SinglePerceptronLearning

June 3, 2021

[1]:
```
                          # Single Perceptron Learning and Decision Boundary␣
 ↪Visualizaiton
                                                                              ␣
 ↪# 2021 Summer, X. Zhu, June 2 2021
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
class1 = pd.read_csv("class1.txt")
class2 = pd.read_csv("class2.txt")
print(class1.shape)
print(class2.shape)
class1.head()
```

```
(100, 2)
(100, 2)
```

[1]:

	weight	height
0	0.132	0.757
1	0.722	0.888
2	0.095	0.804
3	0.633	0.530
4	0.472	0.701

[2]:
```
# add lables to the data. .insert() will directly modify the dataframe
posLabel=1          # positive class Label
negLabel=-1         # negative class label
T=(posLabel+negLabel)/2 # This is the threshold to classify positive vs.␣
 ↪negative
class1.insert(class1.shape[1],'label',posLabel)
class1.head()
```

[2]:

	weight	height	label
0	0.132	0.757	1
1	0.722	0.888	1
2	0.095	0.804	1
3	0.633	0.530	1

```
4   0.472   0.701      1
```

[3]:
```
# add lables to the data. .insert() will directly modify the dataframe
class2.insert(class2.shape[1],'label',negLabel)
class2.head()
```

[3]:

	weight	height	label
0	0.407	0.347	-1
1	0.726	0.761	-1
2	0.644	0.415	-1
3	0.076	0.143	-1
4	0.110	0.010	-1

[4]:
```
class12 = class1.append(class2)
print(class12.shape)
class12.head()
```

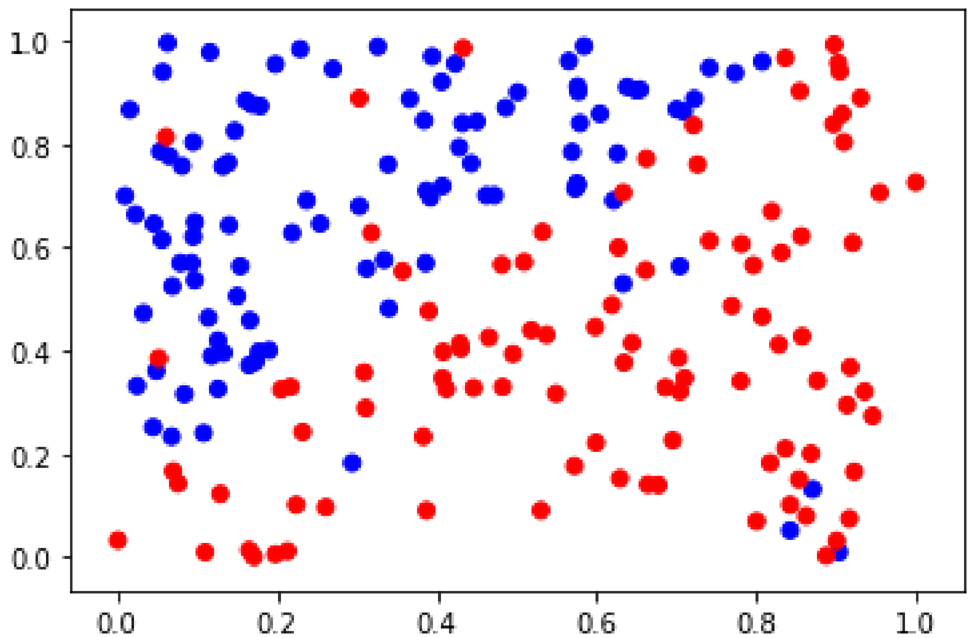
```
(200, 3)
```

[4]:

	weight	height	label
0	0.132	0.757	1
1	0.722	0.888	1
2	0.095	0.804	1
3	0.633	0.530	1
4	0.472	0.701	1

[5]:
```
colors=["red","black","blue"]
plt.scatter(class12.iloc[:,0],class12.iloc[:,1],color=[colors[idx+1] for idx in␣
 ↪class12.iloc[:,2]])
# The dataset is clearly not a linearly searable problem.
```

[5]:
```
<matplotlib.collections.PathCollection at 0x24677b3bd88>
```

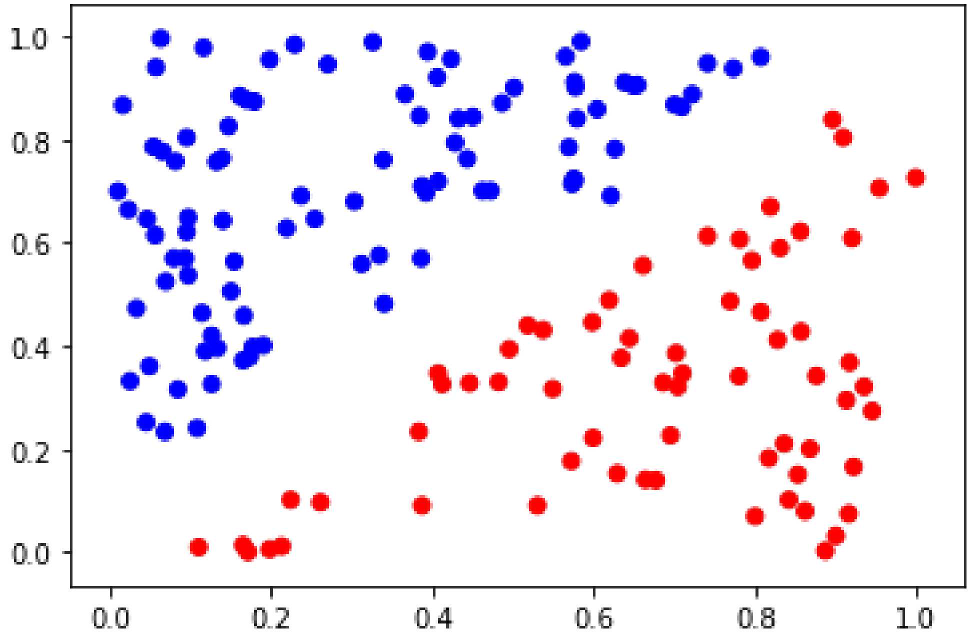


[6]:
```
# the above dataset is not linearly separably. So we are now clean the data to␣
↪make it as a linearly separable problem
index=0
clean=[]
for i, row in class12.iterrows():
    if(((row['weight']-row['height'])*(-(row['label']-T)))>=0.05):
        clean.append(index)
    index=index+1
print(clean)
```

```
[0, 1, 2, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 99, 100, 102, 104, 106, 107, 108,
109, 110, 111, 112, 115, 116, 117, 119, 120, 121, 122, 123, 125, 127, 128, 131,
132, 133, 134, 136, 137, 138, 139, 142, 143, 145, 146, 148, 149, 150, 151, 152,
153, 154, 157, 158, 159, 161, 164, 166, 167, 169, 171, 173, 176, 177, 178, 179,
180, 181, 183, 186, 187, 188, 189, 190, 191, 193, 194]
```

[7]:
```
# Now we show this new dataset, which is clearly linearly separable.
class12_clean=class12.iloc[clean,:]
plt.scatter(class12_clean.iloc[:,0],class12_clean.iloc[:
↪,1],color=[colors[idx+1] for idx in class12_clean.iloc[:,2]])
```

[7]:
```
<matplotlib.collections.PathCollection at 0x24677bff508>
```



[8]:
```
# partitioning the dataset into training vs. test sets
features,labels=class12_clean.iloc[:,0:-1],class12_clean.loc[:,['label']]
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(features, labels,␣
↪test_size=.4, random_state=42)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(95, 2)
(64, 2)
(95, 1)
(64, 1)
```

[9]:
```
# covert data from dataframe into matrix format for arithemetic calculation
X_train_m=np.asmatrix(X_train, dtype = 'float64')
X_test_m=np.asmatrix(X_test, dtype = 'float64')
y_train_m=np.asmatrix(y_train, dtype = 'float64')
y_test_m=np.asmatrix(y_test, dtype = 'float64')
X_train_m
```

[9]:
```
matrix([[0.27 , 0.946],
        [0.654, 0.906],
        [0.621, 0.691],
```

```
[0.412, 0.326],
[0.132, 0.757],
[0.856, 0.622],
[0.139, 0.764],
[0.065, 0.777],
[0.016, 0.867],
[0.312, 0.559],
[0.126, 0.326],
[0.922, 0.166],
[0.366, 0.888],
[0.069, 0.525],
[0.71 , 0.347],
[0.254, 0.646],
[0.45 , 0.844],
[0.876, 0.342],
[0.224, 0.102],
[0.213, 0.012],
[0.572, 0.177],
[0.19 , 0.401],
[0.78 , 0.341],
[0.097, 0.649],
[0.326, 0.989],
[0.573, 0.714],
[0.383, 0.234],
[0.634, 0.377],
[0.446, 0.328],
[0.056, 0.615],
[0.868, 0.201],
[0.836, 0.211],
[0.619, 0.489],
[0.079, 0.57 ],
[0.023, 0.664],
[0.661, 0.556],
[0.598, 0.446],
[0.118, 0.39 ],
[0.198, 0.955],
[0.093, 0.57 ],
[0.579, 0.841],
[0.173, 0.379],
[0.219, 0.628],
[0.054, 0.786],
[0.14 , 0.643],
[0.773, 0.938],
[0.677, 0.14 ],
[0.913, 0.295],
[0.741, 0.948],
[0.063, 0.996],
```

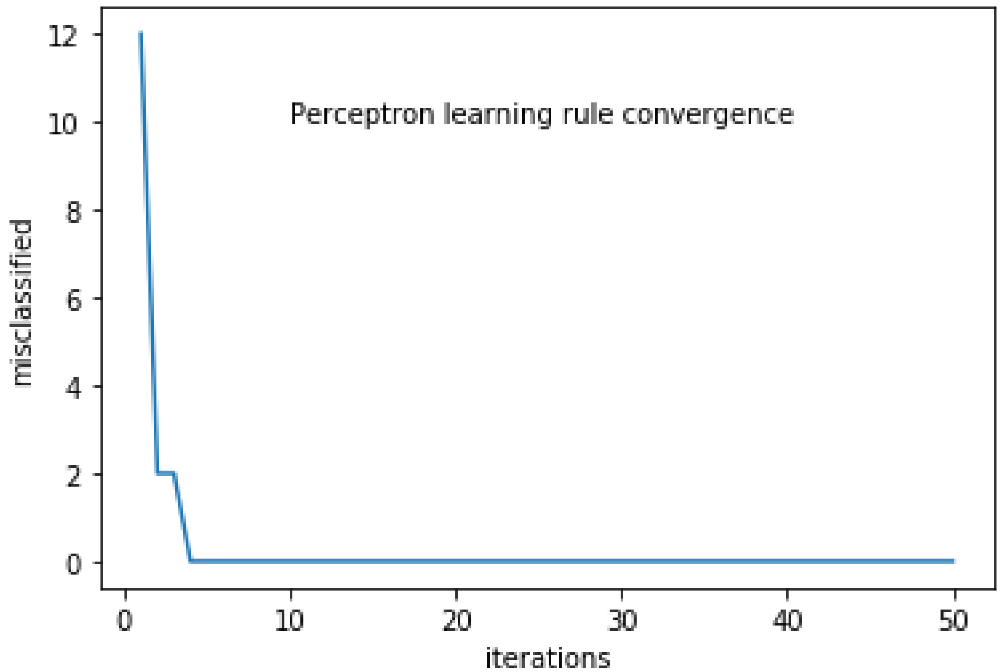
```
[0.909, 0.804],
[0.796, 0.566],
[0.34 , 0.482],
[0.394, 0.97 ],
[0.828, 0.412],
[0.584, 0.99 ],
[0.147, 0.826],
[0.842, 0.102],
[0.472, 0.701],
[0.108, 0.241],
[0.501, 0.901],
[0.576, 0.721],
[0.917, 0.368],
[0.116, 0.978],
[0.261, 0.097],
[0.15 , 0.506],
[0.057, 0.94 ],
[0.704, 0.321],
[0.575, 0.724],
[0.781, 0.607],
[0.097, 0.537],
[0.428, 0.794],
[0.565, 0.961],
[0.709, 0.863],
[0.384, 0.846],
[0.935, 0.321],
[0.817, 0.183],
[0.081, 0.758],
[0.686, 0.329],
[0.722, 0.888],
[0.638, 0.911],
[0.482, 0.33 ],
[0.741, 0.613],
[0.807, 0.466],
[0.862, 0.08 ],
[0.154, 0.564],
[0.407, 0.719],
[0.8  , 0.07 ],
[0.198, 0.006],
[0.169, 0.877],
[0.303, 0.68 ],
[0.896, 0.839],
[0.133, 0.396],
[0.339, 0.761],
[0.629, 0.153]])
```

[10]:
```
def perceptron(features, labels, num_iter, learning_rate):

    # random initialize weight values between rage: [-0.5,0.5]
    w = np.random.rand(features.shape[1]+1)-0.5

    misclassified_ = []
    for epoch in range(num_iter):
        misclassified = 0
        for i, x in enumerate(features):
            x = np.insert(x,0,1)

            v = np.dot(w, x.transpose())
            actual = posLabel if (v > T) else negLabel

            delta = (labels[i] - actual)

            if(delta): # misclassified
                misclassified += 1
                w =w+ (delta*x*learning_rate)

        misclassified_.append(misclassified)
    return (w, misclassified_)
```

[14]:
```
num_iter = 50
eta=0.05
w, misclassified= perceptron(X_train_m, y_train_m, num_iter, eta)
print(misclassified)
```

```
[12, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

[15]:
```
epochs = np.arange(1, num_iter+1)
plt.plot(epochs, misclassified)
plt.xlabel('iterations')
plt.ylabel('misclassified')
plt.text(10,10,"Perceptron learning rule convergence")
plt.show()
```

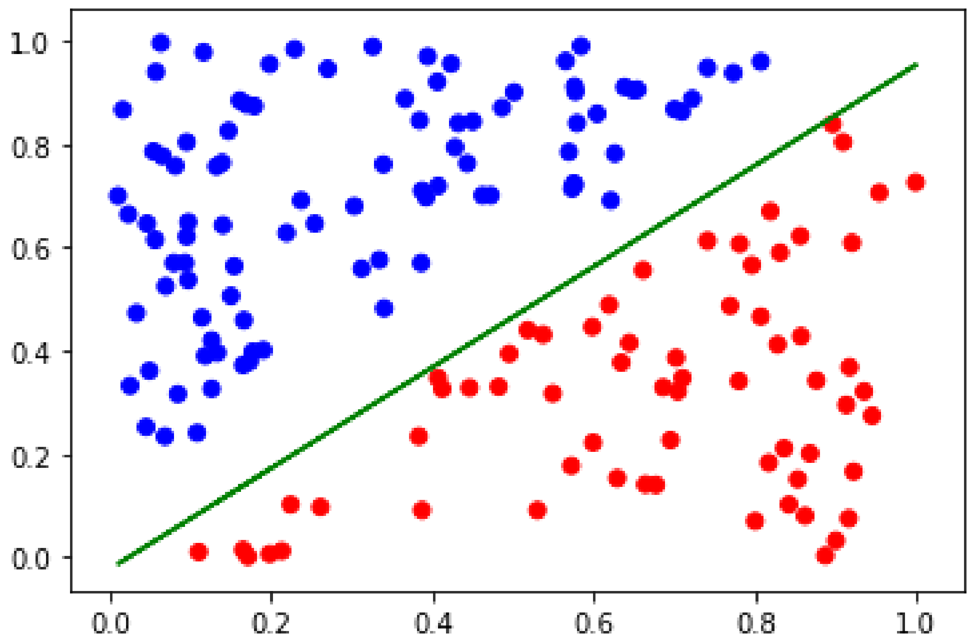


[16]:
```
# Now we create a plot to show learned decision boundaries (find slope and␣
↪intercept)
# The decision boundary line is W2.X2 + W1.X1+ W0=T (where T is threshold,
# which is the middle point between positive and negative class)
# If positive is labled as 1, and negative is labled as -1. The middle point T
# So we have W2.X2 + W1.X1+ W0=T
# The line is X2=-(W1/W2).X1 + (T-W0)/W2
# Therefore the slope is -(W1/W2), and the y-intercept is - W0/W2

print(w)
slope=w[0,1]/w[0,2]*(-1)
intercept=(T-w[0,0])/w[0,2]
print(slope,intercept)
```

```
[[-0.02655938 -0.28170133  0.33008224]]
0.8534277002367575 0.0804629312844326
```

[100]:
```
xvalues=class12_clean.iloc[:,0]
yvalues=xvalues*slope+intercept
plt.scatter(class12_clean.iloc[:,0],class12_clean.iloc[:
↪,1],color=[colors[idx+1] for idx in class12_clean.iloc[:,2]])
plt.plot(xvalues,yvalues,"g-")
```

[100]: `[<matplotlib.lines.Line2D at 0x1a1c8c66148>]`

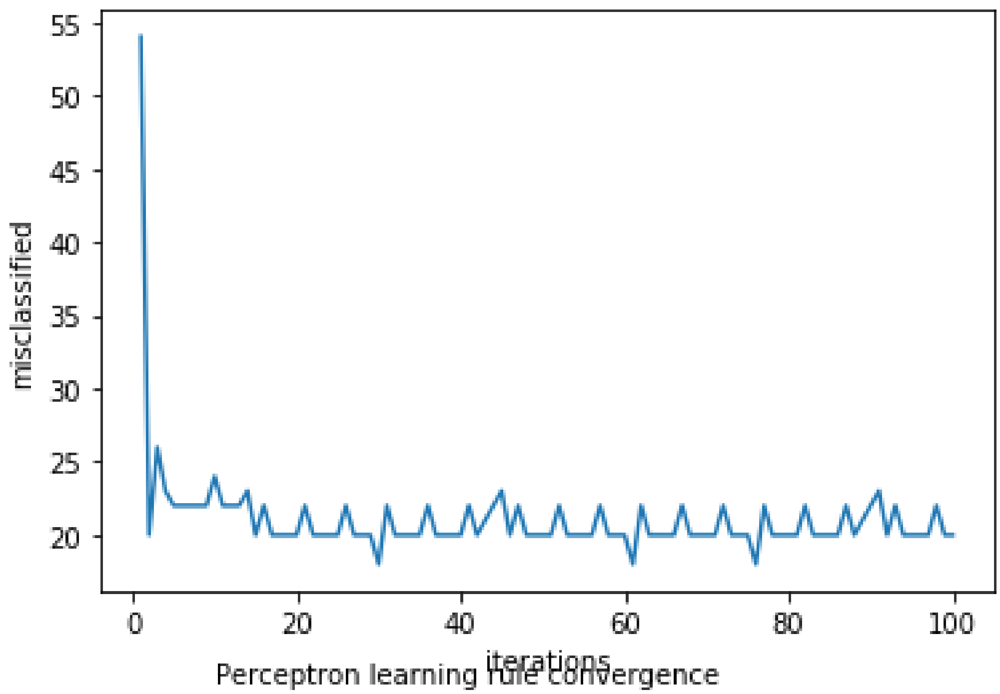



```
[17]: # if we apply perceptron learning to the original nonlinearly separable problem.
      ↪ It would not learn a good decision surfaces.
      features,labels=class12.iloc[:,0:-1],class12.loc[:,['label']]
      from sklearn.model_selection import train_test_split

      X_train, X_test, y_train, y_test = train_test_split(features, labels,␣
      ↪test_size=.4, random_state=42)
      X_train_m=np.asmatrix(X_train, dtype = 'float64')
      y_train_m=np.asmatrix(y_train, dtype = 'float64')

      num_iter = 100
      eta=0.05
      w, misclassified= perceptron(X_train_m, y_train_m, num_iter, eta)
      print(misclassified)
```

```
[54, 20, 26, 23, 22, 22, 22, 22, 22, 24, 22, 22, 22, 23, 20, 22, 20, 20, 20, 20,
22, 20, 20, 20, 20, 22, 20, 20, 20, 18, 22, 20, 20, 20, 20, 22, 20, 20, 20, 20,
22, 20, 21, 22, 23, 20, 22, 20, 20, 20, 20, 22, 20, 20, 20, 20, 22, 20, 20, 20,
18, 22, 20, 20, 20, 20, 22, 20, 20, 20, 20, 22, 20, 20, 20, 18, 22, 20, 20, 20,
20, 22, 20, 20, 20, 20, 22, 20, 21, 22, 23, 20, 22, 20, 20, 20, 20, 22, 20, 20]
```

```
[18]: epochs = np.arange(1, num_iter+1)
      plt.plot(epochs, misclassified)
      plt.xlabel('iterations')
      plt.ylabel('misclassified')
      plt.text(10,10,"Perceptron learning rule convergence")
```

```
plt.show()
```

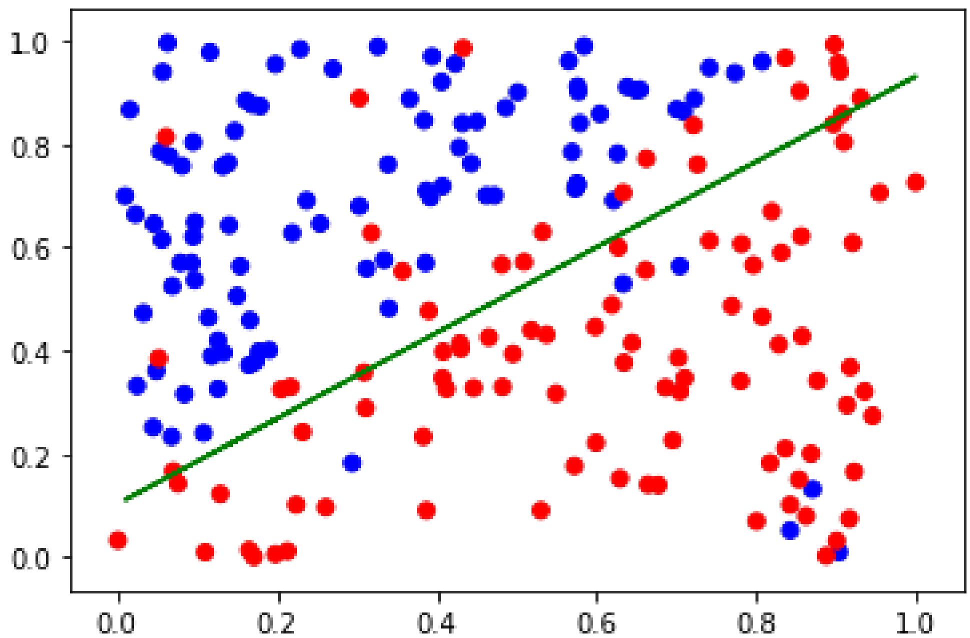


[19]:
```
# Now we create a plot to show learned decision boundaries (find slope and␣
↪intercept)
print(w)
slope=w[0,1]/w[0,2]*(-1)
intercept=(T-w[0,0])/w[0,2]
print(slope,intercept)
```

```
[[-0.04061861 -0.3263444   0.39346011]]
0.8294218086512328 0.10323437897943184
```

[20]:
```
xvalues=class12_clean.iloc[:,0]
yvalues=xvalues*slope+intercept
plt.scatter(class12.iloc[:,0],class12.iloc[:,1],color=[colors[idx+1] for idx in␣
↪class12.iloc[:,2]])
plt.plot(xvalues,yvalues,"g-")
```

[20]: [<matplotlib.lines.Line2D at 0x24679e0ec08>]



1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0