MLA Lab 6 Artificial Neural Network for Machine Learning

43308

Ayan Gadpal

Aim: Creating & Visualizing Neural Network for the given data. (Using Python)(Use Cancer Dataset)
Simply put, an ANN represents interconnected input and output units in which each connection has an associated weight. During the learning phase, the network learns by adjusting these weights in order to be able to predict the correct class for input data.

```
import torch
import numpy as np
import pandas as pd
from torch.autograd import Variable
import torch.nn.functional as F
from sklearn.preprocessing import LabelBinarizer, LabelEncoder
import os
```

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. n the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992,

Saving…

TASK Correctly classify whether the tumor is benign or malignant.

```
data = pd.read_csv('data.csv')
data.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean
0	842302	M	17.99	10.38	122.80	1001.0
1	842517	M	20.57	17.77	132.90	1326.0
2	84300903	M	19.69	21.25	130.00	1203.0
3	84348301	M	11.42	20.38	77.58	386.1
4	84358402	M	20.29	14.34	135.10	1297.0

```
cols = data.columns
x_data = data[cols[2:-1]]
print(len(cols))
x_data.head()
x_data.head()

```

s_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_
17.99	10.38	122.80	1001.0	0.11840	0.2
20.57	17.77	132.90	1326.0	0.08474	0.0
19.69	21.25	130.00	1203.0	0.10960	0.1
11.42	20.38	77.58	386.1	0.14250	0.2
20.29	14.34	135.10	1297.0	0.10030	0.1

```
y_data = data[cols[1]]
le = LabelEncoder()
y_data = np.array(le.fit_transform(y_data))
y_data[:5], le.classes_
```

```
(array([1, 1, 1, 1, 1]), array(['B', 'M'], dtype=object))
```

```
y_data.shape, x_data.values.shape
```

$$((569), \quad (569, \quad 30))$$

```
x_data = Variable(torch.from_numpy(x_data.values))
                                    numpy(y_data))
```

Saving...

```
class Model(torch.nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.l1 = torch.nn.Linear(30,16)
        self.l2 = torch.nn.Linear(16,4)
        self.l3 = torch.nn.Linear(4,1)

        self.sigmoid = torch.nn.Sigmoid()

    def forward(self, x):
        out1 = self.sigmoid(self.l1(x))
        out2 = self.sigmoid(self.l2(out1))
        y_pred = self.sigmoid(self.l3(out2))
        return y_pred

model = Model()
```

```
x_data.float().size(), y_data.float().size()
```

$$((569, \quad 30), \quad (569))$$