Module 2: Training an Artificial Neural Network

`mol-infer/Cyclic`

February 12, 2021

Contents

1 Introduction

In this article, we explain how to proceed to Module 2, i.e., to training an artificial neural network (ANN) in our project (`mol-infer/Cyclic`).

We denote by $D_\pi = \{G_1, G_2, \ldots, G_p\}$ a given set of chemical graphs, and by f a function that maps a chemical graph to a feature vector. We define $\mathcal{F}(D_\pi) \triangleq \{f(G_1), f(G_2), \ldots, f(G_p)\}$. Let us denote by π a chemical property that is considered. For example, π can be boiling point, heat of combustion, Kow, and so on. The input and output of the module are summarized as follows.

Input: A set $\mathcal{F}(D_\pi) = \{x_1, x_2, \ldots, x_p\}$ of molecule feature vectors, a set $\{a(x_1), a(x_2), \ldots, a(x_p)\}$ of observed values $a(x_i)$ for each $G_i \in D_\pi$ (and $x_i = f(G_i)$) for the property π, and values specifying the the architecture of a desired ANN, that is, the number of hidden layers, and the number of nodes in each hidden layer.

Output: An ANN that has architecture as specified in the input, and that estimates $a(x)$ "well" for many feature vectors $x \in D_\pi$.

Concretely, the output consists of weights of the arcs and biases of the nodes in the constructed neural network.

The article is organized as follows.

- Section 2: Some basic terminology and the roles of the files in the package are explained.
- Section 3: A brief example of executing the programs.
- Section 4: Details about the input and the output of the programs.

2 Preparation

2.1 Basic Terminology

Feature vector. A numerical vector that represents features of a molecule. Each dimension corresponds to a *descriptor* that describes such features as the number of atoms of different chemical elements (e.g., carbon, nitrogen, oxygen, etc), the molecular mass averaged over the number of atoms, etc.

Artificial neural network (ANN). A well-known model in machine learning. In this project, we use ANNs to solve the *regression* problem. In this problem, given a data set that consists of numerical vectors and the "correct" values assigned to each of the vectors, we are asked to predict values for vectors not in the data set as accurately as possible. In our project, the numerical vectors are feature vectors that are extracted from molecules and the values assigned to the vectors are property values of a prescribed chemical property. The number of dimensions is assumed to be equal over all the feature vectors.

The architecture of ANNs that we use in the project is restricted to feed-forward networks, which can be represented by a directed acyclic graph. Figure 1 shows an example.

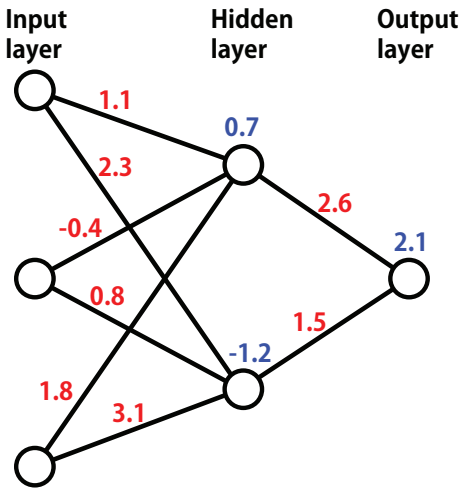

Figure 1: An example of an ANN. A red value indicates a weight, whereas a blue value indicates a bias. The direction of each arc is from left to right.

Layers of an ANN. In the multi-layered perceptron model of ANNs, an ANN consists of several *layers*.

- The first layer is the *input layer*. Each node in the input layer is associated with a dimension of a feature vector (i.e., a descriptor). Then the number of nodes in the input layer is equal to the number of dimensions of a feature vector.

- The subsequent layers (except the last one) are called *hidden layers*. A node in a

hidden layer outputs a value based on the input values, where the input values are the outputs of all nodes in the previous layer.

- The last layer is the *output layer* that has only one node. The output of this node is regarded as the output of the ANN.

In the ANN of Figure 1, the input layer has three nodes. This means that a feature vector should be have three dimensions as well. The ANN has one hidden layer that has two nodes. The output layer has exactly one node.

Weights. In ANNs, each edge is assigned a numerical value, called a *weight*. The weights are determined by *training* the ANN, based on the given data set.

In Figure 1, the weights of edges are indicated in red.

Biases. In ANNs, each node in the hidden layers and the output layer is assigned a numerical value, which is called a *bias*. The biases are determined together with the weight by training the ANN based on the given data set.

In Figure 1, the biases of nodes are indicated in blue.

Activation function. In ANNs, each node except the output one is assigned an activation function. This function computes the output of the node according to the weighted sum of the input values minus the bias of the node.

In this project, we use the Rectified Linear-Unit (ReLU) function for the activation function of each node in the hidden layers. This choice is due to Module 3, where it is assumed that the ReLU function is used as an activation function for ANNs in Module 2.

The process of deciding (appropriate) weights and biases from a given data is called a *learning*.

2.2 Files in the package

Table 1 shows the list of files in the package of Module 2.

Table 1: List of files in the package of Module 2

File	Description
`mol-infer_ANN.py`	A Python script to train an ANN **(To proceed to Module 3,** **it is necessary to run this script)** • Used non-standard libraries: `numpy, pandas, scikit-learn`
`predict_values.py`	A supplementary Python script that predicts the value of a chemical property for molecules in a given data set (It is not necessary to run this script) • Used non-standard libraries: `numpy, pandas`

`Manual_Module_2_Cyclic_jp.pdf`
`Manual_Module_2_Cyclic_jp.tex`
`fig/ANN_sample_jp.eps`
	PDF, LaTeX source of the manual
	and the image file (Japanese version)

`Manual_Module_2_Cyclic_en.pdf`
`Manual_Module_2_Cyclic_en.tex`
`fig/ANN_sample_en.eps`
	PDF, LaTeX source of the manual
	and the image file (English version)

Data files for BP (boiling point) [1]	
`data/BP.sdf`	SDF file containing molecule data. This file is not explicitly treated in Module 2.
`data/BP_fv.csv`	File containing feature vectors that are obtained by applying Module 1 to `BP.sdf`
`data/BP_value.csv`	File containing BP values of molecules
`data/BP_ANN.LOG`	Log of executing `mol-infer_ANN.py` for the data set that consists of `BP_fv.csv, BP_value.csv`
`data/BP_ANN_biases.txt` `data/BP_ANN_weights.txt`	The biases and weights of the constructed NN

(Cont. of Table 1)

File	Description
Data files for HC (heat of combustion) [1]	
data/HC.sdf	(All are analogous to BP)
data/HC_fv.csv	
data/HC_value.csv	
data/HC_ANN.LOG	
data/HC_ANN_biases.txt	
data/HC_ANN_weights.txt	
Data files for KOW (log Kow) [1]	
data/KOW.sdf	(All are analogous to BP)
data/KOW_fv.csv	
data/KOW_value.csv	
data/KOW_ANN.LOG	
data/KOW_ANN_biases.txt	
data/KOW_ANN_weights.txt	
Data files for MP (melting point) [1]	
data/MP.sdf	(All are analogous to BP)
data/MP_fv.csv	
data/MP_value.csv	
data/MP_ANN.LOG	
data/MP_ANN_biases.txt	
data/MP_ANN_weights.txt	

3 A Quick Start

Training an ANN. The following command constructs an ANN from the dataset such that

- the set of feature vectors is written in `data/BP_fv.csv`; and
- the set of property values (BP in this case) is written in `data/BP_value.csv`

```
$ python mol-infer_ANN.py data/BP_fv.csv data/BP_value.csv output 20 10
```

The constructed NN has two hidden layers, where the numbers of nodes in these layers are 20 and 10, respectively. The weights of arcs are output to the file `output_weights.txt` and the biases of nodes are output to the file `output_biases.txt`. **The files containing weights and biases are used in Module 3**.

Predicting the property value of a given molecule. The property value of a molecule can be predicted by using a trained ANN.

The following command invokes a Python script that reads the information (edge weights and node biases) of a trained ANN, and then uses each of the feature vectors given in the file `data/BP_fv.csv` as input to the ANN's prediction function. The edge weights of the trained ANN are read from the file `output_weights.txt`, and node biases are read from the file `output_biases.txt`. The output results are written to the file `predicted.txt`.

The symbol \ at the end of the first line indicates that there is no line break between the two lines, that is, they should be input without pressing the "enter" key.

```
$ python predict_values.py output_weights.txt output_biases.txt \
    data/BP_fv.csv predicted.txt
```

- Given an SDF file, one can generate the feature vectors of the molecules in the SDF by using the program in Module 1.
- This script is supplementary, and it is not directly used in Modules 3 and 4.

4 Details in the Input and Output of the Program

4.1 Input

4.1.1 Feature vectors

The feature vectors should be written in a so-called **FV format** to a csv (comma-separated value) file. The feature vector generator in Module 1 generates a csv file in the FV format

from a given SDF. As long as one uses the csv file generated in Module 1, no problem should occur in Module 2.

We explain the structure of the FV format using a simple example.

```
CID,n,cs,ch,nsH
244,8,6,2,8
307,10,6,4,8
657014,11,7,1,18
16704,9,9,0,10
```

- In the first line, the descriptor names should be written in comma-separated style.
- The first descriptor **must be** CID (Compound ID). The CID values are not used to train an ANN, just for identification.
- In the above example, four descriptors are introduced: `n` (the number of atoms except hydrogens), `cs` (core-size), `ch` (core-height), and `nsH` (the number of hydrogens).
- In each of the subsequent lines, the CID and the feature vector of one molecule should be specified in comma-separated style.
- The molecules **need not be** ordered according to CID.

4.1.2 Property values

The property values should be described in a csv file as follows.

```
CID,a
307,11.2
244,-0.5
657014,98.124
16704,-12.8
117,5.3
```

- **The first line should be `CID,a`.**
- In each of the subsequent lines, CID and the property value of one molecule should be written in comma-separated style.
- The molecules need not be ordered with respect to CID.

4.2 Execution

We use the Python script `mol-infer_ANN.py` to train an ANN, as outlined in Section 3.

4.2.1 Arguments

We here revisit the commands explained in Section 3.

```
$ python mol-infer_ANN.py data/BP_fv.csv data/BP_value.csv output 20 10
```

The meaning of each of the command-line parameters is as follows:

- Parameter 1: A csv file the stores the feature vectors of chemical compounds,
- Parameter 2: A csv file that stores observed values of a chemical property/activity of chemical compounds,
- Parameter 3: Filename where the weights and biases of a trained ANN will be stored,
- Parameter 4: The number of nodes of hidden layers in the ANN, each layer separated by a space.

If the Python script is invoked without supplying command-line arguments, or in the case that there is a significant error in the arguments, then a brief description of the command-line parameters will be printed.

A verbatim output of the output of the program when invoked without any command-line parameters. The backslash symbol (\) indicates that there is no actual line break.

```
$ python mol-infer_ANN.py

usage: mol-infer_ANN.py (TrSet_FeatureVector.csv)(TrSet_TargetValue.csv) \
    (Output_Name)(NN_Architecture)

  - TrSet_FeatureVector.csv ... \
      The CSV file generated by our feature vector generator in Module 1.
  - TrSet_TargetValue.csv ... \
      The CSV file that contains target values.
    = The first column must contain CIDs.
    = The second column must contain target values and be named 'a'.
    = All CIDs appearing in TrSet_FeatureVector.csv must be contained in \
      TrSet_TargetValue.csv.
  - OutputName_{biases,weights}.txt will be output. \
      They contain data on the constructed ANN.
  - NN_Architecture ... Number of nodes in hidden layer(s).
```

If the program is invoked with a proper set of command-line parameters, then the training process of an ANN will begin.

To evaluate the performance of the trained ANN, we use 5-fold cross-validation. The ANN that achieved the highest coefficient of determination (R^2 score) for the test set over the five trials is selected, and its weights and biases are stored in a file. Using the above example of parameter values, where the value of parameter 3 is set to be `output`, the following two files will be written to disk:

- `output_weights.txt` (stores the weights)
- `output_biases.txt` (stores the biases)

These two files are necessary in Module 3.

Caution on the data set The training set for training the ANN is given in the following two files

- a csv file with feature vectors of chemical compounds, and
- a csv file with observed values for a chemical property/activity.

Both files include the CID number of the corresponding chemical compound. One important point to note is that

each of the CIDs in the former file must appear in the latter

however, the opposite is not enforced. In other words, there might be a chemical compound for which an observed value of some property/activity is listed in the latter file, but whose feature vector is not included in the former. Such values will be silently ignored.

4.2.2 Hyper-parameters

We use the Python library `scikit-learn`[1] and its `MLPRegressor` tool to do the training of an ANN as a multi-layered perceptron. After line 133 in the Python script `mol-infer_ANN.py`, an instance of an `MLPRegressor` is initialized, and its hyperparameters can be changed at this point. Some of the various parameters are set as follows:

- `activation`: `'relu'`
  Attention: If the trained neural network is to be used in Module 3 and afterward, the only possible choice here is `'relu'` (Rectified Linear Unit Function - ReLU).
- `alpha`: 10^{-5}
- `early_stopping`: `False`
- `hidden_layer_sizes`: Passed as a command-line parameter at invocation

[1] `https://scikit-learn.org/stable/`

- `max_iter`: 10^{10}
- `random_state`: 1
- `solver`: 'adam'

4.3 Output

We revisit the command issued as below, and explain the resulting output.

```
$ python mol-infer_ANN.py data/BP_fv.csv data/BP_value.csv output 20 10
```

4.3.1 Standard output

When the above command is issued on the command prompt, the results of the computation process will be displayed on the standard output. An example of this output is stored in the file `data/BP_ANN.LOG` which is included together with the program package and this note.

```
src/preparation/BP_fv.csv contains 181 vectors for 107 (=CID+106) features.
src/preparation/BP_value.csv contains 230 target values.
n range = [5,30]
a range = [31.5,470.0]
#instances = 181
#features = 106

D1: train 144, test 37
training time: 3.187196731567383
R2 score train = 0.9935319077425955
R2 score test = 0.7855694851759929
R2 score all = 0.9599908495442584
MAE score train = 3.8570127089010384
MAE score test = 19.756408746147212
MAE score all = 7.10716548999556

D2: train 145, test 36
training time: 5.0139079093933105
R2 score train = 0.9930416804444452
R2 score test = 0.6390572625074291
R2 score all = 0.9339056805642507
MAE score train = 3.7281172621746888
MAE score test = 27.996544836634186
MAE score all = 8.554986834995363
```

```
D3: train 145, test 36
training time: 4.264358758926392
R2 score train = 0.9961346879653636
R2 score test = 0.8566979846124056
R2 score all = 0.9637772344809027
MAE score train = 2.870368503215682
MAE score test = 22.925217956024927
MAE score all = 6.8591783391335435

D4: train 145, test 36
training time: 2.9935202598571777
R2 score train = 0.9909067339023991
R2 score test = 0.8390994594153819
R2 score all = 0.9669036994338265
MAE score train = 4.470398694611737
MAE score test = 18.691072072382227
MAE score all = 7.298819918919681

D5: train 145, test 36
training time: 4.9648661613464355
R2 score train = 0.9946933391220482
R2 score test = 0.8479544758088942
R2 score all = 0.9574271159388575
MAE score train = 3.352287445970431
MAE score test = 21.80973582058148
MAE score all = 7.023382150312958
0.9935319077425955 0.7855694851759929 0.9599908495442584 3.187196731567383
0.9930416804444452 0.6390572625074291 0.9339056805642507 5.0139079093933105
0.9961346879653636 0.8566979846124056 0.9637772344809027 4.264358758926392
0.9909067339023991 0.8390994594153819 0.9669036994338265 2.9935202598571777
0.9946933391220482 0.8479544758088942 0.9574271159388575 4.9648661613464355
Average time = 4.08476996421814
Average R2 test score = 0.7936757335040208
Average MAE test score = 22.235795886354005
```

- First, the number of feature vectors, as well as the number of descriptors are output. Next comes the range (`n range`, in the above example 5 – 30) of the number of non-hydrogen atoms (descriptor `n`) in the chemical compounds in the training set, followed by the range (`a range`, 31.5 – 470.0 above) of the observed chemical property/activity in the training data (in this case, boiling point).

- Following, the summary of each of the five runs of the 5-fold cross-validation is

printed.

- Finally, the average of the calculation time, R^2 score of the test set and MAE score of the test set over the five folds are printed.

- In the above example, the ANN constructed in the third of the five folds had the highest R^2 score of 0.856697..., and therefore the values of its weights and biases are written in the files `output_weights.txt` and `output_biases.txt`, respectively.

- Note that a copy of each of the above files, `output_weights.txt` and `output_biases.txt`, is included in the set of files together with this note.

4.3.2 Edge weights

Following we explain the contents of the file `mol-infer_ANN.py` that contains the values of the weights of the ANN.

For simplicity, let us assume that the ANN depicted in Figure 1 has been obtained after training. The contents of the file with the weights of this ANN would be as follows.

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

- The first row gives the architecture of the ANN, that is the number of nodes in each of its layers. starting with the input layer, then the hidden layers, and finally the output layer.

- Starting from the second line and onward, follow the weights of the ANN. Each row gives the weights of the edges going out of a single node.

4.3.3 Biases of nodes

In a similar manner, we give an explanation of the file that contains the biases of the ANN obtained by executing the Python script `mol-infer_ANN.py`.

For sake of simplicity, again consider the ANN depicted in Figure 1. The values for the node biases of this ANN would be given as follows.

```
0.7
-1.2
2.1
```

Each row contains the bias value of a single node. Note that the nodes of the output layer do not have biases.

References

[1] HSDB in PubChem `https://pubchem.ncbi.nlm.nih.gov` (accessed on February 1st, 2021)