学習後のニューラルネットワークに基づく MILP (Acyclic Branch-height)の構築方法

2020 年 9 月 3 日

目 次

1 概要

この冊子は，学習後のニューラルネットワークを用いて，特定の化学活性を有する化合物の構造情報を計算する混合整数計画問題 (MILP) を構築する方法について説明したものである．

化合物の構造を数値化した特徴ベクトルと化合物の性質を表す値である活性値の組をもとに学習させたニューラルネットワークを用いて，所望の活性値を有するような化合物の特徴ベクトルを計算する MILP を構築する．この冊子では，この MILP の構築方法と Python を用いたプログラムの実装内容について説明する．

また，このフォルダにはこの冊子以外に以下のフォルダやファイルが含まれている．

- フォルダ source_code
  特徴ベクトルを計算する MILP を Python を用いて実装したプログラムが含まれているフォルダである．
  - `infer_acyclic_graphs.py`
    MILP による計算を実行するプログラムである．このプログラムの使い方は第 4 節を参照すること．
  - `ann_inverter.py`
    ANN の逆問題を定式化した MILP を Python で実装したソースコードである．このプログラムの説明は第**??**節を参照すること．
  - `acyclic_graphs_MILP.py`
    計算した特徴ベクトルを満たす化合物が存在するための必要十分条件を実装したソースコードである．このプログラムの説明は第**??**節を参照すること．
- フォルダ test_files
  ニューラルネットワークによる学習のデータが含まれているフォルダである．
  - `rt_biases.txt`
    化学活性 retention time のデータを用いて学習させたニューラルネットワークのバイアスのデータが含まれているテキストファイルである．フォーマットについては第 3 節を参照すること．また，このデータを用いて実験を行なった計算例を第 4 節に示す．
  - `rt_weights.txt`
    化学活性 retention time のデータを用いて学習させたニューラルネットワークのウェイトのデータが含まれているテキストファイルである．フォーマットについては第 3 節を参照すること．また，このデータを用いて実験を行なった計算例を第 4 節に示す．
  - `rt_fv4_new.txt`
    化学活性 retention time のデータと特徴ベクトルのデータが含まれているテキストファイルである．フォーマットについては第 3 節を参照すること．また，このデータを用いて実験を行なった計算例を第 4 節に示す．
  - `rt_fv4_new_max.txt`
    化学活性 retention time のデータと特徴ベクトルのデータが含まれている各要素の最大値である．AD 制約式の上界値が含まれるファイルである．

– `rt_fv4_new_min.txt`
化学活性 retention time のデータと特徴ベクトルのデータが含まれている各要素の最小値である．AD 制約式の下界値が含まれるファイルである．

次に，この冊子の構成について説明する．第 2 節では，この冊子およびプログラム内で使用している用語について説明する．第 3 節では，プログラムの入力と出力について説明する．この節では，プログラムの入力情報と出力情報を具体例を用いて説明する．また，実際に計算機に入力する際に使用するデータ形式や，プログラム実行後に出力されるデータの形式も説明する．第 4 節では，プログラムの計算例について説明する．この節では具体例を用いて，実際に計算機上でプログラムを実行した結果を紹介する．

以上のような流れで，プログラムの実行内容について説明していく．

2 用語の説明

この節では，冊子とプログラムの中で使用する用語について説明する．この節で紹介した用語をもとに冊子とプログラム内の説明を行っていく．

- 特徴ベクトル
特徴ベクトルとは，化合物の構造から予測する活性と関係のありそうな特徴をいくつか抽出・算出し，それらを成分としたベクトルである．このプログラムでは化合物の特徴ベクトルから，その化合物が持つ活性を予測する．

- ニューラルネットワーク
ニューラルネットワークとは，機械学習の手法の一つであり，入力されたベクトルをもとに目的とする値を予測する手法である．この冊子では，化合物の特徴ベクトルを入力した際に，その化合物が持つ活性値の予測値を出力するようなニューラルネットワークを構築する．

- 入力層，中間層，出力層
ニューラルネットワークはいくつかの層で構成されている．まず，特徴ベクトルが入力される層が入力層である．入力層は特徴ベクトルの要素数と同じ数のノードで構成されている．次に，予測値の計算を段階的に行なっていくのが中間層である．中間層は前の層から入力された値をもとに計算を行い，次の層に出力する．最後に，予測値を出力するのが出力層である．この冊子では一つの活性値を出力するため，出力層は一つのノードで構成されている．ある層に含まれるノードは，次の層の全てのノードに接続されている構造を持っている．

- ウェイト (重み)
ニューラルネットワークに含まれているノード間を接続している枝はそれぞれ値を持っており，その値をウェイトと呼ぶ．ニューラルネットワークは，既知の特徴ベクトルと活性値のペアをデータセットとし，それらを関連付けるために学習を行う．ニューラルネットワークの学習によって，ウェイトが決定される．この冊子では，学習後のニューラルネットワークのウェイトを取り出す方法について説明する．

- バイアス
中間層と出力層に含まれているノードはそれぞれ値を持っており，その値をバイアスと呼ぶ．

ウェイトと同様に，ニューラルネットワークの学習によって，バイアスの値が決定される．この冊子では，学習後のニューラルネットワークのバイアスを取り出す方法についても説明する．

- 活性化関数
  ニューラルネットワークには，予測値を計算するために用いられる活性化関数が定められている．ニューラルネットワークの各ノードでは，入力された値を活性化関数に入力し，その関数の出力値をそのノードの出力値としたて計算を行なっていく．この冊子では Rectified Linear Unit 関数 (ReLU) を活性化関数として扱っていく．

- 混合整数計画問題 (MILP)
  一部の決定変数が整数であり，制約式や目的関数が線形の式で表されている問題を混合整数計画問題 (MILP) と呼ぶ．詳細については参考文献 [3] を参照すること．

- グラフ
  点の有限集合と点対である辺の有限集合によって定義される構造をグラフと呼ぶ．この冊子では辺の方向を考慮する有向グラフを扱う．詳細については参考文献 [2] を参照すること．

- 有向木
  長さ 1 以上の閉路を持たず連結である有向グラフを有向木と呼ぶ．有向木において枝 (u, v) が存在するとき，u は v の親，v を u の子と呼ぶ．詳細については参考文献 [2] を参照すること．

- スキーム木
  すべての対応グラフが部分グラフになれるの木構造のグラフ．

3 プログラムの入力と出力

この節では，プログラムの入力と出力について説明する．3.1 節では，プログラムの入力情報について具体例を用いて説明する．3.2 節では，計算機上で入力する際のデータ形式について説明する．3.3 節では，プログラムの出力情報について具体例を用いて説明する．3.4 節では，計算機上でプログラムを実行した際に出力されるデータ形式について説明する．

3.1 プログラムの入力

この節では，プログラムの入力情報について説明する．このプログラムでは計算で使用する三つのテキストファイル, スキーム木が満たすべき数値データと, 出力される化合物が満たすべき数値データを入力とする．

まず，入力する三つのテキストファイルについて説明する．

一つ目は，特徴ベクトルを入力した際に目的とする化学活性の活性値を予測するニューラルネットワークのウェイトのデータが含まれているテキストファイルである．

二つ目は，特徴ベクトルを入力した際に目的とする化学活性の活性値を予測するニューラルネットワークのバイアスのデータが含まれているテキストファイルである．

三つ目は，目的とする化学活性の活性値を予測するニューラルネットワークを学習させるために用いた特徴ベクトルのデータが含まれているテキストファイルである．

次に, スキーム木が満たすべき三つの数値データ $k^*, \mathrm{bn}_{k^*}, \mathrm{bh}_{k^*}$ を入力する．

最後にに，化合物が満たすべき四つの数値データを説明する．入力する四つの数値は，化合物が持つ原子数 n^*，求められる活性値 tv^*, 化合物の構造グラフの最大次数 d_{max} と直径 dia^* である．

このプログラムでは，入力された三つのテキストファイルと入力スキーム木においてのグラフを持つ出力される化合物が満たすべき数値データをもとに，入力した活性値に近い値を持つような特徴ベクトルを算出する．

3.2 入力データの形式

この説では，計算機上で入力する際のデータ形式について説明する．3.1 節で説明した三つのテキストファイルの形式について具体例を用いて説明する．

このプログラムでは学習後のニューラルネットワークを用いて特徴ベクトルの計算を行う．その具体例が図 1 である．図 1 の学習後のニューラルネットワークには，ウェイトとバイアスが記入されている．この情報を二つのテキストファイルで表現する．

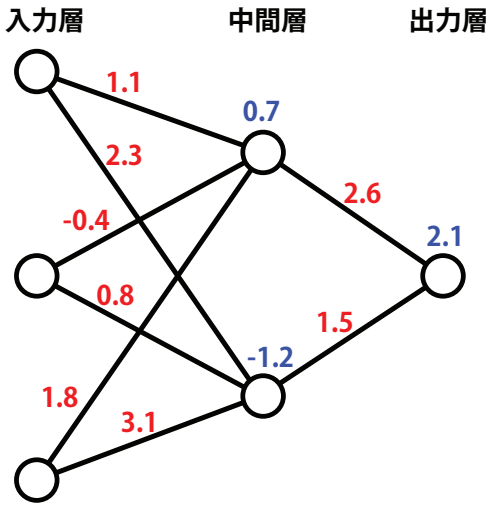

図 1: 学習後のニューラルネットワークの具体例．赤色の数値がウェイト，青色の数値がバイアスを表している．

学習後のニューラルネットワークのウェイトとバイアスの情報は二つのテキストファイルとして出力される．

まず一つ目はウェイトのデータが含まれたテキストファイルである．このテキストファイルでは一行目にニューラルネットワークの構造が記入されている．入力層，中間層，出力層のノード数がそれぞれ記入されている．二行目以降はウェイトのデータが記入されている．各行には入力層のノードから次の中間層の全てのノードへの枝のウェイトが記入されている．続いて，中間層から出力層への枝のウェイトが記入されている．図 1 のウェイトのデータを記入したテキストファイルは以下のようになる．

ウェイトのデータ形式

3 2 1
1.1 2.3
-0.4 0.8
1.8 3.1
2.6
1.5

二つ目はバイアスのデータが含まれたテキストファイルである．各行にバイアスの値が記入されている．記入されている順番は，入力層の一つ目のノードから最後のノード，中間層の一つ目のノードから最後のノード，出力層のノードである．図 1 のバイアスのデータを記入したテキストファイルは以下のようになる．

バイアスのデータ形式

0.7
-1.2
2.1

最後に，特徴ベクトルのデータが含まれたテキストファイルの形式について説明する．このテキストファイルでは，一行目に特徴ベクトルの構成要素が記入されている．二行目以降の各行に特徴ベクトルの数値データが記入されている．このテキストファイルの具体例を以下に示す．

特徴ベクトルのデータ形式

CID,n,M,C,O,H,C1O,C2O,C1C,C2C,#degree1,#degree2,#degree3,#degree4,#degree5,
#degree6,#double_bond,#triple_bond,Diameter,bc_121,bc_122,bc_123,bc_131, bc_132,
bc_141,bc_221,bc_222,bc_223,bc_231,bc_232,bc_241,bc_331,bc_332,bc_341,bc_441,2-
branch_height,2-branch_number
7778,13,126.154,11,2,20,2,1,8,1,4,7,2,0,0,0,2,0,0.769231,0,1,0,3,0,0,5,0,0,2,1,0,0,0,0,0,1,2
86749,11,123.636,10,1,20,1,0,8,1,5,4,1,1,0,0,1,0,0.636364,1,0,0,2,0,2,2,0,0,0,1,2,0,0,0,0,1,2
5282109,13,126.154,11,2,18,2,1,7,2,4,7,2,0,0,0,3,0,0.769231,0,1,0,3,0,0,5,0,0,1,2,0,0,0,0,0,1,2
5319723,11,123.636,10,1,16,1,0,6,3,4,5,2,0,0,0,3,0,0.727273,1,1,0,1,1,0,2,0,0,3,1,0,0,0,0,0,1,2
⋮
⋮

3.3 プログラムの出力

この節では，プログラムの出力情報について説明する．このプログラムでは，与えられた活性値を満たすような特徴ベクトルが存在する場合はその特徴ベクトルを出力する．そなような特徴ベクトルが存在しない場合は，存在しないと出力する．次の節で出力結果の形式について説明する．

3.4 出力データの形式

この説では，計算機上でプログラムを実行した際に出力されるデータ形式について説明する．

まず特徴ベクトルのそれぞれの特徴量が各行に表示される．次に，化学グラフの情報が表示される．各節点が対応する原子が表示される．それぞれの節点には番号が割り振られている．そして，各節点の隣接リストが表示される．リストに含まれる節点番号の原子に結合が存在することを表しており，括弧内の値は結合の多重度を表している．出力結果の具体例を以下に示す．

ターミナル上の出力結果のデータ形式

```
Status:Optimal
Initializing Time: 3.8026208877563477
Solving Time: 20.04114603996277
```

さらに，ターゲット値及び他の引数に応じての名前が変わる `sdf` ファイルが結果として作ります．このファイルは世界中水準の `sdf` フォーマットで一つのグラフを格納しています．

4 プログラムの実行と計算例

この節ではプログラムの実行例を説明する．ここではプログラム `infer_acyclic_graphs.py` の実行方法と結果の具体例を示す．

4.1 環境確認

MacBook Air (13-inch, 2017) において：
プロセッサー 1.8 GHz Intel Core i5
メモリー　8 GB 1600 MHz DDR3
起動ディスク　Macintosh HD
グラフィックス　Intel HD Graphics 6000 1536 MB
のターミナルバージョン 2.9.5 (421.2) でコンパイルすれば問題ないと考えられる。

4.2 実行方法

まず，ターミナル上でディレクトリをフォルダ source_code に変更する．このプログラムを実行するためには，ターミナル上で以下のコマンドを実行する．

`python3 infer_acyclic_graphs.py target value` n^* dia* k^* $d_{\max}$ bn$_{k^*}$ bh$_{k^*}$ `solver_type property`

ここで, solver_type=1 の時, ソルバーとして CPLEX が使われる． solver_type=2 の時, ソルバーとして Coin-OR が使われる．例えば, target value=1900 n^*=15 dia*=10 k^*=2 $d_{\max}$=3 bn$_{k^*}$=3 bh$_{k^*}$=2 solver_type=1 property が retention time とすれば, コマンドが以下のようになる．

```
python3 infer_acyclic_graphs.py 1900 15 10 2 3 3 2 1 rt
```

このコマンドを実行すると計算が実行され，計算結果が出力される．

出力データの形式

```
2
C 120 4 5 8
O 160 2 2 0
3
C C 2 0 3
C O 1 4 0
C C 1 2 5
0 6
4 1
3 1
0 0
10
3
1 2 1 0 1
1 2 2 0 0
1 2 3 0 0
1 3 1 0 2
1 3 2 0 3
1 4 1 0 0
2 2 1 2 0
2 2 2 0 0
2 2 3 0 0
2 3 1 4 1
```

```
2 3 2 0 0
2 4 1 0 0
3 3 1 0 1
3 3 2 0 0
3 4 1 0 0
4 4 1 0 0
```

この具体例を用いて，各行の内容を説明する．数値例とそれぞれの内容の対応を表 1 に示す．

表 1: 入力するテキストファイルの読み方

数値例	内容
2	原子の種類
C 120 4 5 8 O 160 2 2 0	原子のシンボル，質量の十倍，価数，原子の内部節点数，原子の外部節点数
3	原子結合の種類
C C 2 0 3 C O 1 4 0 C C 1 2 5	原子結合(原子，原子，多重度)，内部原子結合の数，外部原子結合の数
0 6 4 1 3 1 0 0	度数が1 の内部節点数，度数が1 の外部節点数 度数が2 の内部節点数，度数が2 の外部節点数 度数が3 の内部節点数，度数が3 の外部節点数 度数が4 の内部節点数，度数が4 の外部節点数
10	直径
3	度数の上限
1 2 1 0 1 1 2 2 0 0 1 2 3 0 0 1 3 1 0 2 1 3 2 0 3 1 4 1 0 0 2 2 1 2 0 2 2 2 0 0 2 2 3 0 0 2 3 1 4 1 2 3 2 0 0 2 4 1 0 0 3 3 1 0 1 3 3 2 0 0 3 4 1 0 0 4 4 1 0 0	度数結合(度数，度数，多重度)，内部度数結合の数，外部度数結合の数

参考文献

[1] N. A. Azam, J. Zhu, Y. Sun, Y. Shi, A. Shurbevski, L. Zhao, H. Nagamochi and T. Akutsu. A Novel Method for Inference of Acyclic Chemical Compounds with Bounded Branch-height Based on Artificial Neural Networks and Integer Programming.

[2] 茨木俊秀，永持仁，石井利昌．グラフ理論ー連結構造とその応用ー．朝倉書店，2010.

[3] 福島雅夫．数理計画入門．朝倉書店，2012.

[4] A Python Linear Programming API, https://github.com/coin-or/pulp.

[5] Optimization with PuLP, http://coin-or.github.io/pulp/.

[6] The Python Papers Monograph, https://ojs.pythonpapers.org/index.php/tppm/article/view/111.

[7] Optimization with PuLP, https://pythonhosted.org/PuLP/.