Using Unlabelled Examples in Constraint Acquisition

Steve Prestwich

Insight Centre for Data Analytics
University College, Cork, Ireland

Abstract. Modeling a combinatorial problem is a hard and error-prone task requiring significant expertise. Constraint acquisition attempts to automate this process by learning constraints from labelled examples, which are known solutions and (usually) non-solutions. However, no current CA system exploits unlabelled examples. The ability to do this would greatly facilitate constraint acquisition in cases where labelled examples are hard to obtain. In this short paper we explore some possibilities.

1 Introduction

A *constraint satisfaction problem* (CSP) has a set of problem variables, each with a domain of possible values, and a set or network of constraints imposed on subsets of the variables. A constraint is a relationship that must be satisfied by any solution, though it can be violated by non-solutions.

Modelling a CSP requires knowledge and experience, but the field of *Constraint Acquisition* (CA) attempts to automate this task. CA has been identified as an important topic in [13] and a possible path to the "Holy Grail" of computer science: the user simply presents a problem to the computer in a manner natural for the user, and the computer proceeds to solve it [6, 7].

In CA we are typically given a set of *instances* (assignments to the variables of the unknown CSP), a set of constraint candidates called the *bias*, and a training dataset E of labelled instances. We wish to learn a subset T of the bias such that positive instances (solutions) satisfy all of T while negative instances (non-solutions) do not. A recent survey of the field is given in [11] but in this short paper we shall only mention a few representative methods.

We explore the possibility of exploiting *unlabelled* instances to improve CA. This work is at an early stage so no computational results are given, but we aim to have some results in time for the workshop.

2 Types of machine learning used in CA

There are several types of machine learning and most CA methods are examples of one type or another. We now briefly survey the types currently used in CA.

2.1 Supervised learning

Version space methods such as CONACQ [2] learn a representation of all possible representative constraint models, containing candidates that are satisfied by all solutions and violated by some non-solutions. The hope is that this set will converge to a single model given enough instances. Finding a version space is an example of *supervised learning*. These CA methods can be subdivided into *passive* and *active* learning systems, and some (such as CONACQ) come in both flavours.

2.2 Weak supervised learning

In *weak supervised learning* noisy data sources are used. The first CA methods to do this are BAYESACQ [4] and SEQACQ [10] which are robust under low noise levels.

2.3 Positive-only learning

Valiant's method [14] uses only examples of solutions and finds a most-specific model: any constraint (actually a clause as it is designed for SAT) that does not violate a known solution is learned. This is an example of *positive-only learning* (*negative-only learning* also exists but does not seem useful for CA). Sometimes non-solutions can be hard to obtain: we might have a few examples of solutions, but have no idea what a non-solution looks like. Positive-only learning is very useful for such cases.

Another positive-only method is MODEL SEEKER [1] which has successfully found high-level constraint models for several applications. Model Seeker also uses extensive knowledge about global constraints, so it could perhaps be viewed as more of an expert system than a machine learning algorithm. Its use of this meta-knowledge enables it to infer constraint models from few examples, in some cases from a single example, so it could also be viewed as an example of *few-shot learning*.

2.4 Note on redundant constraints

In constraint programming redundant constraints can have good or bad effects on performance, so it is not clear whether a CA method should learn them. Version space methods, Valiant's method, BAYESACQ and SEQACQ attempt to learn them.

Model Seeker instead *eliminates* redundant constraints, via a dominance test based on meta-knowledge. It also eliminates "uninteresting" constraints such as `not_all_equal` which have little effect, using hand-coded rules.

3 Using unlabelled instances in CA

In many learning applications, labelled data is hard to obtain while unlabelled data is plentiful. For example, suppose we wish to train a classifier to class websites as "interesting" or "uninteresting". A human would be needed to prepare a dataset, which would take considerable effort. However, there is an almost unlimited supply of unlabelled data in the form of websites, which it would be useful to exploit. In this section we explore some possibilities for CA.

3.1 Transductive learning

In *semi-supervised learning* (SSL) we have a small set of positive and negative labelled examples (which are often hard to obtain) and many unlabelled examples (which are easy to obtain). The aim is to use the unlabelled data to improve learning on the labelled data. Its success depends on the labelled and unlabelled data being drawn from the same distribution. SSL may be subdivided into *inductive* and *transductive* learning. In induction we attempt to learn a classifier to be used on unseen examples, while in transduction we merely attempt to classify the given unlabelled examples. These ideas are well-known in statistics and machine learning but has not been exploited in CA.

Our first idea is to improve CA by using transduction to label the unlabelled examples, giving us a large labelled dataset almost for free. We experimented with this idea using a semi-supervised naive Bayes classifier [3] based on the Expectation Maximisation (EM) algorithm. Unfortunately the results were poor, as each iteration of EM increased classification error. We also tried using naive Bayes to generate binary labels where this could be done with high confidence, leaving other examples unlabeled; but even when transduction was quite accurate, constraint acquisition on the large relabelled data was usually less accurate than on the small labelled data. The classifier was hand-coded so it might have been incorrect, but other researchers have also found EM to make learning less accurate [5, 12] so the result is perhaps unsurprising.

This attempt at using unlabelled data to improve CA failed, but we mention it as an interesting research direction. Other semi-supervised methods such as *feature marginals* [8] might work better.

3.2 Semi-supervised redundancy elimination

In this section we describe the main idea of the paper: using unlabelled data to improve the elimination of redundant constraints.

A supervised method First we describe a simple way of eliminating redundancies, which can be used as a post-processing phase to many CA methods. Having learned a constraint set T we check each $c \in T$ one by one: if $T \setminus \{c\}$ makes c redundant then it is deleted from T. The order in which we test constraints affects the result so we should test least-preferred candidates first, using any preference criteria we like.

Redundancy can be detected as follows: a constraint c is redundant if each instance that violates c also violates another constraint $c' \in T$. So removing c cannot create new solutions of the learned constraint model.

The worst-case time complexity of this method is $O(|E|.|T|^2)$. However, the average time should be typically much less than this, as T will often contain a small subset of the bias, and most constraints in T will not be redundant so an exception might be found quickly.

Adding unlabelled instances Note that class labels are not used in the above test. This allows us to apply it to unlabelled as well as labelled instances. We now show that it is safe:

– SSL can not unlearn valid redundancies that were learned from the labelled data. Suppose we learn a set T of constraints from the labelled data, one of which is c, and that $T \setminus \{c\}$ makes c redundant. Then in each non-solution violating c, at least one $c' \in T \setminus \{c\}$ is also violated. For unlabelled data to invalidate this redundancy, it must contain an instance in which c is violated but every $c' \in T \setminus \{c\}$ is satisfied. But c is redundant wrt $T \setminus \{c\}$ so this can not occur.
– SSL can not learn new invalid redundancies that were not learned on the labelled data. Suppose c is not made redundant by $T \setminus \{c\}$, and that we did not mistakenly learn this from the labelled data. Then there exists a labelled non-solution in which only c is violated, and this counter-example prevents redundancy from being learned from unlabelled data.

Next we show that the idea is potentially useful:

– SSL can unlearn invalid redundancies that were learned from the labelled data. For example, suppose we have a small labelled dataset that is sufficient to learn $T = \{x < y, x < z\}$, and that $\text{dom}(x) = \{1, \ldots, 100\}$ and $\text{dom}(y) = \text{dom}(x) = \{100, \ldots, 200\}$. Then in most non-solutions both constraints will be violated, and we will mistakenly learn that each constraint makes the other redundant and delete one from T. However, if we add sufficiently large unlabelled data then we will find an instance in which $x = y = 100, z > 100$, violating $x < y$ but satisfying $x < z$, thus destroying the illusion that the first makes the second redundant. Similarly, we will find an instance in which $x = z = 100, y > 100$ which invalidates the other redundancy.
– Unfortunately SSL can not learn new valid redundancies that were not learned from the labelled data (though see Section 3.3). Suppose c is not made redundant by $T \setminus \{c\}$, and that we did not learn this from the labelled data. Then there exists a labelled non-solution in which only c is violated, and again this counter-example prevents redundancy from being learned from unlabelled data.

This shows that adding unlabelled data can not cause any harm, but that it might do some good.

3.3 Semi-weak supervised redundancy elimination

The method can be extended to noisy data, by weakening the redundancy test so that it is required to hold in *almost* all instances. In this case, many unlabelled instances indicating redundancy could outweigh labelled counter-examples, and the method could also learn new valid redundancies from unlabelled data. This would be an example of *semi-weak supervised learning* in CA.

3.4 Positive-unlabelled redundancy elimination

Positive-unlabelled (PU) learning is similar to semi-supervised learning, but does not require negative examples. This seems ideal for CA: it might be relatively easy to obtain a few solutions and a large number of unlabelled examples. However, PU learning typically relies on user-specified estimates of class probabilities, which seems impractical for CA: the CSP to be learned might be very tightly-constrained, so that the unlabelled data might contain few or no solutions. Moreover, the labelled solutions might not have been sampled from a distribution, but manually constructed.

However, our redundancy test can also be used as a post-processor for positive-only CA methods. Suppose we learn a most-specific model using Valiant's method, or Model Seeker without a dominance test. We could then detect and remove redundant constraints in the same way as above, using unlabelled instances if available.

We believe that Model Seeker's elimination of "uninteresting" constraints might also be automated by PU learning. An uninteresting constraint such as `not_all_equal` is very weak and unlikely to be violated in even quite a large dataset. A constraint that is hardly ever violated can be removed with little effect on the constraint model. This can be tested on unlabelled as well as labelled instances.

4 Conclusion

In this short paper we proposed a new research direction for constraint acquisition: improving the learned constraint model by exploiting unlabelled as well as labelled instances. A promising first idea is the enhanced elimination of redundant constraints, but we hope to achieve more. This is a work in progress and we aim to have computational results in time for the workshop.

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.

References

1. N. Beldiceanu, H. Simonis. ModelSeeker: Extracting Global Constraint Models from Positive Examples. *Data Mining and Constraint Programming, Lecture Notes in Computer Science* vol. 10101, Springer 2016, pp. 77–95.
2. C. Bessiere, R. Coletta, F. Koriche, B. O'Sullivan. Constraint Acquisition. *Artificial Intelligence* **244**:315–342, 2017.
3. K. M. Branson. A Naive Bayes Classifier Using Transductive Inference for Text Classification. Technical Report, Dept. of Computer Science and Engineering, UCSD, 2001.
4. D. Brown, E. C. Freuder, B. O'Sullivan, S. D. Prestwich. Constraint Acquisition Via Classification. Abstract, IJCAI'19 Workshop on Data Science Meets Optimisation. Also presented at PTHG'19. Submitted for publication.
5. F. G. Cozman, I. Cohen, M. C. Cirelo. Semi-Supervised Learning of Mixture Models. *Proceedings of the 20th International Conference on Machine Learning*, 2003.
6. E. C. Freuder. Constraints: The Ties that Bind. *Proceedings of the 21st National Conference on Artificial Intelligence,* AAAI Press 2006, pp. 1520–1523.
7. E. C. Freuder. Progress Towards the Holy Grail. *Constraints* **23**:158–171, 2018.
8. M. R. Lucas, D. Downey. Scaling Semi-Supervised Naive Bayes With Feature Marginals. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2013, pp. 343–351.

9. K. Nigam, A. McCallum, S. Thrun, T. Mitchell. Learning to Classify Text from Labeled and Unlabeled Documents Using EM. *Machine Learning* **39**(2/3):103–134, 2000.
10. S. D. Prestwich. Robust Constraint Acquisition by Sequential Analysis. *Proceedings of the 24th European Conference on Artificial Intelligence* (to appear).
11. L. De Raedt, A. Passerini, S. Teso. Learning Constraints from Examples. *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018, pp. 7965–7970.
12. A. A. Saeed, G. C. Cawley, A. Bagnall. Benchmarking the Semi-Supervised Naïve Bayes Classifier. *Proceedings of the International Joint Conference on Neural Networks*, 2015, pp. 1–8.
13. B. O'Sullivan. Automated Modelling and Solving in Constraint Programming. *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 2010, pp. 1493–1497.
14. L. G. Valiant. A Theory of the Learnable. *Communications of the ACM* **27**(11):1134–1142, 1984.
15. X.-H. Vu, B. OSullivan. A Unifying Framework for Generalized Constraint Acquisition. *International Journal on Artificial Intelligence Tools* **17**(5):803–833, 2008.