B-016

Take: Python におけるデータマイニング支援ツール
Take: Dataminig Supporting Tool in Python

羽室 行信 * 宇野 毅明 † 中元 政一 * 中原 孝信 ‡ 丸橋 弘明 *
Yukinobu Hamuro Takeaki Uno Masakazu Nakamoto Takanobu Nakahara Hiroaki Maruhashi

1. まえがき

Take はデータマイニングの実施を支援するために開発された Python モジュールである。相関ルール列挙や系列パターンマイニング、データ研磨、クリーク列挙など多様なメソッドを提供する。本モジュールは、データマイニングに関する高度なスキルを持たないユーザでも利用できるように設計されている。本稿では、Take が提供するメソッドの内容とその性能、そして応用例について紹介する。Take は Nysol モジュールの一部として配布されおり [4]、pip など一般的な方法で容易にインストールできる。

2.nysol.take の構成

Take(竹) とは、宇野を中心にして開発されたデータマイニングソフトウェア [6] を Python の拡張モジュール nysol.take として利用できるようにしたものである。モジュール名は開発者の名前の音からとったものである。本モジュールは、大きく 2 種類のメソッドを提供している。一つはオリジナルのソフトウェアを、そのままの形で利用できるようにした core メソッドで、表 1 に示される 7 種類のメソッドが提供されている。

表 1. NYSOL.TAKE.CORE が提供する 7 つのメソッド

#	メソッド名	内容
1	take.core.lcm()	アイテム集合列挙
2	take.core.lcmseq()	系列パターン列挙
3	take.core.mace()	極大クリーク列挙
4	take.core.sspc()	類似アイテムペア列挙
5	take.core.medset()	アイテム集合の共通集合
6	take.core.simset()	simset
7	take.core.grhfil()	グラフの変換

他方は、nysol.mcmd[2] を組み合わせて、より利用しやすくした応用メソッドで、9 つのメソッドを提供している (表 2)。core メソッドでは、アイテムのシンボルとして整数を用いなければならず、また出力も独自フォーマットによるテキストであり、応用メソッドではそれらの制約が取り除かれている。ただし、そのためにシンボルの変換などの余分なオーバーヘッドが発生するために、core メソッドに比べて処理速度は劣る。

表 2. NYSOL.TAKE が提供する 9 つのメソッド

メソッド名	内容	利用 core
mitemset	アイテム集合列挙	1
msequence	系列パターン列挙	2
mtra2gc	トランザクションからの類似度グラフ生成	4
mpolishing	グラフ研磨	4,7
mclique	クリーク列挙	3
mbipolish	二部グラフ研磨	4,7
mbiclique	二部グラフクリーク列挙	1,4,7
mfriends	相互ランク情報によるルール選択	-
mpal	ランク情報によるルール選択	-

3.Online Store データを用いたケース

ここで、Take モジュールを用いた分析ケースを示す。利用したデータは、UCI 機械学習レポジトリの Online Retail データセットである [7]。表 3 に示されるような、主に問屋を対象にギフト商品を販売するオンライン店舗の顧客購買履歴データである。

表 3. ケースで用いたオンライン店舗の売上データ [7]。

InvoiceNo	StockCode	Description	Quantity
536370	21883	STARS GIFT TAPE	24
536370	POST	POSTAGE	3
536390	22174	PHOTO CUBE	48
:	:	:	:
InvoiceDate	UnitPrice	CustoerID	Country
2010/12/1 8:45	0.65	12583	France
2010/12/1 8:45	18	12583	France
2010/12/1 10:19	1.48	17511	United Kingdom
:	:	:	:

3.1. 相互ランク情報による相関ルールの列挙

相関ルール分析は、データマイニングの分野で代表的な分析手法で、特にルールを高速に列挙する技術は飛躍的な進展を遂げてきた。しかしながら、パラメータの設定次第では時に大量のルールが出力され、そこから興味深いルールを抽出するまでにユーザに多大な負担を強いることも少なくない。

その問題を解決する一つの方法として相互ランク情報に基づいたルールの抽出方法が提案されている [1]。Take モジュールでは、mfriends 及び mpal メソッドとして実装されている。この手法の特徴は、相関ルール列挙において 2 アイテムルール $A => B(|A| = 1, |B| = 1)$ のみを列挙し、そこから A, B 相互に関連の強いルールを選択するというものである。$A => B$ 及び $B => A$ の評価指標 (support や confidence) が、それぞれの前件部を共通としてもつルール集合の中でユーザが指定した k 位以内であるとき、アイテム集合 A と B の関連が強いと考える。図 1 は、OnlineStore のデータから、そのようなルールを列挙する Python コードである。

```
1  import nysol.mcmd as nm
2  import nysol.take as nt
3  import networkx as nx
4  import matplotlib.pyplot as plt
5  f=None
6  f <<= nm.mcut(f="InvoiceNo,StockCode",i=iFile)
7  f <<= nm.muniq(k="InvoiceNo,StockCode",o="tra.csv")
8  f.run()
9  nt.mitemset(S=100,tid="invoiceNo",item="StockCode",l=2,u=2,i="tra.csv",O="
       is2").run()
10 f=None
11 f <<= nm.msplit(f="pattern",a="item1,item2",i="is2/patterns.csv")
12 f <<= nm.mcut(f="item1,item2,lift",o="rules.csv")
13 f.run()
14 nt.mfriends(ef="item1,item2", ei="rules.csv", ef="item1,item2", sim="lift",
       rank=5, udout=True, eo=friends.csv").run()
15 f=None
16 f <<= nm.mcal(c="cat(\" \",$s{item1},$s{item2})",a="edges",i="friends.csv")
17 f <<= nm.mcut(f="edges",nfno=True,o="edges.csv")
18 f.run()
19 G = nx.read_edgelist("edges.csv")
20 plt.figure(figsize=(10, 10))
21 nx.draw(G, pos=nx.spring_layout(G),node_size=40,iterations=20)
22 plt.savefig("friends.png")
```

図 1. ルールの相互ランク情報に基づいた 2 アイテム相関ルールの列挙とその可視化を実現するスクリプト。

5-8 行目は nysol.mcmd[4] により InvoiceNo を単位に StockCode をアイテムとするトランザクションデータを作成している。そして 9 行目で、最小サポート 100 の頻出 2-アイテム集合を列挙し、データを整形した後に、mfriends メソッドにより、2 つのアイテムの lift 値がお互いに 5 位以内になるような 2-アイテム集合のみ

*関西学院大学経営戦略研究科
†国立情報学研究所情報学プリンシプル研究系
†専修大学商学部

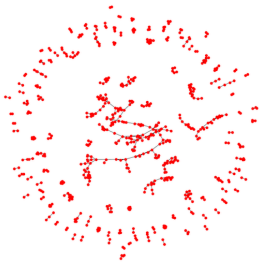

図 2. 図 1 によるルールの視覚化の結果

を選択し (14 行目)、それら 2 アイテムをエッジとするグラフを描画している (15-22)。単純に lift 値のみで頻出アイテム集合を列挙すると、いくつかの特定の商品にエッジが集中するようなグラフになることが多いが、`mfriends` による方法を用いれば、図 2 に見られるように、全体のアイテムの関係性を見渡せるようになる。

3.2. 顧客と商品のバイクラスタリング

顧客 $v \in V$ が商品 $u \in U$ を一定数以上購入していた時に枝を $(v,u) \in E$ を張るようような二部グラフ $G = (V \cup U, E)$ について、枝が密に貼られている 2 つの部の部分集合を抽出することで、商品の購入パターンが似た顧客集合を得ることができる。これはバイクラスタリングと呼ばれる手法である。G 上の密な部分集合の定義としては、極大二部クリークを用いることができるが、現実のデータにおいては例外的な接続関係が多く含まれるために、何の工夫もなければ、多数のクリークが列挙されることとなり、元のデータを小数のグループで表現するというクラスタリングの目的が損なわれてしまう。そこで、与えられた二部グラフ G を「研磨 (polish)」することで、元の性質をできる限り失わずに、劇的にクラスタの数を削減する方法が提案されている [3]。`nysol.take` では、そのような研磨処理を `mbipolish` メソッドで実現できる。図 3 は、OnlineStore のデータから顧客と商品の二部グラフを構成し、それを研磨し極大二部クリークを列挙する Python コードである。

```
1  import nysol.mcmd as nm
2  import nysol.take as nt
3  f=None
4  f <<= nm.mcut(f="StockCode,CustomerID",i=iFile)
5  f <<= nm.mdelnull(f="StockCode,CustomerID")
6  f <<= nm.mcount(k="StockCode,CustomerID",a="freq")
7  f <<= nm.mselnum(f="freq",c='[5,]',o="biG")
8  f.run()
9  nt.mbipolish(ei="biG",ef="StockCode,CustomerID", sim="R", th=0.3, eo="pol")
       .run()
10 nt.mbiclique(ei="biG",ef="StockCode,CustomerID", o="cliq_biG").run()
11 nt.mbiclique(ei="pol",ef="StockCode,CustomerID", o="cliq_pol").run()
```

図 3. 顧客と商品の二部グラフにおけるバイクラスタリング。

図 3 の行 3-8 では、購入回数が 5 回以上の商品と顧客のペアを選択することで二部グラフ G を構成している。9 行目で二部グラフを研磨し、新たなグラフ `pol` を得ている。そして 10 行目では、オリジナルの二部グラフから二部クリークを列挙し、11 行目では研磨後の二部グラフから二部クリークを列挙している。このようにして得られた 2 つのバイクリークの顧客の構成人数の分布を比較すると、研磨によりクラスタ数を劇的に削減できていることがわかる (表 4)。

4. ベンチマークテスト

nysol.take の処理速度を評価するために、Python で利用可能な機械学習モジュール Orange3[5] をベンチマークとした処理速度実験を実施した[‡]。評価に使ったコードは、図 4 に示される通りである。データは先に紹介した OnlineStore データを整形したもので、`InvoiceNo` を単位に `StockCode` をアイテムとするトランザクションデータである (org:トランザクション数 25900,item 数 4070,2.5MB)。さらに、アイテム数はそのままに、オリジナルデータに 3 割のノイズを乗せたデータを追加し、サイズ違いのデータ (s-10:10 倍,s-100:100 倍,s-1000:1000 倍) を用意した。計算方法として、最小サポートが 100(s-10,100,1000 はサイズに比例して設定) の頻出アイテム集合の列挙を取り上げた。実験結果を表 5 に示す。Take の core メソッドの lcm は orange の itemset に比べて、3-4 倍高速であることがわかる。

表 4. 二部クリークの顧客グループ構成人数の分布

size	クラスタ数 (研磨なし)	クラスタ数 (研磨あり)
1-2	2636	205
3-4	9027	31
5-6	6682	17
7-8	3156	4
9-10	1501	2
11-20	1512	6
21	194	0

表 5. ベンチマークテストの結果 (経過秒数)[†]。

program	関数名 (def)	org	s-10	s-100	s-1000
lcm	L1	0.269	1.427	19.39	250.3
orange	O1	0.569	5.598	58.72	778.4

[†] 実験で利用した PC: MacPro(2013),CPU: 2.7GHz 12-Core Intel Xeon E5, メモリ:64GB, ストレージ:USB3 HDD

5. おわりに

本稿で紹介した手法以外にも、一般グラフの研磨や顕在パターン列挙、アイテムの階層構造を考慮したルール列挙など、多様なデータマイニング手法を利用できる。これらの手法は、我々がこれまでに現実の課題において開発/利用してきたものであり、Python ユーザが増加する中、実業務の中で有効に利用されることを期待している。

```
1  import nysol.take.extcore as ntc
2  import Orange
3  from orangecontrib.associate.fpgrowth import *
4  def L1():
5    ntc.lcm(type="Ff",sup=minFreq,i=iFile1,o="xxrsl11")
6  def O1():
7    tbl = Orange.data.Table(iFile2)
8    X, mapping = OneHot.encode(tbl)
9    itemsets =frequent_itemsets(X, minFreq)
```

図 4. ベンチマークに利用した Python スクリプト

謝辞

本研究は、JST CREST(グラント番号: JPMJCR1401) の研究助成を受けている。

参考文献

[1] 岩崎幸子, 中元政一, 中原孝信, 宇野毅明, 羽室行信, グラフ構造による相関ルールの視覚化ツール：KIZUNA,2017 年度人工知能学会 (第 31 回), ウインクあいち,2017/5/24.

[2] 中元政一, 羽室行信, NYSOL: Python における大規模データ前処理支援ツール, FIT2018, 2018.

[3] 中原 孝信, 大内 章子, 宇野 毅明, 羽室 行信, 「データ研磨の 2 部グラフへの適用と Twitter からの意見抽出」,2016 年度人工知能学会（第 30 回）, 北九州国際会議場, 2016.6.6〜6.9.

[4] https://github.com/nysol/nysol_python

[5] https://github.com/biolab/orange3

[6] http://research.nii.ac.jp/ uno/codes-j.htm

[7] http://archive.ics.uci.edu/ml/datasets/online+retail

[‡] https://github.com/nysol/bench にて本稿で使用したベンチマークテストのコードを公開している。