Study about Neural Networks - Percepton insperation

1. Introduction and problem design

This jupyter notebook intends to be as simpler as possible for people understand what neural networks really is. The idea is to separate two datasets with a line (hyperplane) using only one neuron.

2. Import of libraries

In [1]:

```
import numpy as np
from sklearn import datasets as d
from sklearn import model_selection as ms

import keras
from keras.models import Sequential
from keras.layers import Activation, Dense

import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

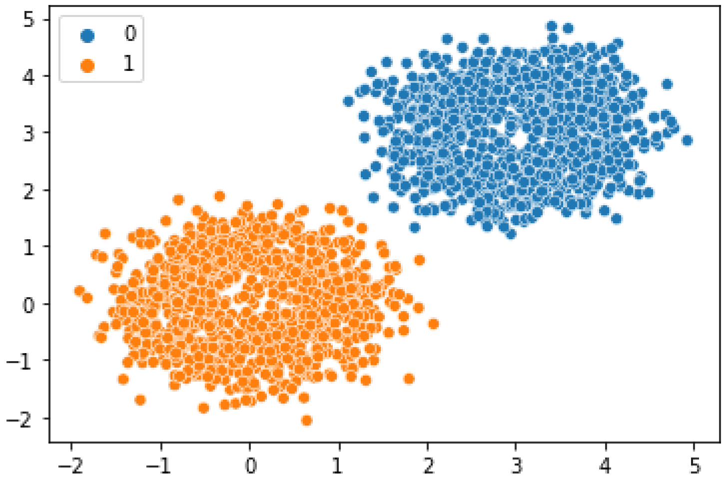
3. Generation of dataset using numpy and sklearn

In [2]:

```
space = 0.2
X = np.append(
    np.array([3,3]) + d.make_circles(n_samples=1000, noise=0.4, factor=.9)[0],
    np.array([0,0]) + d.make_circles(n_samples=1000, noise=0.4, factor=.9)[0],
    #np.array([1,1]) +np.mgrid[-space:space:step, -space:space:step].reshape(2,-
1).T,
    #np.array([0,0]) +np.mgrid[-space:space:step, -space:space:step].reshape(2,-
1).T,
    axis=0)
y = -X[:,0] +3 -X[:,1]
y = np.array([*map(lambda x: 1 if x >= 0 else 0, y)])
```

In [3]:

```
sns.scatterplot(X[:,0], X[:,1], hue=y);
```



3. Neural network modelling

Illustration of Neuron

In [4]:

```
# Splitting the dataset in trainning and validation
x_train, x_val, y_train, y_val = ms.train_test_split(X, y, test_size=0.1, random
_state=1)
```

In [5]:

```
# Creating the simple linear neuron
model = Sequential()
model.add(Dense(1, activation='linear', input_dim=2))

# Hinge loss function was best model for this problem, uncomment the others to t
est
model.compile(loss='Hinge')
# model.compile(loss='mean_squared_error')
# model.compile(loss='BinaryCrossentropy')
# model.compile(loss='MeanSquaredLogarithmicError')
# model.compile(loss='MeanAbsolutePercentageError')

model.fit(x_train, y_train, validation_data=(x_val, y_val), epochs=100)
```

```
Epoch 1/100
57/57 [==============================] - 0s 2ms/step - loss: 2.9530
- val_loss: 2.8550
Epoch 2/100
57/57 [==============================] - 0s 524us/step - loss: 2.77
82 - val_loss: 2.6830
Epoch 3/100
57/57 [==============================] - 0s 522us/step - loss: 2.60
83 - val_loss: 2.5132
Epoch 4/100
57/57 [==============================] - 0s 542us/step - loss: 2.44
05 - val_loss: 2.3444
Epoch 5/100
57/57 [==============================] - 0s 555us/step - loss: 2.27
27 - val_loss: 2.1766
Epoch 6/100
57/57 [==============================] - 0s 546us/step - loss: 2.10
44 - val_loss: 2.0082
Epoch 7/100
57/57 [==============================] - 0s 547us/step - loss: 1.93
97 - val_loss: 1.8438
Epoch 8/100
57/57 [==============================] - 0s 561us/step - loss: 1.77
37 - val_loss: 1.6776
Epoch 9/100
57/57 [==============================] - 0s 545us/step - loss: 1.60
46 - val_loss: 1.5105
Epoch 10/100
57/57 [==============================] - 0s 539us/step - loss: 1.43
84 - val_loss: 1.3461
Epoch 11/100
57/57 [==============================] - 0s 542us/step - loss: 1.27
31 - val_loss: 1.1803
Epoch 12/100
57/57 [==============================] - 0s 547us/step - loss: 1.10
57 - val_loss: 1.0148
Epoch 13/100
57/57 [==============================] - 0s 533us/step - loss: 0.94
04 - val_loss: 0.8489
Epoch 14/100
57/57 [==============================] - 0s 539us/step - loss: 0.77
57 - val_loss: 0.6876
Epoch 15/100
57/57 [==============================] - 0s 537us/step - loss: 0.63
14 - val_loss: 0.5785
Epoch 16/100
57/57 [==============================] - 0s 543us/step - loss: 0.54
24 - val_loss: 0.5165
Epoch 17/100
57/57 [==============================] - 0s 521us/step - loss: 0.48
36 - val_loss: 0.4727
Epoch 18/100
57/57 [==============================] - 0s 514us/step - loss: 0.44
41 - val_loss: 0.4418
Epoch 19/100
57/57 [==============================] - 0s 566us/step - loss: 0.41
28 - val_loss: 0.4132
Epoch 20/100
57/57 [==============================] - 0s 534us/step - loss: 0.38
49 - val_loss: 0.3849
Epoch 21/100
```

```
57/57 [==============================] - 0s 529us/step - loss: 0.35
79 - val_loss: 0.3584
Epoch 22/100
57/57 [==============================] - 0s 536us/step - loss: 0.33
26 - val_loss: 0.3333
Epoch 23/100
57/57 [==============================] - 0s 549us/step - loss: 0.30
74 - val_loss: 0.3092
Epoch 24/100
57/57 [==============================] - 0s 558us/step - loss: 0.28
33 - val_loss: 0.2848
Epoch 25/100
57/57 [==============================] - 0s 536us/step - loss: 0.26
00 - val_loss: 0.2617
Epoch 26/100
57/57 [==============================] - 0s 556us/step - loss: 0.23
89 - val_loss: 0.2404
Epoch 27/100
57/57 [==============================] - 0s 543us/step - loss: 0.21
91 - val_loss: 0.2195
Epoch 28/100
57/57 [==============================] - 0s 548us/step - loss: 0.20
03 - val_loss: 0.1997
Epoch 29/100
57/57 [==============================] - 0s 553us/step - loss: 0.18
31 - val_loss: 0.1821
Epoch 30/100
57/57 [==============================] - 0s 538us/step - loss: 0.16
75 - val_loss: 0.1662
Epoch 31/100
57/57 [==============================] - 0s 541us/step - loss: 0.15
25 - val_loss: 0.1515
Epoch 32/100
57/57 [==============================] - 0s 564us/step - loss: 0.13
89 - val_loss: 0.1385
Epoch 33/100
57/57 [==============================] - 0s 544us/step - loss: 0.12
69 - val_loss: 0.1270
Epoch 34/100
57/57 [==============================] - 0s 533us/step - loss: 0.11
63 - val_loss: 0.1163
Epoch 35/100
57/57 [==============================] - 0s 552us/step - loss: 0.10
66 - val_loss: 0.1075
Epoch 36/100
57/57 [==============================] - 0s 543us/step - loss: 0.09
74 - val_loss: 0.0998
Epoch 37/100
57/57 [==============================] - 0s 568us/step - loss: 0.08
97 - val_loss: 0.0927
Epoch 38/100
57/57 [==============================] - 0s 520us/step - loss: 0.08
26 - val_loss: 0.0855
Epoch 39/100
57/57 [==============================] - 0s 540us/step - loss: 0.07
62 - val_loss: 0.0795
Epoch 40/100
57/57 [==============================] - 0s 532us/step - loss: 0.07
04 - val_loss: 0.0738
Epoch 41/100
57/57 [==============================] - 0s 528us/step - loss: 0.06
```

```
50 - val_loss: 0.0685
Epoch 42/100
57/57 [==============================] - 0s 546us/step - loss: 0.06
01 - val_loss: 0.0636
Epoch 43/100
57/57 [==============================] - 0s 524us/step - loss: 0.05
54 - val_loss: 0.0591
Epoch 44/100
57/57 [==============================] - 0s 549us/step - loss: 0.05
13 - val_loss: 0.0558
Epoch 45/100
57/57 [==============================] - 0s 544us/step - loss: 0.04
81 - val_loss: 0.0522
Epoch 46/100
57/57 [==============================] - 0s 552us/step - loss: 0.04
47 - val_loss: 0.0492
Epoch 47/100
57/57 [==============================] - 0s 527us/step - loss: 0.04
18 - val_loss: 0.0464
Epoch 48/100
57/57 [==============================] - 0s 557us/step - loss: 0.03
89 - val_loss: 0.0437
Epoch 49/100
57/57 [==============================] - 0s 521us/step - loss: 0.03
64 - val_loss: 0.0411
Epoch 50/100
57/57 [==============================] - 0s 534us/step - loss: 0.03
43 - val_loss: 0.0393
Epoch 51/100
57/57 [==============================] - 0s 568us/step - loss: 0.03
23 - val_loss: 0.0376
Epoch 52/100
57/57 [==============================] - 0s 536us/step - loss: 0.03
06 - val_loss: 0.0359
Epoch 53/100
57/57 [==============================] - 0s 535us/step - loss: 0.02
89 - val_loss: 0.0345
Epoch 54/100
57/57 [==============================] - 0s 541us/step - loss: 0.02
76 - val_loss: 0.0332
Epoch 55/100
57/57 [==============================] - 0s 538us/step - loss: 0.02
62 - val_loss: 0.0316
Epoch 56/100
57/57 [==============================] - 0s 545us/step - loss: 0.02
51 - val_loss: 0.0305
Epoch 57/100
57/57 [==============================] - 0s 526us/step - loss: 0.02
41 - val_loss: 0.0295
Epoch 58/100
57/57 [==============================] - 0s 531us/step - loss: 0.02
32 - val_loss: 0.0286
Epoch 59/100
57/57 [==============================] - 0s 551us/step - loss: 0.02
23 - val_loss: 0.0278
Epoch 60/100
57/57 [==============================] - 0s 535us/step - loss: 0.02
14 - val_loss: 0.0273
Epoch 61/100
57/57 [==============================] - 0s 529us/step - loss: 0.02
08 - val_loss: 0.0264
```

```
Epoch 62/100
57/57 [==============================] - 0s 537us/step - loss: 0.02
00 - val_loss: 0.0257
Epoch 63/100
57/57 [==============================] - 0s 541us/step - loss: 0.01
93 - val_loss: 0.0250
Epoch 64/100
57/57 [==============================] - 0s 546us/step - loss: 0.01
87 - val_loss: 0.0243
Epoch 65/100
57/57 [==============================] - 0s 550us/step - loss: 0.01
82 - val_loss: 0.0237
Epoch 66/100
57/57 [==============================] - 0s 557us/step - loss: 0.01
75 - val_loss: 0.0230
Epoch 67/100
57/57 [==============================] - 0s 542us/step - loss: 0.01
69 - val_loss: 0.0224
Epoch 68/100
57/57 [==============================] - 0s 545us/step - loss: 0.01
63 - val_loss: 0.0218
Epoch 69/100
57/57 [==============================] - 0s 551us/step - loss: 0.01
58 - val_loss: 0.0213
Epoch 70/100
57/57 [==============================] - 0s 561us/step - loss: 0.01
54 - val_loss: 0.0208
Epoch 71/100
57/57 [==============================] - 0s 531us/step - loss: 0.01
50 - val_loss: 0.0203
Epoch 72/100
57/57 [==============================] - 0s 516us/step - loss: 0.01
46 - val_loss: 0.0199
Epoch 73/100
57/57 [==============================] - 0s 551us/step - loss: 0.01
42 - val_loss: 0.0194
Epoch 74/100
57/57 [==============================] - 0s 564us/step - loss: 0.01
39 - val_loss: 0.0190
Epoch 75/100
57/57 [==============================] - 0s 546us/step - loss: 0.01
36 - val_loss: 0.0186
Epoch 76/100
57/57 [==============================] - 0s 525us/step - loss: 0.01
33 - val_loss: 0.0182
Epoch 77/100
57/57 [==============================] - 0s 525us/step - loss: 0.01
30 - val_loss: 0.0178
Epoch 78/100
57/57 [==============================] - 0s 530us/step - loss: 0.01
27 - val_loss: 0.0174
Epoch 79/100
57/57 [==============================] - 0s 561us/step - loss: 0.01
25 - val_loss: 0.0171
Epoch 80/100
57/57 [==============================] - 0s 530us/step - loss: 0.01
23 - val_loss: 0.0168
Epoch 81/100
57/57 [==============================] - 0s 541us/step - loss: 0.01
21 - val_loss: 0.0165
Epoch 82/100
```

```
57/57 [==============================] - 0s 535us/step - loss: 0.01
18 - val_loss: 0.0163
Epoch 83/100
57/57 [==============================] - 0s 530us/step - loss: 0.01
16 - val_loss: 0.0159
Epoch 84/100
57/57 [==============================] - 0s 527us/step - loss: 0.01
15 - val_loss: 0.0155
Epoch 85/100
57/57 [==============================] - 0s 545us/step - loss: 0.01
13 - val_loss: 0.0152
Epoch 86/100
57/57 [==============================] - 0s 547us/step - loss: 0.01
11 - val_loss: 0.0150
Epoch 87/100
57/57 [==============================] - 0s 549us/step - loss: 0.01
09 - val_loss: 0.0148
Epoch 88/100
57/57 [==============================] - 0s 563us/step - loss: 0.01
07 - val_loss: 0.0146
Epoch 89/100
57/57 [==============================] - 0s 541us/step - loss: 0.01
05 - val_loss: 0.0144
Epoch 90/100
57/57 [==============================] - 0s 542us/step - loss: 0.01
03 - val_loss: 0.0142
Epoch 91/100
57/57 [==============================] - 0s 543us/step - loss: 0.01
02 - val_loss: 0.0141
Epoch 92/100
57/57 [==============================] - 0s 539us/step - loss: 0.01
00 - val_loss: 0.0139
Epoch 93/100
57/57 [==============================] - 0s 528us/step - loss: 0.00
99 - val_loss: 0.0137
Epoch 94/100
57/57 [==============================] - 0s 553us/step - loss: 0.00
97 - val_loss: 0.0135
Epoch 95/100
57/57 [==============================] - 0s 534us/step - loss: 0.00
96 - val_loss: 0.0134
Epoch 96/100
57/57 [==============================] - 0s 547us/step - loss: 0.00
94 - val_loss: 0.0132
Epoch 97/100
57/57 [==============================] - 0s 555us/step - loss: 0.00
93 - val_loss: 0.0131
Epoch 98/100
57/57 [==============================] - 0s 536us/step - loss: 0.00
92 - val_loss: 0.0129
Epoch 99/100
57/57 [==============================] - 0s 535us/step - loss: 0.00
90 - val_loss: 0.0128
Epoch 100/100
57/57 [==============================] - 0s 537us/step - loss: 0.00
89 - val_loss: 0.0126
```

Out[5]:

```
<tensorflow.python.keras.callbacks.History at 0x7f44a43286d0>
```

4. Exploring the weights of the neuron

In [6]:

```
#Test points
pto_test_x = 0.8
pto_test_y = 0.3
```

In [7]:

```
# Prediction of a point with the help of model
model.predict(np.array([[pto_test_x,pto_test_y]]))
```

Out[7]:

```
array([[1.63439]], dtype=float32)
```

In [8]:

```
# Calculating the prediction with the weight of the linear neuron
# Linear Neuron: x1*w1 + x2*w2 + b = 0
pto_test_x * model.layers[0].get_weights()[0][0] +\
pto_test_y * model.layers[0].get_weights()[0][1] +\
model.layers[0].get_weights()[1]
```

Out[8]:

```
array([1.63439], dtype=float32)
```

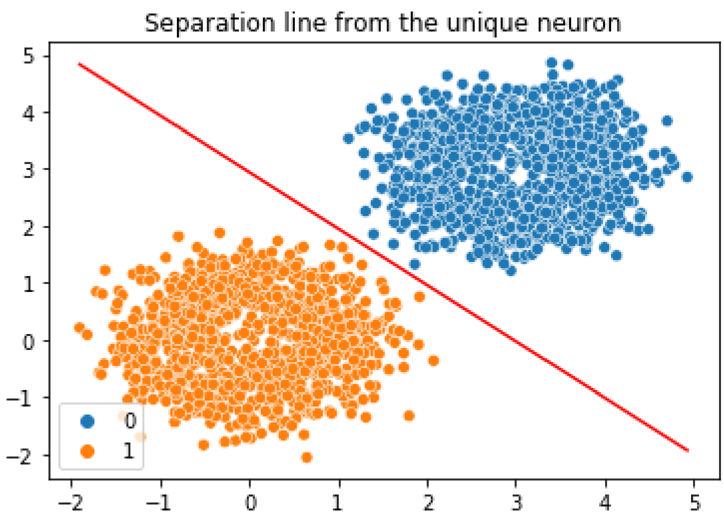
5. Plotting the line of the linear neuron

In [9]:

```
#extract weights and bias from model
weights = model.layers[0].get_weights()[0]
biases = model.layers[0].get_weights()[1]

w1 = weights[0][0]
w2 = weights[1][0]
b = biases[0]

x_line = np.linspace(x_train[:,0].min(), x_train[:,0].max(), 100)

# Linear Neuron: x1*w1 + x2*w2 + b = 0
y_line = -(b + w1*x_line) / w2

sns.scatterplot(X[:,0], X[:,1], hue=y);
plt.plot(x_line, y_line, color='red');
plt.title('Separation line from the unique neuron')
```

Out[9]:

```
Text(0.5, 1.0, 'Separation line from the unique neuron')
```



6. Conclusion

This notebook implemented a simple linear neuron in order to students understand how neural network really works. Starting with simple linear network, it is possible to expand this jupyter notebook to more complex analysis. Besides, users can modify the data and run it on simple CPUs.