Learning Implicitly with Noisy Data in Linear Arithmetic

Alexander Philipp Rader
Imperial College London
alexander.rader20@imperial.ac.uk

Ionela Georgiana Mocanu
University of Edinburgh
i.g.mocanu@ed.ac.uk

Vaishak Belle
University of Edinburgh & Alan Turing Institute
vaishak@ed.ac.uk

Brendan Juba
Washington University in St. Louis
bjuba@wustl.edu

Abstract

Robustly learning in expressive languages with real-world data continues to be a challenging task. Numerous conventional methods appeal to heuristics without any assurances of robustness. While PAC-Semantics offers strong guarantees, learning explicit representations is not tractable even in a propositional setting. However, recent work on so-called "implicit" learning has shown tremendous promise in terms of obtaining polynomial-time results for fragments of first-order logic. In this work, we extend implicit learning in PAC-Semantics to handle noisy data in the form of intervals and threshold uncertainty in the language of linear arithmetic. We prove that our extended framework keeps the existing polynomial-time complexity guarantees. Furthermore, we provide the first empirical investigation of this hitherto purely theoretical framework. Using benchmark problems, we show that our implicit approach to learning optimal linear programming objective constraints significantly outperforms an explicit approach in practice.

1 Introduction

Data in the real world can be incomplete, noisy and imprecise. Approaches from the knowledge representation communities take great care to represent expert knowledge; however, this knowledge can be hard to come by, challenging to formalize for non-experts, and brittle. In contrast, connectionist approaches, such as neural networks, have been particularly successful in learning from real-world data. However, they represent knowledge as distributed networks of nodes, which is neither human-readable nor very explainable (Gunning and Aha, 2019).

In this work, we are concerned with learning in expressive languages, where knowledge is represented as logical formulas. In a logical context, Valiant (2000) recognized that the challenge of learning should be integrated with deduction. He proposed a semantics to capture the quality possessed by the output of (*probably approximately correct*) PAC-learning algorithms, the PAC-Semantics. We will focus on an implicit learning approach in PAC-Semantics, where the step of creating an explicit representation is circumvented. Very recently, the learnability results have been extended to first-order clauses in (Belle and Juba, 2019), and then to fragments of *satisfiability modulo theories* (SMT) in (Mocanu, Belle, and Juba, 2020). We build upon these results in the following ways:

1. Extending the PAC-Semantics framework to be able to handle imprecise data.
2. Proving that the polynomial running time is preserved.
3. Realising the first implementation of the PAC-Semantics framework.
4. Empirically demonstrating the advantages of implicit reasoning regarding speed and noise resistance.

4th Knowledge Representation and Reasoning Meets Machine Learning Workshop (KR2ML 2020), at NeurIPS.

2 Extending the implicit PAC-Semantics framework

In this section, we extend the implicit PAC-Semantics framework from Mocanu, Belle, and Juba (2020) to allow examples to be intervals while maintaining a polynomial running time. The rationale is that real-world data is often imprecise and thus better represented as intervals rather than assignments.

Formulas are expressed in SMT, which is a generalisation of Satisfiability (SAT). It includes function symbols of the form $\{0, 1, +, -, \leq, <, \geq, >, =, \neq\}$, interpreted in the usual way over the reals. The framework is PAC-Semantics, which was introduced by Valiant (2000) to capture the quality possessed by knowledge learned from independently drawn examples from some unknown distribution D. The output produced using this approach does not express validity in the traditional (Tarskian) sense. Instead, the notion of *validity* is then defined as follows:

Definition 1: [$(1-\epsilon)$-validity (Valiant, 2000)] Given a joint distribution D over Σ^n, we say that a Boolean function f is $(1-\epsilon)$-valid if $\Pr_{\boldsymbol{\rho} \in D}[f(\boldsymbol{\rho}) = 1] \geq 1 - \epsilon$.

The reasoning problem of interest is deciding whether a query formula α is $(1-\epsilon)$ valid. Knowledge about the distribution D comes from the set of examples $\boldsymbol{\rho}$, independently drawn from this distribution and from a collection of axioms Δ which constitutes the knowledge base.

Input: Procedure A, query α, variables $\epsilon, \delta, \gamma \in (0, 1)$, list of partial intervals $\{\phi^{(1)}, ..., \phi^{(m)}\}$, knowledge base Δ

Output: *Accept* if there exists a derivation proof S of α from Δ and formulas φ_1, φ_2, ... that are simultaneously witnessed *true* with probability at least $(1 - \epsilon + \gamma)$ on $\boldsymbol{B}(D)$
Reject if $\Delta \Rightarrow \alpha$ is not $(1 - \epsilon - \gamma)$-valid under D

begin
  $B \leftarrow \lfloor \epsilon \times m \rfloor, FAILED \leftarrow 0.$
  foreach k *in* m **do**
    if $A(\alpha, \phi^{(k)}, \Delta)$ *returns UNSAT* **then**
      Increment $FAILED$.
      if $FAILED > B$ **then return Reject;**
  return Accept

Algorithm 1: DecidePAC

Unfortunately, explicitly learning even simple models like DNF formulas is believed to be intractable. (Daniely and Shalev-Shwartz, 2016). However, Khardon and Roth (1997) and Juba (2013) observed that by circumventing the need to produce an explicit representation, learning to reason can be effectively reduced to classical reasoning, leading to a notion of implicit learning. The idea is to answer the query directly using the examples, which is demonstrated in algorithm 1. Mocanu, Belle, and Juba (2020) have shown that implicit reasoning is polynomial-time for SMT queries with partial *assignments* of examples. We now show that polynomial running-time is preserved for partial *intervals* of examples. To turn complete assignments into partial intervals, we introduce the notion of a blurring process:

Definition 2: [Blurring process] Given a full assignment $\boldsymbol{\rho}^{(k)} = \{\boldsymbol{\rho_1}, \ldots, \boldsymbol{\rho_n}\}$, a *blurring* function is defined as $B : \Sigma^n \rightarrow \{\Sigma \cup \{-\infty, +\infty\}\}^{2n}$, which produces a set of intervals $\phi^{(k)}$ consistent with the assignment $\boldsymbol{\rho}^{(k)}$, i.e., with the property that for each $\boldsymbol{\rho}_i$, $B(\boldsymbol{\rho})_{2i-1} \leq \boldsymbol{\rho}_i \leq B(\boldsymbol{\rho})_{2i}$, where $B(\boldsymbol{\rho})_{2i-1}$ is a random value from $(-\infty, \boldsymbol{\rho}_i]$ - lower bound, and $B(\boldsymbol{\rho})_{2i}$ is a random value from $[\boldsymbol{\rho}_i, \infty)$ - upper bound. We refer to elements of the set Σ^{2n} as *partial intervals*, where a full assignment is bound by the lower and upper bound. A *blurring process* **B** is a blur-valued random variable (i.e. a random function).

In this way, we allow the degree of uncertainty of an observation to be given by the width of the interval in which the real value lies. This leads us to the main theorem (Note that a formula φ is *witnessed true* under partial intervals ϕ if it is true under every assignment possible under ϕ.):

Theorem 3: *[Implicit learning] Let Δ be a conjunction of constraints representing the knowledge base and an input query α. We draw at random $m = \frac{1}{2\gamma^2} ln \frac{1}{\delta}$ sets of intervals $\{\phi^{(1)}, \phi^{(2)}, ..., \phi^{(m)}\}$*

from $\boldsymbol{B}(D)$ for the distribution D and a blurring process $\boldsymbol{B}$. Suppose that we have a decision procedure A. Then with probability $1-\delta$:

- *If $(\Delta \Rightarrow \alpha)$ is not $(1-\epsilon-\gamma)$ - valid with respect to the distribution D, Algorithm 1 returns Reject; and*
- *If there exists some KB I such that $\Delta \wedge I \models \alpha$ and I is witnessed true with probability at least $(1-\epsilon+\gamma)$ on $\boldsymbol{B}(D)$, then Algorithm 1 returns Accept.*

Moreover, if A runs in polynomial-time (in the number of variables, size of query, and size of knowledge base), so does Algorithm 1.

With this theorem, we have extended the previous result to deal with the more complex setting of threshold uncertainty. The proof can be found in the appendix. However, like the previous PAC-Semantics results, the approach is geared for answering queries and has been primarily a theoretical framework. In contrast, conventional methods, although not always offering learnability guarantees, provide an explicit hypothesis. One type of problem such methods are used for is finding the optimal objective function value within hard constraints.

To solve such problems using DecidePAC, we have created the "OptimisePAC" algorithm. In essence, we find an optimal value of an objective function f by repeatedly running *DecidePAC(ϕ, bound$\geq$f,ϵ)* with different bounds. This is done in a manner similar to exponential search: we double the bound until we find an approximate area in which the objective value lies. Then we run binary search up to the desired accuracy to find much tighter bounds. The full pseudo-code can be found in the appendix, along with a proof of its functionality. We will use OptimisePAC in the next section to compare it with an explicit approach in practice.

3 Empirical analysis

We compare this line of work with an existing approach, an algorithm proposed by Schede, Kolb, and Teso (2019) that induces linear programs from data examples. Linear programming is an optimisation technique for a linear objective function. The problem consists of an optimisation function $f(x)$, where $x \in \mathbb{R}^n$, and feasibility region expressed as the set of constraints $A \cdot x \leq b$. Such linear programs can be expressed as SMT formulas.

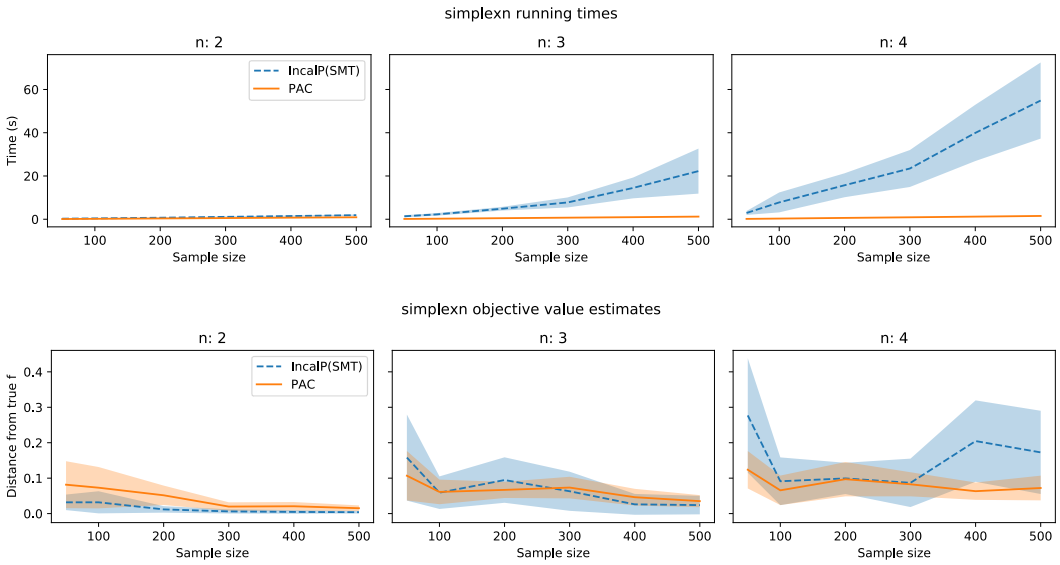
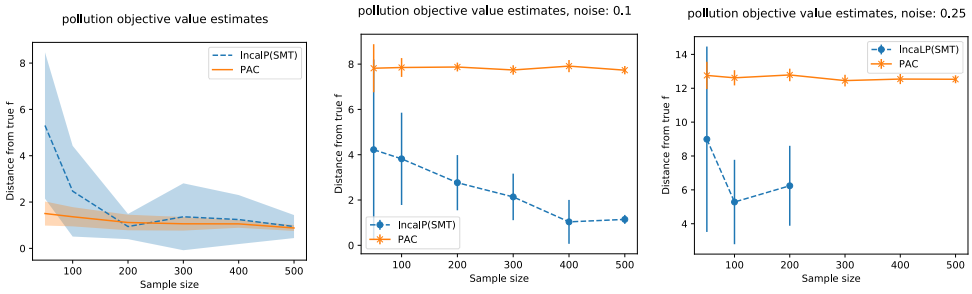


Figure 1: Running times and objective value estimates for simplexn, where n = # dimensions

We use the following benchmark and standard SMT problems for our analysis: **simplexn**, **cuben**, **pollution** and **police** (Hillier and Lieberman, 1995). All these problems consist of a set of hard constraints, which define the feasible region or the boundaries of the shape, and a set of soft constraints



(a) No noise. IncaLP always found a model.

(b) Noise with std of $\frac{0.1}{\sqrt{6}} \approx 0.04$. IncaLP could not find a model 20/60 times.

(c) Noise with std of $\frac{0.25}{\sqrt{6}} \approx 0.1$. IncaLP could not find a model 37/60 times.

Figure 2: Objective values for pollution. True minimum: 32.15.

in the form of a linear objective function. The goal is to find the optimum objective value within the hard constraints. In the case of **pollution**, for example, the objective function represents the amount of pollution and has to be minimised.

The implicit PAC approach and the explicit IncaLP approach differ in how they reach this goal: in the PAC model, we directly compute the optimal objective value from positive examples using OptimisePAC. In contrast, the IncaLP approach first creates an SMT model from positive and negative examples and then finds the optimal objective value for the model by standard MaxSMT techniques.

Results On the theoretical front, one of the advantages of implicit learning is argued to be efficiency. By skipping the step of creating an explicit model, one can give polynomial-time guarantees. As shown in Figure 1, this effect is significant in practice. PAC is able to get similarly good objective value estimates at significantly lower running times for simplexn.

With the extension we introduced in this paper, PAC can now handle noisy data using intervals. If we add Gaussian noise with a standard deviation σ to each data point, we can represent it using intervals of width $4 \log d \cdot \sigma$, where d is the dimensionality. This interval covers about 95% of the density, which is why we set a validity of 95% for DecidePAC. The noise and intervals were capped at the domain boundaries.

Figure 2 shows how PAC compares to IncaLP. We adjusted the noise to the dimensionality, meaning that for a noise value of n, the std of the Gaussian $\sigma = \frac{n}{\sqrt{d}}$. In cases where IncaLP does find a model, it gives an estimate that is closer to the true objective value. However, in 2b, IncaLP failed to find a model on 20/60 runs, and in 2c, it failed on 37/60 runs. Moreover, the PAC estimate is pessimistic on purpose. This gives values farther away from the optimum but ensures that they are reachable. PAC always finds feasible objective values, while IncaLP undershoots the minimum for pollution almost always: Even in the noiseless case, only 10% of found objective values were reachable, for cases with noise, it was 0%. We remark that the empirical behaviour we discussed are similar across the four problems. Graphs for all problems, including tests on data with outliers are in the appendix.

Note that these results do not mean that IncaLP, on the whole, is inferior to our PAC implementation. They do fundamentally different things: IncaLP creates a model, while PAC answers queries implicitly. For some problems, such as objective value optimisation, it is possible to skip the model making process. As we have shown, doing that comes with advantages in speed and noise resistance. However, in some contexts, having an explicit model is desirable, in which case the implicit learning paradigm might be harder to work with. Put simply, OptimisePAC is not a replacement for IncaLP.

4 Conclusion

We proposed a general framework for learning implicitly in linear arithmetic from noisy and imprecise examples. By considering a novel optimisation variant, we were able to empirically compare and outperform an explicit approach in terms of running time and resistance to noise and outliers. A natural direction for the future is to consider whether this implementation strategy extends to other classes of formulas in first-order logic and/or SMT.

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.

References

Belle, V.; and Juba, B. 2019. Implicitly Learning to Reason in First-Order Logic. In *NeurIPS*. URL http://arxiv.org/abs/1906.10106.

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

Gunning, D.; and Aha, D. 2019. DARPA's Explainable Artificial Intelligence (XAI) Program. *AI Magazine* 40(2): 44–58. doi:10.1609/aimag.v40i2.2850. URL https://www.aaai.org/ojs/index.php/aimagazine/article/view/2850.

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

Juba, B. 2013. Implicit learning of common sense for reasoning. In *IJCAI*, 939–946.

Khardon, R.; and Roth, D. 1997. Learning to Reason. *J. ACM* 44(5): 697–725. ISSN 0004-5411. doi:10.1145/265910.265918. URL https://doi.org/10.1145/265910.265918.

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

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

Valiant, L. G. 2000. Robust logics. *Artificial Intelligence* 117(2): 231–253. ISSN 0004-3702. doi:10.1016/S0004-3702(00)00002-3. URL http://www.sciencedirect.com/science/article/pii/S0004370200000023.

Learning Implicitly with Noisy Data in Linear Arithmetic - Appendix

1 Proofs

1.1 Implicit learning

Theorem 1: *[Implicit learning] Let Δ be a conjunction of constraints representing the knowledge base and an input query α. We draw at random $m = \frac{1}{2\gamma^2} ln \frac{1}{\delta}$ sets of intervals $\{\phi^{(1)}, \phi^{(2)}, ..., \phi^{(m)}\}$ from $\boldsymbol{B}(D)$ for the distribution D and a blurring process $\boldsymbol{B}$. Suppose that we have a decision procedure A. Then with probability $1 - \delta$:*

- *If $(\Delta \Rightarrow \alpha)$ is not $(1 - \epsilon - \gamma)$ - valid with respect to the distribution D, Algorithm 1 returns Reject; and*
- *If there exists some KB I such that $\Delta \wedge I \models \alpha$ and I is witnessed true with probability at least $(1 - \epsilon + \gamma)$ on $\boldsymbol{B}(D)$, then Algorithm 1 returns Accept.*

Moreover, if A runs in polynomial-time (in the number of variables, size of query, and size of knowledge base), so does Algorithm 1.

Proof Consider a sound and complete decision procedure A for the language domain aforementioned and the reasoning problem of deciding $\Delta \models \alpha$. By definition of soundness and completeness, $\Delta \models \alpha$ if and only if $A(\Delta \wedge \neg\alpha) = \text{UNSAT}$. Suppose we receive observations about the world as sets of blurred intervals ϕ and we wish to decide entailment of the aforementioned problem with respect to these blurred observations, hence calculate $A(\Delta|_\phi \wedge \neg\alpha|_\phi) = \text{UNSAT}$. This holds iff $A(\Delta \wedge \phi\downarrow \wedge \neg\alpha) = \text{UNSAT}$, iff $A(\Delta \wedge I \wedge \phi\downarrow \wedge \neg\alpha) = \text{UNSAT}$ for any KB I that is witnessed true under ϕ. If $\Delta \Rightarrow \alpha$ is unsatisfied on a point drawn from D, it is not entailed by the blurred example from $B(D)$ either, so FAILED increments when such points are drawn, and does not increment when a suitable I is witnessed true. By Hoeffding's inequality, the returned value satisfies the given conditions with probability $1 - \delta$ for m examples. The decision procedure will return UNSAT in polynomial time $T(n)$ depending on the size of the knowledge base and query. Every iteration costs the time for checking feasibility which is bounded by the time complexity of the decision procedure used for deciding satisfiability. The total number of iterations is $m = \frac{1}{2\gamma^2} \log \frac{1}{\delta}$, corresponding to the number of samples drawn, which gives us the total time bound of $O(T(n) \cdot \frac{1}{\gamma^2} \log \frac{1}{\delta})$. ∎

1.2 OptimisePAC

The pseudocode for the OptimisePAC algorithm is given in algorithm 1. We now prove that the algorithm will find a sufficiently close estimate of the objective value in polynomial time:

Theorem 2: *Let Δ be a conjunction of constraints representing the knowledge base and as input preference function f. We draw at random $m = O(\frac{1}{\gamma^2} \log \frac{1}{\delta})$ partial intervals $\phi^{(1)}, ..., \phi^{(m)}$ from $\boldsymbol{B}(D)$ for a distribution D and a blurring process $\boldsymbol{B}$. Suppose that we have a decision procedure A for linear programming running in time $T(n)$. Then the OptimisePAC algorithm will return a significant bits of a value v^* that is attainable on $I \wedge \Delta$ for all KBs I that are witnessed with probability $1 - \epsilon + \gamma$, and such that for the value u^* obtained by incrementing the ath bit, $\Delta \Rightarrow (f \leq u^*)$ is $(1 - \epsilon - \gamma)$-valid (resp., $f \geq u^*$ with the ath bit decreased if minimising) in time $O(T(n) \cdot m \cdot a)$.*

Proof We will use a theorem due to Talagrand(1994) (Theorem 4.9 of Anthony and Bartlett(1999)):

Input: Preference function f, validity $\epsilon \in (0,1)$, accuracy $a \in \mathbb{Z}^+$, list of intervals $\phi = \{\phi^{(1)}, \phi^{(2)}, ..., \phi^{(m)}\}$, goal $\in$ {"maximise", "minimise"}
Output: t estimated optimal value w.r.t. f
begin
  if *goal = "minimise"* **then** $f \leftarrow -f$;
  if *DecidePAC*$(\phi, 0 \geq f, \epsilon)$ *accepts* **then**
    if *DecidePAC*$(\phi, -1 \geq f, \epsilon)$ *rejects* **then**
      $l \leftarrow -1, u \leftarrow 0$
    else
      $b \leftarrow -2$
      while *DecidePAC*$(\phi, b \geq f, \epsilon)$ *accepts* **do** $b \leftarrow b \times 2$;
      $l \leftarrow b, u \leftarrow b/2$
  else
    if *DecidePAC*$(\phi, 1 \geq f, \epsilon)$ *accepts* **then**
      $l \leftarrow 0, u \leftarrow 1$
    else
      $b \leftarrow 2$
      while *DecidePAC*$(\phi, b \geq f, \epsilon)$ *rejects* **do** $b \leftarrow b \times 2$;
      $l \leftarrow b/2, u \leftarrow b$
  for *a iterations* **do**
    if *DecidePAC*$(\phi, (l+u)/2 \geq f, \epsilon)$ *accepts* **then** $u \leftarrow (l+u)/2$ **else** $l \leftarrow (l+u)/2$;
  if *goal = "minimise"* **then return** $-l$ **else return** l;

Algorithm 1: OptimisePAC

Theorem 3: *[[Anthony and Bartlett(1999), Theorem 4.9]] There are positive constants c_1, c_2, c_3 such that the following holds. Suppose that F is a set of functions defined on the domain X and that F has a finite VC dimension d. Let $\gamma \in (0,1)$ and $m \in \mathbb{Z}^+$. Then the probability that the empirical mean of any $f \in F$ on m examples differs from its expectation by more than γ is at most $c_1 c_2^d e^{-c_3 \gamma^2 m}$.*

Thus for $m \geq \frac{c_3}{\gamma^2}(d \ln c_2 + \ln \frac{c_1}{\delta})$, the bound is at most δ.

Recall, the VC dimension is the size of the largest set of points that can be given all labels by a class ("shattered"). Consider a fixed class of Boolean functions on the blurred samples, and which is parameterized by the objective value bounds b. This function outputs the value 1 whenever $\Delta \wedge \phi \wedge (f(x) \leq b)$ returns UNSAT, and 0 otherwise. We will show that this class has VC-dimension at most 1.

We will show that for any two blurred examples ϕ_1 and ϕ_2, it is not possible to get all the labellings $\{(1,1),(1,0),(0,1),(0,0)\}$ by varying b. Suppose there is some b^* for which ϕ_1 gives the label 1 and ϕ_2 gives 0, meaning that for ϕ_1 the bound $f \leq b*$ does not hold and for ϕ_2 it does. Since $f \leq b^*$ holds for ϕ_2, then for any $b > b^*$, the decision procedure will return 0 for ϕ_2. On the other hand, the bound $f \leq b^*$ will not hold for ϕ_1 for all values $b < b^*$. Thus, in either direction, one of the labels for one of the two remains the same. So, it is not possible to get all possible labellings of ϕ_1 and ϕ_2 and so the VC-dimension is ≤ 1.

Therefore, by Talagrand's bound, given m examples, with probability $1-\delta$ DecidePAC answers correctly for all queries made by OptimisePAC. In particular, we note that the algorithm maintains the invariant that l is the largest value for which $l \geq f$ was rejected by DecidePAC. Since it was not accepted, we see that for any I that is witnessed with probability $\geq 1 - \epsilon + \gamma$, there must exist some x satisfying $I \wedge \Delta$ with $f(x) > l$ (resp., $f(x) < l$ if minimising). Since DecidePAC does not reject with u, $\Delta \Rightarrow f(x) \leq u$ is $(1 - \epsilon - \gamma)$-valid, where u and l only differ by the value of the ath most significant bit. Thus l is as needed.

Finally, the decision procedure A is run each time we call the DecidePAC algorithm. This gives us the total run time stated in the theorem. ∎

2 Experimental setup

We ran tests on all four problems. For each test, we had 10 independent runs on increasing sample sizes from 50 to 500 and increasing dimensions from 2 to 4, if applicable. We have 50% positive and 50% negative

samples. To ensure reproducibility, we ran each test with the seed 111921. The hardware we used was an Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz, with 16 GB of RAM running Scientific Linux 7.6.

As for parameter settings, we chose the SelectDT heuristic for the IncaLP approach, as it is the most appropriate according to their paper. We also ran our experiments using the same initial configurations as their released code. We set the accuracy for OptimisePAC to 60, which is the number of times our intervals are divided by 2. The reason being that we can match the accuracy of double precision floating-point values, which is about 2^{-53}.

We will make the source code public after reviews close.

3 Results

Here we show all the graphs from our experiments. Note that the following hold:

- The true minimum objective value for pollution is about 32.15 and for police it is 3.37.
- The objective functions for simplexn and cuben are randomly generated and vary for each run. They are a linear combination of all variables with coefficients and a constant, each varying from -1 to 1. Thus, the true objective value f is bounded by $-(d+1) \leq f \leq d+1$, where d is the number of dimensions. The goal is to maximise that value.
- For a noise parameter n, the standard deviation σ of the Gaussian noise is adjusted by the dimensionality: $\sigma = \frac{n}{\sqrt{d}}$.
- An outlier is a data point with the wrong label, i.e. a point within the feasible region labelled as outside and vice-versa. An outlier value of 0.01 means that 1% of the points have wrong labels.
- Simplexn and cuben have dimensions ranging from 2 to 4. Pollution has 6 dimensions and police has 5.

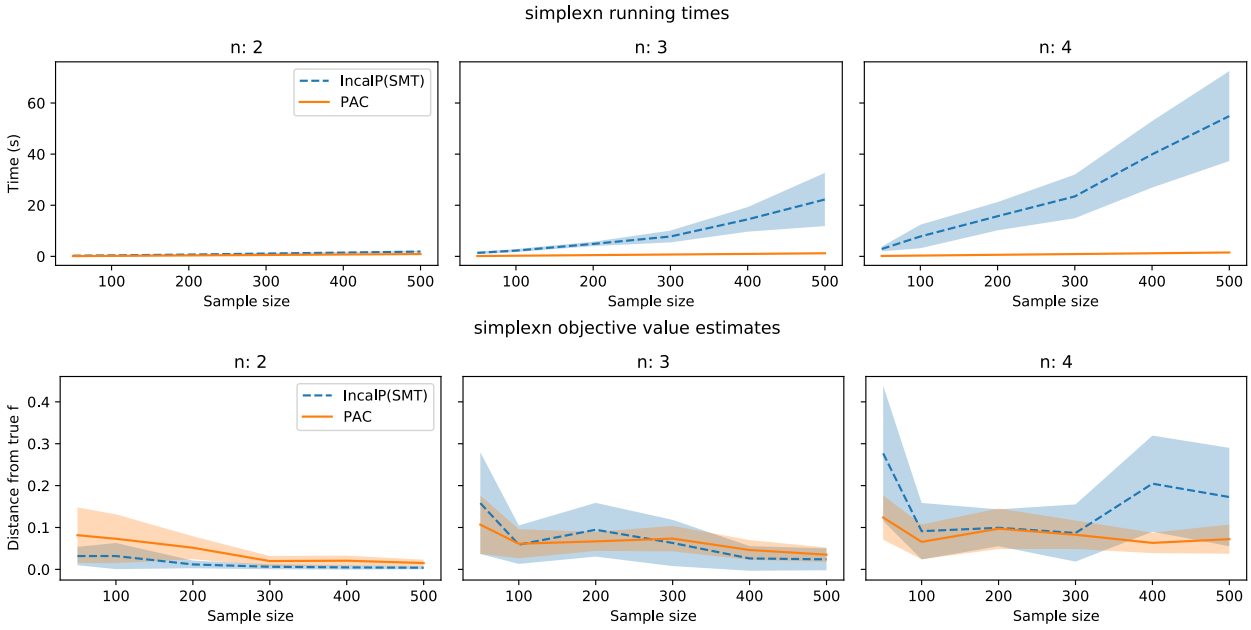


Figure 1: IncaLP always found a model.

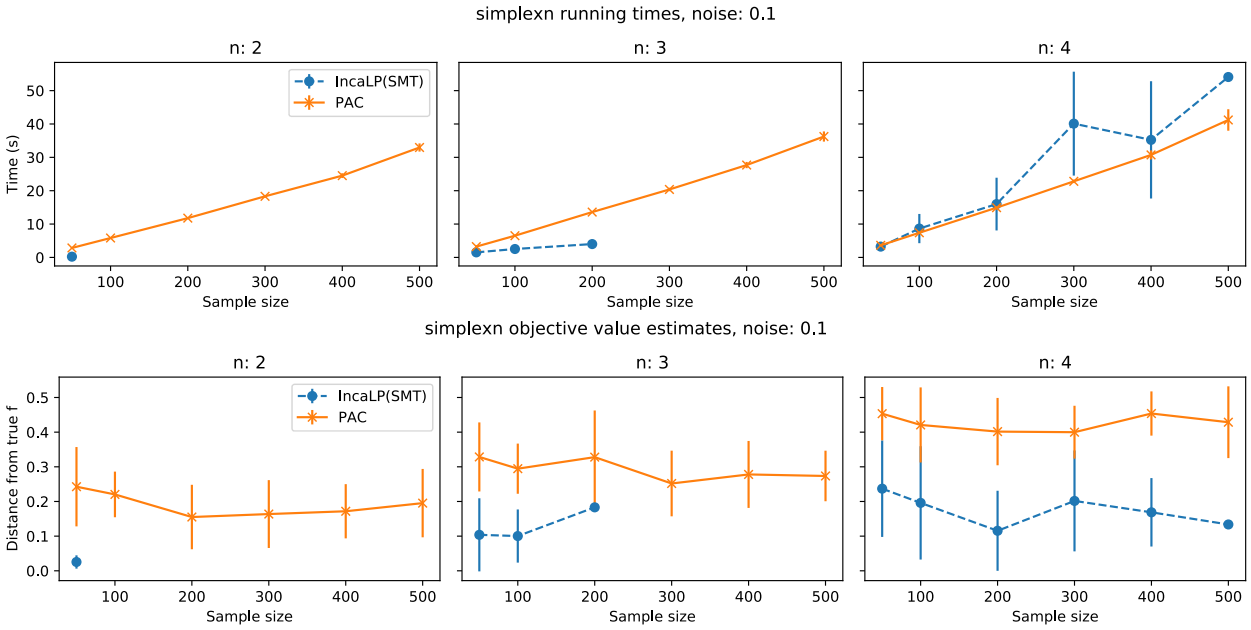


Figure 2: IncaLP failed to find a model 76% of the time.

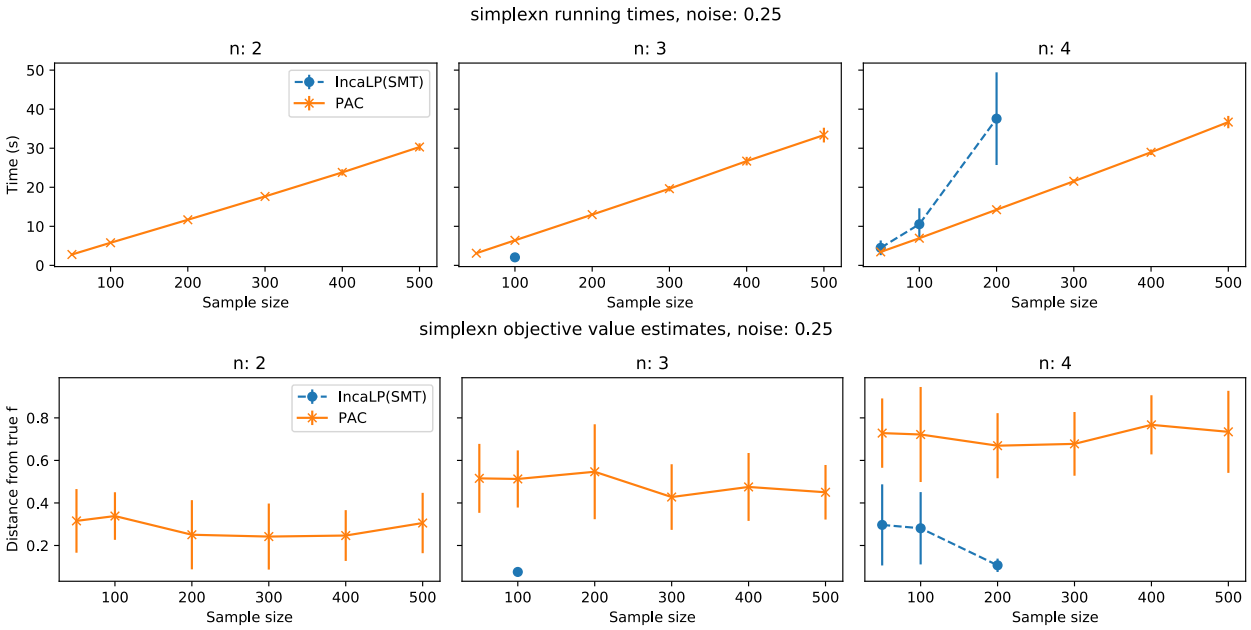


Figure 3: IncaLP failed to find a model 89% of the time.

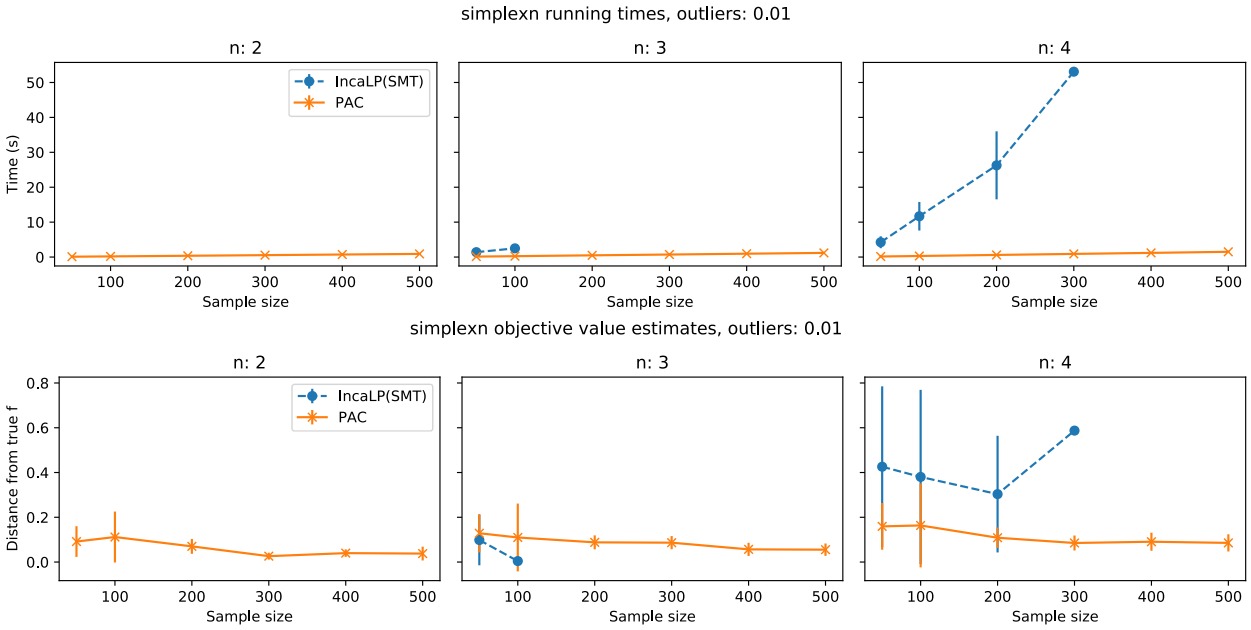


Figure 4: IncaLP failed to find a model 86% of the time.

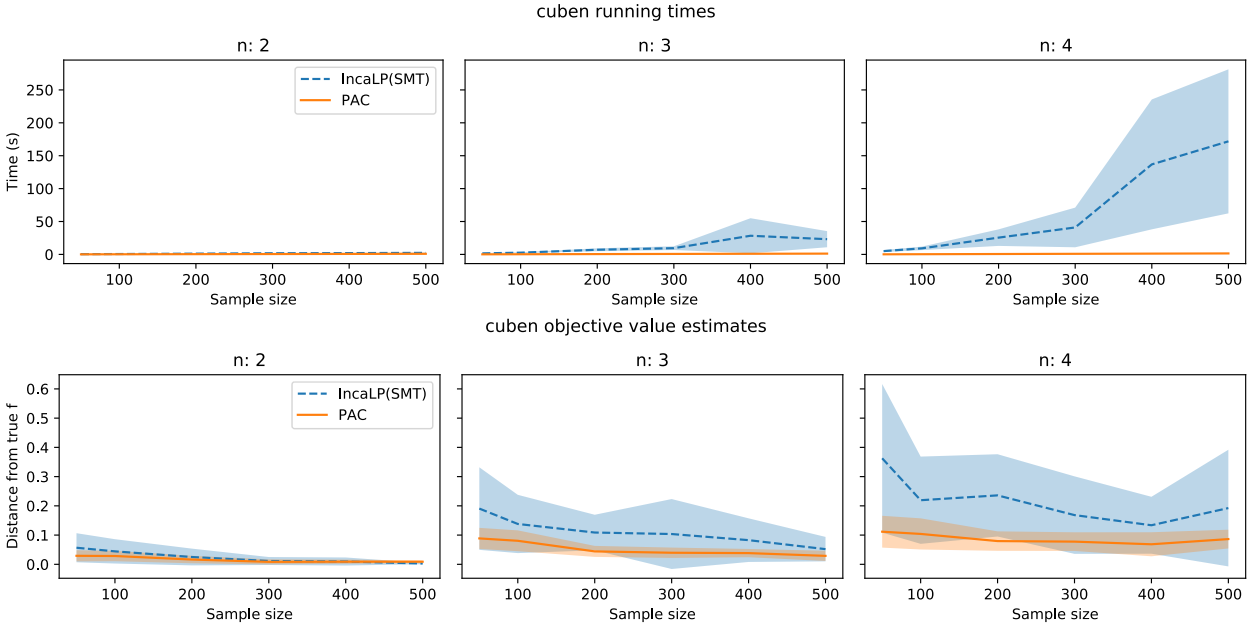


Figure 5: IncaLP always found a model.

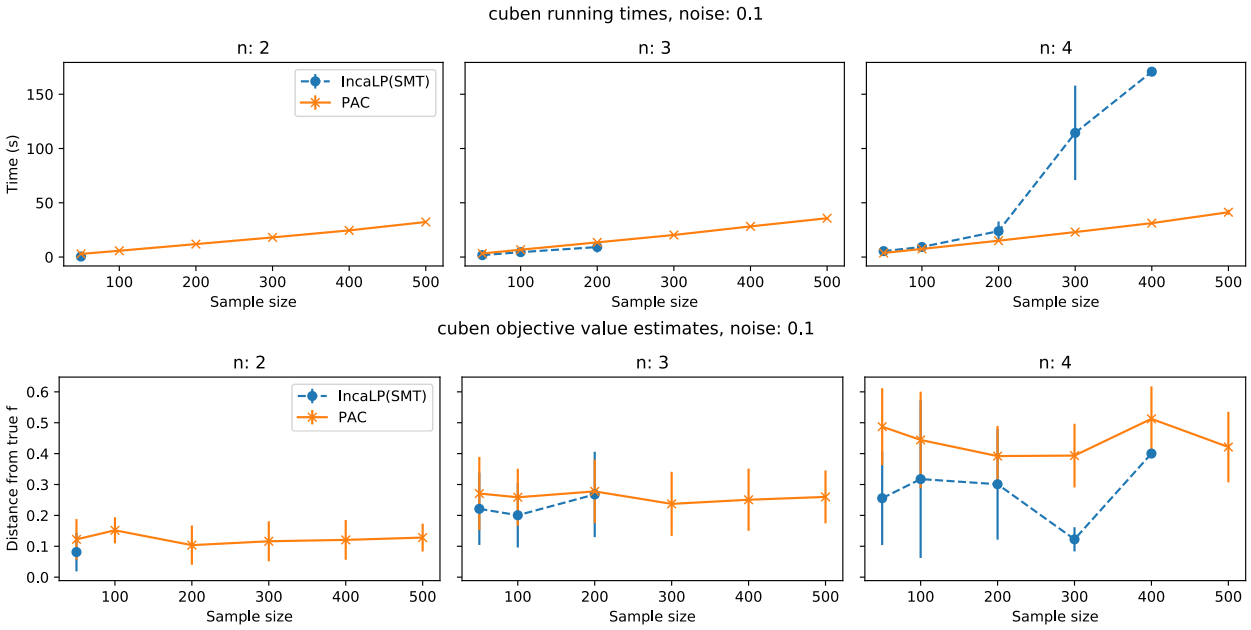


Figure 6: IncaLP failed to find a model 73% of the time.

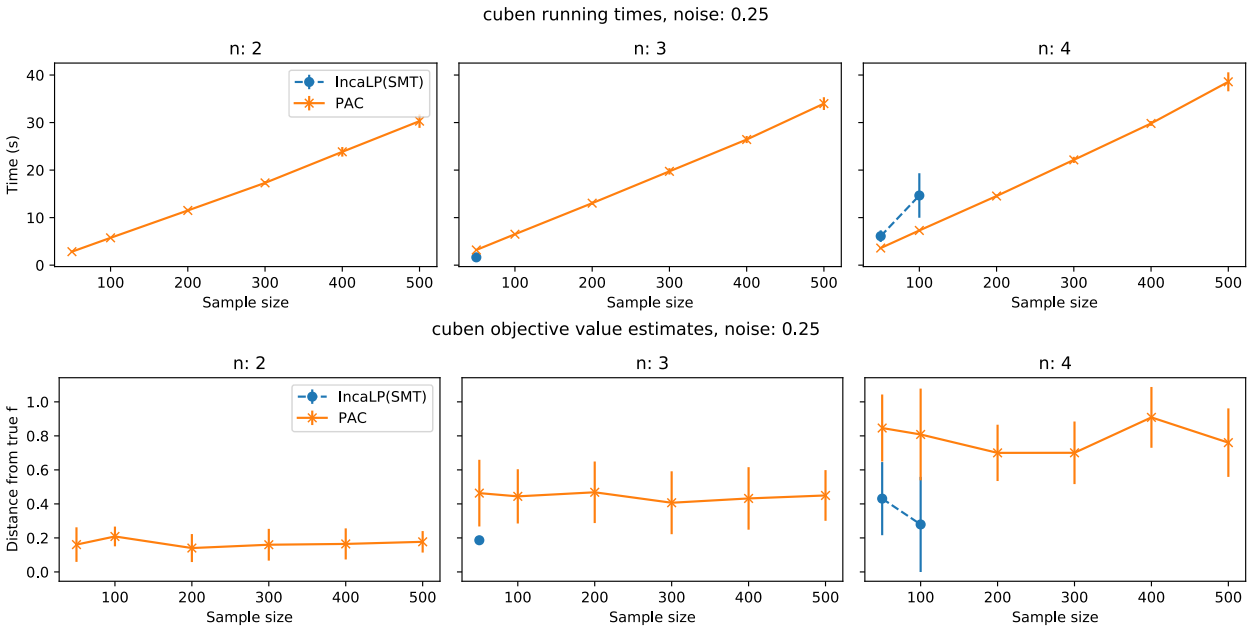


Figure 7: IncaLP failed to find a model 93% of the time.

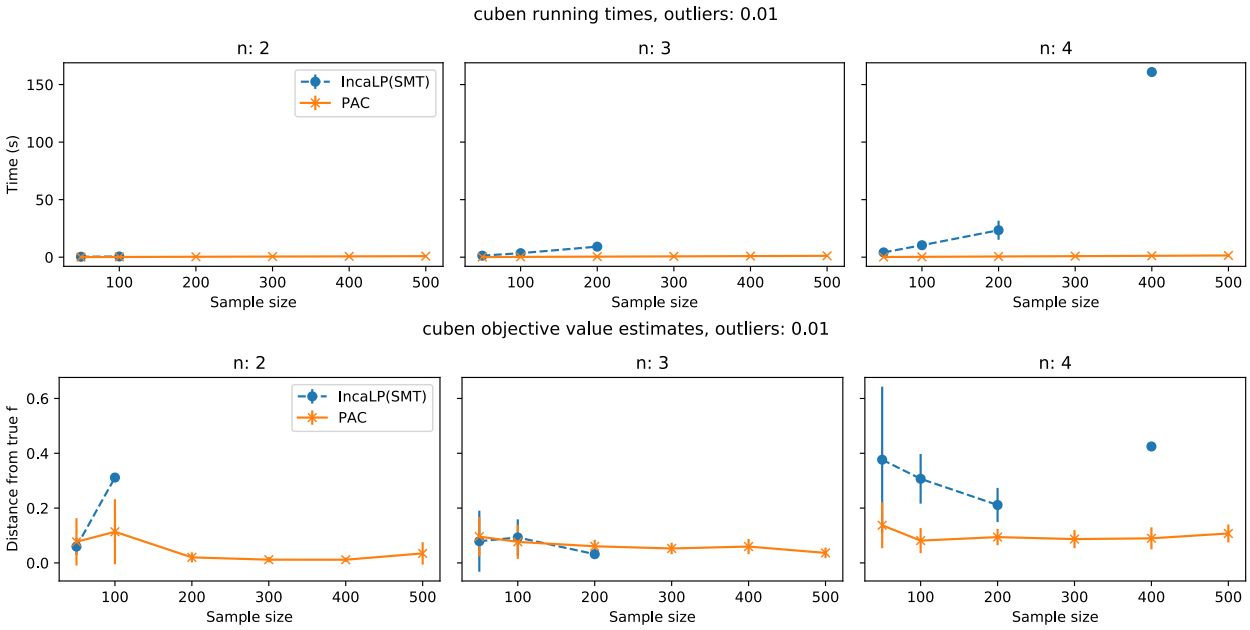


Figure 8: IncaLP failed to find a model 82% of the time.

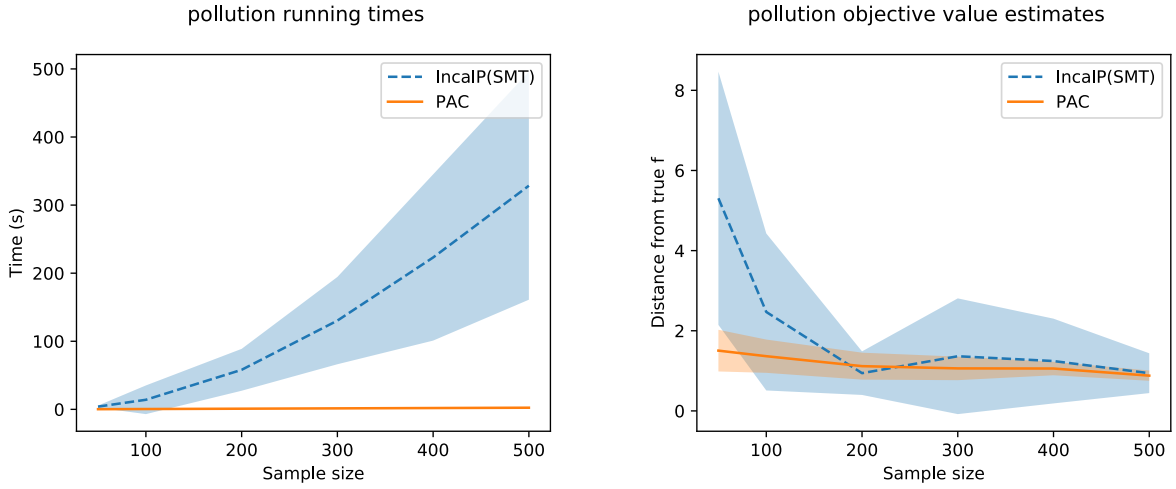


Figure 9: IncaLP always found a model.

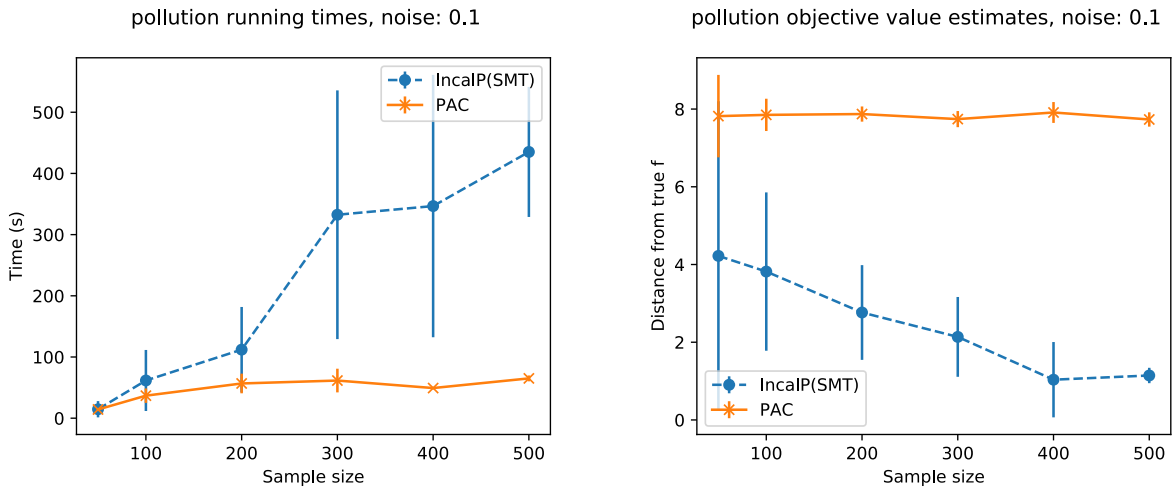


Figure 10: IncaLP failed to find a model 33% of the time.

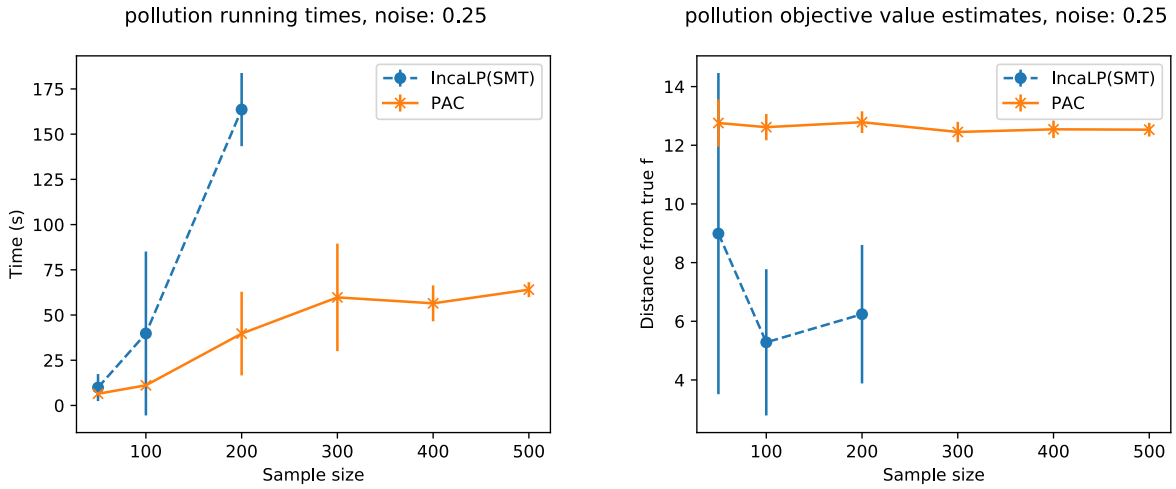


Figure 11: IncaLP failed to find a model 62% of the time.

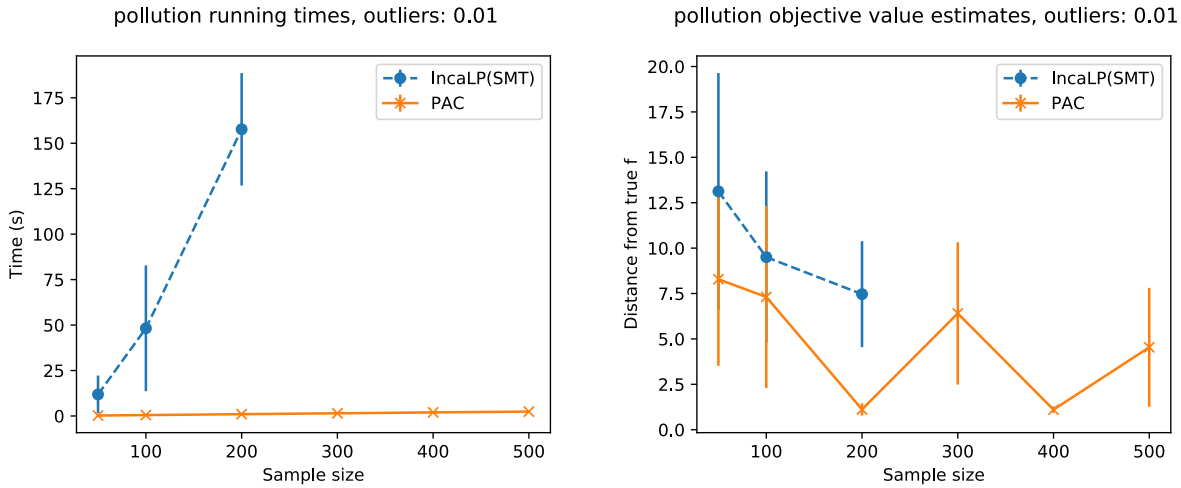


Figure 12: IncaLP failed to find a model 65% of the time.

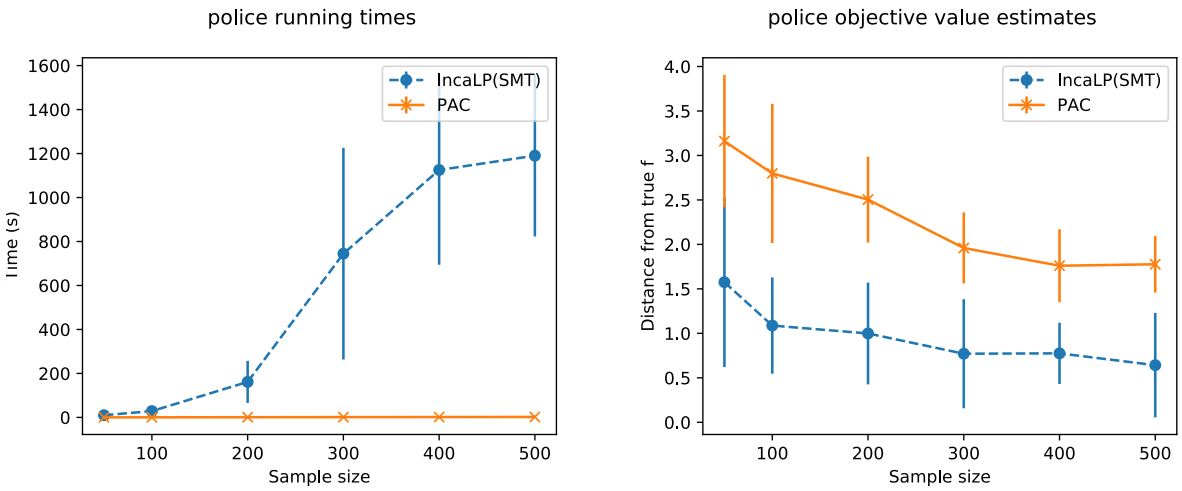


Figure 13: IncaLP failed to find a model 13% of the time.

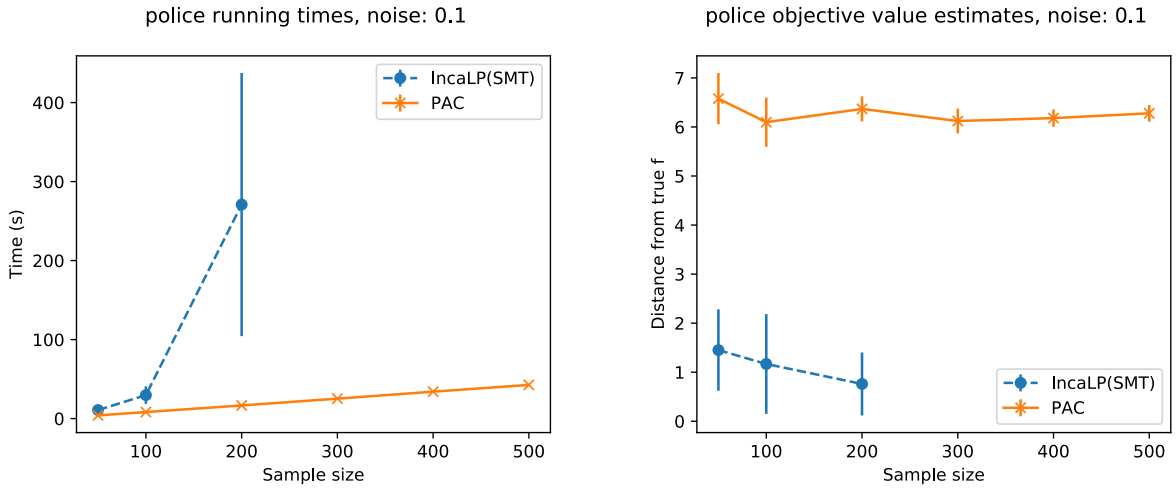


Figure 14: IncaLP failed to find a model 53% of the time.

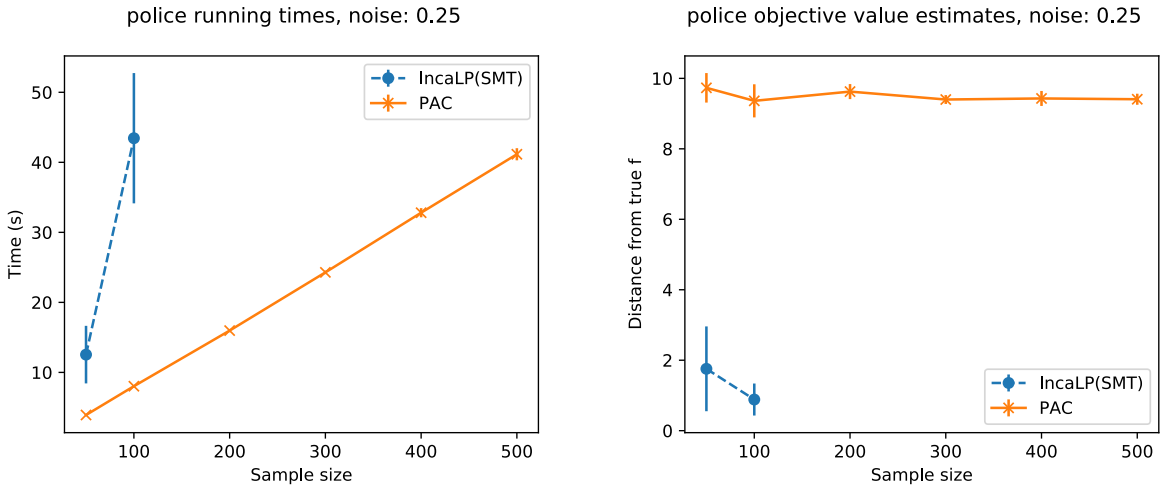


Figure 15: IncaLP failed to find a model 77% of the time.

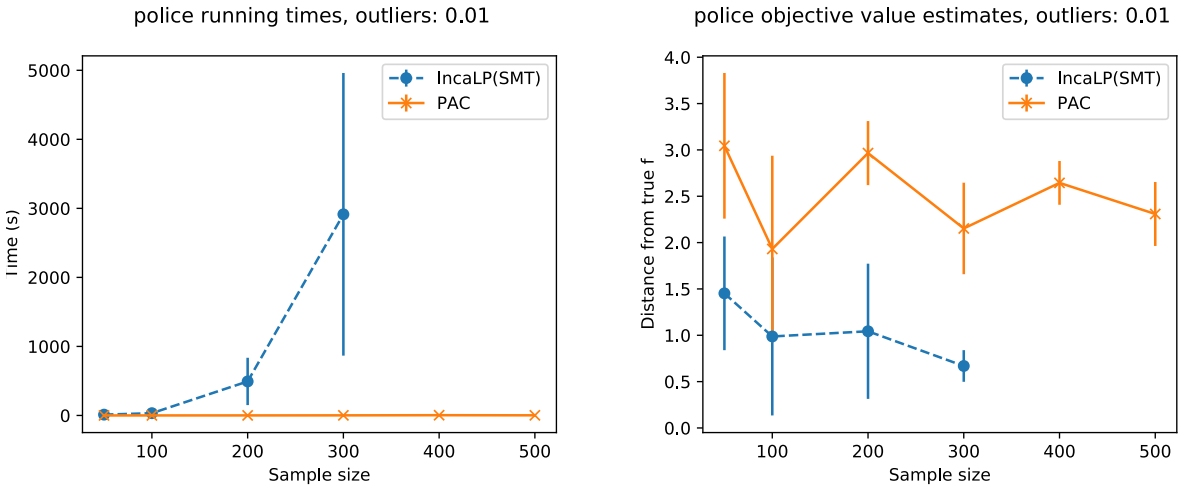


Figure 16: IncaLP failed to find a model 52% of the time.

References

[Anthony and Bartlett(1999)] Anthony, M.; and Bartlett, P. L. 1999. *Neural Network Learning: Theoretical Foundations*. USA: Cambridge University Press, 1st edition. ISBN 052111862X.

[Talagrand(1994)] Talagrand, M. 1994. Sharper bounds for Gaussian and empirical processes. *The Annals of Probability* 22(1): 28–76.