社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－MPS－63 (1)
2007／3／3

モデル化言語を用いた機械学習の抽象化について

丹 羽 竜 哉

産業技術総合研究所 情報技術研究部門

機械学習の側面のひとつとして学習対象の抽象化あるいはモデル化であるという考え方ができる。また、関数型言語を中心として高度な抽象化記述力を持った言語が開発されている。この抽象化記述力を機械学習に用いる事によって、学習結果の抽象的記述の獲得、および機械学習の過程そのものの抽象的モデル化が可能となる。学習結果が抽象的記述される事によって学習対象がいかに抽象化されたかが明確となり、更なる学習過程への応用が期待できる。また、学習過程のモデル化によって学習アルゴリズムの変換や合成が可能となる他、学習の正確さについても検討が可能となる。これらの性質によって得られる利便性について考察する。

Abstraction of Machine Learning by Modeling Language

Tatsuya NIWA

Information Technology Research Institute

National Institute of Advanced Industrial Science and Technology

We see machine learning as abstraction or modeling of learning object. In these days many kinds of programming languages, like functional programming languages, are developed which have strong abstraction description power. It is possible that obtaining abstracted description of learning results and abstracted modeling of the machine learning procedure, using this abstraction description power on the machine learning. From the abstracted description of learning results, it becomes clean how the learning objects are abstracted, and make use toward next learning process. Then, conversion and composition of learning algorithm are enabled by modeling of the learning process, and reliability of learning process can be considered. We discuss about their useful property.

1. はじめに

機械学習とは何か？ その側面のひとつに、学習対象の抽象化、あるいはモデル化であるという捉え方ができる。つまり、機械学習とは、学習対象が持つ性質の特徴を抽出し、それらの間の関係を数理的（抽象的）なモデルに当てはめる事（モデル化）と考える事ができる。従って、抽象化、モデル化とはどういう事なのかを考え直す事によって、改めて機械学習そのものを見直す事が可能であろう。

抽象化やモデル化については、機械学習の分野の他にも言語理論やプログラミング技術などの分野で多くの議論がなされていて、これを使った機械学習の研究も古くからある。例えば、1980 年代には第五世代コンピュータプロジェクト（ICOT）において論理

型言語（Prologや KL1 等）による帰納推論を行なうシステムを研究開発していた事はよく知られているが、これも言語の持つ抽象化記述力を利用した機械学習の一例と言える。但し論理型言語を利用しているため、学習の結果が論理式（ルール）の形式に引きずられる傾向がある事は否めない。もっとも、如何なる言語を用いても、それの表記法に引きずられる事は避けられないものである。

機械学習を利用するためのシステムとして、データマイニング用のプラットフォームがいくつか実用されている。それらは、統計処理用の専用言語（S-Plus, R）や、あるいはオブジェクト指向言語 Java（Weka）を使用しているが、学習アルゴリズムの記述という点において特に工夫はない様に思われる。

プログラミング言語の分野における近年の抽象化の理論の集大成として、関数型言語などを中心に高度な抽象化記述力を持つ計算機言語が開発され、実用化されている。この抽象化記述力を機械学習に利用する事で、学習結果の抽象的記述の獲得、また学習の過程そのものの抽象的モデル化を可能とする事が期待できる。

本稿では、純粋関数型言語 Haskell を利用した機械学習の研究を紹介し、その特徴について検討する。また、それとは別に独自に提案したモデル化言語を用いた研究も、同時に紹介する。

2. 関数型言語と機械学習

関数型言語とは何かという問いには様々な意見があり一定した考え方はない様であるが、多相型、遅延評価、高階関数など、抽象化レベルの高いプログラミングを行う為のしくみのうち幾つかが実装されている言語をいうと合意がある。数学的に自然な記述ができるという特徴があり、同時にプログラミングの誤りを少なくし、可読性を上げてプログラムコードを保守しやすくする事を目指している言語であるが、必ずしも人間にとってプログラミングしやすいとは言えない場合が少なくない様である。これは、人間の思考方法が、関数型言語の特徴である「数学的に自然な記述」という方向性と合っていないからなのではないかと推察できる。もちろん、関数型言語にも実際にプログラミングのしやすさにつながる性質はいくつかあり、それらは他の言語にも取り入れられ実用に供されている。しかしながら、関数型言語が数学的な緻密さを求めるあまり、人間にとってのプログラミングのしやすさという本来の目標から離れてしまっていると言えるのではないだろうか。もっとも、この辺は多くの反論があるかもしれない。

この「数学的に自然な記述」というのは、プログラミングだけでなく機械学習の分野にも大変に有効な考え方である。つまり、機械学習には未知のデータに対して何らかの方法で「正しい」と思われる答えを出す事（汎化）が求められるが、その為には何らかの仮定（事前知識）が必要となる。その仮定のひとつとして、「データは数学的に自然である」という考え方がある。機械学習が汎化を行なう際には、この数学的な自然さを拠り所として行なわれる場合が少なくない。

さて、機械学習に数学的な自然さが必要であるとするなら、関数型言語の持つ数学的に自然な記述という特徴を利用できるであろうと容易に想像できる。本研究の動機は、この辺りにある。

関数型プログラミング言語には Haskell 以外にも LISP/Scheme や ML (Meta Language) 等があるが、Haskell は純粋関数型言語の総合的な実装として、多相型や遅延評価などのしくみのほとんどが実装された言語である。

3. 遺伝的プログラミングによる学習

プログラムの自動合成は、人工知能研究の重要なテーマのひとつであるが、Haskell の強力な抽象化記述力を利用して、これを行なう事が考えられよう。Yu ら[1,2,3]は、Haskell の持つ多相型(Polymorphism)のしくみを利用して、遺伝的プログラミングによる学習、自動プログラム合成を行なっている。Haskell は、強い静的な型付けを持ち、型推論によってコンパイル時に型の整合性が検証される為、実行時にエラーが出ないという特徴があるので、誤ったプログラムの生成が抑えられる。多相型の学習によって、高階関数の合成が可能となる事が示されている。この高階関数により学習された部分関数

の再利用が可能となり、探索効率の向上が図られている。

4. 機械学習の形式言語による記述

Allison[4]は、Haskell を用いた学習モデルの記述を行なっている。これは基本モデルの他に関数モデルや時系列モデル等いくつかの学習のモデルを記述し、それぞれのモデル間の相互の書き換えについて示している。また、MML (Minimum Message Length) という AIC (Akaike Information Criterion) に類似した基準、すなわちモデルの記述長とそのモデルの下でのデータの記述長の合計を最小にする基準を用いる事により、学習モデルの記述長の増大をふせぎ、過学習を避ける事が提案されている。その観点においては、Haskell の持つ多相型や型クラス、高階関数などの性質が関数の再利用に有効であり、モデルの記述長を短くする為に効果的である。

Okita ら[5]は、学習の問題を独自のモデル化言語で記述する事によって、問題そのものを扱い易くする方法を提案している。これは、学習アルゴリズムとデータ（トレーニング、テスト両方）をセットにしたものをペアで記述し、別に用意された幾つかの書き換え規則を用いて、アルゴリズムの変換や並列化、次元数の増減などを行なう事を目指している。学習アルゴリズムそのものを記述するものではなく、知られているアルゴリズムを変換していく事に主眼を置いている。特に並列化した場合の信頼性の範囲を記述する為に、テスト出力データも記述できる点が特徴である。位置付けとしては、プログラム開発における UML (Unified Modeling Language) に相当するものを機械学習システムの開発で用いる事を目指している。また、新しい学習アルゴリズムを開発した時でも、このモデル化言語で記述する事によって直ちに利用できるようにする事も目標としている。

5. 考察

Haskell を用いた２つの例について考えてみると、一方は遺伝的プログラミングによるプログラムの自動合成、他方は関数型言語による機械学習のモデル化と、目的に大きな差異がある。しかし、C等の手続き型言語に較べて短く簡潔に書ける事を利用している事や、関数型言語の持つ性質の多相型や高階関数、再帰を積極的に利用し関数の再利用を図っている事等、共通点も少なくない事がわかる。これは機械学習の前提である事前知識として、Haskell の持つこれらの関数型言語としての性質が利用できる事を示すものに他ならない。但し、「数学的に自然な記述」という観点から考えると、双方向の計算が可能な論理型言語を使用した方が、単方向の関数型言語より自然な記述ができる場合があるという研究結果もあり[6]、関数型言語が最適かどうかについては検討の余地がある。

機械学習は、学習対象のモデルを幾つかの類型に分けて考えるとわかりやすい。また、それぞれの相互間の書き換え規則を形式化できれば、更に理解が深められ、また並列化など環境に合わせて利用する事も期待できる。本稿で紹介した２件の例では、書き換え規則はもちろんモデルの類型もまだまだ不完全なものであり、今後も様々なモデルとその間の書き換え規則の開発が必要であるが、関数型言語あるいはモデル化言語によって形式的な記述を行なう事で、システムとしての機械学習が理解しやすく使いやすくなる事は十分期待できそうである。

更に、学習のモデルの記述と遺伝的プログラミング（GP）によるプログラム合成の記述言語が同一である事によって、モデルを記述するための関数を GP の要素として使う事が可能となる。モデルを記述するための関数は、データの統計的なパラメータを計算するものを多く含んでいる事が予想されるので、これを GP の要素として利用できれば、探索効率の向上が期待できる様に考えられる。

6. 今後の展望

近年、等価変換計算モデル（Equivalent Transformation Computation Model）という新しいプログラミングパラダイムが提案されているそうである。前項で示した通り、必ずしも関数型言語が最適という訳ではなさそうであり、論理型言語等とも併せて、これについても検討してみたい。また、実際に動くシステムを組んで実証する事は、もちろん急務である。

謝辞　本研究の動機を与えて下さり、多くの助言や議論を頂いた産総研情報処理研究部門大蒔和仁元部門長、情報技術研究部門坂上勝彦部門長、およびブリュッセル自由大学 Bernard Manderick 教授、大北剛様その他の皆様に対し心より感謝いたします。

参考文献

[1] Tina Yu and Chris Clack, "PolyGP: A Polymorphic Genetic Programming System in Haskell," Genetic Programming 1998: Proceedings of the Third Annual Conference, pp.416-421, 1998
[2] Tina Yu and Chris Clack, "Recursion, Lambda Abstractions and Genetic Programming," Genetic Programming 1998: Proceedings of the Third Annual Conference, pp.422-431, 1998
[3] Tina Yu, "Polymorphism and Genetic Programming," Proceedings of the Fourth European Conference on Genetic Programming, pp.218-231, 2001
[4] Lloyd Allison, "Models for Machine Learning and Data Mining in Functional Programming," Journal of Functional Programming, 15 (1), pp.15-32, 2005
[5] Tsuyoshi Okita and Tatsuya Niwa, "Modeling Language for Machine Learning," Transactions on Engineering computing and Technology, vol.6, pp.102-105, 2005
[6] 立木秀樹, "実数計算の GHC による実現," コンピュータソフトウェア, vol.18, no.2, pp.40-53, 2001