Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Exe29 - Neural Network Exercise 1

Name: Chuay Xiang Ze

IC Number: 021224070255

Date : 1/8/2023

Introduction : Learning how to apply neural network model to real dataset

Conclusion : Managed to complete tasks related to the topic.

Neural Network Introduction

This exercise is adapted from https://www.kdnuggets.com/2016/10/beginners-guide-neural-networks-python-scikit-learn.html (https://www.kdnuggets.com/2016/10/beginners-guide-neural-networks-python-scikit-learn.html)

We'll use SciKit Learn's built in Breast Cancer Data Set which has several features of tumors with a labeled class indicating whether the tumor was Malignant or Benign. We will try to create a neural network model that can take in these features and attempt to predict malignant or benign labels for tumors it has not seen before. Let's go ahead and start by getting the data!

In [1]:

```
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
data
```

Out[1]:

```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-0
1,
         1.189e-01],
        [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
        [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
        ...,
        [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
        [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
        [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 1, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
        1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
        0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
        1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
        1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
        0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
        1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
        1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
        1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
        1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
        1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
 'frame': None,
 'target_names': array(['malignant', 'benign'], dtype='<U9'),
 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnost
ic) dataset\n--------------------------------------------\n\n**Data Set Ch
aracteristics:**\n\n    :Number of Instances: 569\n\n    :Number of Attrib
utes: 30 numeric, predictive attributes and the class\n\n    :Attribute In
formation:\n        - radius (mean of distances from center to points on t
he perimeter)\n        - texture (standard deviation of gray-scale values)
\n        - perimeter\n        - area\n        - smoothness (local variati
on in radius lengths)\n        - compactness (perimeter^2 / area - 1.0)\n
- concavity (severity of concave portions of the contour)\n        - conca
ve points (number of concave portions of the contour)\n        - symmetry
\n        - fractal dimension ("coastline approximation" - 1)\n\n        T
he mean, standard error, and "worst" or largest (mean of the three\n
worst/largest values) of these features were computed for each image,\n
resulting in 30 features.  For instance, field 0 is Mean Radius, field\n
10 is Radius SE, field 20 is Worst Radius.\n\n        - class:\n
- WDBC-Malignant\n                - WDBC-Benign\n\n    :Summary Statistic
s:\n\n    ===================================== ====== ======\n
Min    Max\n    ===================================== ====== ======\n    r
```

```
adius (mean):                          6.981  28.11\n    texture (mean):
9.71   39.28\n    perimeter (mean):                     43.79  188.5\n
area (mean):                           143.5  2501.0\n    smoothness (mea
n):                   0.053  0.163\n    compactness (mean):
0.019  0.345\n    concavity (mean):                     0.0    0.427\n
concave points (mean):                 0.0    0.201\n    symmetry (mean):
0.106  0.304\n    fractal dimension (mean):             0.05   0.097\n
radius (standard error):               0.112  2.873\n    texture (standard
error):             0.36   4.885\n    perimeter (standard error):
0.757  21.98\n    area (standard error):                6.802  542.2\n
smoothness (standard error):           0.002  0.031\n    compactness (stand
ard error):         0.002  0.135\n    concavity (standard error):
0.0    0.396\n    concave points (standard error):      0.0    0.053\n
symmetry (standard error):             0.008  0.079\n    fractal dimension
(standard error):   0.001  0.03\n    radius (worst):
7.93   36.04\n    texture (worst):                      12.02  49.54\n
perimeter (worst):                     50.41  251.2\n    area (worst):
185.2  4254.0\n    smoothness (worst):                   0.071  0.223\n
compactness (worst):                   0.027  1.058\n    concavity (worst):
0.0    1.252\n    concave points (worst):               0.0    0.291\n
symmetry (worst):                      0.156  0.664\n    fractal dimension
(worst):            0.055  0.208\n    ====================================
====== ======\n\n    :Missing Attribute Values: None\n\n    :Class Distr
ibution: 212 - Malignant, 357 - Benign\n\n    :Creator:  Dr. William H. Wo
lberg, W. Nick Street, Olvi L. Mangasarian\n\n    :Donor: Nick Street\n\n
:Date: November, 1995\n\nThis is a copy of UCI ML Breast Cancer Wisconsin
(Diagnostic) datasets.\nhttps://goo.gl/U2Uwz2\n\nFeatures are computed fro
m a digitized image of a fine needle\naspirate (FNA) of a breast mass.  Th
ey describe\ncharacteristics of the cell nuclei present in the image.\n\nS
eparating plane described above was obtained using\nMultisurface Method-Tr
ee (MSM-T) [K. P. Bennett, "Decision Tree\nConstruction Via Linear Program
ming." Proceedings of the 4th\nMidwest Artificial Intelligence and Cogniti
ve Science Society,\npp. 97-101, 1992], a classification method which uses
linear\nprogramming to construct a decision tree.  Relevant features\nwere
selected using an exhaustive search in the space of 1-4\nfeatures and 1-3
separating planes.\n\nThe actual linear program used to obtain the separat
ing plane\nin the 3-dimensional space is that described in:\n[K. P. Bennet
t and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Two
Linearly Inseparable Sets",\nOptimization Methods and Software 1, 1992, 23
-34].\n\nThis database is also available through the UW CS ftp server:\n\n
ftp ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. to
pic:: References\n\n   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. N
uclear feature extraction \n     for breast tumor diagnosis. IS&T/SPIE 199
3 International Symposium on \n     Electronic Imaging: Science and Techno
logy, volume 1905, pages 861-870,\n     San Jose, CA, 1993.\n   - O.L. Man
gasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and \n
prognosis via linear programming. Operations Research, 43(4), pages 570-57
7, \n     July-August 1995.\n   - W.H. Wolberg, W.N. Street, and O.L. Mang
asarian. Machine learning techniques\n     to diagnose breast cancer from
fine-needle aspirates. Cancer Letters 77 (1994) \n     163-171.',
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter',
        'mean area',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
```

This object is like a dictionary, it contains a description of the data and the features and targets:

In [6]:

```
# find out the attributes in the dataset
data.keys()
```

Out[6]:

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

In [13]:

```
# find out the total instances and number of features
data.data.shape
```

Out[13]:

```
(569, 30)
```

Set up the data (X) and labels (y)

In [32]:

```
X = data.data
y = data.target
```

Train Test Split

Let's split our data into training and testing sets, this is done easily with SciKit Learn's train_test_split function from model_selection:

In [49]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

Data Preprocessing

The neural network may have difficulty converging before the maximum number of iterations allowed if the data is not normalized. Multi-layer Perceptron is sensitive to feature scaling, so it is highly recommended to scale your data. Note that you must apply the same scaling to the test set for meaningful results. There are a lot of different methods for normalization of data, we will use the built-in StandardScaler for standardization.

```
 'filename': 'breast_cancer.csv',
 'data_module': 'sklearn.datasets.data'}
```

In [50]:

```
# Import the StandardScalar library
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# Fit only to the training data
scaler.fit(X_train)
```

Out[50]:

```
▾ StandardScaler

StandardScaler()
```

In [51]:

```
# Now apply the transformations to the data:
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Training the model

Now it is time to train our model. SciKit Learn makes this incredibly easy, by using estimator objects. In this case we will import our estimator (the Multi-Layer Perceptron Classifier model) from the neural_network library of SciKit-Learn!

In [52]:

```
from sklearn.neural_network import MLPClassifier
```

Next we create an instance of the model, there are a lot of parameters you can choose to define and customize here, we will only define the hidden_layer_sizes. For this parameter you pass in a tuple consisting of the number of neurons you want at each layer, where the nth entry in the tuple represents the number of neurons in the nth layer of the MLP model. There are many ways to choose these numbers, but for simplicity we will choose 3 layers with the same number of neurons as there are features in our data set:

In [53]:

```
# create a Multilayerperceptron classifier and call it mlp
mlp = MLPClassifier(hidden_layer_sizes=(30,30,30))
```

Now that the model has been made we can fit the training data to our model, remember that this data has already been processed and scaled:

In [54]:

```
mlp.fit(X_train,y_train)
```

Out[54]:

```
MLPClassifier
MLPClassifier(hidden_layer_sizes=(30, 30, 30))
```

Q: What do you see in the output? What does it tell you?

Predictions and Evaluation

Now that we have a model it is time to use it to get predictions! We can do this simply with the predict() method off of our fitted model:

In [55]:

```
predictions = mlp.predict(X_test)
```

Now we can use SciKit-Learn's built in metrics such as a classification report and confusion matrix to evaluate how well our model performed:

In [57]:

```
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
```

```
[[50  3]
 [ 3 87]]
              precision    recall  f1-score   support

           0       0.94      0.94      0.94        53
           1       0.97      0.97      0.97        90

    accuracy                           0.96       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143
```

Q: what conclusion can you make from the confusion matrix?

Weights and biases

The downside however to using a Multi-Layer Preceptron model is how difficult it is to interpret the model itself. The weights and biases won't be easily interpretable in relation to which features are important to the model itself.

To extract the MLP weights and biases after training your model, you use its public attributes coefs_ and intercepts_.

In [59]:

```
len(mlp.coefs_[0])
```

Out[59]:

```
30
```

In [60]:

```
# Print the intercepts values and interpret it
len(mlp.intercepts_[0])
```

Out[60]:

```
30
```

Q: What do you understand from the two values?