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. Existing Inductive Logic Programming (ILP) [2] 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 study characterise Probabilistic ILP [1] not only from an uncertainty but also from a *non-deterministic* point of view.

In this paper, the new active learning framework is theoretically introduced in Section 2 first. Then we emperically confirm the theoretical result in Noughts and Crosses domain in Section 3. Brief discussions conclude this paper in Section 4.

2 Approximative Generalisation

We propose a new learning framework, called *Approximative Generalisation*, for learning from specialised examples. Our proposal smoothly fits to the standard entailment-based ILP framework.

Presented at the 19th International Conference on Inductive Logic Programming (ILP'09).

2.1 Entailment-based Specialisation

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. Instead of E, We consider a set of specialised examples, E', which is a specialisation of E associated with BK under entailment as follows.

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

Now 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. In this setting, we assume that the learner has domain knowledge to design such a specialisations in BK. The learner may repeat Approximative Generalisation by designing several specialisations.

2.2 Numerically Approximating Examples by Surjection

In the previous subsection, we define the knowledge modifications in a standard entailment relation as shown in (2). In order to transfer the associated Boolean labels of positive or negative examples, we introduce a functional constraint for the specialisation next. Let us consider a class of projections, *surjective* functions, from E to E' as follows.

Definition 1. *A function $f : E \rightarrow E'$ is surjective if and only if its range $f(E)$ is equal to its codomain E'.*

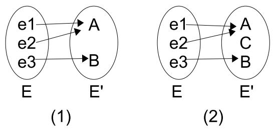

Fig. 1. (1)Surjective and (2)non-surjective functions

Example 1. Figure (1) in Fig. 1 is a surjective function whereas a non-surjective function is shown in Figure (2).

We request the logic-based specialisation, (2) , to satisfy the surjective feature. Now non-deterministic example is defined as follows.

Definition 2 (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 where 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 $e1$ and $e2$ are labelled as "positive example" and "negative example" respectively, $0.5 : A$ is obtained. In ILP, probabilistic examples can capture such a degree of truth in probability. Further discussions on Approximative Generalisation can be found in [4].

3 Example: Noughts and Crosses Domain

3.1 Generating Never-Lose Sequences of Plays

We show an empirical result of this new framework in Noughts and Crosses Domain where our result 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. Noughts and Crosses game is known that there exist a never-lose strategy for each player [5]. That is, the game is always draw if two players know the optimal strategy. However, it becomes a probabilistic one 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.

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 so called *mini-max backup* algorithm [5] 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: $board(p0, p1, p2, p3, p4, p5, p6, p7, p8)$ 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

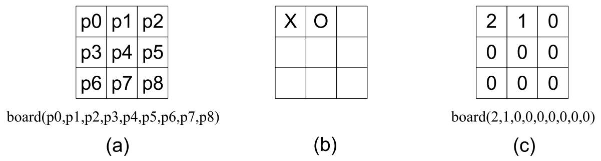

Fig. 2. Logical representation of a state of the 3 × 3 board. (a) shows the mappings between the locations of the grids and arities of the atom. (b) is a state of the board whose logical expression is shown in (c).

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, $corner(mark, p_i)$,

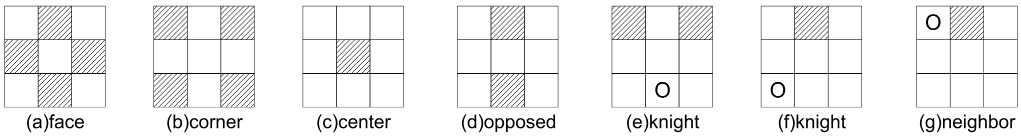

Fig. 3. Relations between grids of the game board

$face(mark, p_i)$, 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 Machine Learning of Noughts and Crosses Game

We study Machine Learning of Noughts and Crosses Game next. As a probabilistic logic, we use Probabilistic Logic Automaton (PLA) [4]. 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).

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 [4] which provides (a) topology learning via Stirling Numbers of the second kind-based topology sampling algorithm, (b) specialisation of ECOLs by adopting Plotkin's lgg algorithm [6], 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 are shown in Figure 4 and Figure 5 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.

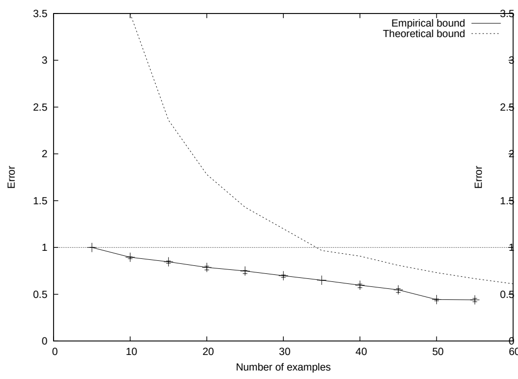

Fig. 4. Predictive Error Rates in $\mathcal{L}_1$

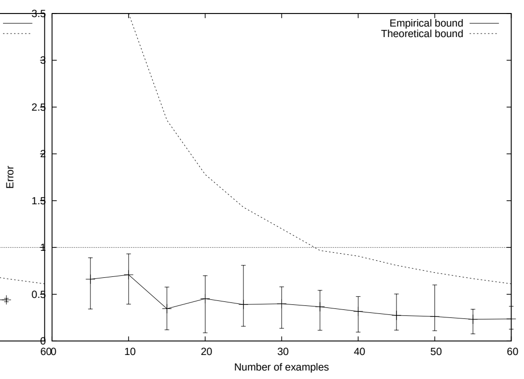

Fig. 5. Predictive Error Rates in $\mathcal{L}_2$

4 Conclusion

In this paper, we present Approximative Generalisation to find *easy-to-learn* knowledge representations formally in PILP. Our experimental result in Noughts and Crosses domain shows a case that the more relational representation in language $\mathcal{L}_2$ has lower sample complexity. This aspect should encourage PILP to tackle more relational applications. A natural but interesting future direction of Approximative Generalisation is about learning with huge data since the surjective function can compress the size of examples. Further discussion is required to characterise PILP from a non-deterministic point of view since our approach only captures the numerical aspect of non-determinicity.

References

1. Luc De Raedt and Kristian Kersting.: Probabilistic Inductive Logic Programming. Lecture Notes in Computer Science3244. pp. 19-36. Springer. 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. Lecture Notes in Artificial Intelligence, 1228. Springer-Verlag, 1997.
4. H. Watanabe.: A Learning Theory Approach for Probabilistic Relational Learning, PhD Thesis (Draft), http://www.doc.ic.ac.uk/~hw3/thesis.pdf.
5. Ken Thompson.: Retrograde analysis of certain endgames. ICCA Journal, vol. 9, No.3, pp131-139, 1986.
6. G. Plotkin.: Automatic Methods of Inductive Inference. PhD thesis, Edinburgh University, UK, 1971.