Generalizing over Several Learning Settings

Anna Kasprzik

University of Trier
kasprzik@informatik.uni-trier.de

Introduction. We recapitulate inference from membership and equivalence queries, positive and negative samples. Regular languages cannot be learned from one of those information sources only [1,2,3]. Combinations of two sources allowing regular (polynomial) inference are MQs and EQs [4], MQs and positive data [5,6], positive and negative data [7,8]. We sketch a meta-algorithm fully presented in [9] that generalizes over as many combinations of those sources as possible. This includes a survey of pairings for which there are no well-studied algorithms.

Definition 1. $T = \langle S, E, obs\rangle$ *($S, E \subseteq \Sigma^*$) is an* observation table *if S is prefix-closed and $obs(s, e) = 1$ if $se \in L$, 0 if $se \notin L$, $*$ if unknown. Let $row(s) := \{(e, obs(s, e))|e \in E\}$. S is partitioned into* RED *and* BLUE. *We call $r, s \in S$* obviously different *(OD; $r <> s$) iff $\exists e \in E$ with $obs(r, e) \neq obs(s, e)$ and $obs(r, e), obs(s, e) \in \{0, 1\}$. T is* closed *iff $\neg\exists s \in$ BLUE $: \forall r \in$ RED $: r <> s$.*

Let $r \equiv_L s$ iff $re \in L \Leftrightarrow se \in L$ for all $r, s, e \in \Sigma^*$. Let $I_L := |\{[s_0]_L | s_0 \in \Sigma^*\}|$. Due to the Myhill-Nerode theorem there is a unique total state-minimal DFA $\mathcal{A}_L$ with I_L states and each state recognizing a different equivalence class.

From a closed and consistent (see [4]) table $T = \langle S, E, obs\rangle$ with $\varepsilon \in E$ we derive a DFA $\mathcal{A}_T = \langle \Sigma, Q_T, q_T, F_T, \delta_T\rangle$ with $Q_T = row($RED$)$, $q_T = row(\varepsilon)$, $F_T = \{row(s) | s \in$ RED, $obs(s, \varepsilon) = 1\}$, and $\delta_T = \{(row(s), a) \mapsto q | \neg(q <> row(sa))$, $s \in$ RED, $a \in \Sigma$, $sa \in S\}$. $\mathcal{A}_T$ has at most I_L states (see [4], Th. 1).

Definition 2. *A finite $X \subseteq L$ is* representative *for L with min. DFA $\mathcal{A} = \langle \Sigma, Q, q_0, F, \delta\rangle$ iff $[\forall(q_1, a) \mapsto q_2 \in \delta : a \in \Sigma \Rightarrow \exists w \in X : \exists u, v \in \Sigma^* : w = uav \wedge (q_0, u) \mapsto q_1 \in \delta] \wedge [\forall q \in F : \exists w \in X : (q_0, w) \mapsto q \in \delta]$. A finite $X \subseteq \Sigma^* \backslash L$ is* separative *iff $\forall q_1 \neq q_2 \in Q : \exists w \in X : \exists u, v \in \Sigma^* : w = uv \wedge [\delta(q_L, u) = q_1 \vee \delta(q_L, u) = q_2] \wedge \exists(q_1, v) \mapsto q_a, (q_2, v) \mapsto q_b \in \delta : [(q_a \in F \wedge q_b \in (Q \backslash F)) \vee (q_b \in F \wedge q_a \in (Q \backslash F))]$.*

All learning algorithms we consider can be seen to start out with a provisional set of classes and converge to the partition by $\equiv_L$ by splitting or merging them according to obtained information. In a table S contains strings whose rows are candidates for *states* in the minimal DFA, and E *experiments* ('*contexts*') proving that two strings belong to distinct classes and represent different states.

Algorithm GENMODEL. The input is a tuple $IP = \langle EQ, MQ, X_+, X_-\rangle$ with Boolean values stating if EQs or MQs can be asked, a positive, and a negative finite sample of L. After initializing T we enter a loop checking if T is closed and,

J.M. Sempere and P. García (Eds.): ICGI 2010, LNAI 6339, pp. 288–292, 2010.

if it is, if we can still find states that should be split up, until there is no more information to process. The algorithm is composed of the following procedures:

INIT initializes the oracle (MQORACLE returns a blackbox for L if $MQ = 1$ and otherwise the prefix automaton for X_+ as an imperfect oracle improved during the process) and T. The set RED contains candidates that were fixed to represent a state in the output, and is initialized by ε (start state), and BLUE contains candidates representing states to which there is a transition from one of the states in RED. WHITE is the set of remaining candidates from which BLUE is filled up. The set of (initial) candidates is given by **POOL**: If $X_+ \neq \emptyset$ POOL returns $\mathit{Pref}(X_+)$, otherwise all strings up to length 2. If $X_- \neq \emptyset \wedge MQ = 1$ POOL builds X_+ from X_-: Let $n_- := |\mathit{Suff}(X_-)|$. In a worst case, every suffix in a separative X_- distinguishes a different of the $(I_L^2 - I_L)/2$ pairs of states. From $n_- \leq (I_L^2 - I_L)/2$ we compute an upper bound for I_L and take all strings up to that length as X_+ as the longest shortest representative of a state in $\mathcal{A}_L$ is at most of length I_L. Note that $|X_+|$ can be exponential with respect to $|X_-|$. We also have **UPDATE** which clears the elements that were moved to BLUE out of WHITE and fills in the cells of T if we have a perfect membership oracle which for $MQ = 1$ is true at any time and for $MQ = 0$ when we have processed all available information, provided that it was sufficient. For the cases with empty samples but $MQ = 1$ we fill up WHITE with all one-symbol extensions of BLUE.

CLOSURE is straightforward, it successively finds all elements preventing the closedness of T, moves them to RED, and calls UPDATE to fill up the table.

NEXTDIST calls FINDNEXT to look for a candidate to be fixed as another state of the output. Then T is modified by MAKEOD such that CLOSURE will move this string to RED. If no such candidate is found FINDNEXT returns $\langle \varepsilon, \varepsilon \rangle$ (this can be seen as a test for the termination criterion). In that case WHITE is emptied if we use queries only, for all other cases the remaining candidates are moved to BLUE in order not to lose the information contained in the pool.

If $MQ = 1$ **FINDNEXT** exploits a counterexample. $EQ = 1$: c is given by the oracle. Else if $X_+ \neq \emptyset$ the learner tries to build c from $T_{ext} = \langle S \cup \text{WHITE}, E \cup \mathit{Suff}(X_+), obs \rangle$. This succeeds if X_+ is representative (see [9]). At least one prefix of c must be a distinct state of the output, but as it may not be in BLUE MINIMIZE is called to replace the BLUE prefix of c until it finds $s'e'$ with $s' \in$ BLUE and e' distinguishes s' from all RED elements: FINDNEXT returns $\langle s'e' \rangle$. If $MQ = 0$ we continue merging states unless there is information preventing it. After the call of MERGENEXT either all BLUE strings correspond to states resulting from a merge or there is s which is a non-mergeable state. FINDNEXT returns $\langle s, \varepsilon \rangle$ as s should be a distinct state of the solution. In cases not covered by these distinctions we cannot reliably find another candidate to move and return $\langle \varepsilon, \varepsilon \rangle$.

MAKEOD is called if FINDNEXT returns $\langle s, e \rangle$ with $s \neq \varepsilon$, i.e., s is to be moved to RED by CLOSURE. If $MQ = 1$ there is a single $r \in$ RED *not* OD from s (RED elements are pairwise OD, and rows of S are complete), and e separates s from r, so add e to E. If $MQ = 0$ $\mathit{row}(s)$ consists of '$*$'s – we have to make s

OD from all $r \in \text{RED}$ "by hand": Find $c \in X_-$ preventing the merge of q_r and q_s via PREVENTMERGE and a suffix e_r of c leading from q_r or q_s to a final state ($X_- \neq \emptyset$ as FINDNEXT returns $\langle \varepsilon, \varepsilon \rangle$ for $MQ = 0$ otherwise). As c should not be accepted e_r separates s from r. Add e_r to E and fill the two cells of T with differing values – note that they do not have to be correct as they are used only once by CLOSURE, and T will be updated completely just before termination.

GENMODEL is intended as a generalization of algorithms for settings where polynomial one-shot inference is possible, which also implies that it is deterministic and does not guess/backtrack. However, note that it behaves in an "intuitively appropriate" way when (polynomial) inference is not possible as well.

We call an information source *non-void* for queries if $MQ = 1/EQ = 1$, for a positive sample if it is representative, and for a negative sample if it is separative.

Theorem 1. *a. Let L be the regular target language. GENMODEL terminates for any input after at most $2I_L - 1$ main loop executions and returns a DFA.*
b. For any input including at least two non-void information sources except for $\langle 1, 0, X_+, X_- \rangle$ with X_+ or X_- void the output is a minimal DFA for L.

See [9] for the proof. Note that Theorem 1b can also be seen from the proofs of the algorithms in [4,6,8]. We comment on the following three cases because to our knowledge there are no such well-studied algorithms for these settings.

$\langle 0, 1, \emptyset, X_- \rangle$: As $\langle 0, 1, X_+, \emptyset \rangle$. We build a positive sample from X_- (see above) which however may be exponential in size with respect to $|X_-|$ so that the number of MQs is not polynomial with respect to the size of the given data.

$\langle 1, 0, X_+, \emptyset \rangle$: Suppose we wanted to handle this case analogously: We would have to test state mergeability in O via EQs. For X_+ representative a positive counterexample reveals the existence of states that should be merged, a negative one of states that should not have been. When we query the result of a merge (even without repairing non-determinism by further merges) and get a positive counterexample we could either repeat the EQ and wait for a negative one but the number of positive ones may be infinite. Or we could query the next merge but when (if) we eventually get a negative one we do not know which of the previous merges was illegitimate. So this strategy is no less complex than ignoring all counterexamples and asking an EQ for the result of every possible *set* of merges, of which there are exponentially many. Therefore, since we cannot proceed as in the cases where inference is possible with a polynomial number of steps or queries this case is eclipsed from GENMODEL by the corresponding case distinctions.

$\langle 1, 0, \emptyset, X_- \rangle$: If X_- is separative negative counterexamples do not carry new information, and the number of negative counterexamples may be infinite. The set of positive counterexamples so far may not be representative so that we cannot reliably detect an illegitimate merge as there may be final states that are not even represented in the current O such that a compatibility check is too weak. If we make the merge we might have to undo it because of another positive counterexample, a situation we want to avoid. Hence we eclipse this case as well.

Note: For input with more than two non-empty sources the algorithm chooses one of the two-source options with priority MQs&EQs > MQ&X_+ > X_+&X_-.

Conclusion. We have aimed to design GENMODEL as modular as possible as an inventory of the essential procedures in existing and conceivable polynomial one-shot regular inference algorithms of the considered kind. This may help to give clearer explanations for the interchangeability of information sources. Practically, an extended GENMODEL (see below) could be used as a template from which individual algorithms for hitherto unstudied scenarios can be instantiated. We have chosen observation tables as an abstract and flexible means to perform and document the process, from which various descriptions can be derived.

GENMODEL offers itself to be extended in several directions. We could try to generalize over the type of objects, such as trees (see [10,6,11,12]), graphs, matrices, or infinite strings. Then there are other kinds of information sources which might be integratable, such as correction queries [13], active exploration [14], or distinguishing functions [15]. The third direction concerns an extension of the learned language class beyond regularity (for example by using strategies as in [16] for even linear languages, or [17] for languages recognized by DFA with infinite transition graphs) and even beyond context-freeness [16,18]. The development of GENMODEL may be of use in the concretization of an even more general model of learning in the sense of polynomial one-shot inference as considered here – also see the very interesting current work of Clark [19].

References

1. Gold, E.: Language identification in the limit. Inf. & Contr. 10(5), 447–474 (1967)
2. Angluin, D.: Queries and concept learning. Mach. L. 2, 319–342 (1988)
3. Angluin, D.: Negative results for equivalence queries. Mach. L. 5, 121–150 (1990)
4. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation 75(2), 87–106 (1987)
5. Angluin, D.: A note on the number of queries needed to identify regular languages. Inf. & Contr. 51, 76–87 (1981)
6. Besombes, J., Marion, J.Y.: Learning tree languages from positive examples and membership queries. In: Gavaldá, R., Jantke, K.P., Takimoto, E. (eds.) ALT 2003. LNCS (LNAI), vol. 2842, pp. 440–453. Springer, Heidelberg (2003)
7. Oncina, J., Garcia, P.: Identifying regular languages in polynomial time. Machine Perception and Artificial Intelligence, vol. 5, pp. 99–108. World Scientific, Singapore (2002)
8. de la Higuera, C.: Grammatical Inference: Learning Automata and Grammars. Cambridge University Press, Cambridge (2010)
9. Kasprzik, A.: Generalizing over several learning settings. Technical report, University of Trier (2009)
10. Drewes, F., Högberg, J.: Learning a regular tree language from a teacher. In: Ésik, Z., Fülöp, Z. (eds.) DLT 2003. LNCS, vol. 2710, pp. 279–291. Springer, Heidelberg (2003)
11. Oncina, J., Garcia, P.: Inference of recognizable tree sets. Technical report, DSIC II/47/93, Universidad de Valencia (1993)
12. Kasprzik, A.: A learning algorithm for multi-dimensional trees, or: Learning beyond context-freeness. In: Clark, A., Coste, F., Miclet, L. (eds.) ICGI 2008. LNCS (LNAI), vol. 5278, pp. 111–124. Springer, Heidelberg (2008)

13. Tîrnăucă, C.: A note on the relationship between different types of correction queries. In: Clark, A., Coste, F., Miclet, L. (eds.) ICGI 2008. LNCS (LNAI), vol. 5278, pp. 213–223. Springer, Heidelberg (2008)
14. Pitt, L.: Inductive inference, DFAs, and computational complexity. In: Jantke, K.P. (ed.) AII 1989. LNCS, vol. 397. Springer, Heidelberg (1989)
15. Fernau, H.: Identification of function distinguishable languages. Theoretical Computer Science 290(3), 1679–1711 (2003)
16. Fernau, H.: Even linear simple matrix languages: Formal language properties and grammatical inference. Theoretical Computer Science 289(1), 425–456 (2002)
17. Berman, P., Roos, R.: Learning one-counter languages in polynomial time. In: SFCS, pp. 61–67 (1987)
18. Yoshinaka, R.: Learning mildly context-sensitive languages with multidimensional substitutability from positive data. In: Gavaldà, R., Lugosi, G., Zeugmann, T., Zilles, S. (eds.) ALT 2009. LNCS, vol. 5809, pp. 278–292. Springer, Heidelberg (2009)
19. Clark, A.: Three learnable models for the description of language. In: Dediu, A.-H., Fernau, H., Martín-Vide, C. (eds.) LATA 2010. LNCS, vol. 6031, pp. 16–31. Springer, Heidelberg (2010)