7

Inverting the Resolution Principle

S. H. Muggleton
The Turing Institute,
Glasgow, UK

Abstract

In this paper we describe the current status of an ongoing research project investigating a novel form of Machine Learning in which the learner's vocabulary is enriched by the machine suggesting useful new descriptive terms for the user to accept or reject. An algorithm called Duce has been shown to be effective along these lines in developing and extending propositional theories within a chess endgame domain and a diagnostic domain of neuro-psychology. By showing that Duce's transformational operators are based on reversing the steps of a resolution proof we show that Duce's learning method is sufficient for learning any propositional theory.

1. INTRODUCTION

Duce [4] is an algorithm which produces hierarchical concept descriptions from large numbers of examples. Whereas the ID [7] and AQ [3] families of inductive algorithms require all necessary attributes to be provided before learning can take place, Duce develops new attributes by incrementally building them from existing ones, testing each against the user for comprehensibility. Duce uses a set of transformations of propositional Horn clauses which successively compress the example material on the basis of generalizations and the addition of new terms. In the following descriptions of three of the six Duce operators, lower-case Greek letters stand for conjunctions of propositional symbols.

1. *Intra-construction* This is the distributive law of Boolean equations. We take a set of rules such as

   $h_1 \leftarrow \alpha\beta$
   $h_1 \leftarrow \alpha\gamma$

   and replace them with the rules

   $h_1 \leftarrow \alpha h_3$
   $h_3 \leftarrow \beta$
   $h_3 \leftarrow \gamma$

The user either names the new concept h_3 or rejects it.

2. *Absorption* This operator is from Sammut and Banerji [10]. Given a set of rules, the body of one of which is completely contained within the bodies of the others, such as

   $h_1 \leftarrow \alpha\beta$
   $h_2 \leftarrow \alpha$

   one can hypothesize

   $h_1 \leftarrow h_2\beta$
   $h_2 \leftarrow \alpha$

   The user can either accept this generalization or reject it.

3. *Identification* This operator has preconditions which are stronger than those of intra-construction. A set of rules which all have the same head, the body of at least one of which contains exactly one symbol not found within the other rules, such as

   $h_1 \leftarrow \alpha\beta$
   $h_1 \leftarrow \alpha h_2$

   can be replaced by

   $h_1 \leftarrow \alpha h_2$
   $h_2 \leftarrow \beta$

   Again the user can either accept this generalization or reject it.

Duce uses the compaction of the rulebase produced by each of the six operators to guide the search for the next operator to apply. In [4] we give the set of characteristic formulas, one for each operator, which predict the exact *symbol reduction* produced by each operator, the number of *symbols* in a rule being equal to the rule-body length plus one for the rule-head. Since Duce applies only operators which give a positive symbol reduction it can easily be shown that the algorithm terminates after a finite number of operator applications.

2. APPLICATION DOMAINS

2.1. KPa7KR application

The first large-scale test of Duce's capabilities [4], was an attempt to reconstruct automatically Shapiro and Kopec's expert system [11] for deciding whether positions within the endgame of King-and-Pawn-on-a7 v. King-and-Rook were won for white or not. A set of 3196 examples was used, and Duce's questions were answered by the chess experts Ivan

Bratko and Tim Niblett. The result was a comprehensible restructuring of the domain, topologically similar to Shapiro and Kopec's original structure, though an order of magnitude more bulky.

2.2 Neuro-psychology application

A second, and previously undescribed structuring experiment was carried out by the author using Duce at Interact Corporation, Canada. In this Duce was used to construct a problem decomposition for deciding on dysfunction of the left parietal brain area of children with learning disabilities. The input to the algorithm consisted of 227 diagnosed cases. Each case contained the results of a battery of approximately 100 binary-valued clinical tests. Each case was marked with a diagnosis of normal/abnormal left parietal lobe by the resident clinical neuro-psychologist, Dr Russell. Using these cases Duce carried out an interactive session in which Russell was asked to answer a total of 53 questions. During and subsequent to the construction of the rulebase, a set of 48 independent cases was used to test the performance of the new rule-set. Since the cases and generated rules were inherently noisy, a majority-vote mechanism was used for rule evaluation. After all 53 questions had been answered, 43 of the 48 test cases agreed with Russell's diagnosis, in other words, 90 per cent agreement. In contrast, an existing expert system developed by Russell had only a 63 per cent agreement rate with Russell's diagnoses over the same test data. While Duce's structured rulebase took 2–3 person-days to build and verify, the equivalent part of the hand-built expert system is conservatively estimated to have taken 2–3 person-months to generate, improve, and verify.

In parallel with the supervised construction of the Duce rulebase, Duce was run on the same cases in unsupervised mode. In this mode, all generalization questions were answered affirmatively and all new concepts were arbitrarily named. Performance in unsupervised mode stabilized after 27 questions to a level of 25 per cent agreement with Russell's diagnoses of the same test cases.

Unlike the endgame experiment in which an exhaustive example set was used, the neuro-psychological example set was relatively sparse. As a consequence, whereas no rejections were necessary in the case of the chess experiment, an average of 10 rejections were required per acceptance with the neuro-psychological data. This seems to indicate the need for expert supervision of Duce where sparse data is involved, and explains the dramatic difference in verification results between the supervised and unsupervised data.

The structure of the rulebase created by Russell working with Duce is shown below. This hierarchical structure contains groups of rules associated with each node of the network. The sub-types implied by this

hierarchy were, according to Russell, 'clinically significant', and relate directly to neuro-psychological sub-types based on Wide-range-achievement-test (WRAT) results in arithmetic, reading, and spelling.

```
LPAb_GoodV
LPAb_PoorA1
  LPAb_PoorA
LPAb_BadA1
  LPAb_BadA1_RS_AST
  LPAb_BadA
    LPAb_BadA_PoorS
      LPAb_BdA_PrS_BadAST_GdSSPT
    LPAb_BadA_BadAST
      LPAb BadA_AST_S1
        LPAb_BadA_AST_S
        LPAb_BadA_AST_S1_V
```

3. THEORY

Duce has shown a considerable amount of sucess within the application domains described in the previous section. However, there are a number of questions concerning the methodology employed within Duce which are of interest both from a theoretical and a practical viewpoint.

1. *Completeness of operators* Six operators are used by Duce to carry out generalizations and introduce new terms. How complete are these? Are they sufficient to learn any arbitrary set of propositional clauses given enough examples? (See Section 3.2.)
2. *Search* Duce presently searches through the set of conjunctions of predicate symbols to find which operator to apply next. For this reason Duce can be myopic in its choice of new terms. The discrepancy in complexity between Duce's solution and human solutions (see Section 2.1) seems to indicate that this problem can be quite severe. New methods of searching for operators are required. (See Section 3.3.)
3. *Extensions to first-order representation* Bain [2] has described a failed attempt to use Duce to learn a simple chess definition of position legality from only positional attributes. Although the definition can be described simply in first-order predicate calculus the cumbersomeness of Bain's description is not eased by adding extra propositional attributes. This gives incentive to an investigation into extending the Duce approach to deal with first-order predicate calculus (see [5]).

3.1 Inverting resolution

Although it is apparent to many researchers in Machine Learning that there is a strong relationship between deductive theorem-proving mechanisms and inductive inference, this idea has rarely been investigated to any greater depth than to notice that the ideas of *logical subsumption* or *logical implication* are central to both. One exception to this is Plotkin [6] who investigated the idea that 'just as unification was fundamental to deduction, so might a converse be of use in induction.'

From this idea Plotkin went on to develop the concept of *least general generalization*, or *anti-unification* of literals and clauses.

Unification is a basic idea within Robinson's [9] theory of resolution. Another important concept within this theory is that of the resolution tautology, or rule of inference. As Plotkin [6] notes, 'It is interesting that ... the similarity between induction and deduction breaks down ... [with anti-unification]. What is useful is not a concept of unification of two clauses, but the deduction principle called resolution.'

We now show that the analogy between deduction and induction can be extended fruitfully, and that in fact the operators used by Duce are merely the inverse of resolution. In a later section, this fact will lead us to a proof of the sufficiency of the Duce operators.

In this section we limit our discussion of resolution to binary resolution of propositional Horn clauses. However, in [5] we extend this analysis to deal with first-order representations. Let C_1 and C_2 be the two clauses

$$C_1 = (h_1 \leftarrow \alpha h_2)$$

$$C_2 = (h_2 \leftarrow \beta)$$

We write the resolvent or resolved product of C_1 and C_2 as

$$C = C_1 \cdot C_2 = (h_1 \leftarrow \alpha\beta)$$

We now define the resolved quotient as follows

$$C_1 = C/C_2$$

Alternatively, the author calls C_1 the *identificant* of C and C_2. Similarly

$$C_2 = C/C_1$$

Again we call C_2 the *absorbant* of C and C_1.

It is now straightforward to define the absorption and identification operators described informally in Section 1.

Definition 1 Given a propositional Horn clause program $P \supseteq \{C, C_1\}$, the absorption operator, *Abs*, transforms P to $P' = (P - \{C\}) \cup \{C/C_1\}$.

Definition 2 Given a propositional Horn clause program $P \supseteq \{C, C_2\}$, the identification operator, *Ident*, transforms P to $P' = (P - \{C\}) \cup \{C/C_2\}$.

We can also define the inverse of both of these operators uniquely.

Definition 3 Given a propositional Horn clause program $P \supseteq \{C_1, C_2\}$, the inverse absorption operator, Abs^{-1}, transforms P to $P' = (P - \{C_2\}) \cup \{C_1 \cdot C_2\}$.

Definition 4 Given a propositional Horn clause program $P \supseteq \{C_1, C_2\}$, the inverse identification operator, $Ident^{-1}$, transforms P to $P' = (P - \{C_1\}) \cup \{C_1 \cdot C_2\}$.

We now give a formal definition of the intra-construction operator of Section 1 and its inverse. Let $A = (h_3 \leftarrow \gamma h_4)$, $BB = \{B_1, \ldots, B_n\} = \{(h_4 \leftarrow \delta_1), \ldots, (h_4 \leftarrow \delta_n)\}$, $CC = \{C_1, \ldots, C_n\} = \{(A \cdot B_1), \ldots, (A \cdot B_n)\} = \{(h_3 \leftarrow \gamma\delta_1), \ldots, (h_3 \leftarrow \gamma\delta_n)\}$.

Definition 5 Given a propositional Horn clause program $P \supseteq CC$, the intra-construction operator, *Intra*, transforms P to $P' = (P - CC) \cup \{A\} \cup BB$.

Definition 6 Given a propositional Horn clause program $P \supseteq (\{A\} \cup BB)$, the inverse intra-construction operator, $Intra^{-1}$, transforms P to $P' = (P - (\{A\} \cup CC)) \cup BB$.

Lemma 1 If the program transformation $P \rightarrow P'$ is carried out by either Abs^{-1}, $Ident^{-1}$, or $Intra^{-1}$ then P subsumes P'.

Proof Follows from the fact that all these operators replace more general clauses with more specific ones. □

The reader may wonder how A and BB are constructed in the definition of *Intra*. As a special case of Plotkin's [6] *least general generalization* (*lgg*) of clauses, we say that γ is the *lgg* of the bodies of clauses within CC ($bodies(CC)$) if and only if γ is the common intersection of propositional symbols of $bodies(CC)$. Given γ and a new predicate symbol h_3, it is straightforward to construct A and BB. It is only through reversal of multiple resolution steps that the introduction of new predicate symbols becomes possible.

3.2. Completeness of Duce operators

In order to ensure that the success of the Duce applications described in Section 2 was not due to some peculiar property of the domains involved we need to show that the operators used by Duce are sufficient to learn any arbitrary set of propositional clauses. Clearly we need to specify some restriction on the allowable forms of examples used, otherwise Duce could merely be presented with any desired solution as its input. Let P be an arbitrary target propositional Horn clause program. The vocabulary used in P ($vocab(P)$) is then simply the set of propositional symbols in P. The primitive vocabulary of P ($prim(P)$) is the set of

symbols not defined in terms of other predicate symbols. Thus $prim(P) = vocab(P) - heads(P)$, where $heads(P)$ is the set of clause heads of clauses within P. Now let E be a set of example propositions from which P can be learned by Duce. A reasonable restriction on allowable forms of examples used would seem to be that

(1) P subsumes E, i.e. any statement which can be derived from E can also be derived from P;

(2) each clause body in $bodies(E)$ is composed only of predicate symbols from $prim(P)$;

(3) the vocabulary of P is an extension of that of E, i.e. $vocab(P) - vocab(E) \neq \{\}$.

If these conditions are met then we will say that E is a *legitimate* example set of P. First we define the abstract algorithm $Duce_{(Abs,Intra)}$ which is a non-deterministic version of the Duce algorithm limited to using only the *Abs* and *Intra* operators. In the following an inverse derivation $E \rightarrow P_1 \rightarrow \ldots \rightarrow P_n$ is a mixed sequence of absorption and intra-construction transformations of the example set E into the propositional Horn clause program $P = P_n$.

Definition 7 The algorithm $Duce_{(Abs,Intra)}(E)$ returns a set of possible Horn clause programs $H = \{P: P \text{ is an inverse derivation of } E\}$.

We can see H as being the hypothesis space of an algorithm which returns a single hypothesis. Angluin [1] introduced the notion of a *characteristic sample* set of examples for language L as being a set of examples which are sufficient to allow the inference of L. Here we use the term somewhat loosely to define a set of examples which induces a hypothesis space containing a given logic program P. If we can show that for any arbitrary propositional Horn clause program P we can generate a characteristic sample set, it follows that there is a sample set from which any P can be induced. This in turn would show that given a large enough set of examples, which in the limit must contain a characteristic sample, the Duce operators are sufficient to learn any propositional Horn clause program.

Definition 8 Given a propositional Horn clause logic program P we say that E is a characteristic sample of P for algorithm $Duce_{(Abs,Intra)}$ if and only if E is a legitimate example set of P and $P \in Duce_{(Abs,Intra)}$.

Before showing how to construct a finite characteristic sample for any logic program we will introduce the auxiliary notion of an isolated reference.

Definition 9 The clause $(h \leftarrow ap) \in P$ is said to reference predicate symbol p.

Definition 10 The clause $C \in P$ contains an isolated reference to predicate symbol p if and only if C is the only clause within P which references p.

Remark 1 If Abs^{-1} is applied to $P \supseteq \{C_1, C_2\}$ to produce $P' = P - \{C_2\} \cup \{C_1 \cdot C_2\}$ then the operator Abs^{-1} reduces the number of clauses which reference predicate symbol $p \in vocab(P)$ by one.

Remark 2 If $Intra^{-1}$ is applied to $P \supseteq (A \cup BB)$ to produce $P' = P - (\{A\} \cup BB) \cup CC$, where $A = (h \leftarrow \alpha p)$ contains an isolated reference to p, $BB = \{(p \leftarrow \beta_1), \ldots, (p \leftarrow \beta_n)\}$ is the set of all clauses containing the predicate symbol p in their heads and $CC = \{(h \leftarrow \alpha\beta_1), \ldots, (h \leftarrow \alpha\beta_n)\}$ then the program P' does not contain the predicate symbol p.

The following algorithm $Char_{(Abs,Intra)}$ can be used to generate a characteristic sample of a given propositional Horn clause logic program P.

algorithm $Char_{(Abs,Intra)}(P)$
  let $i = 0, P_0 = P$
  until P_i is a legitimate example set of P **do**
    if there is an A in P_i such that A contains an isolated reference to p
      P_{i+1} is the result of applying $Intra^{-1}$ to remove p in P_i
    else
      P_{i+1} is the result of applying Abs^{-1} to remove reference A in P_i
    let $i = i + 1$
  done
  let $f = i$
  E is P_f
  return(E)
end *Char*

Now we must show that this algorithm will generate a characteristic sample for any propositional Horn clause logic program.

Theorem 1 $Char_{(Abs,Intra)}(P)$ returns a characteristic sample for any propositional Horn clause logic program P.

Proof Let $E = Char_{(Abs,Intra)}(P)$. According to definition 8 E is a characteristic sample of P for $Duce_{(Abs,Intra)}$ if and only if E is legitimate and $P \in Duce_{(Abs,Intra)}(E)$. Let us assume that E is not a characteristic sample of P.

We will first look at the case in which the **until** loop in $Char_{(Abs,Intra)}$ terminates. According to the loop termination condition, P_f must be a legitimate example set of P. Since each step i in the derivation $P \rightarrow \ldots$

$\rightarrow P_f$ was carried out by either Abs^{-1} or $Intra^{-1}$ it follows that the sequence of transformations $(E = P_f) \rightarrow \ldots \rightarrow P$ is an inverse derivation of P from E. It follows from definition 7 that $P \in Duce_{(Abs,Intra)}(E)$, and thus E is a characteristic sample of P. We must therefore assume that the **until** loop does not terminate.

Let p be some predicate symbol in $vocab(P) - prim(P)$. By definition there must be clauses which reference p in P. These references will be reduced one by one by the **else** statement (Remark 1), with the last reference being removed by the **if** statement, together with all remaining occurrences of p in P (Remark 2). Referenced predicate symbols will be removed one by one until only unreferenced predicate symbols remain for some P_j. From repeated application of Lemma 1, P subsumes P_j. Moreover the vocabulary of P_j will have been successively reduced from that of P by applications of $Intra^{-1}$ (Remark 2). Thus by definition P_j is legitimate and the **until** loop will terminate with $f = j$. This contradicts the assumption and completes the proof. □

We now investigate the size of the characteristic sample set for a given propositional Horn clause logic program.

Theorem 2 Let $E = Char_{(Abs,Intra)}(P)$ and Ps be the set of referenced predicate symbols in P. The size of the characteristic sample set $|E| = |P| - |Ps|$.

Proof From definition 2 Abs^{-1} applies the transformation $P' = P - C_2 \cup C$, and therefore $|P'| = |P|$. From definition 4, $Intra^{-1}$ applies the transformation $P' = (BB \cup \{A\}) \cup CC$, where $|BB| = |CC|$. It follows that for $Intra^{-1}$, $|P'| = |P| - 1$. In the proof of Lemma 1 we have shown that referenced predicate symbols are removed one by one using $Intra^{-1}$. All other transformations employ Abs^{-1}. Since there must therefore be $|Ps|$ applications jof $Intra^{-1}$ it follows that $|E| = |P| - |Ps|$. □

Thus not only have we shown that the operators *Abs* and *Intra* are sufficient to learn any arbitrary propositional program but also, surprisingly, fewer examples are needed to induce a propositional program than there are clauses in that program. This is counter-intuitive to the normal belief in inductive knowledge engineering, in which we expect to use a large number of examples to induce a small number of rules.

3.3. Search: Duce macro-operators

Duce presently searches through the set of conjunctions of predicate symbols to find which operator to apply next. For this reason Duce can be myopic in its choice of new terms. The discrepancy in complexity between Duce's solution and human solutions (see Section 2.1) seems to indicate that this problem can be quite severe. However, by considering the characteristic set generating algorithm of Section 3.2 as being an

inverted strategy for propositional program construction, we have discovered a simple and effective method of improving Duce's present search mechanism. The argument is as follows. The algorithm $Char_{(Abs,Intra)}$ removes intermediate concepts (predicate symbols) one at a time. For every predicate symbol p removed, $Char_{(Abs,Intra)}$ applies Abs^{-1} repeatedly to remove all but the last reference to p. p is finally removed from the vocabulary by application of $Intra^{-1}$. In reverse this strategy becomes

(1) introduce p using *Intra*

(2) apply *Abs* to all clauses with head p.

This can be viewed as a form of macro-operator. Attempts are presently being made to implement this and other macro-operators within Duce. One severe impediment to the approach has been that whereas the symbol reduction effect of the old Duce operators can be efficiently and accurately computed using the characteristic equations of [4], no efficient method of computing the exact effect of macro-operators has been found outside application and measurement. It is estimated that application and measurement would slow the execution of the Duce algorithm by a factor of 1000 on applications the size of the KPa7KR experiment. However, various methods of approximating the evaluation have been tried, the most effective of which led to a 20 per cent compaction of the KPa7KR result [4].

4. DISCUSSION

Although much progress has been made in applying Duce to various problem domains, the present aim of our research is to extend Duce's capabilities. For these purposes it has been necessary to work out the theory underlying the Duce approach in more detail. In so doing we have discovered that Duce is a form of inverse resolution theorem prover. Many useful insights into possible improvements and extensions of the Duce algorithm have resulted from this, some of which are described in Section 3.

The two most interesting adaptions of Duce seem to lie in the directions of changing the underlying knowledge representation to first-order predicate calculus and dealing with noisy data. Although other authors have looked at related problems [8, 12, 10], all such attempts have dealt with learning single predicates, none with the more difficult problem of automatic vocabulary extension.

Acknowledgements

This paper describes work which was funded in part by the British Government's Alvey Logic Database Demonstrator. Research facilities were provided by the Turing Institute,

Glasgow, UK, Interact R&D Corporation, Victoria, BC, Canada and the US Army Office of Research in the Behavioral and Social Sciences through their Science Coordination Office in London.

REFERENCES

1. Angluin, D. (1982). Inference of reversible languages. *JACM*, **29**, pp. 741–65.
2. Bain, M. (1987). *Specification of attributes for computer induction.* TIRM, The Turing Institute, Glasgow.
3. Michalski, R. and Larson, J. (1978). *Selection of most representative training examples and incremental generation of VL1 hypotheses: the underlying methodology and the description of programs ESEL and AQ11.* UIUCDCS-R 78-867, Computer Science Department, Univ. of Illinois at Urbana-Champaign.
4. Muggleton, S. H. (1987). Duce, an oracle based approach to constructive induction. In *IJCAI-87*, pp. 287–92, Kaufmann.
5. Muggleton, S. H. (1987). *Towards constructive induction in first-order predicate calculus.* TIRM, The Turing Institute, Glasgow.
6. Plotkin, G. D. (1971). *Automatic methods of inductive inference.* PhD thesis, Edinburgh University.
7. Quinlan, J. R. (1979). Discovering rules by induction from large collections of examples. In *Introductory Readings in Expert Systems* (ed. D. Michie), pp. 33–46, Gordon and Breach, London.
8. Quinlan, J. R. (1986). Learning from noisy data. In *Machine Learning Volume 2* (eds R. Michalski, J. Carbonnel, and T. Mitchell), Kaufmann, Palo Alto, CA.
9. Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *JACM*, **12** No. 1, 23–41.
10. Sammut, C. and Banerji, R. B. (1986). Learning concepts by asking questions. In *Machine Learning: An Artificial Intelligence Approach* (eds R. Michalski, J. Carbonnel, and T. Mitchell), **2**, pp. 167–92, Kaufmann, Los Altos, CA.
11. Shapiro, A. D. (1987). *Structured induction in expert systems.* Turing Institute Press in association with Addison-Wesley.
12. Shapiro, E. Y. (1982). *Algorithmic program debugging.* PhD thesis, Yale University.