Can ILP be Applied to Large Dataset?

Hiroaki Watanabe and Stephen Muggleton

Imperial College London, 180 Queen's Gate, London SW7 2AZ, UK
Email: {hw3, shm}@doc.ic.ac.uk

Abstract. There exist large data in science and business. Existing ILP systems cannot be applied effectively for data sets with 10000 data points. In this paper, we consider a technique which can be used to apply for more than 10000 data by simplifying it. Our approach is called Approximative Generalisation and can compress several data points into one example. In case that the original examples are mixture of positive and negative examples, the resulting example is ascribed in probability values representing proportion of positiveness. Our longer term aim is to apply on large Chess endgame database to allow well controlled evaluations of the technique. In this paper we start by choosing a simple game of Noughts and Crosses and we apply mini-max backup algorithm to obtain database of examples. These outcomes are compacted using our approach and empirical results show this has advantage both in accuracy and speed. In further work we hope to apply the approach to large database of both natural and artificial domains.

1 Introduction

There exist large data in science and business. Although Inductive Logic Programming (ILP) [2] has been tackling challenging problems [1], existing ILP systems cannot be applied effectively for data sets with 10000 data points unfortunately. A natural approach for handling such large data is to simplify it by reducing the amount of information. We could compress several data points into one example although the resulting example would need to capture proportion of positiveness especially when the original examples are mixture of positive and negative examples. Such a data compression technique is expected to be smoothly integrated into ILP frameworks since it is a part of domain knowledge. The purpose of this study is to investigate if we could achieve high predictive accuracies with simplified data in Probabilistic ILP (PILP) [5].

In this paper, our new technique called Approximative Generalisation is formally introduced in Section 2 first. It characterises PILP not only from an uncertainty but also from a *non-deterministic* point of view. Then we empirically study Approximative Generalisation in Noughts and Crosses domain in Section 3. Brief discussions conclude this paper in Section 4.

2 Approximative Generalisation

2.1 Learning from Specialised Examples

In a standard ILP setting [3], we search the set of hypotheses, H, which satisfies the entailment relation:

$$BK \cup H \models E \tag{1}$$

where E is a set of given examples, and BK is background knowledge. We propose to consider a set of specialised examples, E', which is a specialisation of E associated with BK under entailment as follows.

$$BK \cup E \models E' \tag{2}$$

Then E' satisfies the following relation.

Theorem 1. *Let BK, H, E' be background knowledge, a set of hypotheses, and a set of examples. If (1) and (2) are held, the following entailment relation is also held.*

$$BK \cup H \models E'$$

Proof. From (1), $BK \cup H \models BK \cup E$. From (2), $BK \cup E \models E'$ is held. Therefore $BK \cup H \models E'$.

Theorem 1 shows that the original concept can still be learned with the specialised examples. Now we define Approximative Generalisation as follows.

Definition 1 (Approximative Generalisation). *A set of hypotheses H' is called Approximative Generalisation of E if $BK \cup H \models BK \cup H' \models E'$ such that $BK \cup E \models E'$.*

Example 1. Let us assume the following BK and E.

$$BK = \{human(s), \forall\ X\ mortal(X) \rightarrow need\_grave(X)\} \qquad E = \{mortal(s)\}$$

From $BK \cup E$, we obtain $E' = \{need\_grave(s)\}$. Now $BK \cup \neg E'$ is:

$$BK \cup \neg E' = \{human(s), \neg need\_grave(s), \forall X\ mortal(X) \rightarrow need\_grave(X)\}$$

$$\models \{\exists X(human(X) \wedge \neg need\_grave(X))\} \underset{def}{=} \neg H'.$$

This is equivalent to $H' = \forall X(human(X) \rightarrow need\_grave(X))$. Then H can be $H = \{human(X) \rightarrow mortal(X)\}$ since $\{mortal(X) \rightarrow need\_grave(X)\} \cup \{human(X) \rightarrow mortal(X)\} \models \{human(X) \rightarrow need\_grave(X)\}$.

This example shows that there exist a case such that $BK \cup H' \not\models E$ although $BK \cup H \models BK \cup H' \models E'$.

2.2 Numerically Approximating Examples by Surjection

We further characterise the specialised examples by introducing surjective functions from E to E' in order to transfer the associated Boolean labels of positive or negative examples. Let us consider a class of projections, *surjective* functions, from E to E' as follows.

Definition 2. *A function $f : E \to E'$ is surjective if and only if for every $e' \in E'$ there is at least one $e \in E$ such that $f(e) = e'$.*

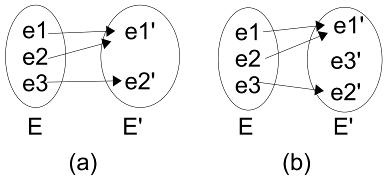

Fig. 1. (a)Surjective and (b)non-surjective functions

Example 2. In Fig. 1, Figure (a) is a surjective function whereas Figure (b) is a non-surjective function.

Now non-deterministic example is defined as follows.

Definition 3 (Non-deterministic Examples). *Let E^+ is a set of given positive examples and E^- a set of given negative examples. For $E = E^+ \cup E^-$ and a surjective function f, a non-deterministic example , $l : f(e)$, is a labelled example in which l is defined as: $l = \frac{|f_{E^+}(e)|}{|f_E(e)|}$ where $|f_{E^+}(e)|$ is a number of positive examples surjected onto $f(e)$ and $|f_E(e)|$ is a number of examples surjected onto $f(e)$.*

For example in Fig. 1, if we assume $e1 \in E^+$ and $e2 \in E^-$, we obtain $0.5 : e1'$.

In PILP, probabilistic examples can capture such a degree of truth in probability. *Now our task is to compute the approximative generalisation of E, H', with non-deterministic examples.* Further discussions on Approximative Generalisation can be found in [7].

3 Example: Noughts and Crosses Domain

In the previous section, we extend the standard ILP setting by adding the two computations: logical specialisation and numerical approximation. In this section, we empirically show such an extension can realise learning from large dataset in PILP.

3.1 Generating Never-Lose Sequences of Plays

We show an empirical result of the new framework in Noughts and Crosses domain where our study shows a more relational representation requires less number of examples. Noughts and Crosses, also called Tic Tac Toe, is a two-person, perfect information, and zero-sum game in which two players, *Nought* (O) and *Cross* (X) take turns marking the space of a 3×3 grid. Nought goes first and the player who succeeds in placing three respective marks in a horizontal, vertical or diagonal row wins the game. Although the setting and the rule are simple, there exist 9! (= 362880) possible ways of placing noughts and crosses on the board without regarding to winning combinations.

Noughts and Crosses game is known that there exist a never-lose strategy for each player [6]. That is, the game is always draw if two players know the optimal strategy. However, it becomes a probabilistic game once one side plays randomly. Let us assume that Nought plays in never-lose strategy whereas Cross plays randomly. Under such a probabilistic setting, we study (1) if a Machine Learning algorithm can obtain the never-lose strategy by cloning the behaviour of Nought and (2) how we can reduce the number of examples by changing knowledge representations from propositional one to relational one via surjective functions.

Before starting Machine Learning, we perform a retrograde analyse of Noughts and Crosses game in order to generate *never-lose* sequences of plays. In two player zero-sum game theory, mini-max is a rule which always selects a decision to *minimise the maximum possible loss.* By "maximum possible loss" we mean one players assumes the opponent always takes his best action which results the maximum loss to the other side. Mini-max algorithm evaluates the game in *forward* direction from the initial plays to the ends. In the Noughts and Crosses case, let us assume Nought plays first. Mini-max rule suggests Cross to always select the play which maximises Nought's loss whereas Nought to select the action to minimise such a loss.

Unfortunately mini-max criteria cannot force a player to win, however, an alternation of mini-max called *mini-max backup* algorithm [6] can do. It is originally developed for retrograde analysis of chess endgames in which the algorithm evaluate the player's actions and board positions in *backward* direction from the ends of the game to the initial plays. The key idea is that we only starts from Nought-won end-positions of the game and generate only sequences of predecessors which *never reach to losing end positions.* We apply this idea to generate the database of all the *never-lose* sequences of plays.

3.2 Two Logical Representations

We study two logical representations of Noughts and Crosses game. A natural way to express the 3×3 board is in the following atom:

$$board(p0, p1, p2, p3, p4, p5, p6, p7, p8)$$

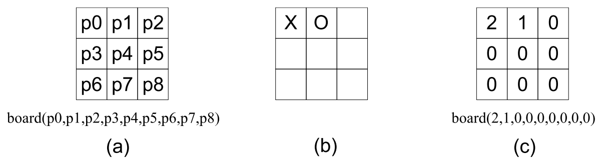

Fig. 2. Logical representation of a state of the 3 × 3 board. (a) shows the mappings between the locations of the grids and arities of the atom. (b) is a state of the board whose logical expression is shown in (c).

where the term p_i is either 1, 2, or 0 to express *nought*, *cross*, and *empty* respectively as shown in Figure 2. The atom, $board(2,1,0,0,0,0,0,0,0)$, expresses the state of the board (b) of Figure 2. Language $\mathcal{L}_1$ is defined as: (a)Predicate: board/9 and (b)Terms: 0,1,2. We also introduce a different relational language, $\mathcal{L}_2$. Figure 3 shows 6 relations in the board. The atoms, $corner(mark, p_i)$,

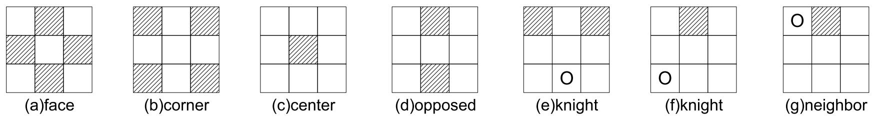

Fig. 3. Relations between grids of the game board

$face(mark, p_i)$, and $center(mark, p_i)$ take the term "*mark*" (either 1 for *nought* or 2 for *cross*) and p_i (i = 0,...,8) to express the *mark* being placed at p_i. The atoms, $opposed(p_i, p_j)$, $knight(p_i, p_j)$, and $neighbor(p_i, p_j)$ represent the relations between two grids, p_i and p_j. These relations are static and do not essentially depend on the plays, however, we only describe them when any *placed* mark has such relations. More precisely, the figures, (e), (f) and (g) in Figure 3, show the relative grids from the placed noughts. If any mark is placed in the shadowed grids, the associated relations are expressed. Language $\mathcal{L}_2$ is defined as follows.

- Predicate: corner/2, face/2, center/2, opposed/2, knight/2, neighbor/2
- Terms: 1, 2, p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8,

For example, the board (b) in Figure 2 can be expressed in the conjunctions of the form as $corner(2, p_0) \wedge face(1, p_1) \wedge neighbor(p_0, p_1)$. Note that $\mathcal{L}_2$ expresses all the marks of *nought* and *cross* on the 3×3 boards even after the specialisation since the second arity of $corner/2$, $face/2$, and $center/2$ tells the grid locations although $\mathcal{L}_2$ cannot express the locations of the empty grid at all.

Logical specialisation from the representation in language $\mathcal{L}_1$ to in language $\mathcal{L}_2$ can be expressed in a logic program. A part of such background knowledge is as follows.

```
corner(X,p0)   :- board(X,_,_,_,_,_,_,_,_), X != 0.
face(X,p1)     :- board(_,X,_,_,_,_,_,_,_), X != 0.
neighbor(X,Y) :- board(X,Y,_,_,_,_,_,_,_), X != 0.
```

3.3 Probabilistic Logic Automaton

We introduce Probabilistic Logic Automaton (PLA) [7] as a probabilistic logic. Intuitively, PLA is a logical extension of Probabilistic Automaton each of whose node can be an interpretation of an existentially quantified conjunction of literals (ECOL).

Definition 4 (Logical State). *Let L be a first-order language for describing ECOLs, F. A logical state q is a pair (n, F) where $n \in \mathcal{N}$ is the name of the logical state.*

Two logical states, (n_1, F_1) and (n_2, F_2), are treated as different if $n_1 \neq n_2$ even if F_1 and F_2 are logically equivalent. Each edge can be associated with disjunctions of ground actions.

Definition 5 (Logical Edge). *A logical edge is (a) a directed edge between two logical states and (b) associated with a set of ground atoms called logical actions.*

We introduce three probability distributions. Probabilistic transition function, $T : S \times \Sigma \rightarrow 2^S$ defines probabilistic state transitions from S to 2^S via Σ which is a set of ground atoms for describing logical actions. S_0 is a probability distribution over a set of initial logical states. We assign probability distribution over a set of logical actions, $B = \{b_{ij}(a_k)\}$, which is a set of probability distributions of taking logical action a_k during the state transition from state n_i to n_j. Now PLA is defined in the following six tuple.

Definition 6 (Probabilistic Logical Automaton). A probabilistic automaton is a 6-tuple

$$PLA = (S, \Sigma, T, S_0, B, G)$$

where S is a finite set of the logical states, Σ is a set of ground atoms for describing logical actions, $T : S \times \Sigma \rightarrow 2^S$ is a probabilistic transition function, S_0 is a probability distribution over a set of initial logical states, B is a set of probability distributions over logical actions at each edge, and G is a set of accept states.

In PLA, an input is a chain of observations of situations and actions as follows.

Definition 7 (Logical Sequence). *A logical sequence: $o_1 a_1 o_2 a_2 ... a_{n-1} o_n$ is an input of PLA in which o_i is ECOLs and a_i is a ground atom.*

Intuitively o_i may be viewed as an observation of a situation at time i and a_i is the action taken at time i. For example, a Nought-won game in a logical sequence is shown in Figure 4 where each state of the board is expressed as a logical node and actions of the players are attached to the directed edges.

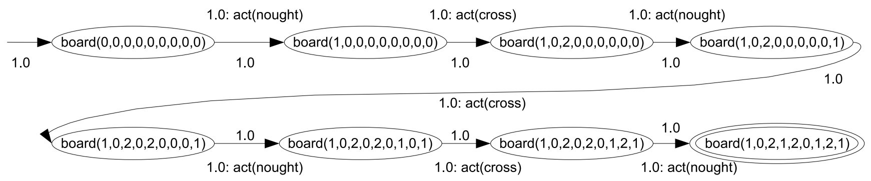

Fig. 4. A positive example of a never-lose sequence of plays in Probabilistic Logic Automaton.

3.4 Machine Learning of Noughts and Crosses Game

We study Machine Learning of Noughts and Crosses game next. We randomly sampled *never-lose* sequences of plays from the database and expressed in PLA based on the language $\mathcal{L}_1$. Let us call this example in $\mathcal{L}_1$ as E_1. Then the logical contents in each node in E_1 is specialised by a logic program a part of which is shown in the previous section. This specialised examples are called E_2. Note that E_1 is a set of positive example whereas E_2 is a set of non-deterministic examples with $l = 1.0$.

We learn the Nought strategy both in $\mathcal{L}_1$ and $\mathcal{L}_2$ by *Cellist* system [7] which provides (a) topology learning via Stirling Numbers of the second kind-based topology sampling algorithm [7], (b) specialisation of ECOLs by adopting Plotkin's lgg algorithm [4, 7], and (c) EM algorithm for estimating probabilistic parameters.

We tested 12 sample sizes, (5 , 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60) for the training examples. Regarding the empirical results, we evaluated predictive accuracies of the generated PLA models using 100 Nought-won test examples. For each number of sample sizes, we calculated the average error of the learned models and plotted in those figures. The best model results 92.3% predictive accuracy (0.077 error) when $m = 55$ in language $\mathcal{L}_2$.

The results[1] are shown in Figure 5 and Figure 6 in which how the error, ε, is decreased by increasing the number of non-deterministic examples. Clearly, the knowledge representation in $\mathcal{L}_2$ shows better predictive accuracies for all the sizes of training data.

4 Conclusion

In this paper, we present Approximative Generalisation for tackling large dataset in PILP. Our approach can compress several data points into one example. If the given examples are mixture of positive and negative examples, the proportion of positiveness is captured in probabilistic examples. We empirically confirmed

[1] The theoretical bounds shown in Figure 5 and Figure 6 are based on our average-case sample complexity analysis [7]. The theoretical bound expresses the delay bound; more than 34 examples are required in theory.

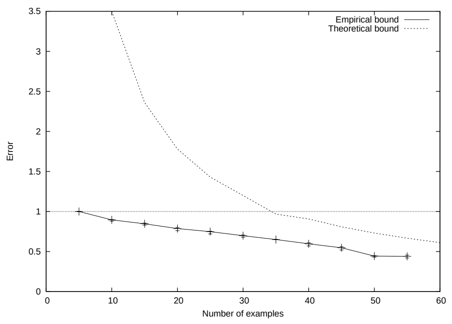

Fig. 5. Predictive Error Rates in $\mathcal{L}_1$

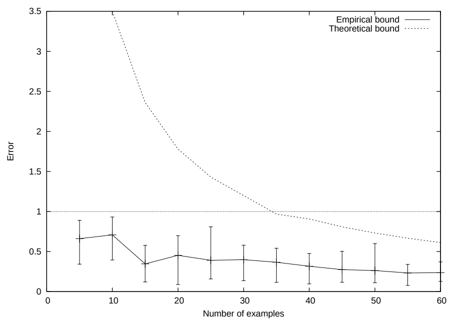

Fig. 6. Predictive Error Rates in $\mathcal{L}_2$

that our technique has advantage both in accuracy and speed in Noughts and Crosses domain even though the compressed examples are more relational than the original examples. This aspect should encourage PILP to tackle more relational applications.

In Approximative Generalisation, the data compression is smoothly encoded in background knowledge. Approximative Generalisation should also be discussed from an Active Learning point of view since we might need to test several data compressions repeatedly for finding better predictive accuracies.

In future work, we hope to apply our technique to large database of both natural and artificial domains including Chess End Game database to allow well controlled evaluation of our technique.

References

1. R.D. King, K.E. Whelan, F.M. Jones, P.K.G. Reiser, C.H. Bryant, S.H. Muggleton, D.B. Kell, and S.G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427:247–252, 2004.
2. S.H. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19(20):629–679, 1994.
3. S.H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1997.
4. G. Plotkin. *Automatic Methods of Inductive Inference*. PhD thesis, Edinburgh University, UK, 1971.
5. Luc De Raedt and Kristian Kersting. Probabilistic inductive logic programming. In *Lecture Notes in Computer Science*, volume 3244, pages 19–36. Springer, 2004.
6. Ken Thompson. Retrograde analysis of certain endgames. *ICCA Journal*, 9(3):131–139, 1986.
7. Hiroaki Watanabe. *A Learning Theory Approach for Probabilistic Relational Learning (DRAFT)*. PhD thesis, Imperial College London, UK, 2009. http://www.doc.ic.ac.uk/∼hw3/thesis.pdf.