EN 813706 Artificial Neural Networks

Final Examination 2021

Faculty of Engineering, Khon Kaen University

Submission: https://autolab.en.kku.ac.th ; assessment: **FE**

Take-home exam with Q&A session:

ZOOM with Meeting ID: 930 5373 0556 Passcode: 035785 (Nov 20th, 2021: 2pm-3pm)

==

* This is a take-home exam: Th 18 Nov - W 24 Nov, 2021. The assessment is accessible from 6:00am of Nov 18th, 2021 to 8:00pm of Nov 24th, 2021.

Submission is taken until Nov 24th, 2021 at 8:15pm with late penalty 50%.

* Maximum submission is unlimited. But, any version OVER 8 will be penalized 10% each.

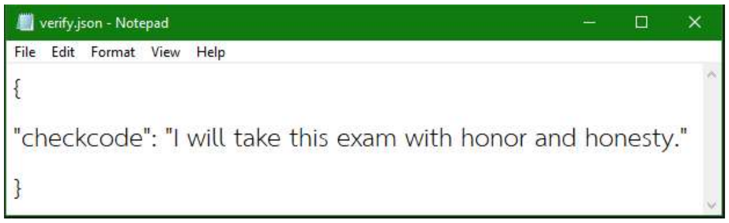
* Personal communication regarding the FE through any mean is prohibited. Only authorized communications (i.e., answer submission and exam administrator's communication) are allowed.

* Submit a program to a (programming) problem with a file with a proper extension. E.g., a python program for P2 should be named "`P2.py`"

* If you need to have data files along with your code, you can have them with naming as follows. Any data file accompanied a code must have its name started with problem id following by an underscore, e.g., `P2_data.npy` for a data file used for P2, `P3_additional.data`, `P3_wb.csv` for P3. See Figure 2 for example.

* All programs and associated files must be packaged together in a tar file. DO NOT HAVE ANY FOLDER IN THE TAR FILE.

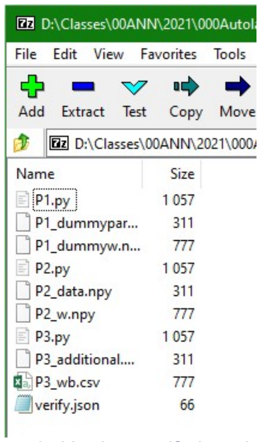
* An integrity affirmation must be declared: write the declaration in `verify.json` as shown in Figure 1 and have `verify.json` in the tar file as shown in Figure 2.



```
{
"checkcode": "I will take this exam with honor and honesty."
}
```

Figure 1. Integrity affirmation in verify.json.

Caution! There is a period/full stop/dot ‘.’ at the end of the declaration.


D:\Classes\00ANN\2021\000Autol
File Edit View Favorites Tools
Add Extract Test Copy Move
D:\Classes\00ANN\2021\000/

Name	Size
P1.py	1 057
P1_dummypar...	311
P1_dummyw.n...	777
P2.py	1 057
P2_data.npy	311
P2_w.npy	777
P3.py	1 057
P3_additional....	311
P3_wb.csv	777
verify.json	66

Figure 2. Having verify.json in the tar file.

* Each question or problem is worth 840 points. There are 2 problems, thus 1680 points are counted as a full score.

* Autograder is run on a docker image with `Python 3.9.4` and `Numpy 1.21.1`. No other non-core python library is available: e.g., no `SciKit-learn` nor `Matplotlib`.

* Problem will be independently graded one at a time. Write each program without dependency on others. E.g. code such as `from P1 import calcA` will give an error when P2 is graded.

==

(**P1**) Given ***Task 1*** (below), prepare a predictive model, i.e., a function to take the input and return the output. The function has interface

```
P1_predict(x)
```

where `x` is a `numpy.array` of shape `(N, D)` and it returns the predicted labels as a list of class labels of length `N` when `N` is a number of data points and `D` is a number of input dimensions.

Note: if input pre-processing and/or output post-processing are needed, they can be done within this function. Also, if the trained weights or other saved parameters are needed, they can be loaded within this function. (See submission example, if you need.)

Task 1 *(Synthetic 4-class classification)*: build a model to classify 2-D input.

This synthetic dataset has 4000 data points each having 3 values: 2 attributes and 1 class label. A class label can be one of 4 classes: 'A', 'B', 'C', and 'D'.

Build and prepare (including train) the model using the provided dataset `P1data.npy`.

See examples below (also provided in the `handout.tar`).

Data loading example

```
dat = np.load('P1data.npy', allow_pickle=True)
X = dat[:,:-1].astype(float)
Y = dat[:, -1]

print(X.shape)
print(Y.shape)
print(X[:5,:])
print(Y[:5])
```

Submission example

```
import numpy as np
from ANN_feAux import *

def P1_predict(x):
  """
  x: np.array of shape (N,D)
  return y: list of classes
  """

  # WRITE YOUR CODE HERE!!!

  # Load params
  loaded_par = np.load('P1_dummypar.npy', allow_pickle=True).tolist()
  # Caution! you have to supply the data file! P1_dummypar.npy


  # Load weights
```

You can use numpy and ANN_feAux.py.

You can load data in. Just name a data with prefix P1_ for P1.

```
    loaded_net = np.load('P1_dummyw.npy', allow_pickle=True).tolist()
    # Caution! you have to supply the data file! P1_dummyweights0.npy

    # Pre-process the input
    px = x * loaded_par['p1']

    # Run network
    Yp = mlp2(loaded_net['c'], loaded_net['v'],
            loaded_net['b'], loaded_net['w'], px, identity)

    # Post-process the output
    postY = Yp + loaded_par['p2']
    class_labels = ['A', 'B', 'C', 'D']
    predicted_labels = [class_labels[int(n)] for n in np.round(postY)]

    return predicted_labels
```

P1.py

Do not forget to properly have output in the final state.

Evaluable

```
import numpy as np
from P1 import P1_predict

if __name__ == '__main__':
  dat = np.load('P1data.npy', allow_pickle=True)
  X = dat[:,:-1].astype(float)
  Y = dat[:, -1]

  plabs = P1_predict(X)

  acc = np.mean(plabs == Y)
  print('Accuracy', acc)
```

===

Caution! if your code cannot be evaluated properly with the evaluable, it is likely that it will not be graded properly on Autograder.

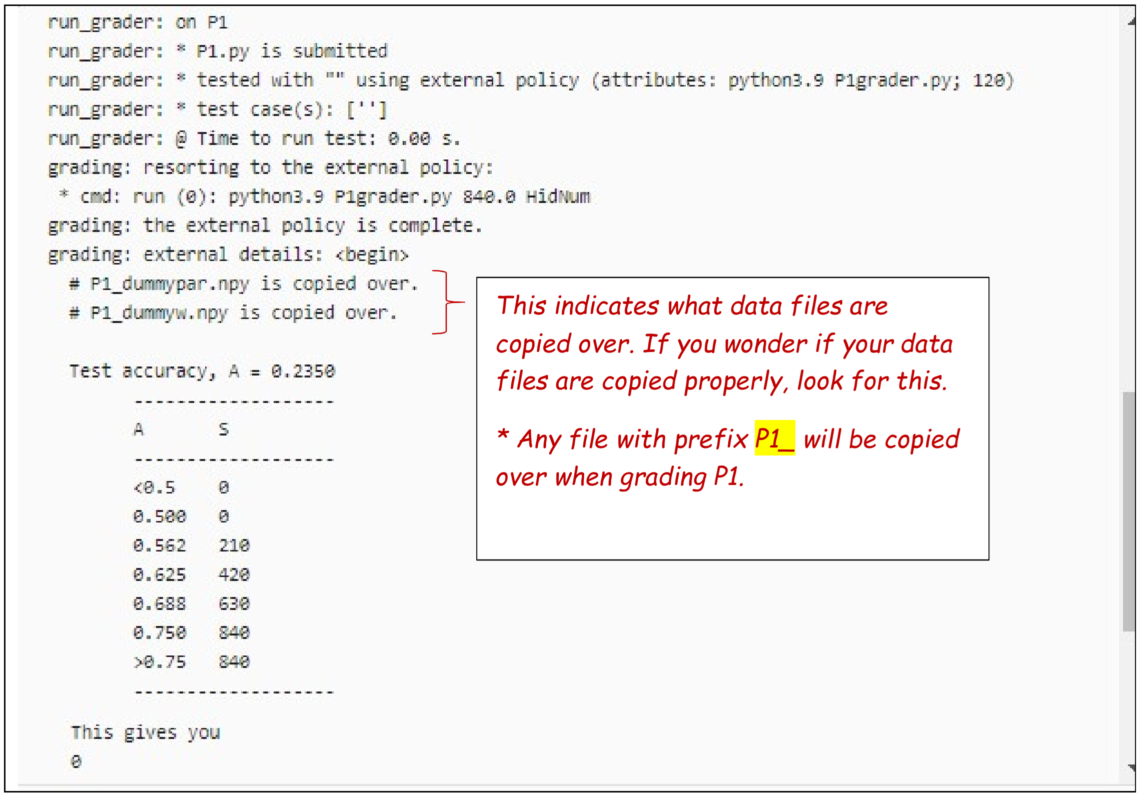
Note: only `P1.py` and your data files (any file with prefix `P1_`) will be copied over to the grading arena. Auxiliary `ANN_feAux.py` is also provided with its fresh copy (not your copy!).

Still confused? You can try:
* Run `evaluable1.py` on the terminal. It should give you something like:

Terminal

```
C:\> python evaluable1.py
Accuracy 0.25025
```

* Or, try submit the example files (`P1.py` along with its data files) to the grader. It should give you something like:



```
run_grader: on P1
run_grader: * P1.py is submitted
run_grader: * tested with "" using external policy (attributes: python3.9 P1grader.py; 120)
run_grader: * test case(s): ['']
run_grader: @ Time to run test: 0.00 s.
grading: resorting to the external policy:
 * cmd: run (0): python3.9 P1grader.py 840.0 HidNum
grading: the external policy is complete.
grading: external details: <begin>
  # P1_dummypar.npy is copied over.
  # P1_dummyw.npy is copied over.

  Test accuracy, A = 0.2350
        -------------------
        A       S
        -------------------
        <0.5    0
        0.500   0
        0.562   210
        0.625   420
        0.688   630
        0.750   840
        >0.75   840
        -------------------

  This gives you
  0
```

This indicates what data files are copied over. If you wonder if your data files are copied properly, look for this.

** Any file with prefix P1_ will be copied over when grading P1.*

Caution! this is also counted as a submission.

(**P2**) Given ***Task 2*** (below), prepare a predictive model, i.e., a function to take the input and return the output. The function has interface

`P2_predict(x)`

where `x` is a `numpy.array` of shape `(N, D)` and it returns the predicted binary output in a `numpy.array` of shape `(N,1)` when `N` is a number of data points and `D` is a number of input dimensions.

Note: if input pre-processing and/or output post-processing are needed, they can be done within this function. Also, if the trained weights or other saved parameters are needed, they can be loaded within this function. (See submission example in P1, if you need.) Recall that data for P2 should have prefix `P2_`.

Task 2: build a model to predict a sign of DR (an undisclosed medical complication).

This dataset contains features extracted to predict whether a test shows signs of DR or not.

Attribute Information:
x_0: the binary result of quality assessment.
x_1: the binary result of pre-screening.
x_2-x_7: the results of a related-sign detection. Each feature value stand for the number of signs found at the confidence levels alpha = 0.5, . . . , 1, respectively.
x_8-x_{15}: result of the same information as x_2-x_7 for another related sign.
x_{16}: the euclidean distance between two key indicators.
x_{17}: another indicator.
x_{18}: the binary result of a related classification.
x_{19}: Class label. 1 = contains signs of DR and 0 = no signs of DR.

Build and prepare (including train) the model using the provided dataset `P2data.npy`.

Evaluable

```
import numpy as np
from P2 import P2_predict

if __name__ == '__main__':
    dat = np.load('P2data.npy', allow_pickle=True)
    X = dat[:,:-1].astype(float)
    Y = dat[:, -1].reshape((-1,1))

    pbin = P2_predict(X)

    acc = np.mean(pbin == Y)
    print('Accuracy = ', acc)

    # Compute F-score
    eps = 1e-12
    TP = np.sum( (pbin == 1)*(Y == 1) , axis=0).item()
    FP = np.sum( (pbin == 1)*(Y == 0) , axis=0 ).item()
    FN = np.sum( (pbin == 0)*(Y == 1) , axis=0 ).item()
    P = TP / (TP + FP + eps)
    R = TP / (TP + FN + eps)
    F = 2*P*R/(P +R + eps)

    print('\nTest F-score, F = {:.4f}'.format(F))
```

==

Note: P2 is graded based on F-score on a test dataset.