Inferring an Acyclic Graph with Bounded Branch-Height from a Trained ANN using MILP

September 7, 2020

Contents

1 Outline

This note explains how to use an implementation of a mixed-integer linear programming (MILP) formulation that can infer a vector of graph descriptors given a target value and the weights and bias values of a trained artificial neural network (ANN).

The MILP is implemented in Python, using the PuLP modeling module of the COIN-OR package [5, 6, 7, 8].

To begin with, we give a list of the files that accompany this note.

- Folder `source_code`
  A folder containing three python scripts that implement an MILP formulation for inferring feature vectors of acyclic chemical graphs from a trained ANN, and files containing minimum and maximum values of each descriptor, used for setting the applicability domain in the MILP formulation.
  - `ann_inverter.py`
    An implementation of an MILP formulation for the Inverse problem on ANNs [1].
  - `acyclic_graphs_MILP.py`
    A python script that contains functions to initialize the variables and prepare the constraints for an MILP formulation for inferring acyclic graphs with a bounded branch-tree height [2].
  - `infer_acyclic_graphs.py`
    A python script that prepares the data and executes the MILP formulation for given input data. Further details on the use of this script are given in Section 4.
  - `AD_min.txt`
    A file containing minimum values over all descriptors, used to set the applicability domain part of the MILP constraints.
  - `AD_max.txt`
    A file containing maximum values over all descriptors, used to set the applicability domain part of the MILP constraints.
- Folder `test_files`
  A folder containing data from a trained ANN. For each of the files, the data format is explained in Section 3, and an actual example is given in Section 4.
  - `6Hc_desc.csv`
    A comma-separated value file containing descriptors used in the training of the ANN.
  - `6Hc_biases.txt`
    A file containing the values of the biases of a trained ANN.
  - `6Hc_weights.txt`
    A file containing the values of the weights of a trained ANN.

The remaining of this note is organized as follows. Section 2 gives an explanation of the used terms and notation. Section 3 explains the input and output data of the program, and Sec. 4 gives a concrete example of input data and the results form the computation.

2 Terms and Notation

This section explains the terms and notation used in this note.

- **Feature vector**
  A *feature vector* stores numerical values of certain parameters, called *descriptors*. In this work, we choose graph-theoretical descriptors, such as number of non-hydrogen atoms, number of vertices of certain degree, etc.

- **Artificial neural network**
  Artificial neural networks are one of the methods in machine learning. They provide a means to construct a correlation function between pairs of feature vectors as input and target data as output.

- **Input, hidden, and output layer**
  We deal with the multilayer perceptron model of feed-forward neural networks. These neural networks are constructed of several *layers*. First comes the *input layer*, where each neuron takes as input one value of the feature vector. Next come the *hidden layers*, where the values from the input layer are propagated in a feed-forward manner, such that each node in one layer is connected to all the nodes of the next layer. Finally, the output is deliver at the *output layer*. We deal with predicting the value of a single target, and hence we assume that the output layer comprises a single node.

- **Weights**
  Each edge connecting two nodes in an ANN is assigned a real value, called a *weight*. Part of the *learning* process of ANNs is to determine values for each of the weights based on known pairs of feature vectors and target values.

- **Biases**
  Each node of an ANN except for the nodes in the input layer is assigned a real value, called a *bias*, which, just like the edge weights, is determined through the learning process.

- **Activation function**
  In an ANN, each node produces an output as a function, called the *activation function*, of its input. We assume that each node has the Rectified Linear-Unit (ReLU) function as its activation function, which can be expressed exactly in the MILP formulation for the inverse problem on ANNs [1].

- **Mixed-Integer Linear Programming (MILP)**
  A type of a mathematical programming problem where all the constraints are given as linear expressions, and some of the decision variables are requested to take only integer values. For more details, see any standard reference, e. g. [3].

- **Graph**
  An abstract combinatorial construction comprising a finite set of *vertices*, and a finite set of *edges*, where each edge is a pair of vertices. We treat *undirected* graphs, i. e., graphs where edges are unordered pairs of vertices. For more information, see e. g. [4].

3 The Program's Input and Output

This section explains the format of the input and the output of the program. Section 3.1 illustrates an example of the program's input format, and Section 3.2 gives a concrete computational example. Following, Section 3.3illustrates an example of the program's output format, and Section 3.4 gives a concrete computational example.

3.1 Program Input

This section gives an explanation of the input to the program.

First and foremost comes the target value for which we wish to infer a chemical graph based on a trained ANN.

Next come several parameters pertaining to the graph structure that we seek to infer:
- the number of vertices n^*,
- the diameter dia^*,
- branch-height parameter k^*,
- maximum degree $d_{\max} \in \{3, 4\}$ of a vertex in the target graph,
- the k^*-branch-leaf-number bl_{k^*}, and
- the k^*-branch-height bh_{k^*}.

Next comes a choice of MILP solver program to be used. We can choose
- 1: CPLEX, a commercial MILP solver [9].

(Note, in this case the parameter `CPLEX_PATH` in the file `infer_acyclic_graphs.py` must be set to the correct path of the CPLEX program executable file.)

- 2: CBC, a free and open-source MILP solver that comes together with the PuLP plugin for python [5].

Finally the input requires three textual files containing
- the descriptor names, in csv format
- the weights and biases of a trained ANN in textual format.

For a common prefix `TT` which the program accepts as a command-line parameter, these files must be saved with file names `TT_desc.csv`, `TT_weights.txt` and `TT_biases.txt` for the files containing the descriptor names, the weights, and the biases of a trained ANN, respectively.

3.2 Input Data Format

This section presents an actual example of an input instance of the program. In particular, we give a concrete example of the three input files mentioned in Section 3.1.

The purpose of this program is to calculate a feature vector that will produce a desired output from a given trained ANN. Figure 1 gives an example of a trained ANN.

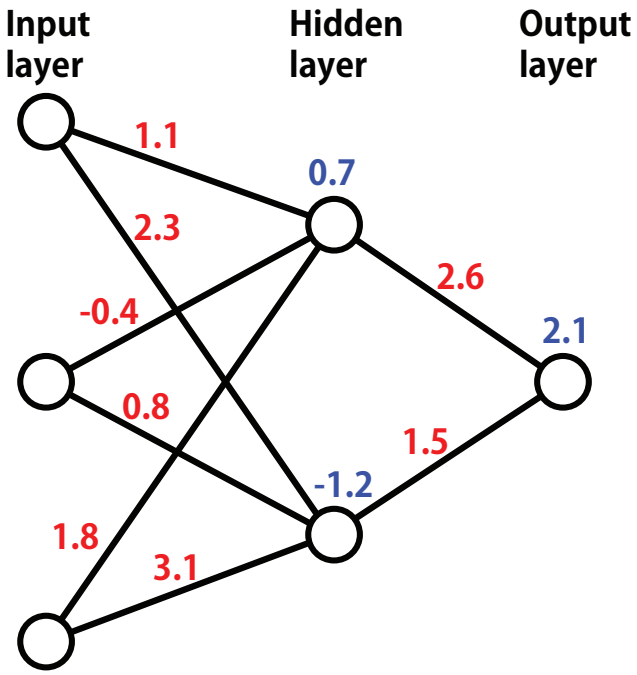

Figure 1: An example of a trained ANN. The ANN's weights are given in red numbers, and its biases in blue.

The information on the trained ANN is written in two text files, containing the information on the ANN's weights and biases, respectively. First, we give the structure of the file that contains the information on the ANN's weights. 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．

Bias values

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 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．

Table 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

References

[1] T. Akutsu and H. Nagamochi. A Mixed Integer Linear Programming Formulation to Artificial Neural Networks, in Proceedings of the 2019 2nd International Conference on Information Science and Systems, pp. 215–220, https://doi.org/10.1145/3322645.3322683.

[2] 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.

[3] J. Matousek and B. Gärtner. Understanding and Using Linear Programming. Springer, 2007.

[4] M. S. Rahman. Basic Graph Theory. Springer, 2017.

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

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

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

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

[9] IBM ILOG CPLEX Optimization Studio 12.8 User Manual. https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/usrcplex.pdf.