Learning Implicitly with Noisy Data in Linear Arithmetic

Alexander Philipp Rader,[1*] Ionela Georgiana Mocanu,[2] Vaishak Belle,[2,3] Brendan Juba[4]

[1]Department of Computing, Imperial College London, UK [2]School of Informatics, University of Edinburgh, UK [3]Alan Turing Institute, UK [4]Washington University in St. Louis, USA

alexander.rader20@imperial.ac.uk, {i.g.mocanu, vaishak}@ed.ac.uk, bjuba@wustl.edu

INTRODUCTION

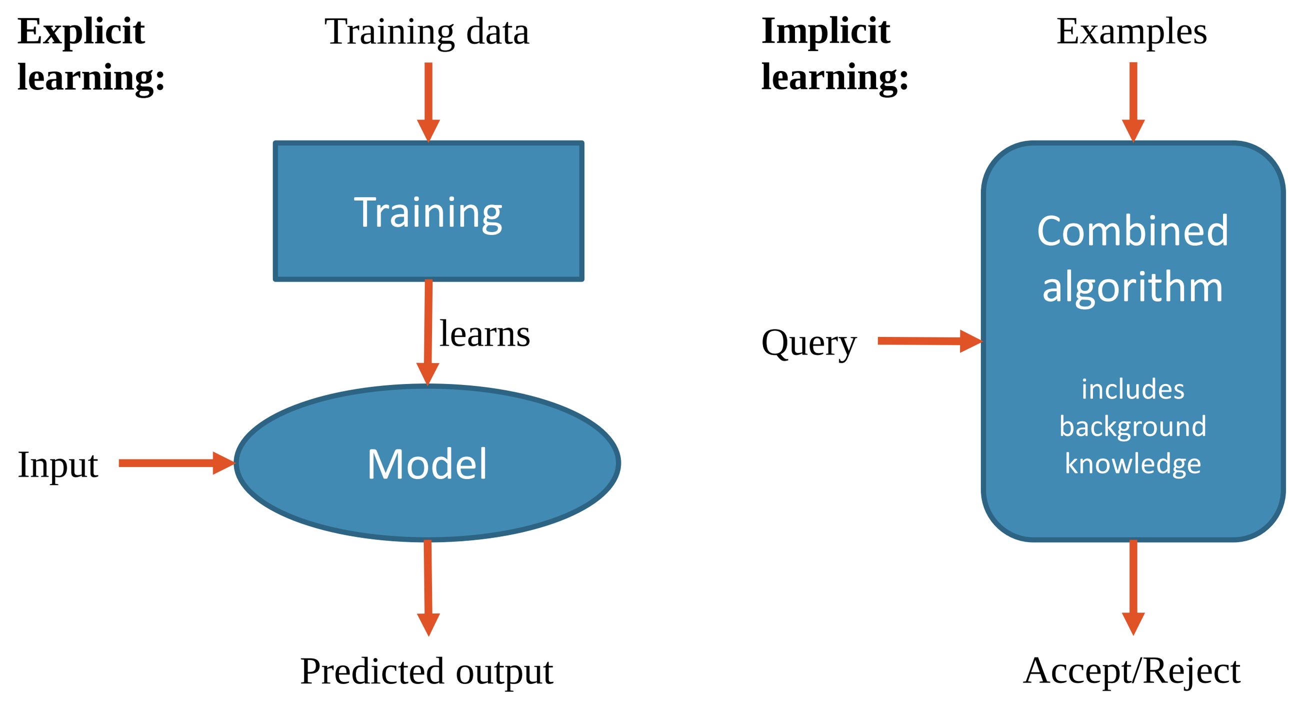
We extend an implicit learning framework to handle noisy data in the language of linear arithmetic. We prove that our extended framework keeps the existing polynomial-time complexity guarantees and provide the first empirical investigation of this hitherto purely theoretical framework.

LINEAR ARITHMETIC IN SMT

- We focus on learning in an expressive language: linear arithmetic in Satisfiability Modulo Theories (SMT)
- Quantifier-free subset of first-order logic with arithmetic operators
- E.g. $(a \geq 0) \land (b < 2a) \land (c = a + b)$
- Has polynomial-time entailment procedures

IMPLICIT LEARNING

- Learning explicit representations for SMT problems is not tractable
- Idea: Answer queries implicitly, i.e. by using examples directly
- No explicit model is created, as illustrated below



PAC-SEMANTICS

- We use the Probably Approximately Correct (PAC) Semantics framework
- Decide-PAC algorithm answers a query implicitly from examples using entailment
  - Hard-coded background knowledge Δ
  - Examples ϕ
  - Query α
  - Query accepted when $\Delta \land \phi \models \alpha$
- If entailment holds for enough examples, DecidePAC returns Accept
  - Query does not have to be fully valid, only $(1-\epsilon)$-valid. I.e. the proportion of accepted examples is at least $(1-\epsilon)$

NEW CONTRIBUTIONS

- Until now, examples had to be exact, i.e. assignments
- Idea: allow examples to be intervals, so we can handle noisy data

Theoretical contributions

- Extended the PAC-Semantics framework to accept interval-valued examples
- Proof that extended framework stays in polynomial time

Optimisation

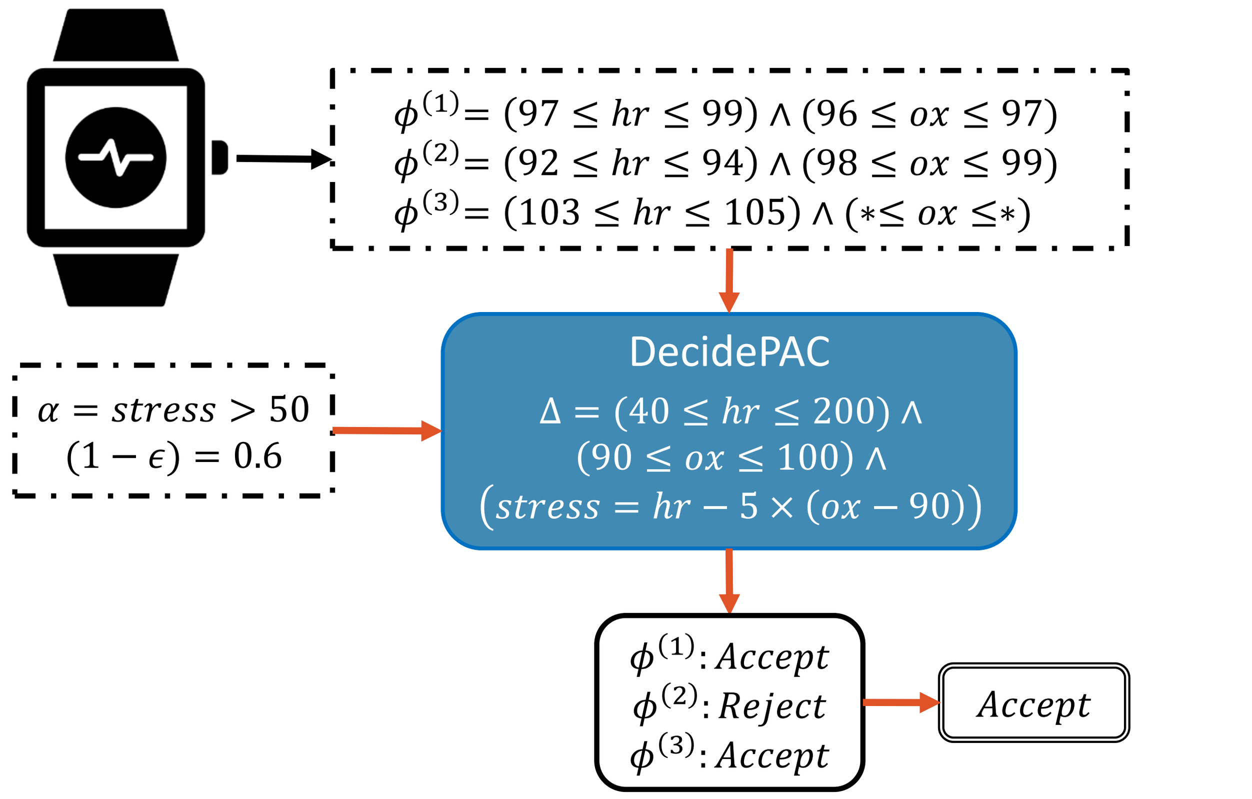
- Adapted framework to solve linear optimisation problems from examples
  - Given hard constraints, what is the optimal objective value?
- Created OptimisePAC
  - Works like exponential search
  - First we run DecidePAC repeatedly to get a rough estimate of optimal objective value
  - Then we run binary search to find optimum to desired accuracy

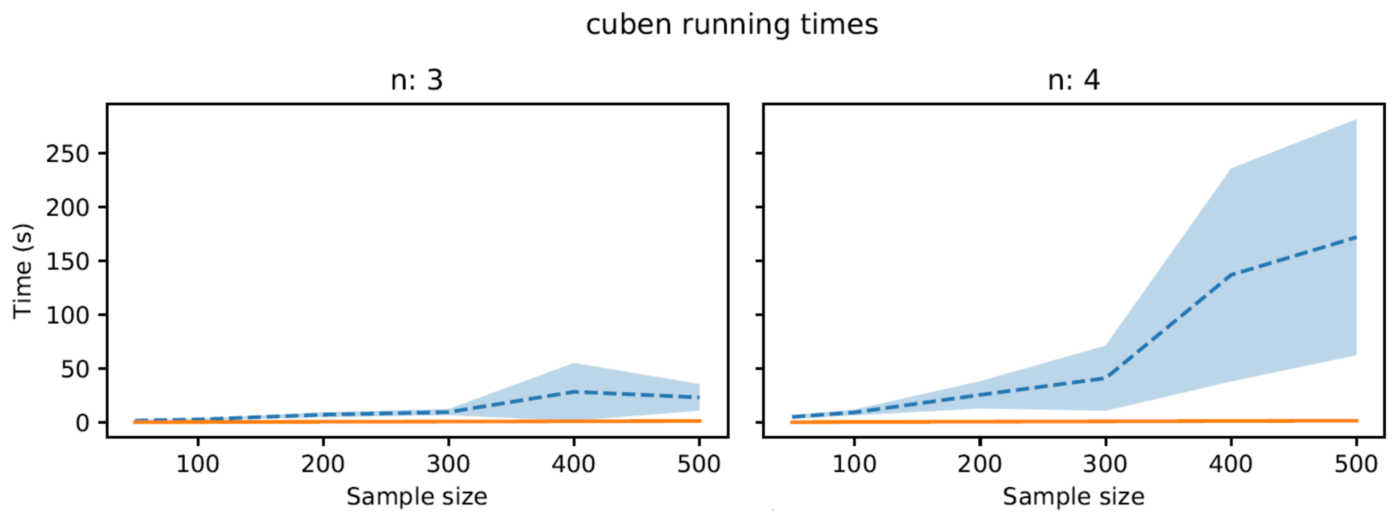
Empirical investigation

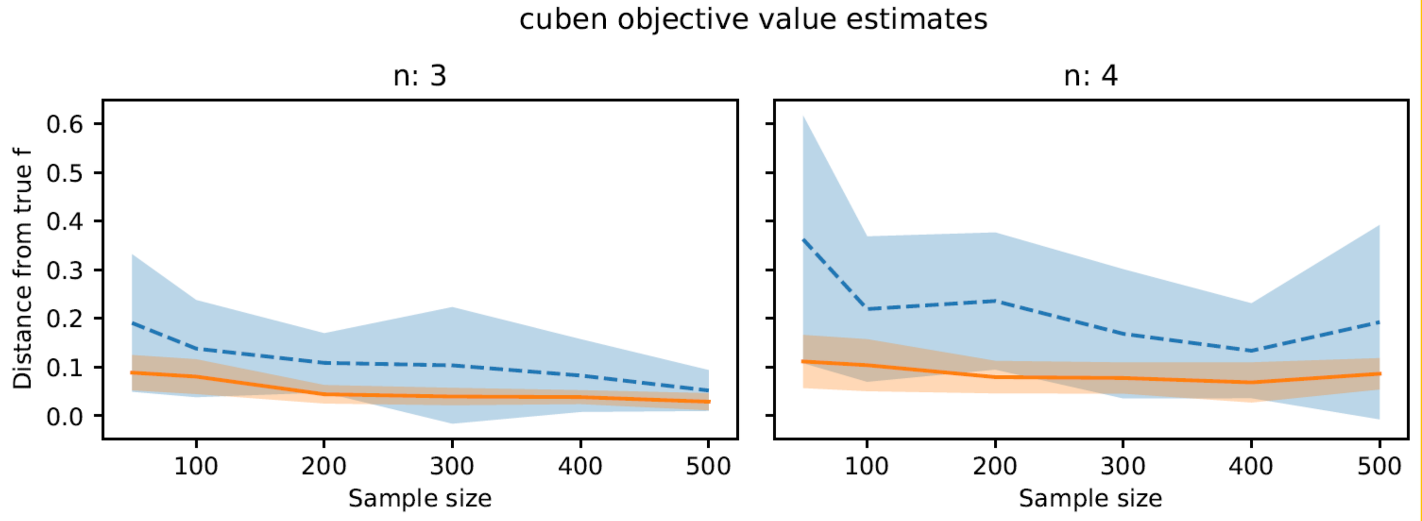
- Created first ever implementation of this framework
- Compared it with an explicit algorithm: IncaLP
  - IncaLP: Create an SMT model of the examples and then find optimum in model
  - PAC: find optimum implicitly using OptimisePAC
- Results for running time, robustness to noise and outliers are shown below

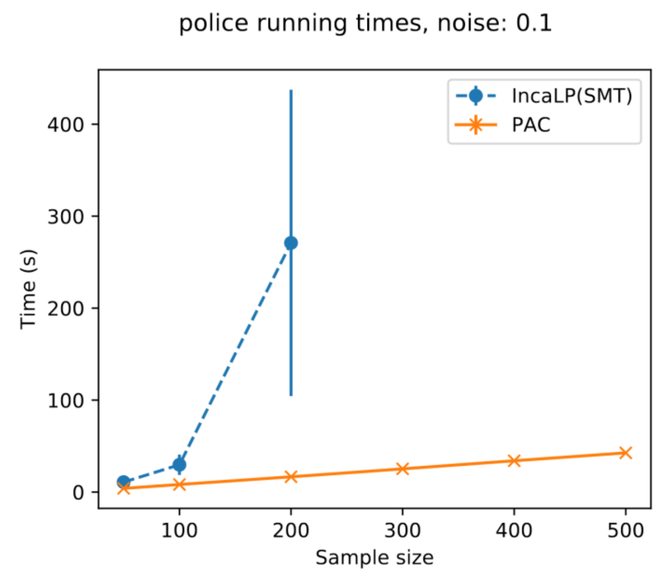
USE CASE EXAMPLE

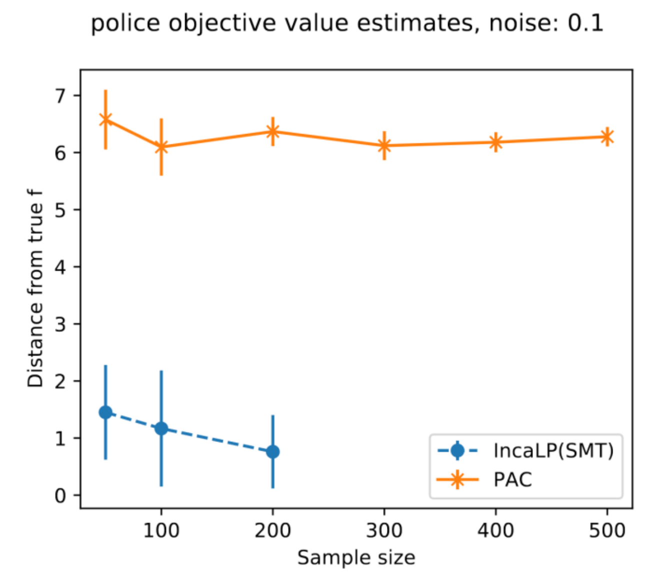
Consider a fitness watch monitoring the heart rate (hr) and blood oxygen (ox) levels of the wearer. It calculates wearer's stress level using formula: $stress = hr - 5 \cdot (ox - 90)$, which is hard-coded into its knowledge base Δ along with bounds for hr and ox. The watch alerts the user if the stress level exceeds 50, encoded as the query $\alpha = stress > 50$. The watch gets regular, but imprecise sensor readings in the form of intervals $\phi^{(k)}$. The illustration below shows that the watch answers the query using the entailment $\Delta \land \phi^{(m)} \models \alpha$ on each example, which works even when data is missing (shown as *).

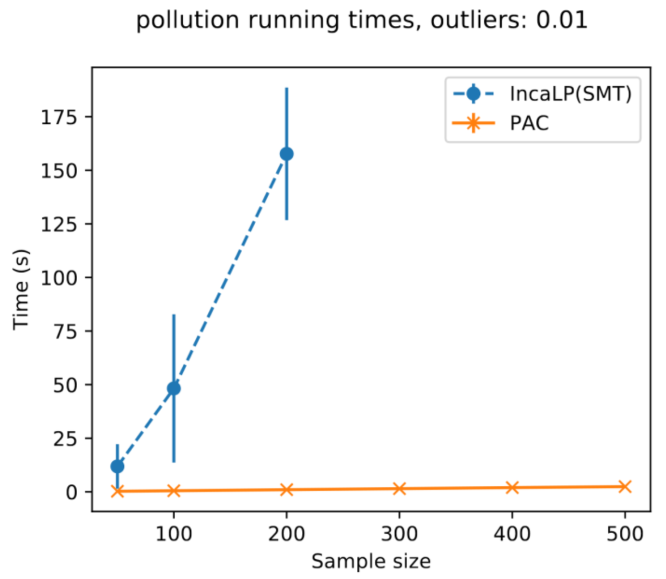


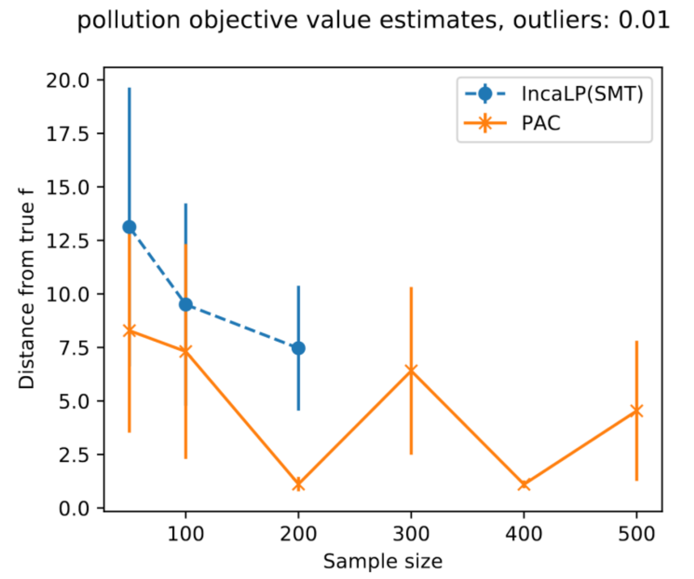












EMPIRICAL RESULTS

- For the noisy case, PAC finds similarly good estimates in significantly lower time
- Running time also grows much more slowly when increasing sample size and dimensionality
- With noise or outliers, PAC always finds an answer, while IncaLP fails to find a model in most cases
- When IncaLP finds a model, estimates can be closer to real optimum but are not always feasible
- PAC always gives feasible estimates

CONCLUSION

- We introduced ability to handle noisy data and solve optimization problem
- We have shown that skipping the step of creating an explicit model can have advantages for running time and robustness to noise and outliers
- Direction for the future: extending the framework to other classes of formulas in first-order logic and/or SMT

REFERENCES

Belle, V.; and Juba, B. 2019. Implicitly Learning to Reason in First-Order Logic. InNeurIPS.

Daniely, A.; and Shalev-Shwartz, S. 2016. Complexity theoretic limitations on learning DNF's. InCOLT, 815–830.

Gunning, D.; and Aha, D. 2019. DARPA's Explainable Artificial Intelligence (XAI) Program.AIMagazine40(2): 44–58.

Hillier, F. S.; and Lieberman, G. J. 1995.Introduction to Mathematical Programming. McGraw-Hill.

Juba, B. 2013. Implicit learning of common sense for reasoning. InIJCAI, 939–946.

Khardon, R.; and Roth, D. 1997. Learning to Reason.J. ACM44(5): 697–725.

Mocanu, I.; Belle, V.; and Juba, B. 2020. Polynomial-time Implicit Learnability in SMT. InProceedings to the 24th European Conference on Artificial Intelligence (ECAI 2020).

Schede, E. A.; Kolb, S.; and Teso, S. 2019. Learning Linear Programs from Data. In2019 IEEE 31stInternational Conference on Tools with Artificial Intelligence (ICTAI), 1019–1026.

Valiant, L. G. 2000. Robust logics. Artificial Intelligence117(2): 231–253.

ACKNOWLEDGEMENTS

Ionela G. Mocanu was supported by the Engineering and Physical Sciences Research Council (EPSRC) Centre for Doctoral Training in Pervasive Parallelism (grant EP/L01503X/1) at the University of Edinburgh, School of Informatics. Vaishak Belle was supported by a Royal Society University Research Fellowship. Brendan Juba was supported by NSF/Amazon award IIS-1939677.

*Work performed while at the University of Edinburgh.