An Analysis on the Generalization Error of the Constraint Acquisition Problem

Eisa Alanazi

Department of Computer Science, Umm Al-Qura University, Makkah, Saudi Arabia

Summary
In this work, we analyze the generalization error for learning a constrained problem, also known as a constraint acquisition problem. We consider the problem of learning constraints over finite and discrete domains (of variables) analyze the generalization error of the well-known version space learning algorithm. We show that a consistent learner would errs at most m(m−1)/2 times for a discrete network with variables having m domain values. Furthermore, we empirically demonstrate the feasibility of building version space learner which outputs a consistent hypothesis of small size even in large constraint networks. This holds true even if the examples were noisy/inconsistent with the given hypothesis.

Key words:
Constraints, Learning, Acquisition, Decision Making

1. Introduction

Constraint Programming (CP) is a powerful paradigm to solve a wide variety of combinatorial search problems. It usually involves specifying a set of relations (called constraints) over a set of decision variables. This specification is then submitted to a special software to search for one or more assignments satisfying the relations. Modeling the problem as a Constraint Satisfaction Problem (CSP) requires an expert in the field. This is due to two main reasons. First, formulating the problem as a CSP requires technical specialty that most of the users lack. The second reason stems from the fact that modeling is tightly correlated with the solving part in CSPs. In other words, even if the user was able to model the problem as a CSP, the modelled problem may not be the best possible modeling for the solving phase. Of course, there might be more than one model of the given problem and choosing the right one depends heavily on the expert choice and the problem tackled. This said, hiring an expert to model the problem seems unrealistic in many situations. For instance, when the modelled CSP represents a possible recommended item of an internet user or compatibility constraints for a personalized matchmaking agent.

As the CSP solving methods are mature enough and generally acceptable by the CP community, the modeling part remains to be one of the main bottlenecks when it comes to adopting CP techniques. For CSPs to be applicable to a wider range of applications, there must be some sort of automation in the modeling part. This automation usually assumes having access to the user data (i.e., past activities) and the goal is to utilize the user profile in order to learn a given CSP. Therefore, learning the constraints from data is an important step towards making CSPs, and CP tools in general, possible candidates in domains where hiring CP experts is unexpected.

In this report, we are interested in inducing a CSP that reflects the user constraints given his past activities. Usually this can be done either by interacting with the user and asking him for labels of carefully chosen instances (active learning) or by assuming a set of labelled instances is given (passive learning). In either case, our goal is clearly different from the concept of constraint reformulation where in the latter the constraints are assumed to be existed but for some reasons, we need to reformulate them in another more efficient way. In our case, there are no constraints in the first place.

The next section provides the necessary background information. Section 3 discusses different methods for constraints acquisition. Section 4 presents a theoretical analysis for the generalization error of constraint acquisition. Empirical evaluation of version space learner is demonstrated in Section 5. Results discussion and some interesting bias in the literature is discussed in Section 6. A conclusion remark and future work is presented in Section 7.

2. Preliminaries

We adopt the notations and the settings as presented in the literature [1,2]. We assume a set of n variables X = {x1, x2, . . ., xn} with the same domain of natural numbers D = {0, 1, 2, . . ., m}. Furthermore, there is a set of relations {r1, r2, . . . rt} constituting our language Γ. The language is of fixed arity k. That is, the scope of any relation ri ∈ Γ is bounded by a constant k ≥ 2.

The bias B_Γ is the set of all possible constraints constructed from Γ. That is, a constraint ci ∈ B_Γ if and only if ci can be represented as a combination of t relations in Γ. Notice that by bounding the relations arity, the size of B_Γ is polynomial in n. In particular, for every k subset of the n variables, there is

Manuscript received August 5, 2020
Manuscript revised August 20, 2020

k!|Γ| possible relations. Thus, $|B_\Gamma|$ = k! $\binom{n}{k}$ t which is polynomial in n. For the easiness of notation, we use rij in short for the relation r(xi,xj) for any relation r in Γ when k = 2.

Example 1 (Bias). Assume X = {x_1,x_2,x_3}, D = {0,1} and Γ = {=,≠}. Then B_Γ = {$=_{12}$, $=_{13}$, $=_{23}$, $\neq_{12}$, $\neq_{13}$, $\neq_{23}$}.

A concept f is a mapping from D^n to {0,1}. Let us denote the map of f(x) to 0 as x being a solution and to 1 by x being a non-solution. We refer to the set of solutions D'⊆ D^n in f by $f^{-1}(0)$

Example 2 (Concept). Consider the previous example and the constraint network C = {=ij}∀i < j ≤ n. The network has only two solutions: s0 = (0,0,...,0) and s1 = (1,1,...,1). A concept f for such network is the one that assigns f(s0) and f(s1) to 0 and the rest to 1.

A representation of a concept f is a constraint network C such that $f^{-1}(0)$ = sol(C) where sol(C) is the set of solutions for C. A concept f is representable by a bias B_Γ if there is a constraint network C = {c_1,c_2,...,c_y} such that $c_i \in B_\Gamma$ for all i ∈ {1,2,...,y} and sol(C) = $f^{-1}(0)$. For instance, Consider the previous example, it is obvious that $f^{-1}(0)$ = {s_0,s_1}. The class concept C_{B_Γ} is the set of all concepts representable in B_Γ. Note that, for every concept f ∈ C_{B_Γ}, there could be many representations of f .

Example 3. Recall the previous example. There are many representations of f. In particular, any constraint network C such that sol(C) = {s_0,s_1} is a representation of f. Two possible networks are $=_{12}$,$=_{23}$ and $=_{12}$, $=_{13}$.

As one may expect, the size of C_{B_Γ} is usually very small compared to the number of possible constraint networks. For example, all over-constrained networks are represented by a single concept which maps every element in D^n to 1. In this work, we fix Γ to {≠,≥,≤} and k to 2. In other words, we are interested in the learnability of binary constraint networks. For simplicity, we use C instead of C_{B_Γ} and B instead of B_Γ if it is clear from the context.

3. Related Work

Recently, Constraint Acquisition has received a considerable attention from the Constraint Programming (CP) community [3, 4, 5, 1, 2]. We classify the existed work into two classes: active and passive acquisition methods. In contrast to the active methods, the passive ones assume no interaction between the learner and the user. In passive methods, the learner has access to a set of training examples. Those examples are fixed apriori and classified into positive (solutions to the target network) and negative examples (non-solutions). Interestingly, the prominent approach for passively learning constraints is through searching the version space [2, 6, 7]. In other words, the learning problem is searching the hypothesis space for a hypothesis (i.e. a constraint networks) that accepts all positive examples and none of the negative ones. Clearly, the number of all consistent constraint networks is very large which urges people in the community to look for efficient ways to represent it. Almost all these attempts adopt the concept of most specific or most general hypotheses or both of them to implicitly maintain the version space. Moreover, as the learning process goes incrementally from most specific to most general, an ordering over the hypothesis space is usually defined. Notably, most of the attempts take the solution set of a given constraint network as a base line of this ordering. That is, a constraint network (or a hypothesis) C_1 is more general than another one C_2 if and only if the solution set of C_2 is a subset of the solution set of C_1. This gives arise to a partial order over the version space. Next, we give more details to the relevant passive methods.

Formulating the problem as a version space learning problem was first described in [6] where an algorithm (called ConAcq) based on implicit representation of the version space was given. In [7], ConAcq was extended to handle redundant constraints in order to solve the problem efficiently. A constraint c_i is redundant with regard to a network C if the removal of c_i eliminates no solution in C. In [2], both attempts have been extended to show how to solve the version space learning through SAT. The main idea of the conacq algorithm is to model the problem of updating the hypothesis space (i.e. current version space) as a satisfiability problem. Therefore, the work developed a clausal theory where consistent hypotheses correspond to the models of the theory. By doing this, any typical operation on the versions space has an equiv- alent operation in the underlying theory. For instance, checking whether the version space is empty corresponds to testing whether the underlying problem is unsatisfiable. The work assumes binary constraints and a set of constraints called library is given. The library works as a bias to the learner (any hypothesis is a combination of the library elements).

More precisely: a set of variables X = {x_1,x_2,...,x_n} and their domains D(x_1), D(x_2), . . . , D(x_n) are known to the learner (called vocabulary). The learner also has an access to a library of binary constraints B. It is assumed implicitly that any constraint of a possible target network is simply a combination of the relations in the library. This is due to the admissibility condition in ConAcq. A constraint c_i is admissible with regard to a library B, if c_i can be represented as conjunction of the constraints in B. A constraint network C is admissible with regard to B if every contraint $c_i \in$ C is admissible.

Example 5 (Admissiblity). Assume X = {x_1, x_2, x_3} each with D(x_i) = {1,2,3} ∀i ∈ {1,2,3} a possible library B is {$x_1 > x_3$,$x_3 > x_2$,$x_1 = x_3$}. A possible admissible network in this example is $x_1 \geq x_3$.

Therefore, the problem is: given a library B and a training set E, find a constraint network that is admissible to B and consistent with E. The work in [4] extend the conacq

algorithm by incorporating arguments. Arguments can be viewed as an additional knowledge to the learner given by an expert. In that work, an argument is a set of constraints provided with each example. The work showed that incorporating arguments has reduced the size of the version space to roughly 50% to 75%.

4. Theoretical Analysis on The Generalization Error of Constraint Acquisition

In this section, we investigate the generalization error of the ConAcq algorithm. We first discuss the generalization error of learning a binary constraint between two variables $x_i, x_j \in X$. Then, we extend the discussion to the binary CSPs case.
Definition 1 (Generalization Error). The generalization error of a concept c, denoted as err(c), with respect to the target c^* is $\sum_{x \in D^n} c(x) \neq c^*(x)$
Consider the case where we have two variables x_i and x_j. The question is: what is the maximum error we could have? Let $|r_{ij}|$ be the number of allowed tuples for a relation $r_{ij} \in B_\Gamma$. Then we have

- $|=_{ij}| = |D| = m$
- $|>_{ij}| = |<_{ij}| = m(m-1)/2$
- $|\geq_{ij}| = |\leq_{ij}| = m + \frac{m(m-1)}{2}$
- $|\neq_{ij}| = m(m-1)$

The problem is trivial: the error is at most $m(m-1)/2$. This happens when for example the learner outputs $>_{ij}$ or $<_{ij}$ where the target is actually $\neq_{ij}$. Therefore, the learner would misclassifies $\frac{1}{2}m(m-1)$ instances at most.

Proposition 1 (Error of learning a binary constraint). Given two variables x_i and x_j with the same domain, a language $\Gamma = \{\neq, \geq, \leq\}$ and a bias B_Γ, ConAcq will learn any binary constraint over x_i and x_j with generalization error at most $\frac{1}{2}m(m-1)$

The VapnikChervonenkis (VC) dimension is an important measure in learning theory. In principle, it gives a way to express the flexibility of a given concept class C. A set of instances $I \subseteq D^n$ is shattered with respect to a concept class C if and only if C shows all the possible labels of I. A class C is called to have a VC dimension of size d (denoted as VCD(C) = d) iff there exists ashattered set of size d but no shattered set of size d + 1. Notice that for any class $VCD(C) \leq \log_2|C|$. To see this: shattering any set of size d, requires at least 2^n distinct concepts to realize the labellings {0, 1} . The VC dimension is trivially 2 for the class of binary constraints over two variables $x_i, x_j \in X$. In the case of |X| = 2, |C| = 6 which makes 2^2 a possible value where $2^3 > |C|$ is not. Consider two instances (y_1, y_2), (y_3, y_4) such that $y_1 = y_2$ and $y_1 = y_3$ but $y_2 \neq y_4$ for any domain values $y_1, y_2, y_3, y_4 \in D$. As $y_2 \neq y_4$ it is either $y_2 > y_4$ or $y_2 < y_4$. Let's assume $y_2 < y_4$. Then:

- {0,0} is realized by $\leq_{ij}$.
- {1,1} is realized by $>_{ij}$.
- {1,0} is realized by $\not=_{ij}$.
- {0,1} is realized by $=_{ij}$.

Proposition 2 (VC Dimension of a binary Constraint). The VC dimension of learning a binary constraint based on bias B is 2.

5. Applied Analysis on The Generalization Error of Constraint Acquisition

We have implemented a Java-based tool for the version space learning. In our tool, we rely on explicit representation for small concept classes (up to four variables with three domain values). In the case of larger CSPs, we represent the concept class implicitly by the most specific and general concepts.
Generating the concept class: We create the concept class C of n variables as follows:

1. Generate all possible undirected graphs over n vertices. There are $2^{n(n-1)/2}$ such graphs.
2. For every undirected graph G(V,E), generate all possible constraint networks. There are $|E|^{|\Gamma|}$ such networks.
3. For every network, maps it to its corresponding concept f.
4. If $f \in C$ then $C = C \cup \{f\}$.

The concepts in C represents all possible concepts arising from binary CSPs over language Γ.
Sampling from the version space: Given an integer r > 0, we sample from the version space as follows:

1. Generate all undirected graphs with n vertices and r edges. There are $\binom{n}{r}$ such graphs.
2. Look for a consistent concept in those graphs.
3. If no concept is found, increase the value of r by one and Repeat (1) to (3).
4. If $r > n(n-1)/2$, report collapsing.

As the number of networks is exponential in n, r acts like a bias parameter for the type of consistent concepts we are

looking for. For instance, if we are interested in consistent concepts represented by simplest networks, we can set r to a small value. Indeed, in the worst case scenario we may generate exponentially many inconsistent networks before reaching a consistent one. To simplify this process we implemented three methods that trying to return simplest,complete or average consistent networks corresponds to initializing r to 1, $n(n-1)/2$ or $n(n-1)/4$ respectively.

In this section, we present an experiment over the number of examples a consistent algorithm needs before converging. Our approach is as follow:

1. Fix a tolerance constant ε to 0.05 and create a random hidden target f^* which represents a constrained network with the following parameters: $n=6, m=3, \Gamma \in \{\Gamma_1, \Gamma_2\}$ where $\Gamma_1 = \{\neq, =, \geq, \leq, >, <\}$ and $\Gamma_2 = \{\neq, =\}$ and density $d \in [0.1, 1)$ with 0.1 increment value for each iteration.
2. Each time we select randomly $10 \leq i \leq 500$ instances and label them according to f^* with increment value of 10 in each iteration.
3. Randomly sample a concept f' from the current version space and check its generalization error.
4. We stop when either i exceeds 500 or when f' converges, $err(f') \leq 0.05 \times 3^6 \approx 36$.

We vary the density of the target network and get the average of examples needed before converging for 10 target networks. Figure 1 shows the average number of examples needed to output a concept f* within ε error from the target f* . In this experiment, we never reached i > 100 which means we always get an ε−approximation of the target without revealing 95% of the possible labels. Furthermore, we look for the simplest consistent networks (initializing r to 1 in the previous section). This due to the fact that simplest explanations are more preferred to complex ones in machine learning (a.k.a Occam's Razon principle).

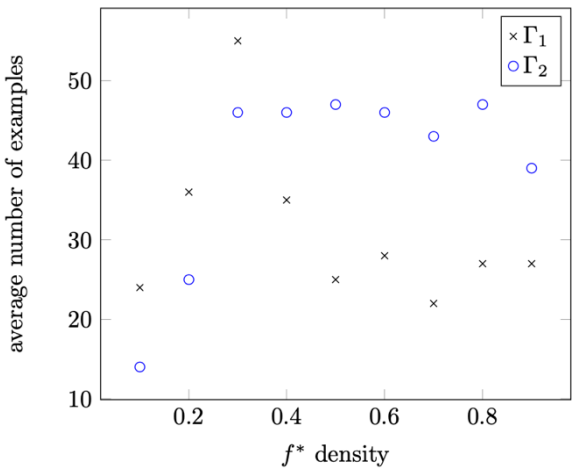

Fig. 1 The average number of examples needed to converge for two languages Γ1 and Γ2.

6. Discussion

Perhaps the first thing that a machine learning practitioner would spot on the literature is the usual bias assumption. Clearly, the goal of the bias is make sure the learner output something sensible to the domain and to escape from the exponential growth when blindly searching for the target network. For instance, in graph coloring problems, a bias would assure that every possible connected variables has distinct colors. Without bias the problem of learning constraints can be viewed as a learning relation problem. In its most general form: the solution set of the target could be any subset of D^n and the constraint networks could be any undirected graph where each constraint is defined as a set of allowed tuples. The bias in the literature is usually defined as a set of arithmetic relations (<, >, =). However, one could define his own bias as set of tuples that any k variables could have. We do not know any class of CSPs that can be describes with the latter bias and finding one class of CSPs with such bias would be interesting. But even with the bias assumption, the experimental results presented in the literature are usually limited to small number of variables (5 to 15)[8, 2]. The target networks gets bigger as we assume further capabilities of the user as in the case of QuiAcq where the problem with over 80 variables were experimented [1, 5].

The assumption of partial queries, i.e., that the user is able to classify a subset of variables values as positive or negative, is too strong. Usually, users do not have the cognitive nor the patience to be able to do this. Moreover, the complexity analysis of QuAcq assumed the example with the most violated constraints has been chosen. In reality, it could be difficult to know which example violating most of the constraints. Although the bias is

polynomial, there could be exponentially many such examples [9].

7. Conclusion

In this work, we discussed different methods in the literature aimed at learning the constraints. We classified the methods based on the interaction nature: passive or active. We then studied the problem of learning binary constraints and showed the generalization error of the version space learner assuming an adversary setting. Future work includes investigating the learnability and the generalization error of actively learning constraints.

References

[1] Robin Arcangioli and Nadjib Lazaar. Multiple constraint acquisition. In Workshop on Intelligent Personalization (at IJCAI), 2015.

[2] Christian Bessiere, Remi Coletta, Frdric Koriche, and Barry OSullivan. A sat-based version space algorithm for acquiring constraint satisfaction problems. In Joo Gama, Rui Camacho, PavelB. Brazdil, AlpioMrio Jorge, and Lus Torgo, editors, Machine Learning: ECML 2005, volume 3720 of Lecture Notes in Computer Science, pages 23– 34. Springer Berlin Heidelberg, 2005.

[3] Srinivas Padmanabhuni, Jia-Huai You, and Aditya Ghose. A framework for learning constraints: Preliminary report. In Grigoris Antoniou, AdityaK. Ghose, and Mirosaw Truszczyski, editors, Learning and Reasoning with Complex Representations, volume 1359 of Lecture Notes in Computer Science, pages 133–147. Springer Berlin Heidelberg, 1998.

[4] K. Shchekotykhin and G. Friedrich. Argumentation based constraint acquisition. In Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on, pages 476–482, Dec 2009.

[5] Christian Bessiere, Remi Coletta, Emmanuel Hebrard, George Katsirelos, Nadjib Lazaar, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. Constraint acquisition via partial queries. In Proceedings of the Twenty-Third International Joint Conference on Artifi- cial Intelligence, IJCAI '13, pages 475–481. AAAI Press, 2013.

[6] Remi Coletta, Christian Bessi`ere, Barry O'Sullivan, Eugene C. Freuder, Sarah O'Connell, and Jo¨el Quinqueton. Semi-automatic modeling by constraint acquisition. In Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings, pages 812–816, 2003.

[7] Christian Bessiere, Remi Coletta, EugeneC. Freuder, and Barry OSullivan. Leveraging the learning power of examples in automated constraint acquisition. In Mark Wallace, editor, Principles and Practice of Constraint Programming CP 2004, volume 3258 of Lecture Notes in Computer Science, pages 123–137. Springer Berlin Heidel- berg, 2004.

[8] Christian Bessi`ere, Remi Coletta, Barry O'Sullivan, and Mathias Paulin. Query-driven constraint acquisition. In IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad,

[9] Christian Bessiere, Remi Coletta, and Nadjib Lazaar. Solve a constraint problem without modeling it. In 26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014, Limassol, Cyprus, November 10-12, 2014, pages 1–7, 2014.

Eisa Alanazi received his B.Sc. degree in Information Systems from King Saud University, in 2007, and the MSc and PhD degree from the University of Regina, Canada in 2011 and 2017 respectively. He is currently an Assistant Professor at the Department of Computer Science, College of Computers and Information Systems in Umm Al-Qura University in Saudi Arabia. His research interests include preference learning and reasoning.