2016年8月9日 河原林ERATO感謝祭III

Scalable Partial Least Squares Regression on Grammar-Compressed Data Matrices

田部井靖生
(東京工業大学/JST-さきがけ)
西郷浩人 (九州大学)
山西芳裕 (九州大学)
Simon.J.Puglisi (ヘルシンキ大学)

KDD'16のResearch Trackに採択

概要

- 解釈可能な統計モデルを大規模データ行列上で効率的に学習させるためのデータ圧縮法を提案
- Partial least squares (PLS)回帰モデル学習を文法圧縮されたデータ行列上で実装
- 大規模データを用いた実験で性能を検証

線形モデルの学習

- データはD次元特徴ベクトル$x \in R^D$として表現されていると仮定
  - 文章中の単語の有る無しを表現した0/1ベクトル (フィンガープリント)
  - 画像における各ピクセルの値を要素とするベクトル など
- 線形モデル：重みベクトル$w \in R^D$とxの内積で応答変数yを求める $y=w^T x$
- N個の特徴ベクトルと応答変数のペアー$(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)$が与えられたら, 重みベクトル$w$を学習する

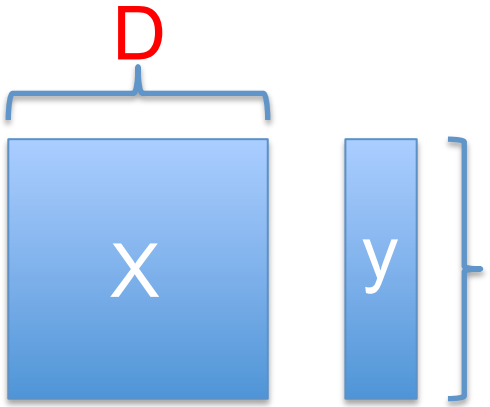
- D, N≒数億からなる大規模かつ超高次元データを仮定

大規模データから解釈可能な統計モデルを学習することは知識発見ための有効な手段

- 解釈可能な統計モデル
  - 線形モデル, 決定木, 決定集合など
- 利点：どの特徴が分類/回帰に有効かなどの解釈が可能
- 学習結果を説明する必要性が機械学習の応用場面で良くある
  - バイオインフォマティクス, ケモインフォマティクス, 多くのビジネスシーン, etc
- KDD'16の関連論文 : Lakkaju et al., Haichuan et al., Ribeiro et al.,Nakagawa et al.,

データ行列のメモリに比べて学習アルゴリズムのメモリは圧倒的に小さい

- 機械学習の入力 = データ行列Xとラベル列y



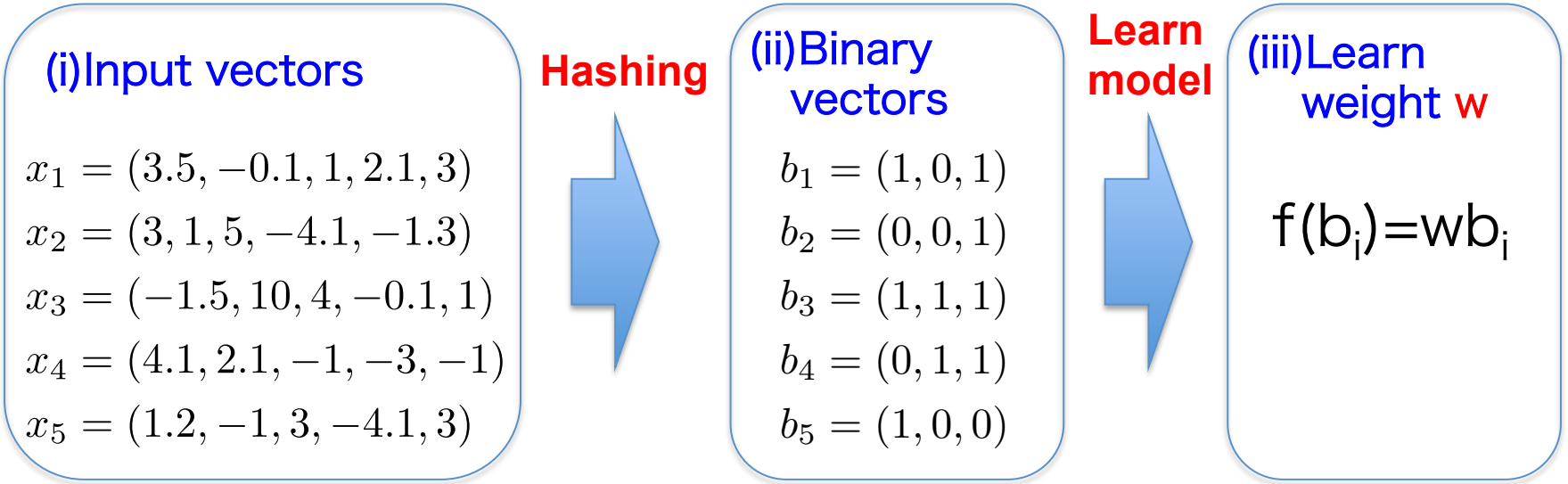
メモリ : O(ND)

- 殆どの学習アルゴリズムのメモリ: O(D)

- データ行列を圧縮した状態でモデル学習できれば巨大な行列上でも効率よく学習可能

乱択ハッシュを用いた機械学習法 [NIPS'11]

- データ行列の各特徴ベクトルを低次元ベクトルにハッシュし、モデルを学習する
  - 入力ベクトルのサイズ >> ハッシュ後のサイズ
  - ２つのベクトル間の距離は近似的に保存される
- 利点：精度を落とさずメモリ効率良く学習可



様々な乱択ハッシュを利用した機械学習法

- 入力ベクトルの形式とシュミレートしたいカーネルに応じて、様々な手法が提案されている
- 欠点：非可逆圧縮のため学習後のモデルの解釈不可

手法	カーネル	入力形式
b-bit MinHash [Li'11]	Jaccard	Fingerprint
Tensor Sketching [Pham'13]	Polynomial	実数ベクトル
0-bit CWS [Li'15]	Min-Max	実数ベクトル
C-Hash [Mu'12]	Cosine	実数ベクトル
Random Feature [Rahimi'07]	RBF	実数ベクトル
RFM [Kar'12]	Dot product	実数ベクトル
PWLSGD [Maji'09]	Intersection	実数ベクトル

解釈可能な統計モデルを学習するために データ圧縮に求められる条件

- スケーラブル
  - 巨大なデータ行列を圧縮したい
- 高い圧縮率
  - メモリ効率よく学習するため
- 可逆圧縮
  - 学習結果のモデル解釈するため
- 学習アルゴリズムに必要な行列操作をサポート
  - データ圧縮した行列上で学習アルゴリズムを実装するため

発表内容

- 研究の動機
  - 大規模データ上の機械学習
- 文法圧縮
  - Re-Pairアルゴリズム
  - Re-Pairアルゴリズムの大規模データへの適応
- 文法圧縮されたデータ行列上のPLS回帰モデルの学習
- 実験
- まとめ

文法圧縮：入力文字列だけを表現するチョムスキー標準形の文脈自由文法

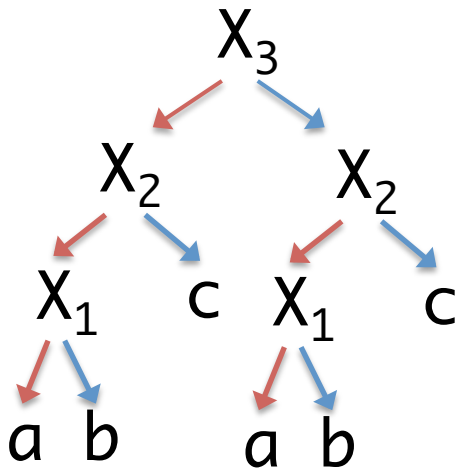
入力　　文法ルール　　構文木 (2分木)

abcabc

$X_3 \rightarrow X_2X_2$

$X_2 \rightarrow X_1c$

$X_1 \rightarrow ab$



- 反復文字列に対して有効な圧縮法
  - 同一の部分列を多く含むほど、少ないルール数で表現可能
- 文法は２分木としての良い表現を持つ
  - 圧縮されたデータに対する様々な操作を実装可能

Re-Pair [J.Lassarson and A.Moffat'99]

- 文字列中で最頻出する隣接文字ペアーを新しい非終端記号に置き換える貪欲アルゴリズム

1. 文字列中で最頻出する隣接文字ペアーabを発見する

2. 新しい文法ルール$X_i \rightarrow ab$を生成し, 文字列中のすべてのabの出現をX_iで置き換える

3. すべての文字ペアーの出現頻度が1になるまでstep1とstep2を繰り返す

Re-Pairの動作例

1. S中の最頻出はaa

   S=aaaaabcaaaaa

2. 文法ルール $X_1 \rightarrow aa$を生成する

3. S中のaaのすべての出現をX_1で置き換える

   S'=X_1X_1abcX_1X_1a

S'を入力として1, 2, 3を繰り返す。

頻度表
aa:8
ab:1
bc:1
ca:1

Re-Pair (続き)

- 利点：シンプルでありながら圧縮率が高い
- 欠点：線形時間で実装するためには多くのデータ構造が必要 (メモリ大)

  Ex：入力文字列の各位置に128bitのポインター，文字ペアーの頻度を管理するヒープやハッシュテーブル etc

省スペース化のための工夫 (その1)

- 位置ポインターを省略
- 頻出するtop-k個の文字ペアーからルールを作成
  - Top-kを十分大きく(k=10000)とれば高速

Re-Pairの省スペース化の工夫(その2)

- Re-Pairの問題点：すべての文字ペアーの出現頻度をハッシュテーブルに保持する必要性

- Re-Pair

文字列中に頻出する文字ペアー

=圧縮に貢献

- ストームマイニング

ストリームデータから頻出するアイテムを選ぶ

例：Lossy/Frequency Countings [VLDB'02, ESA'02]

- 文字列中に頻出するするルールのみを主記憶に残しつつ、高圧縮率を達成することが可能
  - 定数スペース化

データ行列の文法圧縮

- 入力のデータ行列はフィンガープリントを仮定
- 文法圧縮が有効な理由
  1. 同じクラスに属するフィンガープリントは同じ要素を共有する
  2. Gap-encodingにより同じ要素を共有しやすくなる

(i)データ行列

X_1=(1,3,4,7,9,13)
X_2=(2,3,7,9,11)
X_3=(1,3,4,7,9,11)

(ii)Gap-encoding

X_1=(1,2,1,3,2,4)
X_2=(2,1,4,2,2)
X_3=(1,2,1,3,2,2)

(iii)文法圧縮

X_1=(D,2,4)
X_2=(A,4,B)
X_3=(D,B)

D={D→C3,
B→22,
C→1A,
A→21}

Partial Least Squares (PLS) 回帰モデル

- データ行列の複数個の低次元空間に射影した上での線形モデル

$$f(x) = \sum_{i=1}^{m} \alpha_i w_i^\intercal x$$

- w_i：低次元空間へ射影するためのベクトル

$$w_i^\intercal \mathbf{X}^\intercal \mathbf{X} w_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

- α_i：各空間ごとの重みベクトル

- 利点：各空間ごとに予測に効く特徴を観測可

PLS回帰モデルの学習アルゴリズム

- データ行列 : X, レスポンス: y
- 以下のステップを繰り返す

 - w_i: Xを低次元に射影するためのベクトル

1. Xを低次元に射影しt_iに保持 ($t_i=Xw_i$)
2. t_iとyの共分散最大化によりw_{i+1}を学習する

- 行列の足し算と掛け算を文法圧縮したデータ行列上で実現できれば学習アルゴリズムは実装可能
  - データ行列の行アクセスと列アクセス

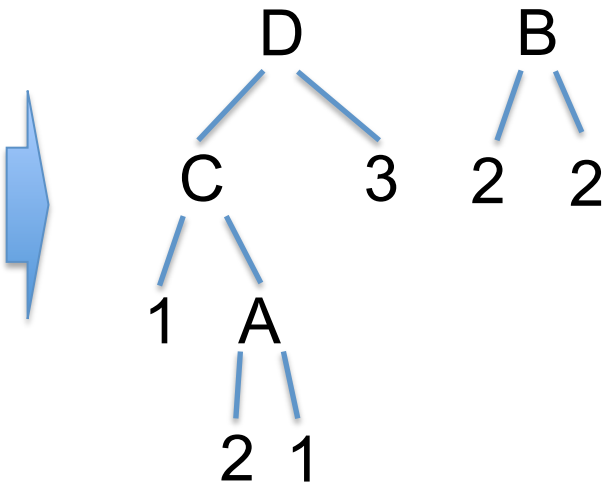
行アクセス

- 復元アルゴリズムと基本的に同じ

1. アクセスしたい行の非終端記号を展開する
   例 : X_1=(1, 2, 1, 3, 2, 4)
2. 前から累積和を計算する
   例 : X_1 = (1, 3, 4, 7, 9, 13)

X_1=(D,2,4)
X_2=(A,4,B)
X_3=(D,B)

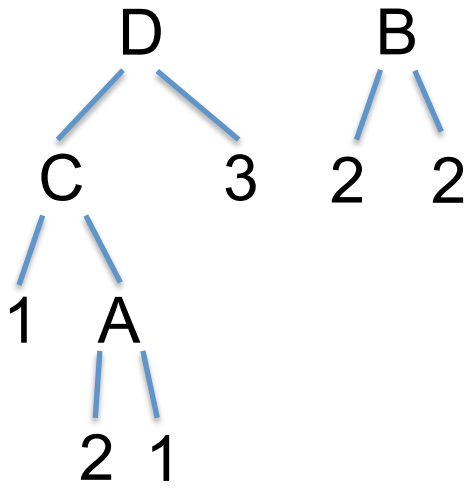
D={D→C3,
B→22,
C→1A,
A→21}



列アクセス

- 各中間ノードより下の葉のラベルの総和を保持
- 各Rowの前から累積和を計算しつつ, 候補となるカラムが有るかチェック
- 非終端記号を根とする部分木を辿って, 求めたい値があるかチェックする

X_1=(D,2,4)
X_2=(A,4,B)
X_3=(D,B)



A	B	C	D
3	4	4	7

実験

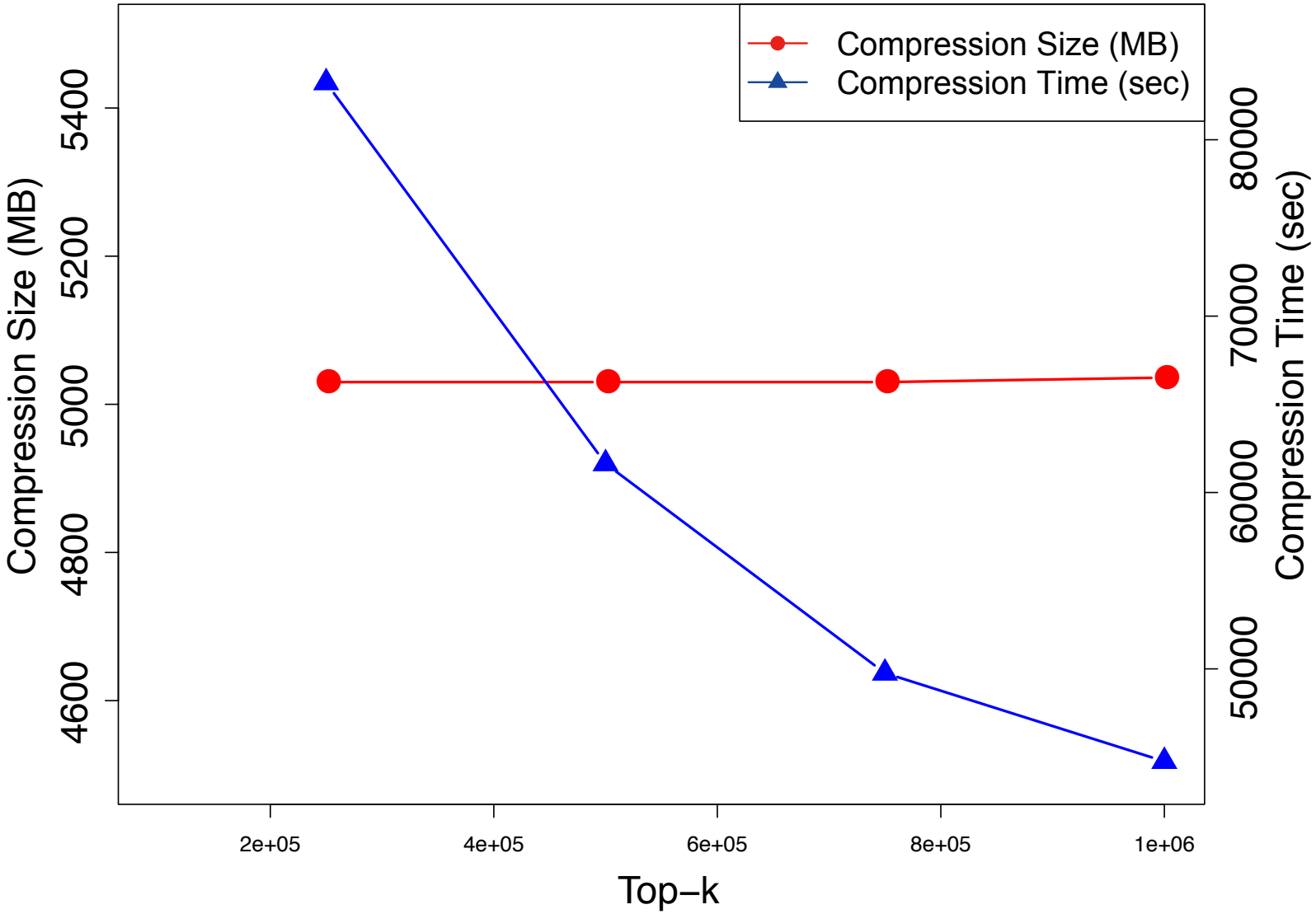
- データセット

Table 2: Summary of datasets.

Dataset	Label type	Number	Dimension	#nonzeros	Memory (mega bytes)
Book-review	binary	12,886,488	9,253,464	698,794,696	2,665
Compound	binary	42,682	52,099,292	914,667,811	3,489
Webspam	binary	350,000	16,609,143	1,304,697,446	4,977
CP-interaction	binary	216,121,626	3,621,623	32,831,736,508	125,243
CP-intensity	real	1,329,100	682,475	28,865,055,991	110,111

- cPLS(提案手法), PCA, b-bit MinHash, SGDを比較
- 評価尺度として、圧縮サイズ, 学習時間, 精度を比較

CP-interation上でのtop-kを変化させたときの実行時間と圧縮サイズ



CP-interaction上でのヒープサイズを変化させたときの圧縮時間、メモリー、圧縮サイズ

- 125GBのデータを圧縮するために、約16GBのメモリーを使用して実行時間は約12時間
- 125GBから約4GBに圧縮

CP-interaction

	10	100	1000	10000
compression size (MB)	-	5199	5139	5036
compression time (sec)	24hours	55919	44853	43756
working space (MB)	-	9914	16650	16635

CP-intensity上でのヒープサイズを変化させたときの圧縮時間、メモリー、圧縮サイズ

- 110GBのデータを圧縮するために、約4GBのメモリーを使用して実行時間は約2時間
- 102GBから約500MBに圧縮
- 各データの非ゼロ要素数 >> データのサイズ (水平ビッグデータ)に強い圧縮法

CP-intencity

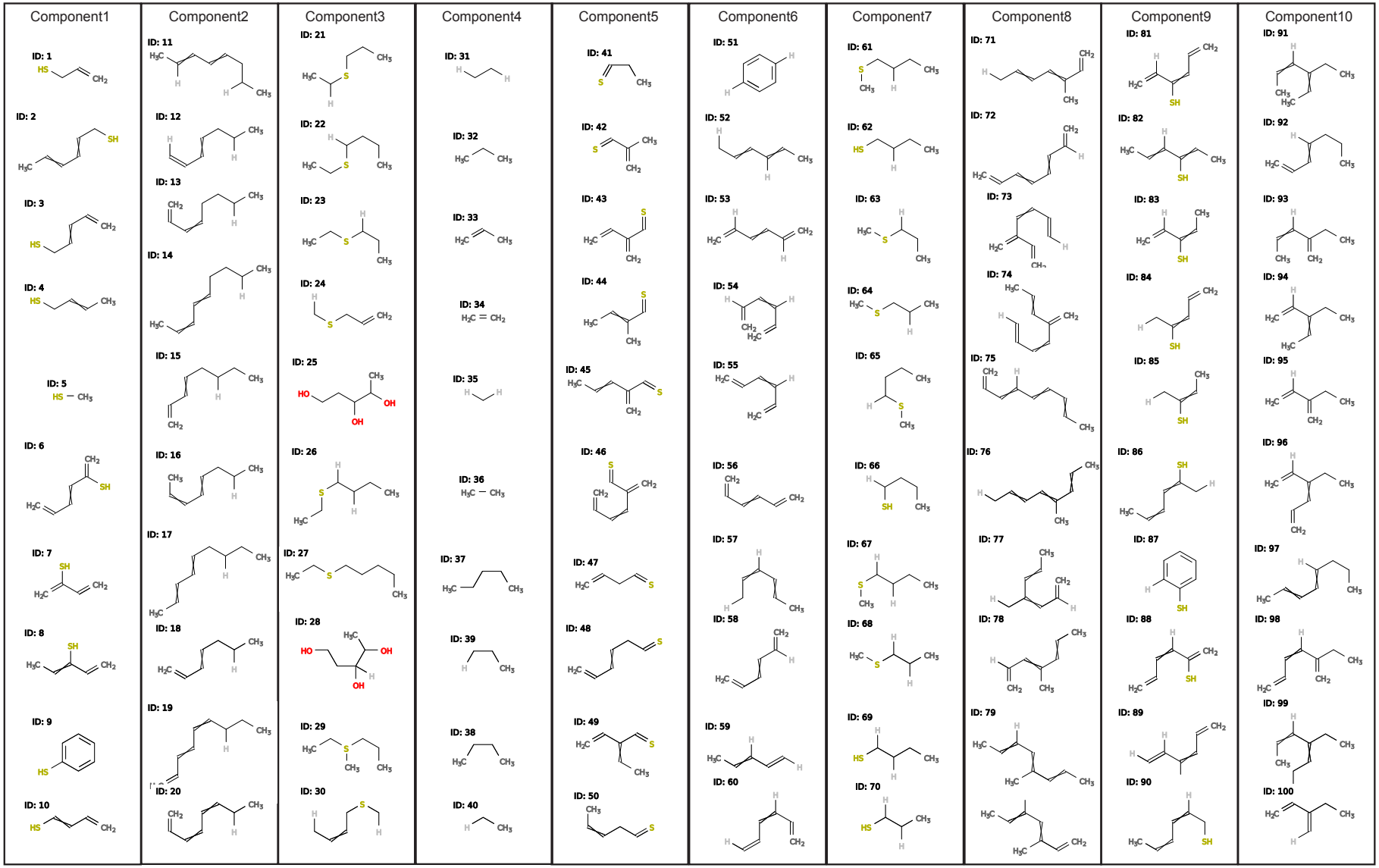
	10	100	1000	10000
compression size (MB)	558	543	540	535
compression time (sec)	8103	6479	6494	6657
working space (MB)	1936	3552	3722	3738

予測精度, メモリー, 学習時間

- Dspace : 圧縮データ行列のメモリ
- Ospace : 最適化のメモリ
- CP-intencity(110GB)上でPLSを学習するのに10GB程度

	cPLS				
Data	m	Dspace(MB)	Ospace(MB)	Ltime(sec)	AUC/PCC
Book-review	100	1288	15082	21628	0.96
Compound	20	786	7955	1089	0.83
Webspam	60	890	7736	4171	0.99
CP-interaction	40	4367	53885	35880	0.77
CP-intensity	60	472	10683	33969	0.67
	PCA-SL				
Data	m/C	Dspace(MB)	Ospace(MB)	Ltime(sec)	AUC/PCC
Book-review	100/1	14747	110	6820	0.70
Compound	25/0.1	12	1	6	0.65
Webspam	50/1	200	2	129	0.99
CP-interaction	-	-	-	>24hours	-
CP-intensity	100/0.1	1521	11	42	0.11
	bMH-SL				
Data	$b/h/C$	Dspace(MB)	Ospace(MB)	Ltime(sec)	AUC/PCC
Book-review	100/16/0.01	2457	110	1033	0.95
Compound	30/16/10	2	1	1	0.62
Webspam	30/16/10	20	2	2	0.99
CP-interaction	30/16/0.1	12366	1854	10054	0.77
CP-intensity	100/16/0.1	253	11	45	0.54

抽出された化合物の部分構造特徴



まとめ

- 圧縮されたデータ行列上での統計モデルの学習法
- 文法圧縮されたデータ行列上でPLS回帰モデルを学習
- 利点
  - PLS回帰モデル学習のメモリ削減
  - 巨大データ行列からの特徴抽出