読み返しによる学習

5L-3

岡　夏樹

新世代コンピュータ技術開発機構

1．背景および問題の概要

我々は本を読むとき，ていねいに一度だけ通読するより，手早く何度か読み返すほうがよく理解できることが多い．これは次のような理由によると思われる．

一つの理由は，ある部分の理解に必要な情報が後になって初めて出てくることがしばしばあるということである．これは著者がいいかげんで書く順序になんの配慮もしなかった場合はもちろん，そうでなくても，著者は書こうとする内容全体についてすでによく知っているから，読者にも分かるつもりで書いてしまったが，その理解のためには実は後に書かれる知識が必要だったということが起こりがちである．また，十分注意したとしても，本に書かれる知識は前後で複雑に関係していることが多く，その場合理解に必要な情報が後から出てくるという状況は避けられない．このような時は，分かりにくいところにはこだわらず読み進み，適当に読み返すことが有効である．

このほかに，ある事柄がいろいろな仕方で繰返し説明されているときにも，分かり易いところを先に理解しておけば，それをもとに，分かりにくかったところを読み返しの際に容易に理解できる可能性がある．

もう一つの理由は，初めに全体の概要をつかむということである．これにより個々の知識間の関係が分かり，知識を適切に位置付けることができる．また，重要な事柄とそうでない事柄との区別や，他の箇所の理解に必要な知識とそうでない知識との区別をすることができ，効率的に学習できる．

三つ目の理由は，人は一度に取得可能なすべての情報を得，またそれを保っているわけではないということである．したがって読み返しの際に新たに情報が得られることは多い．

以上のような状況における読み返しを利用した学習を「読み返しによる学習」(Learning by Rereading) と呼ぶことにする．本研究は人の学習過程の理解のためにそれを模擬する事を目標とするが，工学的に見ると，読み返しにより，学習できる対象の範囲を広げ，学習の効率を上げられる可能性がある．

本からの学習の実現のためには自然言語理解研究の成果を待たねばならないが，以下では，自然言語の問題を含まない，例からの学習に焦点を絞ってさらに具体的に検討する．なお，本論文において「読む」(read)とは，かならずしも本のreadだけでなく一般的にデータの readを意味する．

2．例からの学習における読み返し

例からと直接明示的に教えられることからとの複合による学習を考える．このどちらか一方だけにより知識を表わすのは難しい場合が多く，両者の複合という設定は自然で実際的である．

例の呈示法には一つずつ順に示す方法（逐次法）と一括して示す方法（一括法）とがある．逐次法の利点は，学習しやすいように本質的な例を先に与えることができることと，学習の途中でもそれまでに示された例によりそれなりに学習されていることとである．一方欠点は，学習結果が例の呈示順の影響を大きく受けることと，論理和の形の概念の学習やノイズを含む例による学習が難しいこととである．一括法はこれとほぼ反対の性質を持つ．

前節で述べた，ある部分の理解に必要な情報が後になって出てくるという状況は，例からの学習においては例の呈示順が悪いことに相当する．また，例とその解説（明示的な知識）を合わせて与える場合には，後のほうで解説する要素が前のほうの例に含まれてしまうことがよくある．

ここでは例は逐次示されるとし，呈示順は理想的ではないが，ある程度考慮されているとする．この呈示順を利用することにより，一括法より効率良く学習できる可能性がある．一方，単純な逐次処理と比べて，以下の点を改善することを目標とする．

- 例の呈示順の影響の緩和
- 探索空間が大きい場合の効率的な学習
- 論理和の形の概念の学習
- ノイズを含む例からの学習

このために，以下に示す機能を検討している．

2．1．例の保持と読み出し

全ての例は呈示順に２次記憶に貯えられ，逐次的に順方向あるいは逆方向に読み出すことができる．一度読んだときに以下に述べる保留などのマークを付けておいて，効率的に読み出すこともできる．

2．2．探索の保留

Learning by Rereading

Natsuki OKA

Institute for New Generation Computer Technology

Mitchellの候補消去アルゴリズム[1]やこれと類似の他の学習アルゴリズムは，大きな規則空間に対しては遅過ぎて使い物にならないと言われている．また，G集合より一般的な正の例やS集合より特殊な負の例が呈示されたら，論理和を含む概念であると仮定して，論理和に分割して処理をする方法[1][2]が提案されているが，うまく学習できるかどうかは例の呈示順に強く依存している． Far miss に対応して論理和に分割する方法[3]も提案されているが，これも例の呈示順に強く依存する．

これらの方法に対して，人の学習の仕方は次の点で異なると思われる．人は大きな規則空間では，通常は候補消去アルゴリズムには従わない．例の数が少ないときや，規則空間上で離れた正の例や負の例が呈示されたときのように探索すべき空間が大きいときにはその例の処理を保留するのが普通である．この後，他の例が集まって，次の項目で述べる概要の把握により，あるいは明示的に教えられたことにより，探索空間が狭まったところで保留しておいた例を読み返す．このとき，探索空間が十分小さくなっていれば，候補消去アルゴリズムに従えばよい．

このようにして，論理和の形の概念やノイズを含む例が存在しても，例の呈示順にあまり依存せず，無駄な探索や後戻りを避けて効率的に学習をしていると思われる．

候補消去アルゴリズムと人の通常のやり方との違いは，人の扱える探索空間が狭いことや，普通は規則空間が完全に与えられていないことにも起因する．

2．3．概要の把握

一度の通読で概要を把握しておくと，次に読むときそれを利用して効率良く学習できる．あるいは，初めの方を読むときに概要を把握して，続きはそれによって例を処理しながら読んでもよく，読み進めるうちにその把握内容が適当でないことが分かったら，その時点で変えればよい．

把握すべき項目には次のようなものがある．
- 規則空間中で重要な属性とそうでない属性
- ノイズを含む例
- 論理和の分割の仕方

ここでは多数の例の統計的性質から，これらを推測する方法を提案する．以下に挙げるのは説明のための簡単な例題である．考慮すべき属性の候補とそのとる値（規則空間）が与えられているとする．たとえば，

a={a1,a2},b={b1,b2},c={c1,c2,c3},d={d1,d2}.

学習すべき概念は，たとえば

a=a1 and (b=b1 or c=c1)

のような論理式であり，連言標準形で考えることにする．次のような例とその正負が任意の呈示順で与えられる．

(a=a1, b=b1, c=c1, d=d1)　正，
(a=a1, b=b1, c=c1, d=d2)　正，
:
(a=a1, b=b2, c=c2, d=d1)　負，
:
(a=a2, b=b2, c=c3, d=d2)　負

全ての場合が片寄りなく呈示されるとすると，十分な数の呈示の後では（簡単のため，各属性が等確率でその属性値をとる場合を示すと），次のような統計量が得られる．

r(pos|a=a1)=0.66,　r(pos|a≠a1)=0.0,
r(pos|b=b1)=0.5,　r(pos|b≠b1)=0.17,
r(pos|c=c1)=0.5,　r(pos|c≠c1)=0.25,
r(pos|d=d1)=0.33,　r(pos|d≠d1)=0.33.

ここに，r(pos|a=a1) はa=a1の例が正の例である割合を表わす．

さて一般に，ある属性（たとえばa ）に注目すれば連言標準形の論理式は，次のように表わせる．

(a=x or G) and H.

ここに，G はリテラルの選言からなる任意の論理式，H は連言標準形の任意の論理式である．a=x，G，H が真であるかどうかは，それぞれ独立であると仮定すると，

g=P(pos|a≠x)/P(pos|a=x),　h=P(pos|a=x)

が成立する．ここに，g,h はそれぞれG,H が真である例が呈示される確率，P(pos|a=x)はa=x の例が正の例である確率を表わす．

これらのことから，次のように推測できる．
- (a=a1 or G) and H とすると，g〜0.0，h〜0.66．
  したがって，a=a1 and H　の形であろう．
- (b=b1 or G) and H とすると，g〜0.33，h〜0.5．
  したがって，(b=b1 or G) and H　の形であろう．
- (c=c1 or G) and H とすると，g〜0.5，h〜0.5．
  したがって，(c=c1 or G) and H　の形であろう．
- (d=d1 or G) and H とすると，g〜1.0，h〜0.33．
  したがって，d は無関係であろう．

さらに，以上の情報から概念に含まれている可能性がある(b=b1 or c=c1)について統計情報を調べることもできる．また，以上の方法によりノイズの推測をすることもできる．

上述の仮定が成立していないときには，推測が不正確になるが，重要な属性とそうでない属性，論理和の分割の仕方を大まかにつかむことはでき，これを利用して効率的に学習できる．この段階では，最終的には必要だがそれほど重要でない属性までとらえる必要はなく，次の読み返しの際に考慮すればよい．これを必要なだけ繰り返す．

読み返しの効果は，考慮すべき規則空間が完全には与えられていないときに大きいと考えられるので，将来はこの場合も検討したい．

［参考文献］

[1] Mitchell, T.M. et al., Learning by Experimentation: acquiring and modifying problem-solving heuristics, in: R.S. Michalski et al (Eds.), Machine Learning (Tioga,1983) 163-190.

[2] Bundy, A. et al., An Analytical Comparison of Some Rule-Learning Programs, Artificial Intelligence 27 (1985) 137-181.

[3] Langley, P., Language acquisition through error recovery, CIP Working Paper 432, Carnegie-Mellon University, June 1981.