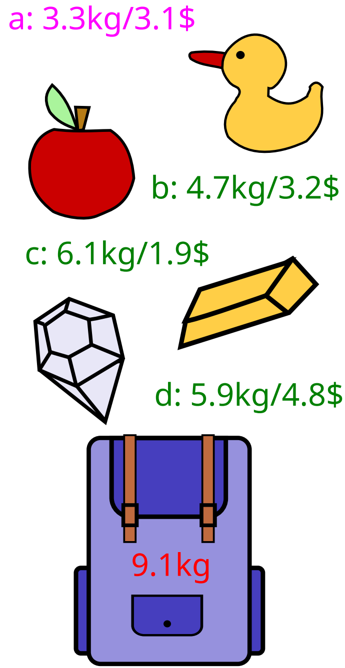


Example

0/1 Knapsack

```
item(a;b;c;d).
weight(a,"3.3";b,"4.7";c,"6.1";d,"5.9").
value(a,"3.1";b,"3.2";c,"1.9";d,"4.8").
load("9.1").

{ pack(I) } :- item(I).
&sum { I } = 1 :- pack(I).
&sum { I } = 0 :- item(I), not pack(I).
&sum { W*I: weight(I,W) } <= L :- load(L).
&maximize { P*I: value(I,P) }.
```



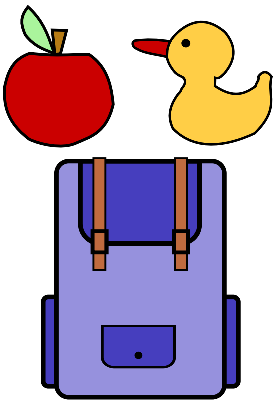
Example

0/1 Knapsack

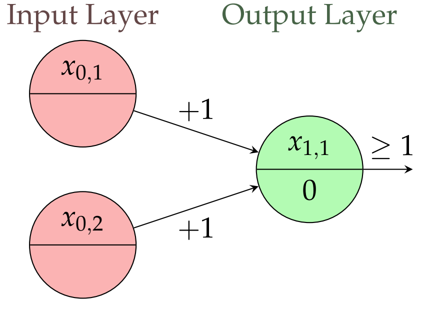
```
Model:
item(a) item(b) item(c) item(d)
load("9.1")
pack(a) pack(b)
value(a,"3.1") value(b,"3.2")
value(c,"1.9") value(d,"4.8")
weight(a,"3.3") weight(b,"4.7")
weight(c,"6.1") weight(d,"5.9")

Assignment:
a=1 b=1 c=0 d=0

Optimization:
6.3
```



A simple artificial neural network with one layer, weight matrix $W_1 = \begin{pmatrix} 1 & 1 \end{pmatrix}$, bias vector $B_1 = \begin{pmatrix} 0 \end{pmatrix}$, and threshold $t = 1$.



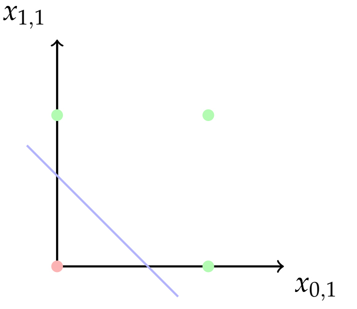


Figure. An ANN to classify an OR gate.

The input variables $X = \{x_{0,1}, x_{0,2}\}$, domains $D_1 = D_2 = \{0, 1\}$, classes $\{\bot, \top\}$, and $\kappa(x_{0,1}, x_{0,2}) = \mathrm{threshold}(\mathrm{relu}(1 \cdot x_{0,1} + 1 \cdot x_{0,2} + 0), 1)$.

```
var(x_{0,1}).            dom(x_{0,1},0).            dom(x_{0,1},1).
var(x_{0,2}).            dom(x_{0,2},0).            dom(x_{0,2},1).
relu(x_{1,1},0).         elem(x_{1,1},1,x_{0,1}).   elem(x_{1,1},1,x_{0,2}).
threshold(x_{1,1},1).
classify(⊤).             input(x_{0,1},1).
```

Approach

ASP encodings

```
{ assign(X,V): dom(X,V) } = 1 :- var(X), not input(X,_).
assign(X,V) :- input(X,V).
&sum { X } = V :- assign(X,V).

&sum { Y } >= 0 :- relu(Y,B).
&sum { neg(Y) } >= 0 :- relu(Y,B).
&sum { W*X: elem(Y,W,X); -Y; neg(Y) } = -B :- relu(Y,B).

{ aux(Y) } >= 0 :- relu(Y,B).
&sum { neg(Y) } <= 0 :-                aux(Y).
&sum {     Y   } >  0 :-                aux(Y).
&sum {     Y   } <= 0 :- relu(Y,_), not aux(Y).

&sum { Y } >= V :- threshold(Y,V), classify(⊥).
&sum { Y } <  V :- threshold(Y,V), classify(⊤).
```

Benchmarks
UCI machine learning datasets

- Build artificial neural networks with ReLU activation function.
- Encoding in the form of a logic programs with Boolean and linear constraints.
- Calculate explanations.

		Time (s)			Size		
Dataset	Features	min	avg	max	min	avg	max
Heart disease	13	0.33	3.84	3.84	4	10	13
Thyroid	16	0.34	18.91	59.77	6	8	16
Breast cancer	9	0.12	5.64	59.8	3	5	9
Diabetes	21	0.41	7.30	59.87	10	17	21
E. coli promoter	57	5.24	5.64	6.90	57	57	57
Voting	16	0.23	13.40	34.95	4	5	9

Table. Time to compute/size of explanations for machine learning datasets generated using ASP.

Benchmarks

Comparison with logic based approaches

	MILP			ASP		
Dataset	min	avg	max	min	avg	max
Heart disease	**7**	**9**	13	**4**	**10**	13
Breast cancer	3	5	9	3	5	9
Voting	**3**	5	11	**4**	5	11

Table. Explanation sizes for MILP and ASP-based approaches.

Benchmarks

Congressional Voting Records from UCI machine learning repository

Data description:

- 16 key votes (Yes or No) of 1984 U.S. House of Representatives
- Classes represent the party of the congressman : Republican or Democrat
- 435 instances with missing data

Neural network:

- 1 hidden layer neural network with 8 hidden nodes
- ReLU activation function
- Accuracy: 96%

Generate explanations:

- Represent neural network in the form of ASP encodings
- length of explanation varies between between 4 and 9 (avg len 5)
- Time varies between 2s and 17s (avg time 13s)

Benchmarks

Congressional Voting Records from UCI machine learning repository

An instance which is classified correctly as republican by our neural network

Features	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Length
Instance	0	1	0	1	1	1	0	0	0	0	0	0	1	1	0	1	16
ASP				1					0		0			1			4
OBDD			0	1	1		0	0	0				1	1	0		9
MILP				1					0		0						3

Voted YES on Physician-fee-freeze (feature num 4)
Voted NO on syn-fuels-corporation-cutback (feature num 11)

In case of ASP, if we set feature 4 and 14 to yes and 9 and 11 to no then the classification is republication, does not matter what values other features take

Benchmarks

Thyroid recurrence prediction dataset

Method	Features
Decision tree	structurally incomplete treatment response, low risk, age
Lime	structurally incomplete treatment response, low risk, intermediate risk
ASP	structurally incomplete treatment response, intermediate risk, euthyroid thyroid function categorization

Table. Explanations by different methods in thyroid recurrence prediction.

- Slight variation in identifying other important factors may come from the strengths and biases of each method, highlighting the importance of employing multiple interpretation techniques for a robust analysis.
- Predictive factors of recurrence in well-differentiated thyroid carcinoma, contributing to the better understanding of the model.

CONCLUSION

- Abductive explanations are model agnostic.
- A proof of concept to encode neural networks in the form of a logic program in ASP.
- Seamless conversion to logic program given inputs, weight matrices and bias vectors.
- Competitive performance on different benchmarks.
- Why specific/individual/local predictions are done!

Perspectives

Methodology

- Other functions, more complex networks (more hidden layers, other architectures).
- Extend to non-binary classification tasks.
- How to change predictions? Generate adversarial examples.
- An alternative dedicated system to encode neural networks?

Reliability

- Robustness of explanations (negligible change -> negligible change in explanation).
- Multiple networks and identify the one with robust explanations.
- Consistency of explanations across similar examples?

Perspectives

Scalability

- Quantized neural networks?
- Parallelization: Partitions of problem representations.

Applicability

- Tool development.
- General approach with large application domain.
- Thrombose prediction with proteomics and clinical dataset (David Tregouet, Inserm, Bordeaux).