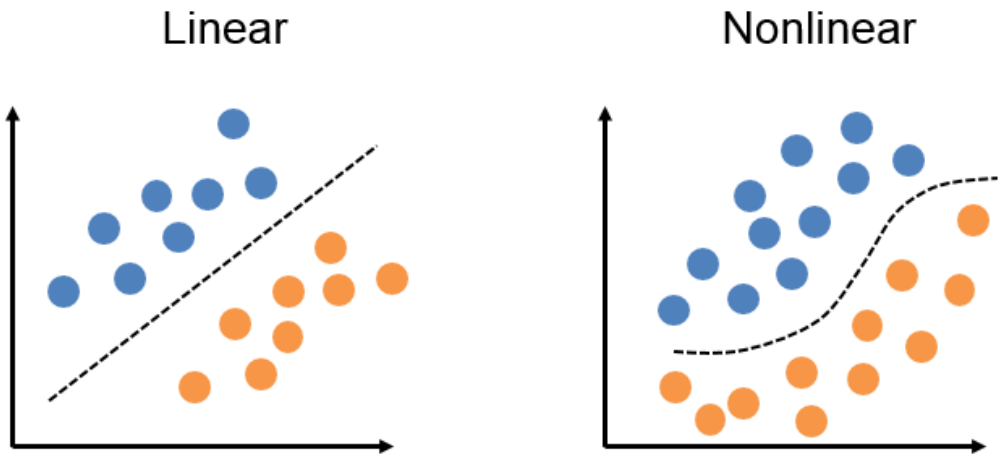
3. Perceptron Code Development

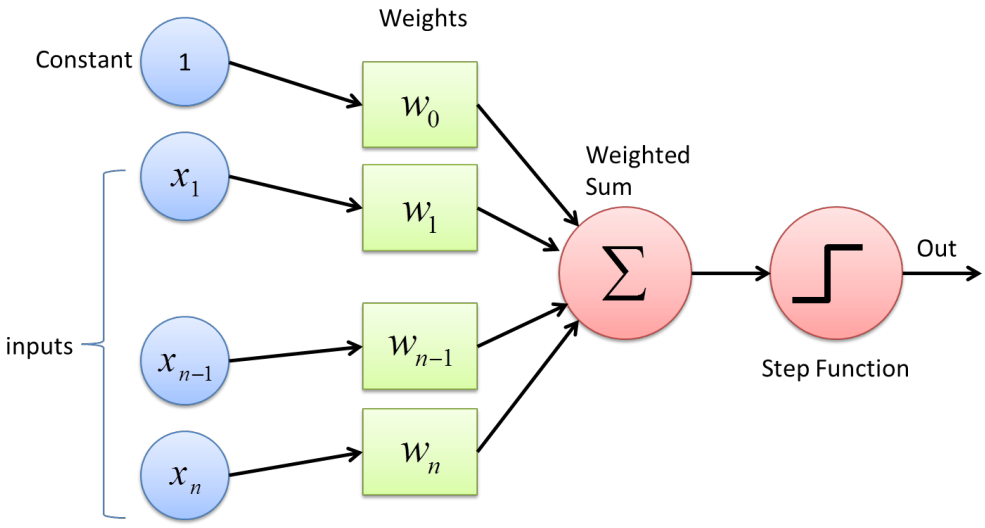
1. Review Teori perceptron

Perceptron adalah bentuk jaringan syaraf yang sederhana. Perceptron digunakan untuk mengklasifikasikan data yang terpisah linear. Seperti pada gambar dibawah ini. Jadi perceptron hanya dapat memisahkan data dengan 2 kategori.



Sumber: https://jtsulliv.github.io/perceptron/

A. Arsitektur Perceptron



Sumber : https://towardsdatascience.com

Detail dari arsitektur perceptron diatas adalah

A. Arsitektur Perceptron

- Memiliki input X_1 sampai input X_n. Input pada jaringan syaraf tiruan disebut dengan *feature* (K). Sedangkan untuk konstan dengan bobot W_0 adalah bias yang biasanya bernilai 1.
- Masing-masing input memiliki bobot yaitu W_1 sampai W_n. Bobot ditentukan diawal dengan melakukan random dari rentang 0-1.
- Weighted sum adalah penjumlah bobot dikali dengan nilai input ditambah bias sesuai dengan rumus berikut:

$$Y = b + \sum_{i=1}^{K} X_i \, . \, W_i$$

- Step function atau disebut dengan activation function digunakan untuk me-map atau mengubah hasil perhitungan weighted sum menjadi Y' atau Y prediksi. Untuk perceptron menggunakan activation linear karena kelasnya yang akan diklasifikasi akan terpisah linear misalnya (0 dan 1) atau (1 dan -1).

B. Algoritma Perceptron

Berikut algoritma untuk melakukan training dan testing. Untuk melakukan training pada umumnya menggunakan 80% dari total dataset dan sisanya 20% dari total dataset untuk testing. Bias pada algoritma perceptron ada 2 macam sumber ada yang melakukan update pada bias dan tidak melakukan update pada bias.

Algoritma Training Perceptron	
1	Inisialisasi bobot (W), bias (b), maksimum iterasi (MaxIter), learning rate (η),
2	Berdasarkan inisialisasi pada tahap ke-1 maka dihitung weighted sum berdasarkan rumus $Y = b + \sum_{i=1}^{K} X_i \, . \, W_i$
3	Menghitung fungsi aktivasi dari weighted sum
4	Jika hasil aktivasi fungsi sama dengan data target maka tidak dilakukan update bobot
5	Jika hasil aktivasi fungsi tidak sama dengan data target maka tidak dilakukan update bobot dengan menghitung error dengan cara menghitung target / label aktual (Y) dikurangi Y prediksi (Y'): $error = Y - Y'$ Kemudian menghitung bobot baru dengan formula $W_{baru} = W_{lama} + \eta . error W_i$ Kemudian menghitung bias baru dengan formula

	$b_{baru} = b_{lama} + \eta.error$
6	Ulangi iterasi sampai MaxIter

Algoritma Testing Perceptron	
1	Bobot hasil pembelajaran/ Training
2	Berdasarkan inisialisasi pada tahap ke-1 maka dihitung weighted sum berdasarkan rumus $Y = b + \sum_{i=1}^{K} X_i . W_i$
3	Menghitung fungsi aktivasi atau Y'dari weighted sum
4	Hitung untuk seluruh data

C. Perhitungan manual perceptron

Lakukan perhitungan manual untuk data set OR berikut menggunakan algoritma perceptron yang telah dijelaskan diatas. Sehingga mendapatkan hasil seperti berikut:

	Training													
	X1	X2	T	b	learning rate	w1	w2	b baru	w1 baru	w2 baru	y	f(y) = y'	f(y)==T??	error = t-f(y)
epoch	0	0	0	1	1	1	1	0	1	1	1	1	FALSE	-1
	0	1	1	0	1	1	1	0	1	1	1	1	TRUE	0
	1	0	1	0	1	1	1	0	1	1	1	1	TRUE	0
epoch	0	0	0	0	1	1	1	0	1	1	0	0	TRUE	0
	0	1	1	0	1	1	1	0	1	1	1	1	TRUE	0
	1	0	1	0	1	1	1	0	1	1	1	1	TRUE	0

Dapat diamati pada hasil perhitungan manual bahwa dengan menggunakan 6 perulangan atau 2 kali epoch mendapatkan model dari dataset OR diatas. Selanjutnya dilakukan testing terhadap model yang telah dibuat diatas, seperti berikut:

Testing										
1	1	1	0	1	1	1	2	1	TRUE	0

D. Pembuatan code perceptron

Untuk code perceptron menggunakan 2 macam data, yaitu data sintetis dengan algoritma OR diatas dan dataset bunga iris.

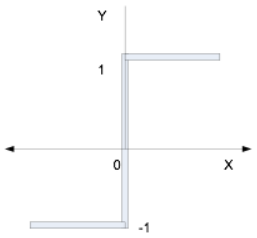
a. Implementasi Code Pada dataset OR

Buka visual code kemudian buat file baru dan simpan dengan nama perceptron_fungsi_or.py. Sesuai dengan algoritma pembelajaran perceptron yang harus dilakukan pertama kali adalah melakukan inisialisasi nilai bias, bobot, iterasi dan learning rate.

```
1   # untuk operasi matriks
2   import numpy as np
3   # untuk melakukan visualisasi
4   import matplotlib.pyplot as plt
5
6   bias = 1
7   weight = np.array([1,1])
8
9   perulangan = 100
10  learning_rate = 0.5
11  errorArray = []
12
```

Selanjutnya membuat dataset training dan dataset testing. Dan membuat fungsi aktivasi dengang Symetric hard limit.

```
13  data_training = np.array([
14      (0,0,0),
15      (0,1,1),
16      (1,0,1)])
17  data_testing = np.array([(1,1,1)])
18
19  def FungsiAktivasi(y):
20      if (y > 0):
21          return 1
22      elif(y==0):
23          return 0
24      else:
25          return -1
```



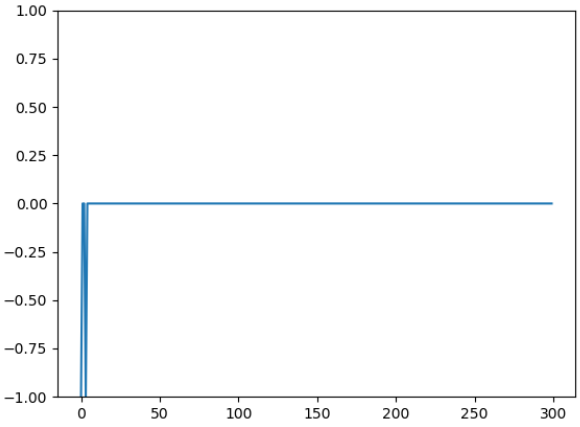
$$y = \begin{cases} 1, & jika\ x > 0 \\ 0, & jika\ x = 0 \\ -1, & jika\ x < 0 \end{cases}$$

Fugsi Symetric Hard limit

Selanjutnya menulis code untuk perhitungan weighted sum dan aktivasi fungsi seperti berikut

```
26  # training
27  for i in range(perulangan):
28      for data in data_training:
29          y = bias+(data[0] * weight[0]) + (data[1]*weight[1])
30          y_prediksi = FungsiAktivasi(y)
31          error = data[2] - y_prediksi
32          errorArray.append(error)
33          if (error!=0):
34              # print (error)
35              weight[0] += (learning_rate*error*data[0])
36              weight[1] += (learning_rate*error*data[1])
37              bias +=  learning_rate*error
38          # else:
39          #     print ('')
40
41
42  print ("model = "+str(weight[0])+";"+str(weight[1])+";")
43
44  plt.ylim([-1,1])
45  plt.plot(errorArray)
46  plt.show()
```

Hasil visualisasi dari matplotlib adalah seperti berikut



Pada proses training berapah nilai bobot dan bias yang didapat ? Isilah tabel dibawah ini!

Weight 1	
Weight 2	
Bias	

Setelah melakukan training pada tahap selanjutnya adalah melakukan testing dari model dengan menulis code dibawah ini.

```
47    # testing
48    for data in data_testing:
49        y = bias+(data[0]*weight[0]) + (data[1]*weight[0])
50        y_prediksi = FungsiAktivasi(y)
51        error = data[2] - y_prediksi
52
53        print ('y_prediksi = '+str(y_prediksi))
54        print ('error = '+str(error))
```

b. Implementasi Code Pada Dataset Bunga Iris.

Pada jobsheet ini akan menggunakan data set bunga iris yang dapat di download pada url berikut https://www.kaggle.com/arshid/iris-flower-dataset. Dataset tersebut terdiri dari 3 kategori, akan tetapi pada jobsheet ini akan menggunakan hanya 2 kelas karena sesuai dengan prinsip dari perceptron yang hanya dapat mengklasifikasi 2 kategori.

Gambar Species Bunga Iris

Dari ketiga species bunga iris pada gambar. Dipilih 2 species pada percobaan ini adalah bunga iris Versicolor dan bunga iris setosa.

Langkah percobaan :

1. Download data iris dan extract datanya seperti pada gambar

1	sepal_length,sepal_width,petal_length,petal_width,species
2	5.1,3.5,1.4,0.2,Iris-setosa
3	4.9,3,1.4,0.2,Iris-setosa
4	4.7,3.2,1.3,0.2,Iris-setosa
5	4.6,3.1,1.5,0.2,Iris-setosa
6	5,3.6,1.4,0.2,Iris-setosa
7	5.4,3.9,1.7,0.4,Iris-setosa
8	4.6,3.4,1.4,0.3,Iris-setosa
9	5,3.4,1.5,0.2,Iris-setosa
10	4.4,2.9,1.4,0.2,Iris-setosa
11	4.9,3.1,1.5,0.1,Iris-setosa
12	5.4,3.7,1.5,0.2,Iris-setosa
13	4.8,3.4,1.6,0.2,Iris-setosa
14	4.8,3,1.4,0.1,Iris-setosa
15	4.3,3,1.1,0.1,Iris-setosa
16	5.8,4,1.2,0.2,Iris-setosa
17	5.7,4.4,1.5,0.4,Iris-setosa
18	5.4,3.9,1.3,0.4,Iris-setosa
19	5.1,3.5,1.4,0.3,Iris-setosa
20	5.7,3.8,1.7,0.3,Iris-setosa
21	5.1,3.8,1.5,0.3,Iris-setosa
22	5.4,3.4,1.7,0.2,Iris-setosa
23	5.1,3.7,1.5,0.4,Iris-setosa
24	4.6,3.6,1,0.2,Iris-setosa
25	5.1,3.3,1.7,0.5,Iris-setosa
26	4.8,3.4,1.9,0.2,Iris-setosa
27	5,3,1.6,0.2,Iris-setosa
28	5,3.4,1.6,0.4,Iris-setosa
29	5.2,3.5,1.5,0.2,Iris-setosa
30	5.2,3.4,1.4,0.2,Iris-setosa

2. Hapus header tabel pada baris pertama
3. Hapus data dengan label / species iris virginica
4. Setelah melakukan proses pembersihan data maka akan didapat data untuk kedua species iris versicolor dan iris setosa dengan total 100 data.
5. Pada tahap selanjutnya akan dipisahkan antara data training dan data testing, karena jumlah keseluruhan data adalah 100 pada 80 data digunakan untuk training dan 20 data digunakan untuk testing.
6. 10 data testing dari species iris setosa dan 10 data testing dari iris versicolor.

   Hasil dari langkah 5 dan 6 kita akan mendapatkan 2 excel yang berisi data training (simpan dengan nama **IRIS-edit-training.xls**) dan testing (simpan dengan nama **IRIS-edit-testing.xls**) seperti berikut:

A	
1	5.1,3.5,1.4,0.2,Iris-setosa
2	4.9,3,1.4,0.2,Iris-setosa
3	4.7,3.2,1.3,0.2,Iris-setosa
4	4.6,3.1,1.5,0.2,Iris-setosa
5	5,3.6,1.4,0.2,Iris-setosa
6	5.4,3.9,1.7,0.4,Iris-setosa
7	4.6,3.4,1.4,0.3,Iris-setosa
8	5,3.4,1.5,0.2,Iris-setosa
9	4.4,2.9,1.4,0.2,Iris-setosa
10	4.9,3.1,1.5,0.1,Iris-setosa
11	5.4,3.7,1.5,0.2,Iris-setosa
12	4.8,3.4,1.6,0.2,Iris-setosa
13	4.8,3,1.4,0.1,Iris-setosa
14	4.3,3,1.1,0.1,Iris-setosa
15	5.8,4,1.2,0.2,Iris-setosa
16	5.7,4.4,1.5,0.4,Iris-setosa
17	5.4,3.9,1.3,0.4,Iris-setosa
18	5.1,3.5,1.4,0.3,Iris-setosa
19	5.7,3.8,1.7,0.3,Iris-setosa
20	5.1,3.8,1.5,0.3,Iris-setosa
21	5.4,3.4,1.7,0.2,Iris-setosa
22	5.1,3.7,1.5,0.4,Iris-setosa
23	4.6,3.6,1,0.2,Iris-setosa
24	5.1,3.3,1.7,0.5,Iris-setosa
25	4.8,3.4,1.9,0.2,Iris-setosa
26	5,3,1.6,0.2,Iris-setosa
27	5,3.4,1.6,0.4,Iris-setosa
28	5.2,3.5,1.5,0.2,Iris-setosa
29	5.2,3.4,1.4,0.2,Iris-setosa
30	4.7,3.2,1.6,0.2,Iris-setosa
31	4.8,3.1,1.6,0.2,Iris-setosa
32	5.4,3.4,1.5,0.4,Iris-setosa

Isi data training yang berjumlah 80 data

1	4.5,2.3,1.3,0.3,Iris-setosa
2	4.4,3.2,1.3,0.2,Iris-setosa
3	5,3.5,1.6,0.6,Iris-setosa
4	5.1,3.8,1.9,0.4,Iris-setosa
5	4.8,3,1.4,0.3,Iris-setosa
6	5.1,3.8,1.6,0.2,Iris-setosa
7	4.6,3.2,1.4,0.2,Iris-setosa
8	5.3,3.7,1.5,0.2,Iris-setosa
9	5,3.3,1.4,0.2,Iris-setosa
10	7,3.2,4.7,1.4,Iris-versicolor
11	5.5,2.6,4.4,1.2,Iris-versicolor
12	6.1,3,4.6,1.4,Iris-versicolor
13	5.8,2.6,4,1.2,Iris-versicolor
14	5,2.3,3.3,1,Iris-versicolor
15	5.6,2.7,4.2,1.3,Iris-versicolor
16	5.7,3,4.2,1.2,Iris-versicolor
17	5.7,2.9,4.2,1.3,Iris-versicolor
18	6.2,2.9,4.3,1.3,Iris-versicolor
19	5.1,2.5,3,1.1,Iris-versicolor
20	5.7,2.8,4.1,1.3,Iris-versicolor

Isi data testing yang berjumlah 20 data

7. Setelah memproses data secara manual selanjutnya adalah membuat coding untuk membuat model data bunga iris.

Perceptron manual

```
#### untuk operasi matriks
import numpy as np
#### untuk visualisasi
import matplotlib.pyplot as plt
#### untuk menghitung akurasi score dari perhitungan manual
from sklearn.metrics import accuracy_score

#### Inisialisasi bias, bobot, perulangan, learning rate
bias = np.random.random_sample()
weight = np.random.random_sample((4,)).astype(float)

perulangan = 1000
learning_rate = 0.8

#### mengatur path sesuai dengan path csv
path =str('/Users/arie/Documents/_Work/Komputasi Kognitiv/Minggu ke 3/')
data_training = np.genfromtxt(path+'IRIS-edit-
training.csv',delimiter=',',dtype='unicode')
data_testing = np.genfromtxt(path+'IRIS-edit-
testing.csv',delimiter=',',dtype='unicode')

#### preprocess data iris
#### mengubah  datairis yang berlabel Iris-versicolor menjadi 1
#### mengubah  datairis yang berlabel Setosa menjadi 0
def Preprocess(dataIris):
    for singleData in dataIris:
        if (singleData[4] == 'Iris-versicolor'):
            singleData[4] = 1
        else:
            singleData[4] = 0
    return dataIris

#### Fungsi aktivasi dengan hard limit
def FungsiAktivasi(y):
    if (y > 0):
        return 1
    elif(y<=0):
        return 0

#### data training dan testing diubah kedalam bentuk float
data_training  =  Preprocess(data_training).astype(float)
data_testing   =  Preprocess(data_testing).astype(float)

####### training
for i in range(perulangan):
    for data in data_training:
```

```
        # print(type (data[0]))
        y = bias + np.dot(data[0:4],weight)
        # y = bias+(data[0] * weight[0]) + (data[1]*weight[1]) +
(data[2]*weight[2]) + (data[3]*weight[3])
        y_prediksi = FungsiAktivasi(y)
        error = data[4] - y_prediksi
        if (error!=0):
            # print (error)
            weight[0] = weight[0]   + (learning_rate*error*data[0])
            weight[1] = weight[1]   + (learning_rate*error*data[1])
            weight[2] = weight[2]   + (learning_rate*error*data[2])
            weight[3] = weight[3]   + (learning_rate*error*data[3])
            ####### boleh diupdate atau tidak berdasarkan sumber
            bias      = bias + learning_rate*error

# # testing
y_predict = []
for data in data_testing:
    y = bias + np.dot(data[0:4],weight)
    # y = bias+(data[0] * weight[0]) + (data[1]*weight[1]) + (data[2]*weight[2]) +
(data[3]*weight[3])
    y_prediksi = FungsiAktivasi(y)
    y_predict.append(y_prediksi)

print (accuracy_score(data_testing[:,4], np.array(y_predict)))
```

8. Membandingkan implementasi manual dengan library scikit learn.

a. Untuk menggunakan library terbut lakukan instalasi dengan pip

```
pip install scikit-learn

atau

conda install scikit-learn
```

b. Implementasi dengan scikit-learn

Implementasi perceptron dengan scikit learn

```
# sklearn implementation
import numpy as np
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score

path =str('/Users/arie/Documents/_Work/Komputasi Kognitiv/Minggu ke 3/')
```

```
data_training = np.genfromtxt(path+'IRIS-edit-
training.csv',delimiter=',',dtype='unicode')
data_testing = np.genfromtxt(path+'IRIS-edit-
testing.csv',delimiter=',',dtype='unicode')

def Preprocess(data):
    for singleData in data:
        if (singleData[4] == 'Iris-versicolor'):
            singleData[4] = 1
        else:
            singleData[4] = 0
    return data

data_training = Preprocess(data_training).astype(float)
data_testing = Preprocess(data_testing).astype(float)

x_traning = data_training[:,0:4]
y_training = data_training[:,4]

x_test = data_testing[:,0:4]
y_test =  data_testing[:,4]
print(y_test)

clf = Perceptron(random_state=None, eta0= 0.1, shuffle=False, penalty=None,
class_weight=None, fit_intercept=False)

clf.fit(x_traning, y_training)

y_predict = clf.predict(x_test)

print ("sklearn perceptron coeffs:")
print (clf.coef_)

print ("Akurasi sklearn Perceptron :")
print (accuracy_score(y_test, y_predict))
```

9. Tugas

a. berapakah weight1, weight2, weight3, weight4, yang didapat pada implementasi perceptron manual dan dengan sklearn? Untuk implementasi manual buat code untuk print bobot

b. Ganti fungsi aktivasi pada implementasi manual dengan menggunakan fungsi aktivasi Symetric Hard limit.

c. Bagaimana akurasi score ketika diubah dengan aktivasi Symetric hard limit

d. Lakukan perubahan learning rate pada implementasi manual menjadi 0.1 apakah ada pengaruh terhadap bobot yang diperoleh?
e. Buat langkah-langha untuk cara debugging pada python