Robust Explainable AI: the Case of Counterfactual Explanations

27 October 2023

Francesco Leofante
f.leofante@imperial.ac.uk
About me

Imperial College Research Fellow
Centre for Explainable AI

Contacts:

• f.leofante@imperial.ac.uk
• https://fraleo.github.io/
• @fraleofante
Agenda

• Explainable AI

• Counterfactual explanations and recourse

• Robustness
  • what does it mean?
  • why is it needed?
  • how can we achieve it?
Explainable AI (XAI)

XAI methods span a wide range of topics within AI and beyond, e.g.

- automated planning
- machine learning
- human computer interaction
Explainable AI (XAI)

XAI methods span a wide range of topics within AI and beyond, e.g.

- automated planning
- machine learning
- human computer interaction

Today we will focus on explaining deep neural networks (DNNs)

- **high-level** concepts rather than specific algorithms
- **fictional** use case and explanations


**Supervised learning**

**Training set**

- Age: 25
  - Amount: £40K
  - Duration: 36M
  - **denied**

- Age: 32
  - Amount: £20K
  - Duration: 24M
  - **accepted**

- Age: 82
  - Amount: £26K
  - Duration: 34M
  - **denied**

- Age: 54
  - Amount: £14K
  - Duration: 24M
  - **accepted**
Supervised learning

Training set

• Age: 25
  • Amount: £40K
  • Duration: 36M
  - denied

• Age: 32
  • Amount: £20K
  • Duration: 24M
  - accepted

• Age: 82
  • Amount: £26K
  • Duration: 34M
  - denied

• Age: 54
  • Amount: £14K
  • Duration: 24M
  - accepted

Deep neural network
(using your favourite algorithm)
Supervised learning

Deep neural network
(using your favourite algorithm)

Training set

- Age: 25
  - Amount: £40K
  - Duration: 36M
  - Predicted class: denied

- Age: 32
  - Amount: £20K
  - Duration: 24M
  - Predicted class: accepted

- Age: 82
  - Amount: £26K
  - Duration: 34M
  - Predicted class: denied

- Age: 54
  - Amount: £14K
  - Duration: 24M
  - Predicted class: accepted

New instance

Predicted class: denied
Supervised learning

Focus: explaining model predictions

Training set

• Age: 25
• Amount: £40K
• Duration: 36M

• Age: 32
• Amount: £20K
• Duration: 24M

• Age: 82
• Amount: £26K
• Duration: 34M

• Age: 54
• Amount: £14K
• Duration: 24M

New instance

• Why is it denied?
• Why not accepted?
• How do I get accepted?
• And many more questions…
Challenge

- Age: 30
- Amount: £15K
- Duration: 24M

DNNs are black boxes!

Loan denied
Challenge

• Age: 30
• Amount: £15K
• Duration: 24M

DNNs are black boxes!

Post-hoc explainability: counterfactual explanations
Counterfactual explanations (CXs)

Original instance

• Age: 30
• Amount: £15K
• Duration: 24M

Loan denied
Counterfactual explanations (CXs)

Original instance
- Age: 30
- Amount: £15K
- Duration: 24M

Loan denied

Counterfactual explanation
- Age: 30
- Amount: £10K
- Duration: 24M

The application would have been accepted had you asked for £10K instead of £15K
Computing a CX

• Given an input $x_F$ and a binary classifier $\mathcal{M}$ such that $\mathcal{M}(x_F) = c$

• A distance function $d$
Computing a CX

- Given an input $x_F$ and a binary classifier $\mathcal{M}$ such that $\mathcal{M}(x_F) = c$
- A distance function $d$

A counterfactual explanation $x$ is computed as:

$$\arg \min \limits_{x} d(x_F, x)$$

subject to $\mathcal{M}(x) = 1 - c$
Computing a CX

Most approaches solve relaxation defined as:

$$\arg\min_x \ell(M(x), 1 - c) + \lambda \cdot d(x_F, x)$$
Computing a CX

Most approaches solve relaxation defined as:

$$\arg\min_x \ell(M(x), 1 - c) + \lambda \cdot d(x_F, x)$$

where:

- $\ell$ is a differentiable loss function which minimises the gap between current and desired prediction
Computing a CX

Most approaches solve relaxation defined as:

$$\arg \min_x \ell(\mathcal{M}(x), 1 - c) + \lambda \cdot d(x_F, x)$$

where:

- $\ell$ is a differentiable loss function which minimises the gap between current and desired prediction
- $\lambda$ controls distance trade-off
Is minimising distance always good?

CXs are often indistinguishable from adversarial examples!

Brittle explanations ahead!

Threats
1. Model perturbations
2. Model multiplicity
3. Noisy execution
Robust XAI

Threats
1. Model perturbations
2. Model multiplicity
3. Noisy execution

Rethinking CX algos to mitigate these risks.
Brittle explanations ahead!

Threats
1. Model perturbations
2. Model multiplicity
3. Noisy execution
Model perturbations

\( t_0 \)
Model perturbations

t_{0}
Model perturbations

$t_0$  $t_1$
Model perturbations

\[ t_0 \quad t_1 \quad t_n \]
Model perturbations

$t_0$  $t_1$  $t_n$
Model perturbations

t_0 \rightarrow t_1 \rightarrow t_n \rightarrow t_{n+1}
Model perturbations

$t_0$ $t_1$ $t_n$ $t_{n+1}$
Model perturbations

$t_0$ $t_1$ $t_n$ $t_{n+1}$

DENIED
Implications

Model shifts may occur as a result of data shifts
Implications

Model shifts may occur as a result of data shifts

Dilemma
Implications

Model shifts may occur as a result of data shifts

Dilemma

• **Trust** the old CX, although possibly contradicted by new data

accepted
Implications

Model shifts may occur as a result of data shifts

Dilemma

• **Trust** the old CX, although possibly contradicted by new data

• **Trash** the old CX, possibly upsetting end users
Our solution

We use interval abstractions to obtain formal robustness guarantees.
Our solution

We use interval abstractions to obtain formal robustness guarantees.

A **model shift** $S$ is a function mapping an DNN into another one s.t.

- the two DNNs have same topology and,
- their differences (in parameter space) are bounded.
Our solution

We use interval abstractions to obtain formal robustness guarantees.

A **model shift** $S$ is a function mapping an DNN into another one s.t.

- the two DNNs have same topology and,
- their differences (in parameter space) are bounded.

Define set of **plausible model shifts** as:

$$\Delta = \{ S \mid \| M - S(M) \| \leq \delta \}$$
Our solution

• Plausible model shifts induce a family of DNNs…
• Need a way to reason about them concisely!
Our solution

- Plausible model shifts induce a family of DNNs...
- Need a way to reason about them concisely!

Enter the interval neural network $\mathcal{I}$
Our solution
Our solution
Our solution

Robustness decreases with shift magnitude - for robust methods as well!
Our solution

Robustness of base methods increased - 100% in some cases.
Brittle explanations ahead!

Threats
1. Model perturbations
2. Model multiplicity
3. Noisy execution
Model multiplicity

Situation where models of equal accuracy differ in the process by which they reach a given prediction.
Model multiplicity

- Age: 30
- Amount: £15K
- Duration: 24M
Model multiplicity

- Age: 30
- Amount: £15K
- Duration: 24M
Model multiplicity

- Age: 30
- Amount: £10K
- Duration: 24M
Model multiplicity

- Age: 30
- Amount: £10K
- Duration: 24M
Model multiplicity

- Age: 30
- Amount: £10K
- Duration: 24M
Implications

• Disagreeing models might raise concerns about the **justifiability** of CXs

• Different models might offer **better/worse recourse** options

Increase by £50

That’s not enough!

Erm, I’ll leave you alone now…
Our solution

We use tools from **relational verification**.

- Introduce a **novel product construction** tailored for the problem.
- Use this construction to **study the complexity** of generating robust CFXs under model multiplicity.
- Propose an approach to **generate robust CFXs** via MILP.
Sequential products

\[\begin{align*}
i &:= 0; \\
\text{while } (i < N) \text{ do} & \\
& \quad j := N - 1; \\
& \quad \text{while } (j > i) \text{ do} \\
& \quad \quad \text{if } (a[j - 1] > a[j]) \text{ then} \\
& \quad \quad \quad \text{swap}(a, j, j - 1); \\
& \quad \quad j-- \\
& \quad i++
\end{align*}\]

Program c

Sequential products

\[
i := 0; \\
\text{while } (i < N) \text{ do} \\
\quad j := N - 1; \\
\quad \text{while } (j > i) \text{ do} \\
\quad \quad \text{if } (a[j-1] > a[j]) \text{ then} \\
\quad \quad \quad \text{swap}(a, j, j - 1); \\
\quad \quad j--; \\
\quad i++; \\
\]

\[
i := 0; \\
\text{while } (i < N) \text{ do} \\
\quad j := N - 1; \\
\quad \text{while } (j > i) \text{ do} \\
\quad \quad \text{if } (a[j-1] > a[j]) \text{ then} \\
\quad \quad \quad \text{swap}(a, j, j - 1); \\
\quad \quad j--; \\
\quad i++; \\
\]

Program c

\[
i := 0; \\
\text{while } (i < N) \text{ do} \\
\quad j := N - 1; \\
\quad \text{while } (j > i) \text{ do} \\
\quad \quad \text{if } (a[j-1] > a[j]) \text{ then} \\
\quad \quad \quad \text{swap}(a, j, j - 1); \\
\quad \quad j--; \\
\quad i++; \\
\]

Program c’

\[
i := 0; \quad i' := 0; \\
\text{while } (i < N) \text{ do} \\
\quad j := N - 1; \quad j' := N - 1; \\
\quad \text{while } (j > i) \text{ do} \\
\quad \quad \text{if } (a[j-1] > a[j]) \text{ then} \\
\quad \quad \quad \text{swap}(a, j, j - 1); \\
\quad \quad \text{if } (a'[j'-1] > a'[j']) \text{ then} \\
\quad \quad \quad \text{swap}(a', j', j' - 1); \\
\quad \quad j--; \quad j'--; \\
\quad i++; \quad i'++; \\
\]

Product program P

\hspace{1cm} \times \hspace{1cm} \equiv

\text{Example taken from: Relational Verification Using Product Programs. Barthe et al, FM’11.}
Sequential products

\[\begin{align*}
\text{Program } c & : \\
i & := 0; \\
& \text{while } (i < N) \text{ do} \\
& \quad j := N - 1; \\
& \quad \text{while } (j > i) \text{ do} \\
& \quad \quad \text{if } (a[j - 1] > a[j]) \text{ then} \\
& \quad \quad \quad \text{swap}(a, j, j - 1); \\
& \quad \quad j -- \\
& \quad i ++ \\
\end{align*}\]

\[\begin{align*}
\text{Program } c' & : \\
i & := 0; \\
& \text{while } (i < N) \text{ do} \\
& \quad j := N - 1; \\
& \quad \text{while } (j > i) \text{ do} \\
& \quad \quad \text{if } (a[j - 1] > a[j]) \text{ then} \\
& \quad \quad \quad \text{swap}(a, j, j - 1); \\
& \quad \quad j -- \\
& \quad i ++ \\
\end{align*}\]

\[\begin{align*}
\text{Product program } P & : \\
i & := 0; \quad i' := 0; \\
& \text{while } (i < N) \text{ do} \\
& \quad j := N - 1; \quad j' := N - 1; \\
& \quad \text{while } (j > i) \text{ do} \\
& \quad \quad \text{if } (a[j - 1] > a[j]) \text{ then} \\
& \quad \quad \quad \text{swap}(a, j, j - 1); \\
& \quad \quad \text{if } (a'[j' - 1] > a'[j']) \text{ then} \\
& \quad \quad \quad \text{swap}(a', j', j' - 1); \\
& \quad \quad j --; \quad j'-- \\
& \quad i ++; \quad i'++ \\
\end{align*}\]
Additionally, we use omitted due to space constraints and can be found in the is constructed. A proof of equivalence of (P1) and (P2) is mean that ruling it out as a potential CFX. Namely, classification outcome of some model is undefined for Intuitively, (P1) and (P2) are equivalent. Then (P1) holds for each such that is a robust counterfactual for . Hence, we obtain that is a CFX for .

Proof. From (P2) we obtain (P1). Let be a robust counterfactual for . Then (P1) holds. Since checking the latter is NP-complete and the product extending the above upper bound to sets of arbitrary models representing piecewise linear functions. For such sets of models checking the existence of a robust counterfactual is NP-complete.

Our solution

A pictorial representation of and vice versa. Function and is parameterised by the matrix . The weights matrix is the block diagonal and uses identity activation function and is parameterised by the matrix . We set the bias vector in its and uses identity activation function and is parameterised by the matrix .

The input layer of is constructed. A proof of equivalence of (P1) and (P2) is mean that ruling it out as a potential CFX. Namely, classification outcome of some model is undefined for Intuitively, (P1) and (P2) are equivalent. Then (P1) holds for each such that is a robust counterfactual for . Hence, we obtain that is a CFX for .

Proof. From (P2) we obtain (P1). Let be a robust counterfactual for . Then (P1) holds. Since checking the latter is NP-complete and the product extending the above upper bound to sets of arbitrary models representing piecewise linear functions. For such sets of models checking the existence of a robust counterfactual is NP-complete.
Our solution

Property of the product

(P1) $v = 0$ and $u^j > 0$ for all $j \in \{1, \ldots, n\}$

(P2) $x'$ is a robust counterfactual for $x$ across $\mathcal{M}$. 
Our solution

Result #1:

**Thm.** Determining whether there exists a robust counterfactual for a set of structurally equivalent piece-wise linear models is NP-complete.
Our solution

Result #1:

*Thm.* Determining whether there exists a robust counterfactual for a set of structurally equivalent piece-wise linear models is NP-complete.

Result #2:

*Thm.* Determining whether there exists a robust counterfactual for a set of piece-wise linear models is NP-complete.
Our solution

Result #1:

**Thm.** Determining whether there exists a robust counterfactual for a set of structurally equivalent piece-wise linear models is NP-complete.

Result #2:

**Thm.** Determining whether there exists a robust counterfactual for a set of piece-wise linear models is NP-complete.

Result #3:

- The product network is itself a neural network
- We extend standard MILP encodings for CFX computation to generate robust CFXs under model multiplicity.
Brittle explanations ahead!

Threats
1. Model perturbations
2. Model multiplicity
3. Noisy execution
Noisy execution

- Age: 30
- Amount: £15K
- Duration: 24M
Noisy execution

- Age: 30
- Amount: £15K
- Duration: 24M

- Age: 30
- Amount: £10K
- Duration: 24M
Noisy execution

- Age: 30
- Amount: £15K
- Duration: 24M

- Age: 30
- Amount: £10K
- Duration: 24M

- Age: 30
- Amount: £9.9K
- Duration: 24M
Noisy execution

- Age: 30
- Amount: £15K
- Duration: 24M

- Age: 30
- Amount: £10K
- Duration: 24M

- Age: 30
- Amount: £9.9K
- Duration: 24M

DENIED
Implications

Recourses are often noisily implemented in real-world settings

- Noise may **invalidate** CX
- **Jeopardise** explanatory function
- **Reduce** trust

Our solution

We proposed to use formal verification to identify robust CXs
Our solution

We proposed to use formal verification to identify robust CXs

• Given a CX $x$ and model $M$
Our solution

We proposed to use formal verification to identify robust CXs

- Given a CX $x$ and model $\mathcal{M}$
- Check **local robustness** of $\mathcal{M}$ around $x$ using verifiers
Our solution

We proposed to use formal verification to identify robust CXs

- Given a CX $x$ and model $\mathcal{M}$
- Check **local robustness** of $\mathcal{M}$ around $x$ using verifiers
Our solution

We proposed to use formal verification to identify robust CXs

• Given a CX $x$ and model $M$

• Check **local robustness** of $M$ around $x$ using verifiers

• CX **guaranteed to be robust** when safe radius identified
Summing up

• CX generation methods focus on **minimising distance**

• This may result in **brittle explanations**

• We have examined **lack of robustness** in three scenarios:
  • model shifts, model multiplicity and noisy execution

• Can we borrow ideas from other areas of CS to fix this?
Thank you!

Contacts:

• f.leofante@imperial.ac.uk
• https://fraleo.github.io/
• @fraleofante