S.SRIHARI - 2018103601:CSE BATCH-P

PREPARATORY QUESTIONS

Week 3 PERCEPTRON

6-3-2021

Desktop/SEM-6/Machine_Learnir × | pima-indians-diabetes.csv - Jupy × | PIMA-INDIAN-DIABETES-PERCEP × | AND-GATE-PERCEPTRON - Jupyt × | OR-GATE-PERCEPTRON - Jupyter ×

localhost:8888/notebooks/Desktop/SEM-6/Machine_Learning/lab/lab-week3/AND-GATE-PERCEPTRON.ipynb

jupyter AND-GATE-PERCEPTRON Last Checkpoint: 25 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help   Trusted   Python 3

Run   Code

In [25]:

```
import numpy as np

class pcn:
    def __init__(self,inputs,targets):
        if np.ndim(inputs)>1:
            self.nIn = np.shape(inputs)[1]
            print("nIn = ",self.nIn)
        else:
            self.nIn = 1
    
        if np.ndim(targets)>1:
            self.nOut = np.shape(targets)[1]
            print("nOut = ",self.nOut)
        else:
            self.nOut = 1
        self.nData = np.shape(inputs)[0]

        # Initialise network
        # nIn is the number of input nodes
        # np.random.rand(a,b) return an a x b matrix
        self.weights = np.random.rand(self.nIn+1,self.nOut)*0.1-0.05
        print("Weights",self.weights) # of dimension 3 x 1

    def pcntrain(self,inputs,targets,eta,nIterations):
        # Add the inputs that match the bias node
        #print("Inputs sent to pcntrain\n",inputs)
        inputs = np.concatenate((inputs,-np.ones((self.nData,1))),axis=1)
        #axis is used for the concatenate function thus we concat along axis = 1 i.e. columns
        #print("Inputs after concat in pcntrain\n",inputs)# Training
        
        for n in range(nIterations):
            self.activations = self.pcnfwd(inputs);
            self.weights -= eta*np.dot(np.transpose(inputs),self.activations-targets)

        
```

Desktop/SEM-6/Machine_Learnin × | pima-indians-diabetes.csv - Jupy × | PIMA-INDIAN-DIABETES-PERCEP × | AND-GATE-PERCEPTRON - Jupyt × | OR-GATE-PERCEPTRON - Jupyter ×

localhost:8888/notebooks/Desktop/SEM-6/Machine_Learning/lab/lab-week3/AND-GATE-PERCEPTRON.ipynb

jupyter AND-GATE-PERCEPTRON Last Checkpoint: 27 minutes ago (autosaved)

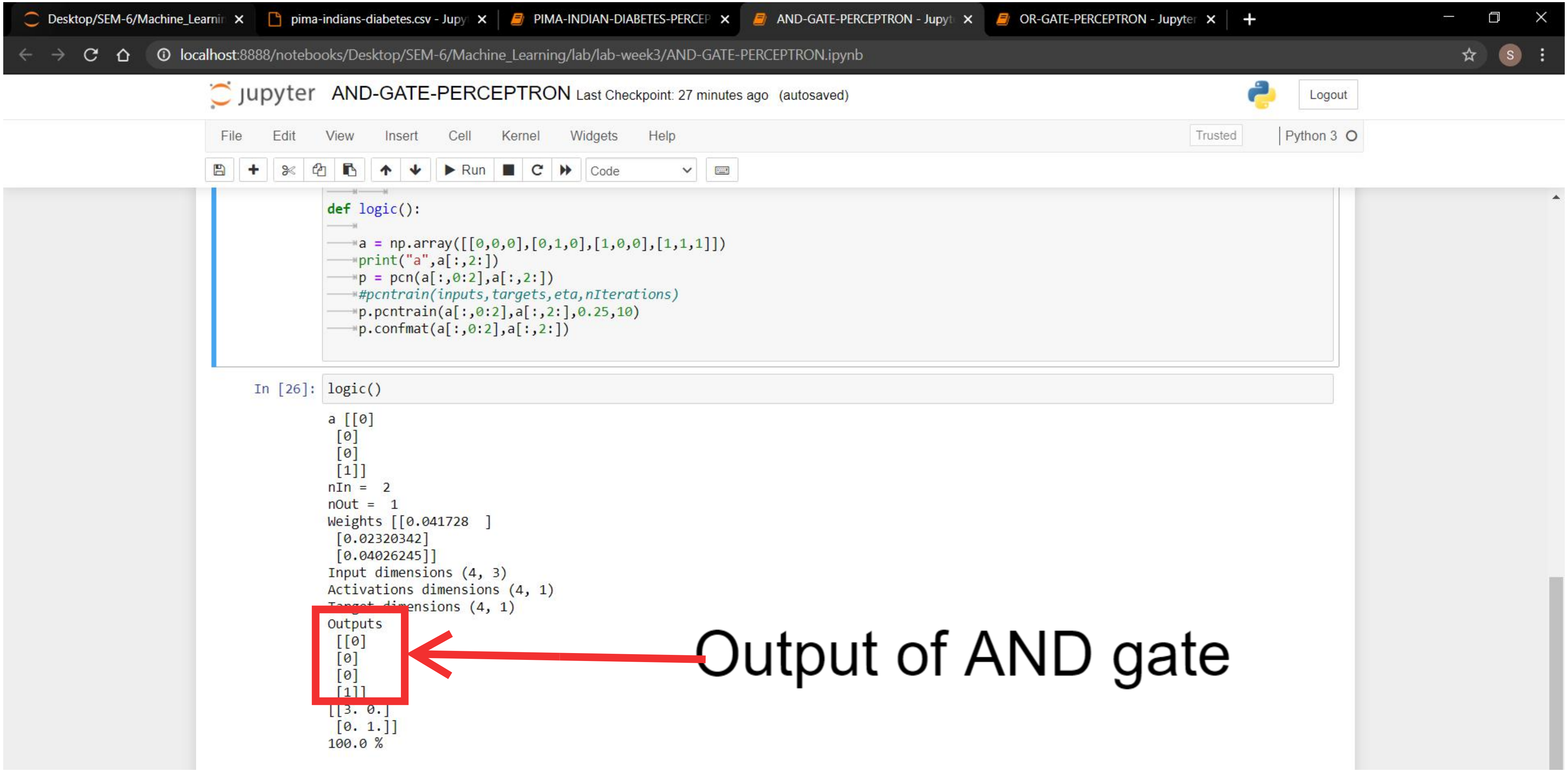
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

```
        print("Input dimensions",np.shape(inputs))
        print("Activations dimensions",np.shape(self.activations))
        print("Target dimensions",np.shape(targets))

    def pcnfwd(self,inputs):
        # Compute activations
        activations =  np.dot(inputs,self.weights)
        #print("Activations in pcnfwd",activations.shape)
        # Threshold the activations
        return np.where(activations>0,1,0)


    def confmat(self,inputs,targets):
        # Add the inputs that match the bias node
        inputs = np.concatenate((inputs,-np.ones((self.nData,1))),axis=1)
        #print("Inputs sent to confmat\n",inputs)
        outputs = np.dot(inputs,self.weights)

        nClasses = np.shape(targets)[1] + 1
        #print("No of classes",nClasses)


        outputs = np.where(outputs>0,1,0)

        print("Outputs\n",outputs)

        cm = np.zeros((nClasses,nClasses))
        for i in range(nClasses):
            for j in range(nClasses):
                cm[i,j] = np.sum(np.where(outputs==i,1,0)*np.where(targets==j,1,0)) #At diag i=j, hence if op=ip diag val inc.

        print(cm)
        print(np.trace(cm)/np.sum(cm)*100,"%")#  this is the accuracy
```


Desktop/SEM-6/Machine_Learnir
pima-indians-diabetes.csv - Jupy
PIMA-INDIAN-DIABETES-PERCEP
AND-GATE-PERCEPTRON - Jupyt
OR-GATE-PERCEPTRON - Jupyter
localhost:8888/notebooks/Desktop/SEM-6/Machine_Learning/lab/lab-week3/AND-GATE-PERCEPTRON.ipynb
jupyter AND-GATE-PERCEPTRON Last Checkpoint: 27 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
def logic():
a = np.array([[0,0,0],[0,1,0],[1,0,0],[1,1,1]])
print("a",a[:,2:])
p = pcn(a[:,0:2],a[:,2:])
#pcntrain(inputs,targets,eta,nIterations)
p.pcntrain(a[:,0:2],a[:,2:],0.25,10)
p.confmat(a[:,0:2],a[:,2:])
In [26]: logic()
a [[0]
[0]
[0]
[1]]
nIn = 2
nOut = 1
Weights [[0.041728]
[0.02320342]
[0.04026245]]
Input dimensions (4, 3)
Activations dimensions (4, 1)
Target dimensions (4, 1)
Outputs
[[0]
[0]
[0]
[1]]
[[3. 0.]
[0. 1.]]
100.0 %
Output of AND gate

ONSPOT QUESTIONS

Week 3 PERCEPTRON

6-3-2021

PIMA DATASET

~/week3/pima-indians-diabetes.csv - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

perceptron.py | pima_perceptron.py | and_gate.py | pima-indians-diabetes.csv

```
A,B,C,D,E,F,G,H,I
6,148,72,35,0,33.6,0.627,50,1
1,85,66,29,0,26.6,0.351,31,0
8,183,64,0,0,23.3,0.672,32,1
1,89,66,23,94,28.1,0.167,21,0
0,137,40,35,168,43.1,2.288,33,1
5,116,74,0,0,25.6,0.201,30,0
3,78,50,32,88,31,0.248,26,1
10,115,0,0,0,35.3,0.134,29,0
2,197,70,45,543,30.5,0.158,53,1
8,125,96,0,0,0,0.232,54,1
4,110,92,0,0,37.6,0.191,30,0
10,168,74,0,0,38,0.537,34,1
10,139,80,0,0,27.1,1.441,57,0
1,189,60,23,846,30.1,0.398,59,1
5,166,72,19,175,25.8,0.587,51,1
7,100,0,0,0,30,0.484,32,1
0,118,84,47,230,45.8,0.551,31,1
7,107,74,0,0,29.6,0.254,31,1
1,103,30,38,83,43.3,0.183,33,0
1,115,70,30,96,34.6,0.529,32,1
3,126,88,41,235,39.3,0.704,27,0
8,99,84,0,0,35.4,0.388,50,0
7,196,90,0,0,39.8,0.451,41,1
9,119,80,35,0,29,0.263,29,1
11,143,94,33,146,36.6,0.254,51,1
10,125,70,26,115,31.1,0.205,41,1
7,147,76,0,0,39.4,0.257,43,1
1,97,66,15,140,23.2,0.487,22,0
13,145,82,19,110,22.2,0.245,57,0
5,117,92,0,0,34.1,0.337,38,0
5,109,75,26,0,36,0.546,60,0
3,158,76,36,245,31.6,0.851,28,1
3,88,58,11,54,24.8,0.267,22,0
6,92,92,0,0,19.9,0.188,28,0
10,122,78,31,0,27.6,0.512,45,0
4,103,60,33,192,24,0.966,33,0
11,138,76,0,0,33.2,0.42,35,0
9,102,76,37,0,32.9,0.665,46,1
2,90,68,42,0,38.2,0.503,27,1
4,111,72,47,207,37.1,1.39,56,1
3,180,64,25,70,34,0.271,26,0
```

Line 1, Column 1

Implementation of Perceptron

Desktop/SEM-6/Machine_Learnir × | pima-indians-diabetes.csv - Jupy × | PIMA-INDIAN-DIABETES-PERCEP × | AND-GATE-PERCEPTRON - Jupyt × | OR-GATE-PERCEPTRON - Jupyter ×

localhost:8888/notebooks/Desktop/SEM-6/Machine_Learning/lab/lab-week3/PIMA-INDIAN-DIABETES-PERCEPTRON.ipynb

jupyter PIMA-INDIAN-DIABETES-PERCEPTRON Last Checkpoint: an hour ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [30]:

```
import numpy as np

class pcn:
    def __init__(self,inputs,targets):
        if np.ndim(inputs)>1:
            self.nIn = np.shape(inputs)[1]
            #print("nIn = ",self.nIn)
        else:
            self.nIn = 1

        if np.ndim(targets)>1:
            self.nOut = np.shape(targets)[1]
            #print("nOut = ",self.nOut)
        else:
            self.nOut = 1
        self.nData = np.shape(inputs)[0]

        # Initialise network
        # nIn is the number of input nodes
        # np.random.rand(a,b) return an a x b matrix
        self.weights = np.random.rand(self.nIn+1,self.nOut)*0.1-0.05
        #print("Weights",self.weights) # of dimension 3 x 1

    def pcntrain(self,inputs,targets,eta,nIterations):
        # Add the inputs that match the bias node
        #print("Inputs sent to pcntrain\n",inputs)
        inputs = np.concatenate((inputs,-np.ones((self.nData,1))),axis=1) #axis is used for the concatenate function thus we conc
        #print("Inputs after concat in pcntrain\n",inputs)# Training

        for n in range(nIterations):

            self.activations = self.pcnfwd(inputs);
            self.weights -= eta*np.dot(np.transpose(inputs),self.activations-targets)
```

Desktop/SEM-6/Machine_Learnin × | pima-indians-diabetes.csv - Jupyt × | PIMA-INDIAN-DIABETES-PERCEP × | AND-GATE-PERCEPTRON - Jupyt × | OR-GATE-PERCEPTRON - Jupyter ×

localhost:8888/notebooks/Desktop/SEM-6/Machine_Learning/lab/lab-week3/PIMA-INDIAN-DIABETES-PERCEPTRON.ipynb

jupyter PIMA-INDIAN-DIABETES-PERCEPTRON Last Checkpoint: an hour ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

```python
    def pcnfwd(self,inputs):
        # Compute activations
        activations =  np.dot(inputs,self.weights)
        #print("Activations in pcnfwd",activations.shape)

        # Threshold the activations
        return np.where(activations>0,1,0)


    def confmat(self,inputs,targets):

        # Add the inputs that match the bias node
        inputs = np.concatenate((inputs,-np.ones((self.nData,1))),axis=1)
        #print("Inputs sent to confmat\n",inputs)
        outputs = np.dot(inputs,self.weights)

        nClasses = np.shape(targets)[1] + 1
        #print("No of classes",nClasses)


        outputs = np.where(outputs>0,1,0)

        #print("Outputs\n",outputs)

        cm = np.zeros((nClasses,nClasses))
        for i in range(nClasses):
            for j in range(nClasses):
                cm[i,j] = np.sum(np.where(outputs==i,1,0)*np.where(targets==j,1,0))

        print(cm)
        print("Accuracy is ",np.trace(cm)/np.sum(cm)*100,"%")
```

localhost:8888/notebooks/Desktop/SEM-6/Machine_Learning/lab/lab-week3/PIMA-INDIAN-DIABETES-PERCEPTRON.ipynb

jupyter PIMA-INDIAN-DIABETES-PERCEPTRON Last Checkpoint: an hour ago (autosaved)

Loading Dataset

In [31]:

```
import pylab as pl
import numpy as np
import pandas as pd


pima = pd.read_csv('pima-indians-diabetes.csv',sep=",")
pima.head()
np.shape(pima)
pima
```

Out[31]:

	A	B	C	D	E	F	G	H	I
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

Desktop/SEM-6/Machine_Learnir × | pima-indians-diabetes.csv - Jupy × | PIMA-INDIAN-DIABETES-PERCEP × | AND-GATE-PERCEPTRON - Jupyt × | OR-GATE-PERCEPTRON - Jupyter ×

localhost:8888/notebooks/Desktop/SEM-6/Machine_Learning/lab/lab-week3/PIMA-INDIAN-DIABETES-PERCEPTRON.ipynb

jupyter PIMA-INDIAN-DIABETES-PERCEPTRON Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [32]:

```
import pylab as pl
import numpy as np
import pandas as pd

pima = pd.read_csv('pima-indians-diabetes.csv',sep=",")
print(type(pima))
pima = pima.values
print(type(pima))
# Plot the first and second values for the two classes
indices0 = np.where(pima[:,8]==0)
indices1 = np.where(pima[:,8]==1)

pl.plot(pima[indices0,0],pima[indices0,1],'go')
pl.plot(pima[indices1,0],pima[indices1,1],'rx')
pl.xlabel("Feature-1")
pl.ylabel("Feature-2")

# Perceptron training on the original dataset
print("Output on original data")
p = pcn(pima[:,:8],pima[:,8:9])
p.pcntrain(pima[:,:8],pima[:,8:9],0.25,100)
p.confmat(pima[:,:8],pima[:,8:9])
print("----------------------------------------------------------------------")
print("mean",pima.mean(axis=0))
print("var",pima.var(axis=0))
print("max",pima.max(axis=0))
print("min",pima.min(axis=0))
print("----------------------------------------------------------------------")
# Various preprocessing steps
pima[np.where(pima[:,0]>8),0] = 8

pima[np.where(pima[:,7]<=30),7] = 1
pima[np.where((pima[:,7]>30) & (pima[:,7]<=40)),7] = 2
pima[np.where((pima[:,7]>40) & (pima[:,7]<=50)),7] = 3
pima[np.where((pima[:,7]>50) & (pima[:,7]<=60)),7] = 4
pima[np.where(pima[:,7]>60),7] = 5
```

Desktop/SEM-6/Machine_Learnin | pima-indians-diabetes.csv - Jupy | PIMA-INDIAN-DIABETES-PERCEP | AND-GATE-PERCEPTRON - Jupyt | OR-GATE-PERCEPTRON - Jupyter

localhost:8888/notebooks/Desktop/SEM-6/Machine_Learning/lab/lab-week3/PIMA-INDIAN-DIABETES-PERCEPTRON.ipynb

jupyter PIMA-INDIAN-DIABETES-PERCEPTRON Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
pima[:,:8] = pima[:,:8]-pima[:,:8].mean(axis=0)
pima[:,:8] = pima[:,:8]/pima[:,:8].var(axis=0)

print("mean",pima.mean(axis=0))
print("var",pima.var(axis=0))
print("max",pima.max(axis=0))
print("min",pima.min(axis=0))
print("----------------------------------------------------------------------")
trainin = pima[::2,:8]
testin = pima[1::2,:8]
traintgt = pima[::2,8:9]
testtgt = pima[1::2,8:9]

# Perceptron training on the preprocessed dataset
print("Output after preprocessing of data")
p1 = pcn(trainin,traintgt)
p1.pcntrain(trainin,traintgt,0.25,100)
p1.confmat(testin,testtgt)



pl.show()
```

Desktop/SEM-6/Machine_Le… | pima-indians-diabetes.csv - J… | PIMA-INDIAN-DIABETES-PE… | AND-GATE-PERCEPTRON - J… | OR-GATE-PERCEPTRON - Ju… | How to add axis labels to a p…

localhost:8888/notebooks/Desktop/SEM-6/Machine_Learning/lab/lab-week3/PIMA-INDIAN-DIABETES-PERCEPTRON.ipynb

jupyter PIMA-INDIAN-DIABETES-PERCEPTRON Last Checkpoint: an hour ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

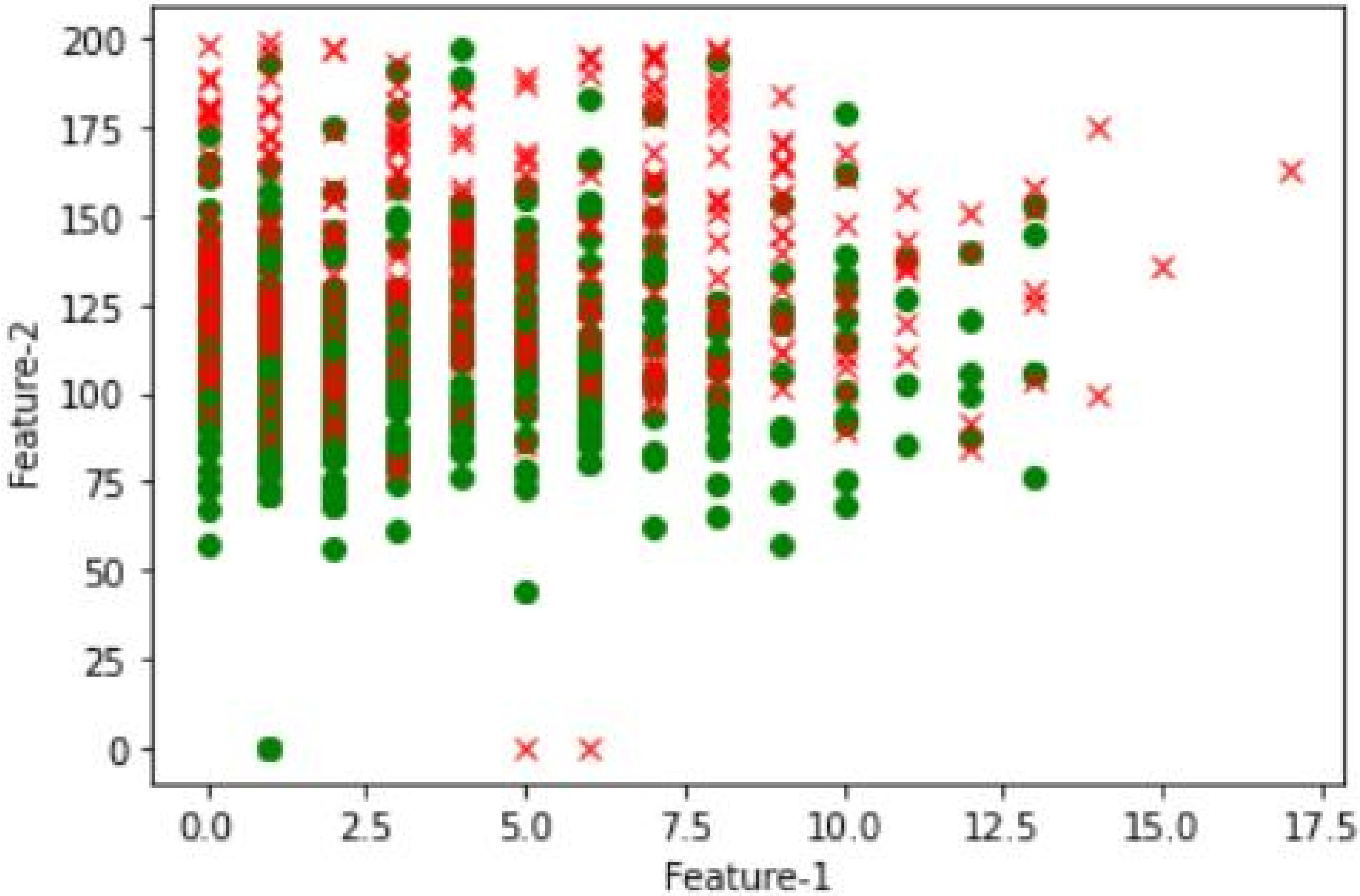
Run Code

```
pl.show()
```

Output

```
<class 'pandas.core.frame.DataFrame'>
<class 'numpy.ndarray'>
Output on original data
[[ 39.   6.]
 [461. 262.]]
Accuracy is  39.19270833333333 %
----------------------------------------------------------------------
mean [  3.84505208 120.89453125  69.10546875  20.53645833  79.79947917
  31.99257812   0.4718763   33.24088542   0.34895833]
var [1.13392724e+01 1.02091726e+03 3.74159449e+02 2.54141900e+02
 1.32638869e+04 6.20790465e+01 1.09635697e-01 1.38122964e+02
 2.27186415e-01]
max [ 17.    199.    122.     99.    846.     67.1     2.42   81.      1.  ]
min [ 0.     0.     0.     0.     0.     0.     0.078 21.     0.   ]
----------------------------------------------------------------------
mean [-2.31296463e-18  0.00000000e+00 -7.22801448e-19  3.03576608e-18
 -3.56883215e-19  3.29597460e-17  7.03141249e-16 -8.21102445e-17
  3.48958333e-01]
var [1.28606527e-01 9.79511306e-04 2.67265734e-03 3.93480965e-03
 7.53926816e-05 1.61084948e-02 9.12111683e+00 7.90302374e-01
 2.27186415e-01]
max [ 0.5716962   0.07650519  0.14136896  0.3087391   0.05776591  0.56552772
 17.76906384  2.48924667  1.        ]
min [-0.45715601 -0.11841756 -0.18469524 -0.08080705 -0.0060163  -0.51535228
 -3.59259177 -0.67196283  0.        ]
----------------------------------------------------------------------
Output after preprocessing of data
[[216.  85.]
 [ 35.  48.]]
Accuracy is  68.75 %
```

Plotting 2 features at a time



Viva Questions – Machine Learning Lab

Week 3 PERCEPTRON

6-3-2021

1. How did you take training and testing datasets for consideration?

- Answer:
  - It would be unfair to both train and test on the same set of data and then check the accuracy. So I have taken all the even numbered records in the csv file dataset as training dataset. All the odd numbered records in the csv file is taken for the testing dataset.
  - Thus, I have split the PIMA dataset as 50:50 for training and testing.
  - These lines of code achieve this purpose.

```
trainin = pima[::2,:8]
testin = pima[1::2,:8]
traintgt = pima[::2,8:9]
testtgt = pima[1::2,8:9]
```

trainin = pima[::2,:8] – This extracts all records starting from 0 till the end incrementing in steps of 2 for which all the columns indexed from 0 to 7 are extracted. This forms the training input.

traintgt = pima[::2,8:9] - This extracts all records starting from 0 till the end incrementing in steps of 2 for which all the columns indexed 8 are extracted. This forms the training target.

Similarly all odd numbered records are taken as the test-input and test-target.

```
trainin = pima[::2,:8]
testin = pima[1::2,:8]
traintgt = pima[::2,8:9]
testtgt = pima[1::2,8:9]
```

2. Explanation of step functions using code:

- The following code is used for the weight update. The weight update stops when the outputs are the same as the targets. i.e the error becomes 0. But in many practical scenarios it takes a large amount of time to achieve zero error. In some cases that might not even be achieved. Hence we fix the no. of Iterations here in the variable nIterations.

```
def pcntrain(self,inputs,targets,eta,nIterations):
    # Add the inputs that match the bias node
    #print("Inputs sent to pcntrain\n",inputs)
    inputs = np.concatenate((inputs,-np.ones((self.nData,1))),axis=1)
    #axis is used for the concatenate function thus we concat along axis = 1 i.e. columns
    #print("Inputs after concat in pcntrain\n",inputs)# Training

    for n in range(nIterations):

        self.activations = self.pcnfwd(inputs);
        self.weights -= eta*np.dot(np.transpose(inputs),self.activations-targets)

```

```python
def pcnfwd(self,inputs):
    # Compute activations
    activations =  np.dot(inputs,self.weights)
    #print("Activations in pcnfwd",activations.shape)

    # Threshold the activations
    return np.where(activations>0,1,0)
```

- We concatenate -1 to the input to account for the bias node. This concatenation is done along the columns hence we specify axis=1.
- For nIterations times we perform the weight update rule:
- Here we propagate forward computing the activations which is the dot product of inputs(Dimension: 4 X 3) and weights vectors (Dimension: 3 X 1). Thus activations vector has dimension: 4 X 1.
- At the end of this forward propagation the entries of activations vector is updated to 1, if its existing value is more than 0; Else it is updated as 0.

- Here eta = learning rate = 0.25 and dimensions as specified below:

Matrix	Dimension
inputs	4 X 3
activations	4 X 1
target	4 X 1

- To make the matrices compatible for multiplication we take the transpose. Now the dot-product of input and (activations-targets) is taken.
- Hence weight is finally updated as mentioned here.

```
        for n in range(nIterations):

            self.activations = self.pcnfwd(inputs);
            self.weights -= eta*np.dot(np.transpose(inputs),self.activations-targets)

```