Differentiable Inductive Logic Programming (∂ILP) for Fraud Detection

Boris Wolfson, Dr. Erman Acar (Data Science Master thesis)

Universiteit van Amsterdam

∂ILP

- *"Learning Explanatory Rules from Noisy Data"*, Richard Evans, Edward Grefenstette
- Learns set of rules $\alpha \leftarrow \alpha_1, \ldots, \alpha_m$
- Required input: Positive, Negative examples and a set of facts
- Neurosymbolic AI - Deep NN + Symbolic Reasoning

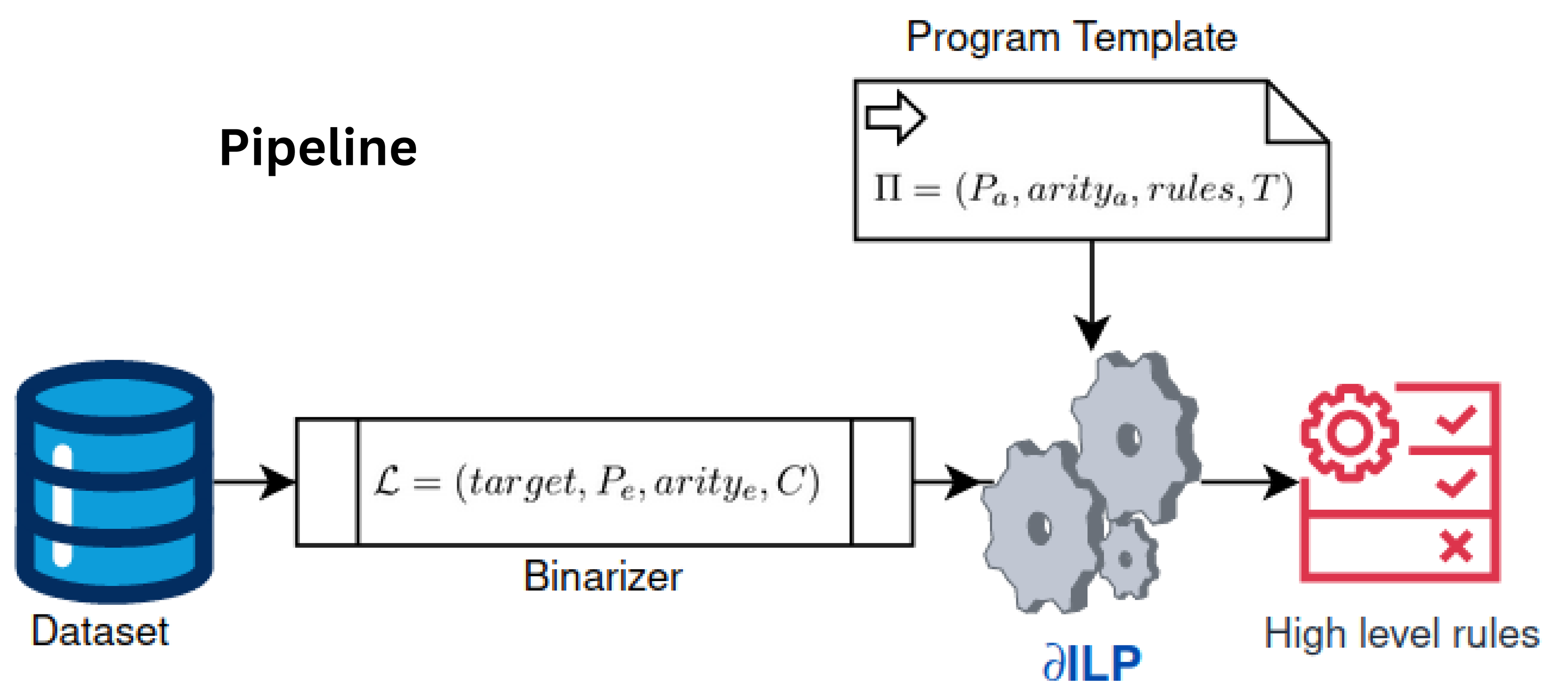
Objectives:

- Compare performance to classical rule generation methods: Deep Symbolic Regression, Decision Tree
- Test Recursive Structures for the fraudulent relationships detection

Contributions

- Pipeline for converting tabular dataset to required input format
- Set of templates for different programs

Pipeline



Program Template

- Rules (τ_p^1, τ_p^2) define predicate p template
- $\tau = (n_\exists, int)$ defines the range of clauses *C* to generate, each clause consists of two atoms
- $n_\exists$ Number of existential predicates
- *int* A flag to use an intensional (auxiliary) predicate in generated clauses
- rules are defined for each predicate P
- *arity* $_a$ The arity of an auxiliary predicate P_a
- Generates a set of clauses:

$$\begin{array}{cccc} C_p^{1,2}, C_p^{1,1} & C_p^{1,2}, C_p^{1,2} & \ldots & C_p^{1,2}, C_p^{|cl(\tau_p^1)|,1} \\ C_p^{2,2}, C_p^{1,1} & & \ldots & . \\ \ldots & & \ldots & . \\ C_p^{|cl(\tau_p^2)|,2}, C_p^{1,1} & \ldots & \ldots & C_p^{|cl(\tau_p^2)|,2}, C_p^{|cl(\tau_p^1)|,1} \end{array}$$

Induction as Satisfiability

For each P_a, *∂ILP* learns a weight matrix $\boldsymbol{W}_p$, *to find* a set of clauses best explaining Positive, Negative instances of a Target predicate

Inference Steps

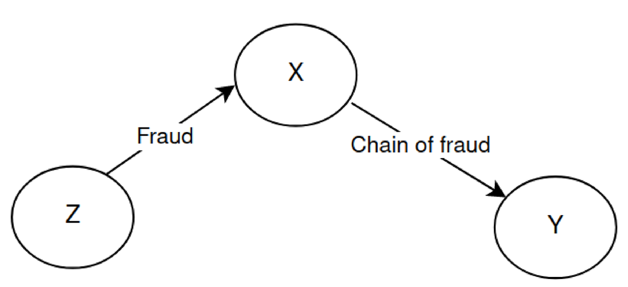
edge(a,b) connected$(X,Y) \leftarrow$ edge(X,Y)
edge(b,c) connected$(X,Y) \leftarrow$ edge(X,Z), connected(Z,Y)
edge(c,a)

$$\begin{aligned} C_{R,1} &= \{\text{edge}(a,b), \text{edge}(b,c), \text{edge}(c,a)\} \\ C_{R,2} &= C_{R,1} \cup \{\text{connected}(a,b), \text{connected}(b,c), \text{connected}(c,a)\} \\ C_{R,3} &= C_{R,2} \cup \{\text{connected}(a,c), \text{connected}(b,a), \text{connected}(c,b)\} \\ C_{R,4} &= C_{R,3} \cup \{\text{connected}(a,a), \text{connected}(b,b), \text{connected}(c,c)\} \end{aligned}$$

Examples

Chain of fraud

- Transaction(X,Y) - X sends transaction to Y
- Fraud(Z,X) - Transaction from Z to X is Fraud
- Fraud_Chain(X,Y) - X and Y are in a chain of a fraud event
- Template $\tau^1_{target} = (n_\exists = 1, int = 0)$
- Rule $Fraud\_Chain(X,Y) \leftarrow Fraud(Z,X), Transaction(X,Y)$



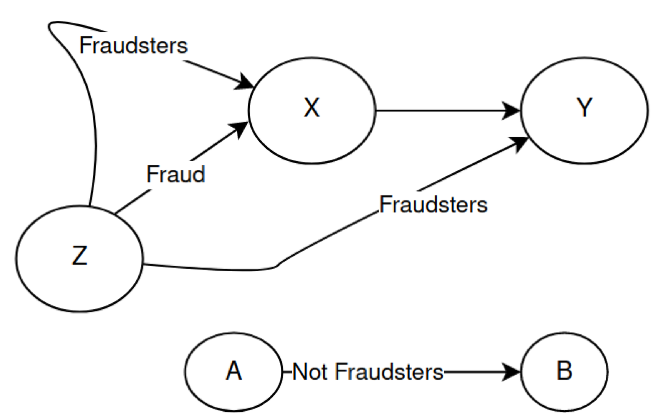
	orig	destination	Fraud_chain--orig--destination
0	16058	16066	False
1	16065	16052	False
2	16036	16067	False
3	16086	16014	True
4	16043	16004	False
5	16011	16067	True
6	16002	16011	False
7	16051	16086	False
8	16080	16077	False

Fraud Relationship

- Fraudsters(X,Y) - X and Y are Fraudsters
- Fraud(X,Y) - Transaction from X to Y is Fraud
- Template $\tau^1_{target} = (n_\exists = 0, int = 1)$

  $\tau^2_{target} = (n_\exists = 1, int = 1)$
- Rule $Fraudsters(X,Y) \leftarrow Fraud(X,Y)$

  $Fraudsters(X,Y) \leftarrow Fraud(Z,Y), Fraudsters(Z,X)$



	X	Y	Fraudsters--X--Y	Fraud--X--Y
0	1	2	True	True
1	2	3	True	True
2	3	4	True	True
3	2	1	True	True
4	1	3	True	False
5	1	4	True	False
6	1	1	True	False
7	2	4	True	False
8	2	2	True	False

PaySim Dataset

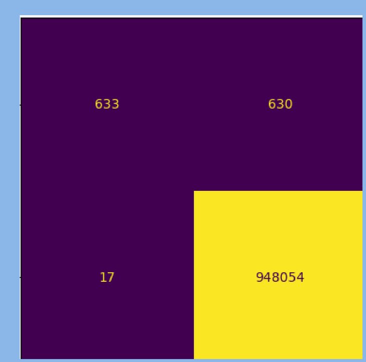
- Binarised Columns from Decision Trees Thresholds
- X - transaction id
- Template

$$\begin{aligned} \tau^1_{target} &= (n_\exists = 0, int = 1) \\ \tau^2_{target} &= (n_\exists = 0, int = 1) \\ \tau^1_{p1} &= (n_\exists = 0, int = 1) \\ \tau^2_{p1} &= None \\ \tau^1_{p2} &= (n_\exists = 0, int = 0) \\ \tau^2_{p2} &= None \end{aligned}$$

- Rule

$isFraud(X_0) \leftarrow NOT\{oldbalanceDest > -0.007\}(X_0), pred2(X_0)$
$isFraud(X_0) \leftarrow pred2(X_0), amount > 1.297(X_0)$
$pred1(X_0) \leftarrow NOT\{oldbalanceDest > -0.007\}(X_0), pred2(X_0)$
$pred2(X_0) \leftarrow external\_dest(X_0), type\_TRANSFER(X_0)$

Performance	∂ILP	DT	DSC
Accuracy	0.999	0.999	0.999
Precision	0.973	0.971	**0.984**
Recall	0.501	0.665	0.501
F1	0.662	0.789	0.664
MCC	0.698	**0.803**	0.702



Pros:

- **∂ILP** can generalize from a small amount of data; not data-hungry
- Creates recursion predicates
- Provides shorter explainable rules than Decision Tree

Cons:

- Hard to scale
- Did not outperform other techniques
- Required to define Program Template, and to convert dataset to binary dataset

Amsterdam Data Science

https://github.com/wolfbrr/ThesisDS