Constraint Acquisition and the Data Collection Bottleneck

Steven Prestwich

Department of Computer Science,
University College Cork, Ireland
s.prestwich@cs.ucc.ie

Abstract

The field of constraint acquisition (CA) aims to remove the "modelling bottleneck" by learning constraints from examples. However, it gives rise to a "data collection bottleneck" as humans must prepare a suitable (labelled) dataset. A recently published paper described an unsupervised CA method called MineAcq that can learn standard CA benchmarks. In this paper we summarise the results, and apply MineAcq to a new, noisy, unlabelled dataset that was not designed for CA.

Constraint acquisition

Constraint Programming is the area of Artificial Intelligence (AI) concerned with modelling and solving combinatorial problems. A *constraint satisfaction problem* (CSP) has a set of problem variables, each with a domain of possible values, and a network of constraints imposed on subsets of the variables. Modelling a CSP requires knowledge and experience, and can be difficult even for experts. An approach to automating this task is *Constraint Acquisition* (CA) in which constraints are learned from examples.

Though CA can remove the modelling bottleneck, it introduces what is sometimes called a *data collection bottleneck* in machine learning (ML): the task of gathering labelled instances to form a training dataset. This affects CA because all current methods used some form of *supervised learning* which requires (sometimes implicitly) labelled data. Moreover, most CA methods require rather a lot of data even for small problems.

One way to eliminate the data collection bottleneck would be to eliminate the need for labels, by designing new CA methods based on some form of *unsupervised learning*. In many ML applications this is the key to scalability, and some experts believe that the future of AI lies in unsupervised learning (Hao 2019).

MineAcq

A recent paper (Prestwich 2021) proposed an unsupervised CA method called MineAcq, inspired by data mining techniques. MineAcq can learn from positive-only, negative-only and positive-negative data; does not need labels; can learn constraint models for over-constrained problems; and is robust under data errors. Because it is unsupervised it could, for example, use data scraped from the Web.

MineAcq uses ideas from Association Rule Mining (ARM). ARM aims to extract interesting correlations, patterns and associations among sets of items in databases. Candidate association rules are tested using *measures of interestingness*, and any rule that passes the test is learned. MineAcq uses two measures of the interestingness of a constraint, corresponding to the *conviction* and *Ralambondrainy measure* in ARM, with runtime parameters denoted by τ and ρ.

In (Prestwich 2021) MineAcq learned a 9×9 Sudoku model, a 10×10 Latin square model, a Golomb ruler of length 12, random 3-SAT problems, and a partial ordering. It was faster than all other methods apart from SeqAcq and BayesAcq.

A new application

In this paper we choose a dataset that was not designed with CA in mind, but is simple enough so that we know the target constraint model.

LED display problem

The LED Display Domain Data Set is available from the UCI Machine Learning Repository (Dua and Graff 2017) in the form of a dataset generation program. This problem was designed as a 10-class classification problem, to test classification trees on noisy data (Breiman et al. 1984).

As a SAT problem the target model has 7 Boolean variables which we shall call A–G, each true if a light-emitting diode (LED) is illuminated in a digital display with the layout shown in Figure 1. There are $2^7 = 128$ possible configurations, 10 of which are used to represent digits 0–9. A dataset contains a specified number of examples such as

```
1 0 1 0 0 1 0 7
```

meaning that variables A, C and F set to true (1) represents digit 7. A specified noise level is used to randomly flip each attribute value; typically a noise level of 10% is used, causing a misclassification rate of 26%. We generate several datasets for this classification problem, and remove all labels to treat them as unsupervised CA problems.

```
   A            0          1          2
  ---          ---        ...        ---
B|   |C       |   |      :   |      :   |
 | D |        |   |      :   |      :   |
  ---          ...        ...        ---      ...
E|   |F       |   |      :   |      |   :
 | G |        |   |      :   |      |   :
  ---          ---        ...        ---
```

Figure 1: LED digital display

The target model

As this was designed as a classification problem, what does it mean to learn a clausal (or constraint) model from it? There was presumably no such model in the designers' minds, so what model are we trying to learn? Like association rule mining methods, MineAcq tries to learn data regularities in the form of clauses (or constraints). An instance is a SAT solution if and only if it satisfies all these clauses. Note that there might not be a solution: it is possible that no instance can exhibit all observed regularities in the dataset. In that case we learn an over-constrained CSP, which could be treated as a max-CSP. But for the LED dataset we are trying to learn a clausal model with 10 solutions corresponding to the digits 0–9.

To allow us to check the correctness of our results on noisy data, we first find the target model. This is easily done by generating a dataset with 0% noise so that all instances are solutions of the target CSP (like some other methods, MineAcq can handle positive-only datasets). MineAcq with $\tau = \infty$ and $\rho = 0$ learned 13 2-literal clauses:

$$\begin{array}{c} A \vee C \quad A \vee \neg E \quad A \vee F \quad A \vee \neg G \\ B \vee C \quad \neg B \vee F \quad C \vee D \quad C \vee F \\ C \vee G \quad D \vee F \quad E \vee F \quad \neg E \vee G \quad F \vee G \end{array}$$

and 108 3-literal clauses which reduce (by subsumption) to 12:

$$\begin{array}{c} A \vee B \vee \neg D \quad A \vee \neg B \vee D \quad \neg A \vee \neg B \vee G \\ A \vee \neg D \vee \neg G \quad \neg A \vee \neg D \vee G \quad B \vee D \vee \neg E \\ \neg B \vee D \vee E \quad B \vee D \vee \neg G \quad B \vee \neg D \vee G \\ \neg B \vee D \vee G \quad B \vee \neg E \vee \neg F \quad D \vee E \vee \neg G \end{array}$$

This clausal model has exactly 10 solutions, corresponding to the 10 digits, and is our target model.

Note that in general we might need to learn clauses longer than 3 literals, which would quickly become computationally expensive. However, for this application 2- and 3-literal clauses are sufficient. In future work we aim to adapt ARM methods to scale up to longer clauses.

Learning from noisy data

We experimented with different noise levels in the data. With 1% noise, 10^4 instances, $\tau = 2.5$ and $\rho = 0.01$ we learn the target in 0.08 seconds. With 3% noise we learn the target with $\rho = 0.03$ (data noise does not affect runtime). With 5% noise we get the right result using $\rho = 0.05$. With 10% noise there seem to be no parameter settings that yield the correct result. Increasing the number of instances reduces the error, but even with 10^7 instances (runtime 168 seconds) the best result we achieved failed to learn one of the target clauses. Moreover, the noiser the data the more careful we must be to choose the correct value of ρ, though given enough data the method works for a range of values.

Conclusion

Based on these and previous experiments, we make some observations about MineAcq. (i) It can work on datasets that were not designed for CA, which bodes well for its applicability to automatically gathered data. (ii) It can handle noise, given sufficient data and appropriate parameter values. (iii) The noisier the data the narrower the range of appropriate parameter values. (iv) For very noisy data a great deal of data might be required, which also increases MineAcq runtime. However, many ML experts believe that *simple models and a lot of data trump more elaborate models based on less data* (Halevy, Norvig, and Pereira 2009).

To the best of our knowledge, MineAcq is the first unsupervised CA method. It is similar in spirit to Process Mining in which event logs are mined for useful information (van der Aalst 2016). This has been used to learn constraints for scheduling problems (Senderovich, Booth, and Beck 2019) but it is not a general CA method.

Acknowledgements

This material is based upon works supported by the Science Foundation Ireland under Grant No. 12/RC/2289-P2 which is co-funded under the European Regional Development Fund. We would also like to acknowledge the support of the Science Foundation Ireland CONFIRM Centre for Smart Manufacturing, Research Code 16/RC/3918.

References

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and Regression Trees*. Wadsworth International Group: Belmont, California.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.

Halevy, A.; Norvig, P.; and Pereira, F. 2009. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*, 24: 8–12.

Hao, K. 2019. The AI Technique That Could Imbue Machines With the Ability to Reason. *MIT Technology Review*.

Prestwich, S. D. 2021. Unsupervised Constraint Acquisition. In *33rd International Conference on Tools with Artificial Intelligence*.

Senderovich, A.; Booth, K. E. C.; and Beck, J. C. 2019. Learning Scheduling Models from Event Data. In *29th International Conference on Automated Planning and Scheduling*.

van der Aalst, W. 2016. *Process Mining: Data Science in Action*. Springer.