株式会社 インテージヘルスケア

CzeekS ver. 2.0
ユーザーマニュアル

第 2版

CzeekS

【商標について】

本マニュアルに記載の社名、商品名等は各社の商標または登録商標である場合があります。また、本マニュアルに記載のシステム名、製品名等には、商標表示を付記しておりません。

改定履歴

第 1 版　2019 年 9 月 10 日

- 初版発行

第 2 版　2019 年 12 月 12 日

- 初期 DB の内容の変更に伴う修正
- サブコマンド “sample” のオプション変更に伴う修正

目次

1　はじめに

「ある 1 つの化合物がたった 1 つのタンパク質とのみ活性を有するという事は稀なことである」という考え方は近年では常識になっています。このような化合物-タンパク質の複雑な関係の情報を我々はケミカルゲノミクス情報と呼称しており、このような情報は ChEMBL 等に代表される化合物の生物活性のデータベースとして整備されつつあります。これらの情報を機械学習によるパターン認識によって未知化合物の活性を予測・スクリーニングする手法が CGBVS（Chemical Genomics-Based Virtual Screening）です。CzeekS はこの CGBVS を行うためのツール群で、以下の機能を提供しています。

- 化合物のスコア計算
- CGBVS 学習モデルの作成
- 学習モデルの管理機能
- 化合物のフィンガープリント（ECFP4）の計算
- ターゲットタンパク質の既知化合物との類似度計算

本マニュアルの 2 章では CzeekS のインストール方法について説明します。3 章ではサンプルデータを使って化合物のスクリーニング方法について説明します。また、発展的な使用方法として化合物の選択性の予測や、ターゲット予測についても説明します。4 章では CGBVS 用の初期 DB から学習モデルの構築方法について説明します。5 章では独自のデータを追加して学習モデルを構築する方法について説明します。6 章では DB ファイルの詳細について説明します。DB ファイル中のデータの取り出しや、CzeekS のツールでは対応出来ないような細かいデータ操作の方法について解説します。7 章は CzeekS のコマンドのレファレンスです。

以下のような計算機環境において CzeekS の利用を想定しています。CzeekS は OpenMP による並列計算に対応していますので、CPU コア数が多い方が効率よく計算することができます。また、計算機は 1 台から CzeekS を利用することができます。

表 1　動作環境

CPU	4 コア以上のマルチコア CPU (Intel, AMD)
メモリ	16Gb 以上
HDD	20Gb 以上の空き容量
OS	CentOS7.x 64bit (Linux カーネル 2.6)
外部ツール	alvaDesc ver. 1.0

サンプルデータの機械学習に掛かる時間（1 ノード）。

表 2　SVM による機械学習時間

動作環境	ベクトル次元	化合物/タンパク質	正例数/負例数	計算時間
Intel Core i7-7700, 8 threads, 32GB memory		269 / 39	39753 / 39753	3154 [s]
Intel Xeon Gold 6154, 24 threads, 192GB memory		234 / 193	62597 / 62597	3907 [s]
Intel Xeon Gold 6154, 24 threads, 192GB memory		276 / 206	117346 / 117346	15178 [s]
Intel Xeon Gold 6154, 36 threads, 192GB memory		277 / 351	135896 / 135896	24578 [s]

一つのタンパク質に対して、1 万個化合物のスクリーニング時間（1 ノード）は下記となります。

表 3　CGBVS スコアの計算時間

動作環境	モデル	化合物数/タンパク質数	正例数/負例数	計算時間
Intel Xeon E3-1231 v3, 8 threads, 32GB memory	Transporter	37656 / 126	44111 / 5318	122 [s]
Intel Core i7-4790, 8 threads, 32GB memory	Transporter	37656 / 126	44111 / 5318	110 [s]
Intel Xeon Gold 6154, 24 threads, 192GB memory	Transporter	37656 / 126	44111 / 5318	83 [s]

2　CzeekS のインストールと設定

2.1　アーカイブの展開とライセンスファイルの配置

アーカイブファイル “CzeekS_******.tgz” を以下の様に tar コマンドで展開して下さい。いずれのディレクトリで展開しても構いませんが、/usr/local の下や、czeeks 等のユーザーを作成し、/home/czeeks の下に展開することをお勧めします。尚、本マニュアルでは/home/czeeks 下にファイル展開したと仮定して説明を進めます。

```
$ tar xvzf CzeekS_******.tgz
CzeekS/
CzeekS/example/
CzeekS/example/H3_mols.csv
CzeekS/example/H3_mols.fp
CzeekS/example/H3_mols.sdf
CzeekS/example/H3_mols.smi
CzeekS/example/H3_positive.csv
...
```

展開されるファイル構成は以下の通りです。添付のライセンスファイル “czeeks_license.dat” を/home/czeeks/CzeekS/exec の下に上書きコピーして下さい。

```
CzeekS
|-- example                 // サンプルデータ等を収めたディレクトリ
|   |-- H3_mols.csv         // ヒスタミンH3受容体のリガンドの記述子（100個）
|   |-- H3_mols.fp          // ヒスタミンH3受容体のリガンドのフィンガープリント（100個）
|   |-- H3_mols.sdf         // ヒスタミンH3受容体のリガンドのSDファイル（100個）
|   |-- H3_mols.smi         // ヒスタミンH3受容体のリガンドのSMILES（100個）
|   |-- H3_posi.csv         // ヒスタミンH3受容体のリガンドの正例データ（100個）
|   |-- sample_mols.csv     // テスト用化合物の記述子
|   |-- sample_mols.fp      // テスト用化合物のフィンガープリント
|   |-- sample_mols.sdf     // テスト用化合物のSDFファイル
|   `-- sample_mols.smi     // テスト用化合物のSMILES
`-- exec                    // 実行ファイル等を収めたディレクトリ
    |-- 2D_941_sdf.drs      // alvaDesc用のスクリプトファイル
    |-- 2D_941_smi.drs      // alvaDesc用のスクリプトファイル
    |-- SVMlearn            // SVM学習の実行ファイル
    |-- calc_FP_ECFP4       // ECFP4計算の実行ファイル
    |-- calc_alvaDesc.sh    // DRAGON7実行用スクリプト
    |-- cgbvs               // CGBVSの実行ファイル
    |-- czeeks_license.dat  // 空のライセンスファイル
    `-- uniprot.txt         // UniProtからのタンパク質情報
```

2.2　ライセンス認証

本製品を使用するためにはライセンス認証が必要です。以下の手順で認証を行い、発行されたライセンスファイルをインストールして下さい。

1. 計算機情報の確認本製品の CzeekS/exec 以下にある cgbvs コマンドでライセンス認証に必要な文字列（計算機情報の SHA1 ハッシュ値）を確認して下さい。発行されるライセンスファイルはこのコマンドで文字列を確認した計算機上でのみ有効です。

2. ライセンスファイルの発行 cgbvs コマンドを引数無しで実行します。有効なライセンスが無い場合以下のようなメッセージが表示されます。

```
$ cd CzeekS/exec
$ ./cgbvs
The license file is invalid.

SHA1 digest of this machine is "d5d984743bf44bed09ac863a4d9eb17d2b5841a1"
please send the above string to the address below in order to obtain a license
...
```

確認した SHA1 ハッシュ値の文字列を弊社宛にメール等でお知らせください。ライセンスファイルを発行しメールにて送付いたします。

3. ライセンスファイルのインストール弊社より受け取ったライセンスファイルを以下の 2 通りのいずれかの方法でインストールして下さい。
   （a）環境変数 CzeekS に設定したディレクトリに “czeeks_license.dat” という名前でライセンスファイルを置きます。
   （b）環境変数 CzeekS_LICENSE にフルパスでライセンスファイル名を設定します。
   ※両方同時に設定されている場合は後者の方を優先します。

2.3 環境変数の設定

アーカイブファイルを展開し、ライセンスファイルを配置した後に、以下のように環境変数を設定して、.bashrc ファイルにも以下の内容を記述して下さい。

```
$ export CzeekS=/home/czeeks/CzeekS/exec
$ export PATH=$PATH:$CzeekS
$ export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
$ export CzeekS_LICENSE=/home/czeeks/CzeekS/exec/czeeks_license.dat // ライセンスファイル
$ export ALVADESC=/home/czeeks/usr/bin // alvaDescがインストールされたパス
```

環境変数 CzeekS_LICENSE については、フルパスのファイル名を指定してください。ただし、環境変数 CzeekS で指定したディレクトリにライセンスファイルが存在する場合は、CzeekS_LICENSE を設定する必要はありません。環境変数 ALVADESC については、alvaDesc の実行ファイルである “alvaDescCLI” の存在するディレクトリを指定してください。正しく.bashrc 内に記述した後に、source コマンドで.bashrc の設定内容を反映させて下さい。

2.4 動作確認

CzeekS の動作確認は、SHA1 ハッシュ値の確認と同じ方法で行います。引数無しで cgbvs コマンドを実行したとき、以下のようなメッセージが表示されれば認証出来ています。

```
$ cgbvs
CzeekS ver. 2.0.0

Usage:
```

```
cgbvs <subcommand> [options] [args]

Available subcommands:
  add           : add data to db file
  add_model     : add SVM model to db file
  comment       : add comments to db file
  create        : create an empty db file
  delete        : delete data from db file
  del_model     : delete SVM model from db file
  learn         : create input file for SVM
  predict       : perform CGBVS calculation
  prepare       : specify protein group to be used for data sampling
  sample        : sample positive and negative data
  similarity    : perform calculation of Tanimoto coefficient
  shrink        : delete non-essential data from db file to reduce file size
  status        : display db file information summary
  vacuum        : apply SQLite vacuum command to DB file
  validation    : perform validation of SVM models
```

3　化合物スクリーニングとターゲット予測

3.1　CGBVS モデル

CzeekS にはサンプルのモデルファイルが添付されております。あくまでも CzeekS の説明のためのサンプルなので、実際のインシリコスクリーニングに用いること避けて下さい。CzeekS ではモデルファイルのファイル名の拡張子は.db であり、これは SQLite というデータベースマネジメントシステムのファイル形式となっています。　以下ではモデルファイルのことを “DB ファイル” と呼ぶこともあります。このサンプルのモデルファイルは公的データベースの ChEMBL に由来する GPCR のデータを使って構築しております。また、モデル構築に用いた初期 DB ファイルも添付しています。これらのデータについては 4 節と 5 節で解説します。

CGBVS ではパターン認識の手法としてサポートベクターマシン（SVM）を用いています。SVM は正例と負例の 2 クラスを分類する方法で、機械学習するには正負両方のデータが必要です。しかしながら化合物-タンパク質ペアで活性有りという情報は潤沢に存在しても、化合物-タンパク質ペアが実験的に活性無しと確認されている情報は公的データベース等には僅かしか在りません。したがって、負例としての化合物-タンパク質ペアの相互作用情報を仮想的に生成して、機械学習を行っております。仮想的負例は正例ペアをランダムに組み換えることにより生成されております。ランダム性が入るため複数の負例セットで学習モデルを生成して化合物の予測スコアを負例セットの数だけ算出し、最終的にそれらの平均値をスコアとしております。

CzeekS で算出される CGBVS のスコアは 2 種類あります。1 つは SVM の決定関数値の平均値で、$-\infty \sim +\infty$ までの範囲をとります。もう一方はこの決定関数値をシグモイド関数で正規化したものの平均値で、$0 \sim 1$ までの範囲をとります。CzeekS では通常は後者の正規化した方のスコアを表示します。このスコアは、ターゲットのタンパク質に対して化合物が活性を持つ “確率値” としての意味を持ちます。つまり、CGBVSスコアと活性値とは比例関係にはありませんのでご注意ください。

上記で説明した CGBVS モデルの情報を “cgbvs status” コマンドで確認することができます。先ず以下のコマンドでサンプルモデルの DB ファイルを確認して下さい。DB ファイルに登録されている化合物数やタンパク質数、学習したモデルについての情報が一覧表示されます。

```
$ cgbvs status gpcr_sample.db
[title]
  sample GPCR Model

[compound descriptor]
  alvaDesc ver. 1.0
  # of data = 22442
  # of descriptors = 941

[protein descriptor]
  PROFEAT 2016
  # of data = 805
  # of descriptors = 1437
```

```
[fingerprint]
  ECFP4 by using RDKit
  # of data = 22442

[Interactions]
  # of positive interactions = 20102
  # of negative interactions = 5635

[sampling]
  ============================================================
   ID  positives  negatives  virtual negatives  negative sets
  ------------------------------------------------------------
    1      20102       5635              14467             5
  ============================================================

[CGBVS model]
  ============================================================
   ID  models  dim_c  dim_p       C        gamma_c    gamma_p
  ------------------------------------------------------------
    1       5    259    216  3.000E+00  3.000E-03  1.000E-03
  ============================================================
```

出力された [sampling] の表について、ID はデータ全体から学習するためのデータをサンプリングする度に連番で付与される ID です。positives はサンプリングされた正例数で negatives は負例数です。virtual negatives は仮想的に生成された負例数です。negative sets はランダムに生成される仮想的負例セットの数です。最後の [CGBVS model] の表は学習モデルの詳細を示しています。ID はサンプリングの ID 番号で、models は負例セットの数です。dim_c はスケーリング後の化合物記述子数で、dim_p はスケーリング後のタンパク記述子です。C、gamma_c、gamma_p は SVM のパラメータです。

上述の “cgbvs status” コマンドの最後に “protein” を指定すると、以下のように計算できるタンパク質の一覧表が出力されます。

```
$ cgbvs status gpcr_sample.db protein
[protein list]
  ================================================================================
     protein    positives  negatives  accession                name
  --------------------------------------------------------------------------------
   5HT1A_HUMAN        100         75  P08908         5-hydroxytryptamine receptor 1A
   5HT1B_HUMAN        100          9  P28222         5-hydroxytryptamine receptor 1B
   5HT1D_HUMAN        100          9  P28221         5-hydroxytryptamine receptor 1D
   5HT1E_HUMAN         77          8  P28566         5-hydroxytryptamine receptor 1E
   5HT1F_HUMAN        100          8  P30939         5-hydroxytryptamine receptor 1F
   5HT2A_HUMAN        100         33  P28223         5-hydroxytryptamine receptor 2A
   5HT2B_HUMAN        100          9  P41595         5-hydroxytryptamine receptor 2B
   5HT2C_HUMAN        100         94  P28335         5-hydroxytryptamine receptor 2C
   5HT4R_HUMAN        100          8  Q13639          5-hydroxytryptamine receptor 4
   5HT5A_HUMAN        100        100  P47898         5-hydroxytryptamine receptor 5A
   5HT6R_HUMAN        100         12  P50406          5-hydroxytryptamine receptor 6
   5HT7R_HUMAN        100         34  P34969          5-hydroxytryptamine receptor 7
    AA1R_HUMAN        100        100  P30542                   Adenosine receptor A1
   AA2AR_HUMAN        100        100  P29274                  Adenosine receptor A2a
   AA2BR_HUMAN        100         99  P29275                  Adenosine receptor A2b
    AA3R_HUMAN        100        100  P0DMS8                   Adenosine receptor A3
...
```

出力される表中の protein が結合予測計算時に使用するタンパク質の ID です。この ID と accession はタンパク質のデータベースである UniProt (http://www.uniprot.org/) で使用さ

れている ID です。また positives の列は DB ファイルに登録されているタンパク質ごとの活性化合物数（正例数）を表しており、negatives の列は非活性化合物数（負例数）を示しています。化合物構造の多様性にもよりますが、一般的には化合物数が大きいほど予測精度も高くなる傾向があります。活性・非活性はデータの活性値によりますが、その条件は DB の設定により異なります。サンプルの DB ファイルでは、活性有りの基準は 30 μM 以下です。

3.2 化合物スクリーニング（化合物の記述子計算からスコアリングまで）

【化合物の記述子計算】

ターゲットとするタンパク質に対しての化合物の予測計算を始める前に、化合物構造（SD ファイル）から記述子を計算する必要があります。化合物記述子の種類は必ず DB ファイル中のものと一致させなければなりません。さらに記述子計算時の化合物の処理条件（脱塩や電荷中性化など）も一致させる必要があります。CzeeS では、化合物記述子はディレクトリ exec 下のスクリプトファイル “cald_alvaDesc.sh” を用いて計算しており、化合物は脱塩・電荷中性化を行っております。

化合物の SD ファイル（SMILES でも可）から alvaDesc で記述子を計算する場合は下の様なコマンドで実行できます。このコマンドでは計算された記述子が CSV 形式で標準出力されます。

```
$ calc_alvaDesc.sh sample_mols.sdf > output.csv
$ cat output.csv
ZINC00074638,315.320,8.522,24.952,38.109,25.091,...
ZINC00075927,269.300,8.416,21.796,32.563,22.216,...
ZINC00492910,300.390,7.152,25.928,42.138,27.228,...
ZINC02759964,339.170,10.941,21.362,32.153,21.784,...
ZINC03518134,264.360,6.778,22.928,39.138,24.228,...
...
```

記述子ファイルの CSV 形式で、“（化合物 ID）,（記述子 1）,（記述子 2）,…” のように 1 行に 1 化合物ずつ、化合物 ID と記述子の数値を並べて記述して下さい。特に calc_alvaDesc.sh スクリプトを使わず計算する場合はフォーマットに注意して下さい。

【スコアリング】

化合物の記述子が用意できれば、“cgbvs predict” コマンドで予測計算が実行できます。CzeekS にはサンプル記述子ファイルとして “sample_mols.csv” があり、上述のコマンドで実行した内容と同一です。例えば、アドレナリン β2 受容体に対するスコアの計算は以下のようなコマンドで実行でき、結果は標準出力されます。

```
$ cgbvs predict gpcr_sample.db ADRB2_HUMAN sample_mols.csv
compound        ADRB2_HUMAN
```

```
ZINC00074638    0.30761743
ZINC00075927    0.41729564
ZINC00492910    0.66687655
ZINC02759964    0.48933271
ZINC03518134    0.45773668
ZINC03912658    0.21275703
ZINC04143221    0.47365943
...
```

このコマンドの引数 2 は CGBVS モデルの DB ファイルを指定します。引数 3 はターゲットのタンパク ID を指定して、最後の引数 4 に化合物記述子のファイル名を指定します。ここで引数 3 にて指定できるタンパク ID は前述の “cgbvs status (dbfile) protein” コマンドでご確認ください。尚、計算結果をファイルにする場合はリダイレクトして下さい。

【複数タンパクについてスコアリング】

ターゲットタンパクを指定する引数 3 では、コンマ区切りで複数のタンパク ID を並べて指定することによって複数のタンパクについてのスコアを計算することができます。タンパク ID 数には特に上限は設けておりません。例えば β 1、β 2 受容体の両方のスコアを計算したい場合は以下のようにコマンドを実行します。

```
$ cgbvs predict gpcr_sample.db ADRB1_HUMAN,ADRB2_HUMAN sample_mols.csv
compound        ADRB1_HUMAN     ADRB2_HUMAN
ZINC00074638    0.03987265      0.30761743
ZINC00075927    0.07638203      0.41729564
ZINC00492910    0.45341149      0.66687655
ZINC02759964    0.06134940      0.48933271
ZINC03518134    0.28673856      0.45773668
ZINC03912658    0.19891787      0.21275703
ZINC04143221    0.07831798      0.47365943
...
```

この様に複数タンパクについてのスコアがタブ区切りで表示されます。複数タンパクを指定すれば、化合物の選択性も考慮したスクリーニングも容易です。また、% 記号をワイルドカードとして利用することもできます。例えば、α 受容体も含めた全てのアドレナリン受容体に対してのスクリーニングは以下のようなコマンドで実行できます。

```
$ cgbvs predict gpcr_sample.db ADA%,ADR% sample_mols.csv
compound  ADA1A_HUMAN ADA1B_HUMAN ADA1D_HUMAN ADA2A_HUMAN
ADA2B_HUMAN ADA2C_HUMAN ADRB1_HUMAN ADRB2_HUMAN ADRB3_HUMAN
ZINC00074638  0.05711626  0.05538899  0.27683547  0.01723265
0.10989224  0.05202779  0.03987265  0.30761743  0.01528607
ZINC00075927  0.08758092  0.07610408  0.33834918  0.08277456
0.21185069  0.21552622  0.07638203  0.41729564  0.11404149
ZINC00492910  0.28799951  0.43182245  0.81362560  0.09418648
0.24360570  0.20492162  0.45341149  0.66687655  0.83097088
...
```

【表示形式】

“cgbvs predict” のオプションで CGBVS スコアの表示内容を変更することができます。-d オプションをつけると正規化されたスコアではなく、SVM の決定関数の平均値を出力できます。

```
$ cgbvs predict -d gpcr_sample.db ADR% sample_mols.csv
compound  ADRB1_HUMAN ADRB2_HUMAN ADRB3_HUMAN
ZINC00074638  -0.84748056 -0.24838904 -1.09524675
ZINC00075927  -0.65214708 -0.08245668 -0.53104639
ZINC00492910  -0.04439970 0.20269411  0.42458627
ZINC02759964  -0.70399970 -0.00395215 -0.68510799
ZINC03518134  -0.23484752 -0.03898457 -0.45582779
ZINC03912658  -0.35507470 -0.33290417 -0.35623274
ZINC04143221  -0.64635067 -0.02058411 -0.63536403
...
```

また-v オプションで決定関数値と正規化スコアの両方を出力します。

```
$ cgbvs predict -v gpcr_sample.db ADR% sample_mols.csv
compound  protein score decision
ZINC00074638  ADRB1_HUMAN 0.03987265  -0.84748056
ZINC00074638  ADRB2_HUMAN 0.30761743  -0.24838904
ZINC00074638  ADRB3_HUMAN 0.01528607  -1.09524675
ZINC00075927  ADRB1_HUMAN 0.07638203  -0.65214708
ZINC00075927  ADRB2_HUMAN 0.41729564  -0.08245668
ZINC00075927  ADRB3_HUMAN 0.11404149  -0.53104639
ZINC00492910  ADRB1_HUMAN 0.45341149  -0.04439970
ZINC00492910  ADRB2_HUMAN 0.66687655  0.20269411
ZINC00492910  ADRB3_HUMAN 0.83097088  0.42458627
...
```

このように、1 行に化合物-タンパク質ペアに対する 2 種のスコア値を表示する形式で表示されます。

3.3 ターゲット予測

【CGBVS におけるターゲット予測とは】

前節では CGBVS で複数タンパク質についてスコアを算出できることについて説明しましたが、この考え方を拡張して計算できる全てのタンパク質についてスコアを算出すれば化合物のターゲット探索にも応用することができます。

“cgbvs predict” のターゲットを指定する引数に “all” を指定すれば、DB ファイルに登録されていて、学習されている化合物が 1 個以上ある全てのタンパクについてスコアを計算できます。また、学習されている化合物が無いタンパクに対してもスコアを算出したい場合は-a オプションをつけて下さい。（計算できるタンパクは “cgbvs status -a (dbfile) protein” で確認できます）たとえばサンプルの “sample_mols.csv” 中の ZINC10454282 という ID の化合物について全スコアを計算する例は以下のようになります。

```
$ grep ZINC10454282 sample_mols.csv > test.csv
$ cgbvs predict -t gpcr_sample.db all test.csv
```

```
protein ZINC10454282
5HT1A_HUMAN 0.10916306
5HT1B_HUMAN 0.15878529
5HT1D_HUMAN 0.07889011
5HT1E_HUMAN 0.14655586
5HT1F_HUMAN 0.08803487
5HT2A_HUMAN 0.05324092
5HT2B_HUMAN 0.30258845
5HT2C_HUMAN 0.08713960
5HT4R_HUMAN 0.10030115
5HT5A_HUMAN 0.04370872
5HT6R_HUMAN 0.03649251
5HT7R_HUMAN 0.06133398
AA1R_HUMAN  0.00940896
...
```

この例では、-t オプションを付けて行方向にタンパク ID を並べて表示させています。出力をリダイレクトしてスコア高い順にソートすれば確率の高いターゲットのリストが得られます。

```
$ cgbvs predict -t gpcr_sample.db all test.csv > out
$ sort -k2 -nr out | head
TSHR_HUMAN      0.97850009
TS1R3_HUMAN     0.96211287
TS1R1_HUMAN     0.95630239
MTR1B_HUMAN     0.89510040
GLP1R_HUMAN     0.87863621
TS1R2_HUMAN     0.78310599
TAAR1_HUMAN     0.74047184
ADRB2_HUMAN     0.70227043
GALR3_HUMAN     0.67855099
MTR1A_HUMAN     0.67590262
...
```

先頭のタンパク ID の TSHR_HUMAN は以下のようにして確認できます。

```
$ cgbvs status -p gpcr_sample.db | grep -e "MTR1..*"
   TSHR_HUMAN        100        100  P16473     Thyrotropin receptor
```

また 4 列目の “P16473” を検索ワードとして UniProt(http://www.uniprot.org/) などで検索すれば、このタンパク質の詳細を得ることが出来ます。

3.4 構造類似度（Tanimoto 係数）の計算

CzeekS では化合物のフィンガープリントから、Tanimoto 係数（Similarity）を計算することができます。Tanimoto 係数を計算する対象は、ターゲットとして指定したタンパク質と結合する化合物群（DB ファイル中）です。複数の化合物との Tanimoto 係数を計算し、最大値のものを表示します。コマンド操作は cgbvs コマンドで “predict” の代わりに “cgbvssimilarity” を用いて計算します。以下にその手順について示します。

```
$ calc_FP_ECFP4 sample_mols.sdf > test.fp  // フィンガープリント計算です。test.
    fpとsample_mols.fpは同一のものになります
$ cgbvs predict -s gpcr_sample.db ADRB2_HUMAN test.fp
compound  ADRB2_HUMAN
ZINC00074638  0.18947368
ZINC00075927  0.21875000
ZINC00492910  0.45161290
ZINC02759964  0.36923077
```

```
ZINC03518134  0.22058824
ZINC03912658  0.12987013
ZINC04143221  0.15789474
ZINC05766699  0.27868852
ZINC10006603  0.15789474
...
```

フィンガープリントファイル test.fp の内容は以下のようになります。

```
$ head sample_mols.fp
ZINC00074638,10565946 12100272 26234434 92620239 98513984 171813213 ...
ZINC00075927,10565946 19852674 118059098 158975758 378442094 422715066 ...
ZINC00492910,43357009 98513984 353395765 411967733 711358894 711855525 ...
ZINC02759964,98513984 271903915 312306367 399277278 422715066 714893496 ...
ZINC03518134,6755136 98513984 237615532 571978829 623728226 847961216 ...
ZINC03912658,159041780 271903915 293971392 307525970 398876281 407363943 ...
ZINC04143221,271903915 380690222 422715066 592593828 619920801 793509949 ...
ZINC05766699,26234434 353395765 454919045 601159014 621263118 847336149 ...
ZINC10006603,20230819 156142916 171200514 269171368 271903915 540046244 ...
ZINC10454282,222262242 321980664 717512901 847961216 864674487 864942730 ...
...
```

書式は、1 列目に化合物 ID を記述してコンマで区切り、2 列目にフィンガープリントの内容を記述します。フィンガープリント部分の書式は、1 が立っているビットのビット番号（n 桁ビット列の左端を最下位 1 とし、右端を最上位 n とする）をスペース区切りで記述してください。

4　CGBVS モデルの作成

4.1　初期 DB ファイル

CzeekS では標準で CGBVS モデルを作成するための初期 DB を用意しています。この初期 DB は ChEMBL(https://www.ebi.ac.uk/chembl/) にて公開されている化合物のアッセイデータを基にして作成されています。具体的には CGBVS モデルの作成に必要な部分を抽出・加工し、化合物とタンパク質の記述子を計算して DB ファイルに格納しています。使用している ChEMBL のバージョンは 25 です。（2019 年 9 月現在）この初期 DB ファイルの状態は以下のようになっています。

```
$ cgbvs status cgbvs_init.db
[title]
  curated ChEMBLdb 25

[compound descriptor]
  alvaDesc ver. 1.0
  # of data = 1844709
  # of descriptors = 941

[protein descriptor]
  PROFEAT 2016
  # of data = 20106
  # of descriptors = 1437

[fingerprint]
  ECFP4 by using RDKit
  # of data = 1870301

[Interactions]
  # of positive interactions = 0
  # of negative interactions = 0

[sampling]
No sampled positives

[CGBVS model]
No models
```

CzeekS での CGBVS モデル作成は、この初期 DB を上書き加工しながら進行します。元々の初期 DB は修正されないように別途保存しておいて下さい。

4.2　初期 DB の準備

CGBVS 用の初期 DB には ChEMBL の大部分のデータが入っており、ファイルサイズも十数 GB と大きくなっています。このデータの全てを用いて CGBVS モデルを作成することは非常に計算コストが掛かります。したがって、目的に合わせてタンパク数を絞ることになります。CzeekS では標準で幾つかのタンパクグループを用意しています。また、自作したタンパクリストを取り込んで利用することも可能です。

このマニュアルでは例として核内受容体モデルの作成を行います。先ず以下のコマンドで初期

DB をコピーして下さい。

```
$ cp cgbvs_init.db nuclear.db
```

今後この節では、コピーした “nuclear.db” を利用します。

【タンパクグループの確認】

”cgbvs prepare“コマンドで以下のように確認することが出来ます。

```
$ cgbvs prepare nuclear.db
[Protein Group]
  ================================================
   ID            Description            proteins
  ------------------------------------------------
    1  GPCR                                  805
    2  Kinase                                427
    3  Ion channel                           367
    4  Protease                              504
    5  Nuclear receptor                       48
    6  Transporter                           344
    7  Cytochrome P450                        48
    8  PPI related targets from TIMBAL        54
  ================================================
```

出力の ID はタンパクグループ ID を示しており、Description はグループの説明で、proteins はそのグループ中のタンパク数を示しています。

【タンパクグループの選択】

タンパクグループを選択するには以下のコマンドを実行します。

```
$ cgbvs prepare nuclear.db 5
```

ここで、第 3 引数の “5” はタンパクグループ ID の 5 番を示しています。登録されているタンパクグループを利用する場合はこれで完了です。指定したグループにあるタンパクのうち、DB にアッセイデータが存在するタンパクのみが利用出来るデータとして準備されます。それらのタンパクリストは “cgbvs status (dbfile) protein” コマンドで確認することができます。

```
$ cgbvs status nuclear.db protein
[protein list]
  ================================================================================
     protein    positives  negatives  accession                  name
  --------------------------------------------------------------------------------
   ANDR_HUMAN         2321        117  P10275     Androgen receptor
   COT2_HUMAN          213          6  P24468     COUP transcription factor 2
   ERR1_HUMAN          117          0  P11474     Steroid hormone receptor ERR1
   ERR2_HUMAN           45          0  O95718     Steroid hormone receptor ERR2
   ERR3_HUMAN           38          0  P62508     Estrogen-related receptor gamma
   ESR1_HUMAN         3016        281  P03372     Estrogen receptor
   ESR2_HUMAN         2195        173  Q92731     Estrogen receptor beta
   GCR_HUMAN          2696         84  P04150     Glucocorticoid receptor
   HNF4A_HUMAN         207          1  P41235     Hepatocyte nuclear factor 4-alpha
...
```

一方、リストに表示されないタンパク質を指定する場合は、タンパクリストのファイルを作成して読み込ませます。タンパクの ID としては UniProt で定義されている “accession number”

を記述して下さい。例えばキナーゼの一種である PAK1–6 のデータのみを利用する CGBVS モデルを構築する場合は以下のようなタンパクリストになります。

```
$ cat protein_list
Q13153
Q13177
O75914
O96013
Q9P286
Q9NQU5
```

この例では “protein_list” というファイル名で作成します。そして以下のコマンドを実行して下さい。

```
$ cgbvs prepare nuclear.db protein_list
```

リストを読み込ませる場合は第 3 引数に番号ではなく、ファイル名を指定します。準備されたタンパク質のリストは前と同様に “cgbvs status (dbfile) protein” を実行します。

```
$ cgbvs status nuclear.db protein
[protein list]
================================================================================
  protein    positives  negatives  accession                  name
--------------------------------------------------------------------------------
 PAK1_HUMAN         325          0  Q13153     Serine/threonine-protein kinase PAK 1
 PAK2_HUMAN          57          1  Q13177     Serine/threonine-protein kinase PAK 2
 PAK3_HUMAN          41          0  O75914     Serine/threonine-protein kinase PAK 3
 PAK4_HUMAN         480         34  O96013     Serine/threonine-protein kinase PAK 4
 PAK5_HUMAN          22          0  Q9P286     Serine/threonine-protein kinase PAK 5
 PAK6_HUMAN          21          0  Q9NQU5     Serine/threonine-protein kinase PAK 6
================================================================================
```

上記のように “nuclear.db” ファイルは上書きされて PAK のみのデータが準備されていることが分かります。

核内受容体に戻す場合は以下のコマンドを再度実行して下さい。

```
$ cgbvs prepare nuclear.db 5
```

【DB ファイルの縮小】

初期 DB は ChEMBL のデータの大部分を入れているので、ファイルサイズが十数 GB と少々大きくなっています。以下のように “-S” オプションを付けると選択されたタンパク質に関するデータ以外を削除して DB ファイルの大きさを縮小することが出来ます。

```
$ cgbvs prepare -S nuclear.db 5
```

DB ファイルを縮小するとコマンドの動作が早くなり、作業が軽くなります。

【活性の有無の基準】

標準では、活性値が 30 μM 以下で活性有り、50 μM 以上で活性無しとしています。活性値が 30–50 μM のものはグレーなので省いています。しかしながら、もっと強い活性値のもので

CGBVS モデルを構築したい場合は、“-p” と “-n” オプションを利用します。“-p” で活性有りの基準値を nM 単位で指定して、“-n” で活性なしの基準値を nM 単位で指定します。例えば 1 μM 以下で活性有りで、10 μM 以上で活性無しとする場合は以下の通りにコマンド入力して下さい。

```
$ cgbvs prepare -p 1000 -n 10000 nuclear.db 5
```

4.3 データサンプリング

CGBVS に使用するタンパク質を選択出来たら、実際に SVM での機械学習に使用するデータのサンプリングを行います。まずは DB ファイルの状態確認してみます。以下に出力の最後の方を示します。

```
$ cgbvs status nuclear.db
...
[Interactions]
  # of positive interactions = 39753
  # of negative interactions = 17488

[sampling]
No sampled positives

[CGBVS model]
No models
```

“positive interactions” と “negative interactions” の数が実際に SVM で学習できるデータの最大数となります。上記の例では正例数が 39,753 個なので、極端にスペックの高い計算機でなくても全てのデータを学習することが可能です。しかしながら、この正例数が非常に多い場合は全てを学習させることは困難です。そのような場合は現実的に計算できるデータ数をサンプリングすることになります。大よその目安としては、正例数が 15 万個程度が上限です。

一般的に ChEMBL などの公的 DB の場合は、活性有り（正例）のデータ数が活性無し（負例）のデータ数よりも非常に多くなっています。したがって、CzeekS では正例の数を基準にデータサンプリングします。動作手順としては以下のようになります。

1. 正例のサンプリング
2. 負例のサンプリング
3. 負例数が正例数に満たない場合は仮想的に生成する。

ここで、正例がサンプリングされた時点でサンプリングの ID 番号が連番で付与されます。

【サンプリング】

“cgbvs sample (dbfile) execute” コマンドでサンプリングされます。以下に実行例を示します。

```
$ cgbvs sample nuclear.db execute
number of sampled positives = 39529
number of sampled negatives = 17461
generating virtual negative set 1 ... Done
```

```
generating virtual negative set 2 ... Done
generating virtual negative set 3 ... Done
generating virtual negative set 4 ... Done
generating virtual negative set 5 ... Done
```

そしてサンプリングの状況を確認するには次のコマンドを実行して下さい。

```
$ cgbvs sample nuclear.db status
[sampling]
  ============================================================
   ID  positives  negatives  virtual negatives  negative sets
  ------------------------------------------------------------
    1      39529      17461              22068              5
  ============================================================
```

この出力例では、正例数が 39,753 個で負例数が 17,488 個で、仮想的な負例が 22,265 個が生成されたことが分かります。つまり学習に使用可能なデータが全て選択されたことになります。

データを部分的にサンプリングする場合は “-s” と “-t” オプションを使用します。“-s” オプションは 1 タンパク質あたりの（正例の）化合物数の上限を指定します。一方、“-t” オプションは 1 タンパク質あたりの（負例の）化合物数の上限を指定します。また、両オプションにはパーセンテージも指定でき、数字（0 から 100 まで）に % を付けます。例えば “-s 50%” を指定した場合は 1 タンパク質あたり半数の正例をサンプリングします。

それではコマンド例を見てみます。先ずは先程のサンプリングを一旦削除します。

```
$ cgbvs sample nuclear.db delete
```

続いて、以下のコマンドで 1 タンパク質あたり 100 化合物を上限にサンプリングします。

```
$ cgbvs sample -s 100 -t 100 nuclear.db execute
number of sampled positives = 3400
number of sampled negatives = 1675
generating virtual negative set 1 ... Done
generating virtual negative set 2 ... Done
generating virtual negative set 3 ... Done
generating virtual negative set 4 ... Done
generating virtual negative set 5 ... Done
```

正例数が 3400 個で負例数が 1675 個で先程よりも少なくなっていることが分かります。このマニュアルではこのサンプリングの状態でモデル構築を進めます。

4.4 SVM での機械学習

【SVM 用ファイルの作成】

SVM は “SVMlearn” という別の実行コマンドを使います。したがって、そのコマンド用のファイルを作成する必要が有ります。“cgbvs learn” コマンドでそれらのファイルを作成することが出来ます。

```
$ cgbvs learn -C 99% -P 99% nuclear.db
Calculating scaling parameters for compound descriptors
ID = 1  # of vectors = 4526
PC  sdev^2
```

```
1 176.9729850061
2 56.5011835584
3 46.2475816981
...
Export to input_1 ... Done
Export to input_2 ... Done
Export to input_3 ... Done
Export to input_4 ... Done
Export to input_5 ... Done
```

オプションの “-C” と “-P” は化合物とタンパク質の記述子を主成分分析によって、ベクトル次元を圧縮することを意味しています。この例では、化合物とタンパク質のそれぞれの記述子ベクトルの累積寄与率が 99 %になるまで主成分を採用することを意味しています。通常はこの条件でモデルを構築して下さい。

作成されるファイルは “input_1”–“input_5” の 5 個です。仮想的に生成される負例セットの数だけ出力されます。

【SVM の実行】

SVM 用の入力ファイルが作成できたら SVM を実行します。“input_1” についての SVM は以下のコマンドの様に計算して下さい。

```
$ SVMlearn -c 1 -gc 0.001 -gp 0.001 input_1 model_1
     itr       nSV      vKKT                 Objective
       1       561      5015    -1.584705825333126E+02
       2      1055      5250    -5.643601198250631E+02
       3      1366      5305    -7.508984758028218E+02
       4      1818      5329    -1.092259249753438E+03
       5      2280      4855    -1.492645222080593E+03
...
...
...
 Accuracy=  0.820851688693098      miss                   244
            39.382 [s]
                    5 fold cross-validation [%] =   82.4382592444267
 writing model file
            39.538 [s]
```

ここで、SVM のパラメータとして C = 1(-c), $\gamma_c = 0.001$(-gc), $\gamma_p = 0.001$(-gp) で計算しています。γ_c は化合物側の RBF カーネルに対するパラメータで、γ_p はタンパク質側の RBF カーネルに対するパラメータです。上記の出力例で 5-fold の交差検定値が 82.438 となっています。この計算を残りの “input_2”–“input_5” の 4 個についても行って下さい。

この例では SVM のパラメータは適当に設定していますが、実際は交差検定値のをなるべく高めるようなパラメータセットを設定して下さい。

【SVM モデルのインポート】

5 個の入力ファイルについて SVM 計算が完了すれば、“model_1”–“model_5” の 5 個のファイルが出来ているはずです。これらのファイルを DB ファイルに取り込み CGBVS の計算が出来るようにします。model ファイルの取り込みは “cgbvs add_model” コマンドを利用します。以下の

例では model ファイル名にワイルドカードを使用しています。

```
$ cgbvs add_model nuclear.db model_*
add model_1
add model_2
add model_3
add model_4
add model_5
```

最後に DB ファイルの状態を確認します。

```
$ cgbvs status nuclear.db
[title]
  curated ChEMBLdb 25

...

[sampling]
  ============================================================
   ID  positives  negatives  virtual negatives  negative sets
  ------------------------------------------------------------
    1       3408       1678               1730              5
  ============================================================

[CGBVS model]
  ============================================================
   ID  models  dim_c  dim_p       C       gamma_c    gamma_p
  ------------------------------------------------------------
    1       5    269     39  1.000E+00  1.000E-03  1.000E-03
  ============================================================
```

“CGBVS model” の項目にモデルに関する情報が追加されています。

4.5 モデルバリデーション

SVM モデルを追加出来たら、モデルのバリデーションを行います。ここでのバリデーションとは、DB ファイル内のデータを作成した SVM モデルで予測して、どの程度の精度で予測が出来ているか計算することです。必ずしも実行しなくても CGBVS 計算は可能ですが、作成されたモデルの性能を知らずに未知の化合物の予測計算はすべきではありません。バリデーションは “cgbvs validation (dbfile) execute” コマンドで以下のように実行します。

```
$ cgbvs validation nuclear.db execute
interactions = 39753
posi : 3840 / 39753
posi : 7680 / 39753
posi : 11520 / 39753
...
nega : 19200 / 25954
nega : 23040 / 25954
nega : 25954 / 25954
```

計算終了後、続いて “cgbvs validation (dbfile) summay” コマンドでバリデーション結果を表示します。以下にその表示例を省略なしで示します。

```
$ cgbvs validation nuclear.db summary
All data values
[ROC]
AUC = 0.834135
```

```
[statistics]
cutoff = 0.500000
TP = 24401
FP = 1843 (1432)
FN = 15352
TN = 24111 (7034)
NP(TP+FN) = 39753
NN(TN+FP) = 25954 (8466)

[statistical measures]
Sensitivity = 0.613815 (0.613815)
Specificity = 0.928990 (0.976498)
Positive predictive value = 0.929774 (0.983435)
Negative predictive value = 0.610977 (0.526597)
Accuracy = 0.738308 (0.724620)


Validation scores for data not present in the training set
[ROC]
AUC = 0.833286

[statistics]
cutoff = 0.500000
TP = 21359
FP = 332 (0)
FN = 14986
TN = 15457 (0)
NP(TP+FN) = 36345
NN(TN+FP) = 15789 (0)

[statistical measures]
Sensitivity = 0.587674 (0.587674)
Specificity = 0.978973 (0.978973)
Positive predictive value = 0.984694 (0.984694)
Negative predictive value = 0.507736 (0.507736)
Accuracy = 0.706180 (0.706180)
```

出力の見方としては、前半が DB ファイル中の全てのデータでの各種統計量を表示しており、後半が SVM 学習に用いていないデータでの各種統計量となっております。表示している統計量については以下の表 4 にまとめます。

表 4　各種統計量の説明

統計量	説明
AUC	ROC 曲線下の面積
cutoff	正負を判定する CGBVS スコアの閾値
TP	真陽性 (True Positive)
FP	偽陽性 (False Positive)　括弧内の数値は仮想負例の数
FN	偽陰性 (False Negative)
TN	真陰性 (True Negative)　括弧内の数値は仮想負例の数
NP	正例の総数
NN	負例の総数　括弧内は仮想負例の数の数値
Sensitivity	感度　括弧内は仮想負例を除いた場合の数値
Specificity	特異度　括弧内は仮想負例を除いた場合の数値
Positive predictive value	陽性的中率　括弧内は仮想負例を除いた場合の数値
Negative predictive value	陰性的中率　括弧内は仮想負例を除いた場合の数値
Accuracy	精度　括弧内は仮想負例を除いた場合の数値

5　独自データを追加したモデル作成

5.1　必要なデータとフォーマット

独自データを DB ファイルに追加して、CGBVS の学習モデルの作成に必要な情報は以下の 2 点です。

1. 化合物の記述子情報
2. 化合物-タンパク質ペアの相互作用情報

ただし、初期 DB に入っていないタンパク質を含む独自データの場合は、タンパク質の記述子情報も必要になります。初期 DB には UniProt に登録されている大分部のタンパク質（ヒトのみ）の記述子が入っているので、大抵の場合は必要ないはずです。

上記の必要情報はコンマ区切り（CSV）のファイル形式となります。ファイル書式の詳細についてサンプルデータを例に説明します。このサンプルデータの化合物数は 100 個で、対象のタンパク質はヒスタミン H3 受容体のみになります。先ずは化合物の記述子情報についてです。サンプルの “H3_mols.csv” を見てみます。

```
$ head H3_mols.csv
CHEMBL106612,167.24,6.6896,14.413,25.0507,15.2786,28.7803,0.5765,1.002,0.6111,...
CHEMBL1076267,239.36,6.2989,21.235,37.9124,22.7787,43.6505,0.5588,0.9977,0.5994,...
CHEMBL1079747,281.45,5.9883,25.8154,46.5632,28.0629,53.8961,0.5493,0.9907,0.5971,...
CHEMBL1080089,225.33,6.438,19.7082,35.0288,21.0173,40.2353,0.5631,1.0008,0.6005,...
CHEMBL1084877,479.6,7.6127,40.1053,63.5339,41.7392,71.9082,0.6366,1.0085,0.6625,...
CHEMBL1085142,426.63,7.4847,37.0675,56.2032,40.0459,63.4804,0.6503,0.986,0.7026,...
CHEMBL1085782,388.56,6.8168,35.122,56.3377,37.6141,64.185,0.6162,0.9884,0.6599,...
CHEMBL1094142,365.61,6.5287,31.8573,55.3995,35.0289,64.0185,0.5689,0.9893,0.6255,...
CHEMBL1098215,363.48,6.99,30.9135,52.4032,32.4549,59.4452,0.5945,1.0078,0.6241,...
CHEMBL1202895,344.42,7.3281,27.9439,47.9995,28.8697,54.4169,0.5946,1.0213,0.6142,...
```

このファイルは化合物の SD ファイル “H3_mols.sdf” から “calc_alvaDesc.sh” スクリプトを利用して作成されたものです。計算の手順は第 3 節で説明したものと同じです。化合物と同様にタンパク質の記述子情報についても CSV 形式で、第 1 列がタンパク ID（accession number）で続く 2 列目以降に数値を記述します。タンパク質の記述子計算は PROFEAT（`http://bidd2.nus.edu.sg/cgi-bin/profeat2016/protein/profnew.cgi`）の WEB サービスを使って FASTA ファイルから計算できます。計算方法などの詳しい情報は PROFEAT のページを参照して下さい。

次に化合物-タンパク質ペアの相互作用情報についてです。サンプルの “H3_posi.csv” というファイルを見てみます。

```
$ head H3_posi.csv
CHEMBL1822856,Q9Y5N1
CHEMBL375571,Q9Y5N1
CHEMBL3898186,Q9Y5N1
CHEMBL1934048,Q9Y5N1
CHEMBL383324,Q9Y5N1
```

```
CHEMBL257392,Q9Y5N1
CHEMBL473662,Q9Y5N1
CHEMBL395131,Q9Y5N1
CHEMBL179234,Q9Y5N1
CHEMBL571448,Q9Y5N1
```

このファイルの書式は、第 1 列に化合物 ID を記述し、第 2 列にタンパク質 ID を記述します。タンパク質 ID の “Q9Y5N1” はヒスタミン H3 受容体の accession です。このように化合物-タンパク質のペアを行方向に記述していきます。

5.2 データの追加

必要なファイルが準備できれば、次は DB ファイルにそれらを追加します。データの追加方法は “cgbvs add” コマンドを使います。ここでサンプルの GPCR モデルである “gpcr_sample.db” にデータ追加することを例にして解説します。元のファイルは保存しておきたいので、コピーします。

```
$ cp gpcr_sample.db training.db
```

現在の状態は第 3 節で説明した通り、“cgbvs status training.db” コマンドで確認できます。その出力も第 3 節にある通りです。

記述子の追加

化合物の記述子は以下のコマンドで追加できます。サンプルとして用意している “H3_mols.csv” を追加します。

```
$ cgbvs add training.db H3_mols.csv compound
```

追加したことを確認すると以下の通りです。

```
$ cgbvs status training.db
[title]
  sample GPCR Model

[compound descriptor]
  alvaDesc ver. 1.0
  # of data = 22542  // 化合物記述子数
  # of descriptors = 941
...
```

化合物記述子数を第 3 節の状態出力と比較すると、100 個増えていることが分かります。タンパク質記述子を追加する場合は以下のように、データタイプに protein を指定します。

```
$ cgbvs add training.db descriptors.csv protein
```

相互作用情報の追加

化合物の記述子は以下のコマンドで追加できます。サンプルとして用意している “H3_posi.csv” を追加します。

```
$ cgbvs add training.db H3_posi.csv positive
```

追加したことを確認すると以下の通りです。

```
$ cgbvs status training.db
...
[Interactions]
  # of positive interactions = 20202  // 正例の相互作用情報
  # of negative interactions = 5635
...
```

正例の相互作用情報数を第 3 節の状態出力と比較すると、100 個増えていることが分かります。

上記の正例の追加方法だけだと、データサンプリングをやり直す必要があり、サンプリングされた正例・負例の状態が以前と大きく変わってしまいます。特に仮想的負例はランダムなので再現は困難です。サンプリングの状態を崩さずにデータを追加したい場合は “-i” オプションを利用してデータを追加します。第 3 節の状態出力からサンプリング ID は 1 のみであることが分かります。したがって、ここで指定できる ID は “1” だけです。

```
$ cgbvs add -i 1 training.db H3_posi.csv positive
            ~~~~
```

追加したことを確認すると以下の通りです。

```
$ cgbvs status training.db
...
[sampling]
  ============================================================
   ID  positives  negatives  virtual negatives  negative sets
  ------------------------------------------------------------
    1      20202       5635              14467              5
  ============================================================
...
```

この出力から positive が 100 個増えていることが分かります。

負例の相互作用情報についても正例とほとんど同様に追加することが出来ます。負例の場合はコマンドにおけるデータタイプの指定をを “positive” から “negative” にするだけです。

5.3 SVM での機械学習

必要なデータの追加が出来てしまえば、機械学習の手順は第 4 節で説明したものとほとんど同じです。

【SVM 用ファイルの作成】

以下のように “cgbvs learn” コマンドを実行します。前節の異なる点として、“-i” オプションでサンプリング ID を指定しています。

```
$ cgbvs learn -C 99% -P 99% -i 1 training.db
Calculating scaling parameters for compound descriptors
ID = 1  # of vectors = 22542
```

```
PC  sdev^2
1 245.9505443902
2 65.2768673770
3 41.0514044383
...
Export to input_1 ... Done
Export to input_2 ... Done
Export to input_3 ... Done
Export to input_4 ... Done
Export to input_5 ... Done
```

“input_1”–“input_5” の 5 個のファイルが作成されます。

【SVM の実行 ∼SVM モデルのインポート】

SVM による機械学習も前節で説明した通りです。今回はパラメータが C = 1, $\gamma_c = 0.003$, $\gamma_p = 0.001$ とします。

```
$ SVMlearn -c 3 -gc 0.003 -gp 0.001 input_1 model_1
$ SVMlearn -c 3 -gc 0.003 -gp 0.001 input_2 model_2
$ SVMlearn -c 3 -gc 0.003 -gp 0.001 input_3 model_3
$ SVMlearn -c 3 -gc 0.003 -gp 0.001 input_4 model_4
$ SVMlearn -c 3 -gc 0.003 -gp 0.001 input_5 model_5
```

全ての SVM 計算が終了した後は DB ファイルに SVM モデルファイルをインポートします。

```
$ cgbvs add_model training.db model_*
```

【モデルバリデーション】

最後にモデルバリデーションします。実行方法は前節と同じです。

```
$ cgbvs validation training.db execute
$ cgbvs validation training.db summary
```

5.4 アッセイデータを追加する方法

独自データを追加する際にアッセイデータから追加することも出来ます。以下示すようなコンマ区切りかタブ区切りでテキストファイルを用意します。

```
CHEMBL496553,Q9Y5N1,Ki,=,106.0,nM,CHEMBL1003672,description of assay
CHEMBL495724,Q9Y5N1,Ki,=,2.6,nM,CHEMBL1004592,description of assay
CHEMBL564405,Q9Y5N1,Ki,=,1.514,nM,CHEMBL1036178,description of assay
CHEMBL571448,Q9Y5N1,Ki,=,26.3,nM,CHEMBL1036178,description of assay
CHEMBL507084,Q9Y5N1,Ki,>,5000.0,nM,CHEMBL1036547,description of assay
...
```

上記ファイルの各列の意味の次の通りです。

第 1 列 化合物の ID

第 2 列 タンパク質の ID (accession number)

第 3 列 活性タイプ（IC50, EC50, Ki, Kd, ...）

第 4 列 活性値との関係（=, <, >）
第 5 列 活性値
第 6 列 活性値の単位（原則 nM）
第 7 列 アッセイ ID、任意で良い
第 8 列 アッセイに関する説明、任意で良い

上記のような活性情報を記述子したファイルが用意できれば、“cgbvs add” コマンドでデータ追加できます。例えば、“training.db” に “assay_data.csv” を追加するには、以下のようなコマンドになります。

```
$ cgbvs add training.db assay_data.csv assay
```

コンマ区切りではなくタブ区切りの場合は “-t” オプションを追加して下さい。

6　DB ファイルの仕様とデータの取扱い

CzeekS の DB ファイルは SQLite(`https://www.sqlite.org/index.html`) のデータベースとなっています。したがって SQLite のツールで SQL によりデータを操作すれば、細かい設定などが可能です。幾つか SQL を使用して DB ファイルを操作する例を紹介します。したがってこの節は SQLite や SQL を使える方向けの内容となっています。DB ファイルのテーブル定義についてはこの節の最後に記述しています。

6.1　SQLite を利用した操作例

ここでは操作する DB ファイルの例として初期 DB“cgbvs_init.db” を操作します。誤ってデータを改変してしまう恐れがあるので、バックアップしてから実行して下さい。また、マニュアルでは端末からコマンドで SQLite を扱うことを前提に説明します。

【SQLite のインストール】

SQLite のインストールを簡単に説明します。OS が CentOS7 や RedHat の場合は、root 権限で以下のようにコマンドを実行します。

```
$ su -
# yum install sqlite
```

また、OS が Ubuntu の場合は以下のようにコマンドを実行します。

```
$ sudo apt install sqlite3
```

【SQLite の簡単な使用方法】

SQLite の起動は “sqlite3 (dbfile)” というコマンドになります。起動するとプロンプトが “sqlite>” となります。この状態になれば SQL コマンドを実行できます。ヘルプを見る場合は “.help” または、“.help (.command)” です。テーブル一覧の表示は “.tables” です。タブ区切り表示にする場合は “.mode tabs” で、CSV にする場合は “.mode csv” です。ファイルに出力する場合は “.output (filename)” です。そして終了は “.quit” です。詳細は SQLite のページを参照してください。

【SMILES の出力】

DB に登録されている SMILES を出力する場合は以下のようになります。

```
$ sqlite3 cgbvs_init.db
sqlite> .mode tabs
sqlite> .header off
```

```
sqlite> .output output.smi
sqlite> select * from smiles;
sqlite> .quit
```

これで DB ファイルに登録されている SMILES の全てが出力されます。

特定のタンパク質に関する化合物の SMILES は以下のようになります。例としてヒスタミン H3 受容体（Q9Y5N1）に作用する化合物で、活性値が 10nM 未満のものを出力しましょう。

```
$ sqlite3 cgbvs_init.db
sqlite> .mode tabs
sqlite> .header off
sqlite> .output output.smi
sqlite> select a.* from smiles a, assay b where a.chem=b.chem and b.prot='Q9Y5N1'
   ...> and b.standard_value<10;
sqlite> .quit
```

同じ化合物に対して複数の活性値が存在するので、データの重複が見られます。重複を許さない場合は “select distinct a.* from ...” のように、distinct を付けて SQL を実行します。

【アミノ酸配列の出力】

DB ファイルに登録されている全てのタンパク質のアミノ酸配列の出力は次のようになります。

```
$ sqlite3 cgbvs_init.db
sqlite> .mode tabs
sqlite> .header off
sqlite> .output output.txt
sqlite> select * from sequence;
sqlite> .quit
```

特定のタンパク質の場合は where 句を利用します。

```
$ sqlite3 cgbvs_init.db
sqlite> .mode tabs
sqlite> .header off
sqlite> .output output.txt
sqlite> select * from sequence where prot='Q9Y5N1';
sqlite> .quit
```

これでヒスタミン H3 受容体の配列のみが “output.txt” ファイルに出力されます。

出力されたファイルの形式は単純なタブ区切りなので、FASTA 形式にする場合は以下のような簡単な Perl スクリプトで変換します。スクリプト名は “convert.pl” とします。

convert.pl

```
#!/usr/bin/perl

while (<STDIN>) {
  chomp;
  @v=split(/\t/);
  print ">",$v[0],"\n";
  for ($i=0; $i<length($v[1]); $i+=60) {
    print substr($v[1], $i, 60),"\n";
  }
}
```

このスクリプトで SQLite からの出力ファイルを FASTA 形式に変換します。

```
$ ./convert.pl < output.txt > output.fasta
```

【アッセイデータのの出力】

初期 DB には ChEMBL のアッセイデータを入れています。例としてヒスタミン H3 受容体に関するデータを全てをカンマ区切りで出力するには以下のようにします。

```
$ sqlite cgbvs_init.db
sqlite> .mode tabs
sqlite> .header off
sqlite> .output output.txt
sqlite> select * from cgtable where prot='Q9Y5N1';
sqlite> .quit
```

6.2 テーブル定義

CzeekS の DB ファイルのテーブル定義です。テーブルの一覧は表 5 に示します。

表 5 DB ファイル中のテーブル一覧

テーブル名	概要
comment	記述子などに付けるコメントを格納する
uniprot	UniProt からのタンパク情報
cgtable	アッセイデータ
smiles	化合物の SMILES
sequence	タンパク質のアミノ酸配列
cgroup	化合物グループ
clist	化合物リスト
pgroup	タンパク質グループ
plist	タンパク質リスト
ctable	化合物記述子
ptable	タンパク質記述子
fptable	化合物のフィンガープリント
model_cgbvs	SVM モデルの各種パラメータ
model_cgbvs_x	SVM モデル用のマトリックスデータ
posi	正例を管理するテーブル
nega	負例を管理するテーブル
scores	バリデーション結果を保存するテーブル
scaling	記述子のスケーリング用パラメータ

ここからは各テーブルの詳細について示します。

comment

No.	列名	型	主キー	概要
1	class	text		コメントの分類
2	item	text		コメントの項目
3	comment	text		コメントの内容

uniprot

No.	列名	型	主キー	概要
1	accession	text	◯	accession number
2	uid	text		UniProt の entry name
3	organism	text		生物名（ヒトは “Homo sapiens”）
4	protein	text		タンパク名

cgtable

No.	列名	型	主キー	概要
1	chem	text		化合物 ID
2	prot	text		タンパク質 ID (accession)
3	standard_type	text		活性の種類（IC50, EC50, Ki など）
4	standard_relation	text		活性値との関係（=, <, > など）
5	standard_value	real		活性値
6	standard_units	text		活性値の単位（nM など）
7	assay	text		ChEMBL のアッセイ ID
8	description	text		アッセイに関する情報

smiles

No.	列名	型	主キー	概要
1	smiles	text		SMILES 文字列
2	chem	text	◯	化合物 ID

sequence

No.	列名	型	主キー	概要
1	prot	text	◯	タンパク ID (accession)
2	sequence	text		アミノ酸配列

cgroup

No.	列名	型	主キー	概要
1	cgid	int	◯	化合物グループ ID
2	description	text		グループ名、説明

clist

No.	列名	型	主キー	概要
1	cgid	int	◯	化合物グループ ID
2	chem	text	◯	化合物 ID

pgroup

No.	列名	型	主キー	概要
1	pgid	int	◯	タンパク質グループ ID
2	description	text		グループ名、説明

plist

No.	列名	型	主キー	概要
1	pgid	int	◯	タンパク質グループ ID
2	prot	text	◯	タンパク質 ID (accession)

ctable

No.	列名	型	主キー	概要
1	chem	text	◯	化合物 ID
2	c0001	real		記述子 1
3	c0002	real		記述子 2
4	c0003	real		記述子 3
...	...	...		...

ptable

No.	列名	型	主キー	概要
1	prot	text	◯	タンパク質 ID (accession)
2	p0001	real		記述子 1
3	p0002	real		記述子 2
4	p0003	real		記述子 3
...	...	...		...

fptable

No.	列名	型	主キー	概要
1	chem	text	○	化合物 ID
2	fp	text		フィンガープリント

model_cgbvs

No.	列名	型	主キー	概要
1	posi_id	int		正例のサンプリング ID
2	nega_id	int		負例セットの ID
3	nsv	int		サポートベクターの数
4	ndc	int		スケーリング後の化合物ベクトルの次元数
5	ndp	int		スケーリング後のタンパク質ベクトルの次元数
6	cker	int		化合物側のカーネルタイプ
7	pker	int		タンパク質側のカーネルタイプ
8	c	real		SVM の Cost
9	gc	real		化合物側の γ
10	gp	real		タンパク質側の γ
11	uc	blob		パラメータの予備領域（化合物）
12	up	blob		パラメータの予備領域（タンパク質）
13	a	blob		SVM の係数
14	y	blob		クラスベクトル（正例は $+1$、負例は -1）
15	b	real		分離超平面の切片
16	pa	real		シグモイドのパラメータの A
17	pb	real		シグモイドのパラメータの B
18	px	int		使用していない
19	cv	real		SVM の 5-fold クロスバリデーション値

model_cgbvs_x

No.	列名	型	主キー	概要
1	posi_id	int		正例のサンプリング ID
2	nega_id	int		負例セットの ID
3	ii	int		分割 ID
4	x	blob		SVM 用のマトリックスデータ

posi

No.	列名	型	主キー	概要
1	posi_id	int		正例のサンプリング ID
2	chem	text		化合物 ID
3	prot	text		タンパク質 ID (accession)

nega

No.	列名	型	主キー	概要
1	posi_id	int		正例のサンプリング ID
2	nega_id	int		負例セットの ID
3	chem	text		化合物 ID
4	prot	text		タンパク質 ID (accession)

scores

No.	列名	型	主キー	概要
1	posi_id	int		正例のサンプリング ID
2	chem	text		化合物 ID
3	prot	text		タンパク質 ID (accession)
4	bind	int		正例は +1、負例は −1
5	score	real		CGBVS スコア
6	decision	real		決定関数値（負例セットの平均）

scaling

No.	列名	型	主キー	概要
1	posi_id	int		正例のサンプリング ID
2	type	text		0 は化合物、1 はタンパク質
3	nd	text		記述子数
4	method	int		スケーリング方法
5	npc	real		PCA で圧縮した後の次元数
6	mean	blob		記述子の平均値のベクトル
7	sdev	blob		記述子の標準偏差のベクトル
8	min	blob		記述子の最大値のベクトル
9	max	blob		記述子の最大値のベクトル
10	matrix	blob		PCA のローディングの行列

7　cgbvs コマンドレファレンス

7.1　使用方法

cgbvs　＜サブコマンド＞　[オプション]　＜引数＞
利用可能なサブコマンドは次の通りです。

add, add_model, comment, create, delete, del_model, learn, predict, prepare, sample, similarity, shrink, status, vacuum, validation

オプションや引数はそれぞれのサブコマンド毎に異なります。

7.2　サブコマンドの説明

add：データを追加します。

（書式）

cgbvs　add　[オプション]　<DB ファイル >　< データファイル >　< データタイプ >

（説明）

記述子情報や相互作用ペア情報などのデータファイル（CSV 形式）を db ファイルに取り込み現存のデータに追加します。また、引数の < データタイプ > はデータファイルの種類（化合物の記述子情報や相互作用ペア情報など）を指定します。指定できるターゲットの種類は次の通りです。

compound	化合物の記述子
protein	タンパク質の記述子
fingerprint	化合物のフィンガープリント
assay	アッセイデータ
smiles	化合物の SMILES
positive	正例の相互作用ペア
negative	負例の相互作用ペア

（オプション）

-h	ヘルプの表示
-t	データファイル形式をタブ区切りとして読込みます。
-n	DB ファイル中の既存データを一旦削除してからデータファイルを追加します。
-i <int>	正例・負例の追加の際にサンプリング ID を指定します。

add_model：SVM モデルを追加します。

（書式）

cgbvs　add_model　[オプション]　<DB ファイル >　<SVM モデルファイル >...

（説明）

SVM の機械学習で出力されるモデルファイルを DB ファイルに取り込みます。取り込みの際に対応する学習データのサンプリング ID が自動的に付与されます。SVM モデルファイル名の一部にワイルドカード “*” を指定して一度に複数のモデルファイルを取り込むことも可能です。既に同じサンプリング ID がモデルデータが存在する場合は、一旦削除してから SVM モデルを取り込みます。

（オプション）

-h　　ヘルプの表示

comment：コメントを記入します。

（書式）

cgbvs comment [オプション] <DB ファイル > < コメント > < データタイプ >

（説明）

<DB ファイル > で指定した DB ファイルに、< データタイプ > で指定したものについてコメントを記入します。記入するかどうかは任意ですが、通常は化合物やタンパク質の記述子としてどの様なものを使ったかを記入してください。指定できるデータタイプの種類は次の通りです。

title	DB ファイルのタイトル
compound	化合物の記述子
protein	タンパク質の記述子
fingerprint	化合物のフィンガープリント

（オプション）

-h	ヘルプの表示

create：空の DB ファイルを作成します。

（書式）

cgbvs　create　[オプション]　<DB ファイル >　[<uniprot>]

（説明）

何もデータを登録していない空の DB ファイルを作ります。通常は初期 DB ファイルから CGBVS モデルを構築するため、このコマンドを使用する機会はほとんど無いでしょう。<uniprot> はタンパク質に関する accession やタンパク質名に関するデータを指定すると読込みます。

（オプション）

-h	ヘルプの表示

delete：データを削除します。

（書式）

cgbvs　delete　[オプション]　<DB ファイル >　< データタイプ >

（説明）

<DB ファイル > で指定した DB ファイルから、< データタイプ > で指定した種類の情報を削除します。指定できるデータタイプの種類は次の通りです。

compound	化合物の記述子
protein	タンパク質の記述子
fingerprint	化合物のフィンガープリント
assay	アッセイデータ
smiles	化合物の SMILES
positive	正例の相互作用ペア
negative	負例の相互作用ペア

（オプション）

-h	ヘルプの表示

del_model：SVM モデルを削除します。

（書式）

cgbvs　del_model　[オプション]　<DB ファイル >　< サンプリング ID>

（説明）

<DB ファイル > で指定した DB ファイルから、< サンプリング ID> で指定した番号の SVM モデルを削除します。登録されているモデルの一覧は “cgbvs status” で確認できます。また、< サンプリング ID> に “all” を指定すると、すべての SVM モデルを削除します。

（オプション）

-h	ヘルプの表示

learn：機械学習のための入力ファイル作成を行います。

（書式）

cgbvs learn [オプション] <DB ファイル > [< 化合物スケーリング >] [< タンパク質スケーリング >]

（説明）

DB ファイル内に登録されているデータ（化合物記述子、タンパク記述子、相互作用ペア）から、SVM で機械学習するための入力ファイルの生成します。“-i” オプションでサンプリング ID を指定すると、対応する相互作用ペアを利用したファイルが出力されます。ID の指定が無い場合はサンプリング ID の最大値が自動的に選択されます。< 化合物スケーリング > と < タンパク質スケーリング > はそれぞれのスケーリング方法を指定します。指定が無い場合はオートスケーリング法（分布の平均を 0、標準偏差を 1 に変換する方法）が選択されます。

standard	オートスケーリングします。
minmax	最大値を 1、最小値を 0 になるように変換します。
none	スケーリングしません。通常は選択しないでください。

（オプション）

-h	ヘルプの表示
-i <int>	正例・負例のサンプリング ID を指定します。
-C <arg>	スケーリング後の化合物記述子について主成分分析で次元削減を行います。値に整数を指定した場合はその値だけ主成分を選択します。また、実数値に “%” を付けた場合は累積寄与率で主成分を選択します。
-P <arg>	スケーリング後のタンパク質記述子について主成分分析で次元削減を行います。値に整数を指定した場合はその値だけ主成分を選択します。また、実数値に “%” を付けた場合は累積寄与率で主成分を選択します。

predict：CGBVS スコアを算出します。

（書式）

cgbvs predict [オプション] <DB ファイル > < タンパク質 ID> < 化合物記述子ファイル >

（説明）

<DB ファイル > で指定した CGBVS モデルを用いて、< タンパク質 ID> で指定したターゲットに対して < 化合物記述子ファイル > で指定したファイル中の化合物の予測スコアを算出します。“-i” オプションでサンプリング ID が指定されない場合は、サンプリング ID の最大値が自動的に選択されます。

予測したい化合物は予め記述子を計算して所定の形式でファイルする必要があります。また < タンパク質 ID> はコンマ区切りで複数指定することが可能です。また、“%” を文字列のワイルドカードとして利用でき、“all” を指定とすることで DB ファイルに正例又は負例が登録されている全てのタンパクについてスコアを算出します。尚、利用可能なタンパク質の一覧については “cgbvs status” で確認して下さい。

（オプション）

-h	ヘルプの表示
-i <int>	正例・負例のサンプリング ID を指定します。
-a	正例・負例の無いタンパク質であっても記述子が登録されていればスコアを計算します。
-t	縦にタンパク質 ID で、横に化合物 ID を並べた形式で結果を出力します。
-v	行列形式ではなく、化合物とタンパク質のペア毎のスコア値と決定関数値を出力します。
-d	CGBVS スコアの代わりに決定関数値を出力します。

prepare： アッセイデータから正例・負例を作成します。

（書式）

cgbvs　prepare　[オプション]　<DB ファイル >　[< タンパク質リスト・ID>]

（説明）

タンパク質のグループと活性値の設定条件より、アッセイデータから CGBVS モデルに利用できる正例・負例を作成します。< タンパク質リスト・ID> にタンパク質グループの ID 番号を指定した場合は、そのグループのタンパク質が設定されます。また、タンパク質 ID（accession）を記述したファイルを指定した場合は、そのファイルに記述しているタンパク質が設定されます。尚、タンパク質グループ ID の一覧を確認するには < タンパク質リスト・ID> を指定せずにコマンドを実行して下さい。“-i” で ID 番号を指定した場合は < タンパク質リスト・ID> に指定したタンパク質リストを DB ファイルに登録します。

既定の正例・負例に分ける活性値の条件は、正例が 30μM 以下、負例が 50μM 以上です。これらを変更する場合は “-p” または “-n” オプションで nM 単位で設定して下さい。

（オプション）

-h	ヘルプの表示
-i <int>	タンパク質グループ ID を指定します。
-m <string>	タンパク質グループ名を指定します。
-d <int>	指定した ID のタンパク質グループを削除します。
-n <real>	負例とするための活性値の条件を nM 単位で指定します。
-p <real>	正例とするための活性値の条件を nM 単位で指定します。
-S	正例・負例の設定後にファイルサイズを縮小（shrink）します。
-s <int>	指定したタンパク質グループの周辺のタンパク質を、記述子のユークリッド距離が近い順に指定された数だけ選択します。

sample：正例・負例をサンプリングします

（書式）

cgbvs　sample　[オプション]　<DB ファイル >　[< 処理 >]

（説明）

“cgbvs prepare” コマンドで作成した正例・負例から、実際に SVM でモデルを作成する学習セットをサンプリングします。サンプリング方法は大きく 3 通りあります。1 つ目は全ての正例・負例を選択する。2 つ目はタンパク質毎の化合物数の上限を指定してサンプリングする。3 つ目は前回のサンプリングの学習結果（バリデーション）を基にして新たな学習セットをサンプリングする方法です。指定できる処理の種類は次の通りです。

status	現在のサンプリングの状態を表示します。（既定）
execute	サンプリングを実行します。
active	バリデーション結果を基にサンプリングします。
delete	サンプリングされたデータセットを削除します。
virtual	仮想的負例のみを再生成します。

（オプション）

-h	ヘルプの表示
-i <int>	正例・負例のサンプリング ID を指定します。
-n <int>	仮想的負例のセット数を指定します。規定値は 5 です。この値を 0 にした場合は仮想的負例を生成しません。
-s <arg>	1 タンパク質あたりのサンプリングする正例の化合物数の上限または割合（%）を指定します。
-t <arg>	1 タンパク質あたりのサンプリングする負例の化合物数の上限または割合（%）を指定します。

similarity：構造類似度（similarity）を算出します。

（書式）

cgbvs similarity [オプション] <DB ファイル > < タンパク質 ID> < フィンガープリント >

（説明）

<DB ファイル > で指定した CGBVS モデルを用いて、< タンパク質 ID> で指定したターゲットに対して < フィンガープリント > で指定したファイル中の化合物との構造類似度を算出します。各タンパク質には複数の正例化合物がありますが、それら化合物との類似度の最大値です。ここで、構造類似度はフィンガープリントの谷本係数となります。

予測したい化合物は予め記述子を計算して所定の形式でファイルする必要があります。また < タンパク質 ID> はコンマ区切りで複数指定することが可能です。また、“%” を文字列のワイルドカードとして利用でき、“all” を指定とすることで DB ファイルに正例又は負例が登録されている全てのタンパクについてスコアを算出します。“-a” オプションを使用して正例が無い場合を計算すると、算出される値は “-1” となります。尚、利用可能なタンパク質の一覧については “cgbvs status” で確認して下さい。

（オプション）

-h	ヘルプの表示
-a	正例・負例の無いタンパク質であっても記述子が登録されていればスコアを計算します。
-t	縦にタンパク質 ID で、横に化合物 ID を並べた形式で結果を出力します。
-v	行列形式ではなく、化合物とタンパク質のペア毎のスコア値と決定関数値を出力します。

shrink：不要なデータを削除して DB ファイルのサイズを縮小します。

（書式）

cgbvs　shrink　[オプション]　<DB ファイル >　[< データタイプ >]

（説明）

CGBVS モデル作成に使用されていない記述子やサンプリング ID の小さいデータを削除することで、DB ファイルのサイズを減らします。データタイプの指定が無い場合は全ての項目についてデータ削除します。選択できるデータタイプの一覧は以下の通りです。

all	全てのタイプ
compound	化合物の記述子とフィンガープリントと SMILES
protein	タンパク質の記述子と配列情報
assay	アッセイデータ
model	SVM モデル

（オプション）

-h	ヘルプの表示

status：DB ファイル中のモデルの状態を表示します。

（書式）

cgbvs　status　[オプション]　<DB ファイル >　[< カテゴリー >]

（説明）

DB ファイルに登録されているモデルや相互作用データについての内容を、一覧表として出力します。カテゴリーの指定無しの場合は、全体的な情報が出力されます。“-i” オプションでサンプリング ID が指定された場合は、その ID についての正例・負例を加味した情報が表示されます。指定できるカテゴリーの一覧は以下の通りです。

summary	全体的な情報を表示します。
model	モデルに関する部分を表示します。
compound	化合物の一覧を表示します。
protein	タンパク質の一覧を表示します。

（オプション）

-h	ヘルプの表示
-a	正例・負例のない情報も表示します。
-i <int>	正例・負例のサンプリング ID を指定します。
-w <int>	表の列幅の最大文字数を指定します。指定が無い場合は 60 です。

vacuum：SQLite の vacuum コマンドを実行します

（書式）

cgbvs　vacuum　[オプション]　<DB ファイル >

（説明）

SQLite の DB は追記型なので、データを削除してもファイルから完全に削除されません。したがって追加・削除を繰り返すと次第にファイルサイズを大きくなります。この vacuum コマンドを実行すれば不要なデータを完全に削除するので、ファイルサイズを縮小することができます。この機能は以下のような SQLite のコマンドでも実行できます。

“sqlite3 <dbfile> vacuum“

（オプション）

-h	ヘルプの表示

validation：作成したモデルのバリデーションをします

（書式）

cgbvs　validation　[オプション]　<DB ファイル >　[< 処理 >]

（説明）

登録されている正例・負例の CGBVS スコアを計算することによって、DB ファイルに取り込んだ SVM モデルのバリデーションを行います。また、“-d” オプションを利用すると、他の DB ファイルに登録してある正例・負例の CGBVS スコアの計算やバリデーションが実行できます。指定できる処理の種類は次の通りです。

summary	バリデーションの結果を表示します。
execute	バリデーションを実行します。
export	CGBVS スコアと決定関数値を出力します。

（オプション）

-h	ヘルプの表示
-c <real>	CGBVS スコアのカットオフ値を設定します。“summary” を選択した場合のみ有効です。
-o	ROC 曲線から最適なカットオフ値で各種統計量を計算します。
-i <int>	正例・負例のサンプリング ID を指定します。
-r	ROC 曲線の描画のためのデータファイルを出力します。
-d <string>	他の DB ファイル名を指定することにより、その DB ファイルに登録されている正例・負例の CGBVS スコアを計算します。