LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

BAB	: EXTREME LEARNING MACHINE
NAMA	: DIMAS TRI MUSTAKIM
NIM	: 205150200111049
TANGGAL	: 21/11/2022
ASISTEN	: ANDIKA IRZA PRADANA

A. Praktikum

1. Buka Google Colaboratory melalui tautan ini.
2. Tulis kode berikut ke dalam setiap *cell* pada *notebook* tersebut.
    a. Fungsi *Training* ELM

```
import time
import numpy as np

def elm_fit(X, target, h, W=None):
  start_time = time.time()
  if W is None:
    W = np.random.uniform(-.1, .1, (h, len(X[0])))

  Hinit = X @ W.T
  H = 1 / (1 + np.exp(-Hinit))
  Ht = H.T
  Hp = np.linalg.inv(Ht @ H) @ Ht
  beta = Hp @ target
  y = H @ beta

  mape = sum(abs(y - target) / target) * 100 / len(target)

  execution = time.time() - start_time
  print("Waktu eksekusi: %s detik" % execution)

  return W, beta, mape, execution
```

    b. Fungsi *Testing* ELM

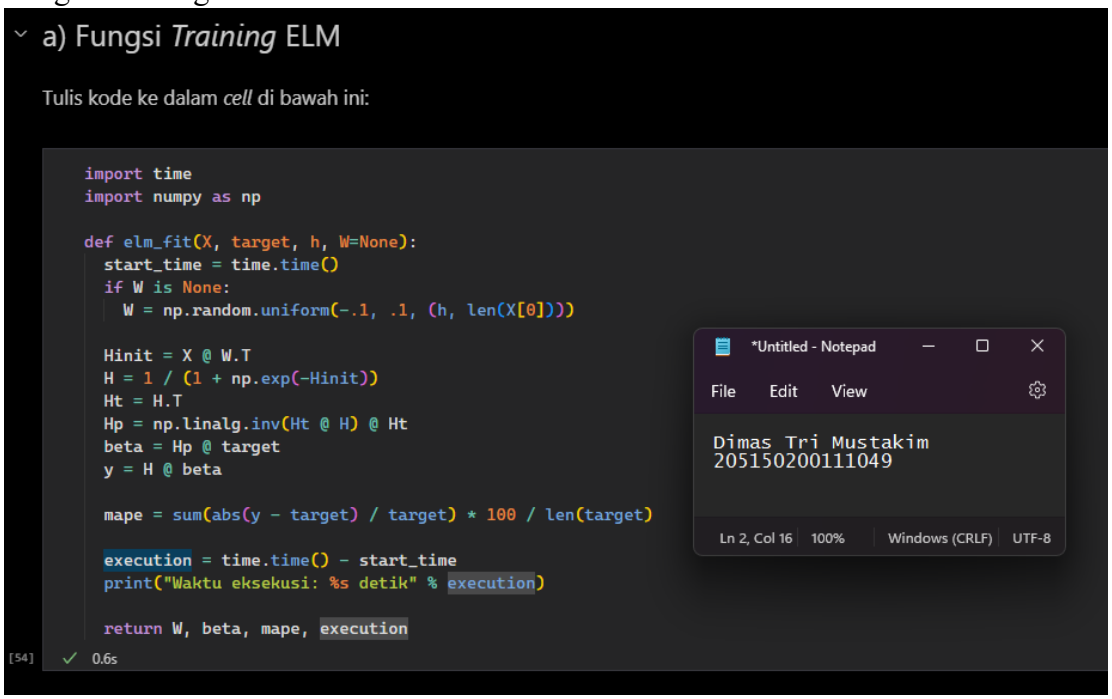
```
def elm_predict(X, W, b, round_output=False):
  Hinit = X @ W.T
  H = 1 / (1 + np.exp(-Hinit))
  y = H @ b
  if round_output:
    y = [int(round(x)) for x in y]
  return y
```

    c. Percobaan Klasifikasi Dataset Iris

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import minmax_scale
from sklearn.metrics import accuracy_score

iris = datasets.load_iris()
X = minmax_scale(iris.data)
Y = iris.target
Y += 1

X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=.3)
W, b, mape, _ = elm_fit(X_train, y_train, 3)
print('MAPE:', mape)

output = elm_predict(X_test, W, b, round_output=True)
accuracy = accuracy_score(output, y_test)

print('Output:', output)
print('True :', y_test)
print('Accuracy:', accuracy)
```

B. Screenshot

a. Fungsi Training ELM



b. Fungsi Testing ELM

b) Fungsi *Testing* ELM

Tulis kode ke dalam *cell* di bawah ini:

```
def elm_predict(X, W, b, round_output=False):
  Hinit = X @ W.T
  H = 1 / (1 + np.exp(-Hinit))
  y = H @ b
  if round_output:
    y = [int(round(x)) for x in y]
  return y
```

[55] ✓ 0.3s

*Untitled - Notepad

File Edit View

Dimas Tri Mustakim
205150200111049

Ln 2, Col 16 100% Windows (CRLF) UTF-8

c. Klasifikasi Dataset Iris

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import minmax_scale
from sklearn.metrics import accuracy_score

iris = datasets.load_iris()
X = minmax_scale(iris.data)
Y = iris.target
Y += 1

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=.3)
W, b, mape, _ = elm_fit(X_train, y_train, 3)
print('MAPE:', mape)

output = elm_predict(X_test, W, b, round_output=True)
accuracy = accuracy_score(output, y_test)

print('Output:', output)
print('True :', y_test)
print('Accuracy:', accuracy)
```

[56] ✓ 0.3s

*Untitled - Notepad

File Edit View

Dimas Tri Mustakim
205150200111049

Ln 2, Col 16 100% Windows (CRLF) UTF-8

```
Waktu eksekusi: 0.0 detik
MAPE: 15.331135580891765
Output: [2, 2, 1, 2, 3, 2, 1, 1, 2, 1, 3, 1, 2, 2, 1, 3, 1, 2, 1, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2, 3, 1, 1, 1, 2, 2, 2, 2, 2, 3, 2, 3, 1, 4]
True : [2 2 1 2 3 1 1 1 2 1 3 1 2 2 1 3 1 3 1 3 1 2 3 3 3 1 2 1 2 2 3 1 1 1 2 2 2
 2 3 2 3 2 3 1 3]
Accuracy: 0.8222222222222222
```

C. Analisis

1. Lakukan klasifikasi dengan menggunakan dataset Iris seperti pada contoh di atas. Ubahlah nilai pengaturan sebagai berikut:
    a. Rasio data latih: 70% dan data uji: 30%
    b. Jumlah hidden neuron: 3;5;7;10;30

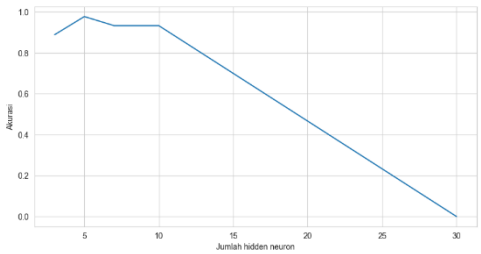
    Lakukanlah pengujian menggunakan jumlah hidden hidden neuron yang berbeda dan bandingkan hasilnya. Analisa kemampuan algoritma ELM untuk mengklasifikasikan dataset Iris tersebut.

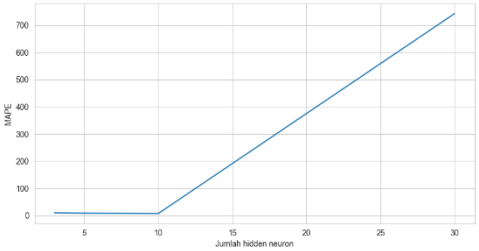
    Jawab:

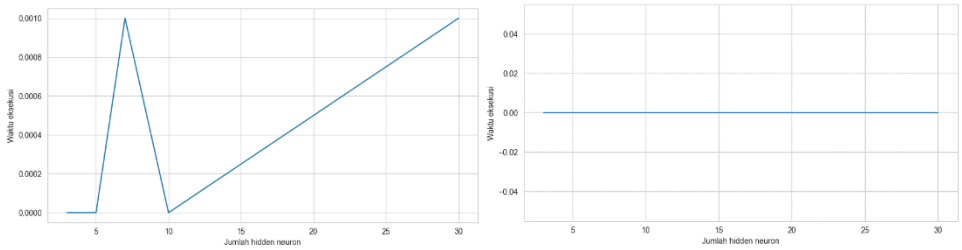
    Akurasi Pada percobaan ini saya melakukan percobaan klasifikasi dataset iris menggunakan beberapa jumlah hidden neuron kemudian mencatat hasilnya. Parameter hasil yang saya catat adalah akurasi, mape, dan waktu eksekusi. Dari percobaan ini saya dapatkan bahwa dengan mengubah jumlah hidden neuron, nilai akurasi dapat naik (jumlah neuron 5 memiliki akurasi lebih tinggi dibandingkan jumlah 3), tetapi juga bisa turun (jumlah neuron 7 lebih rendah akurasinya dibanding jumlah 5). Hal tersebut membuktikan bahwa semakin banyak jumlah neuron tidak berarti akurasi akan semakin baik.

    Dari percobaan ini juga terdapat fenomena dimana ELM tidak bisa melakukan klasifikasi ketika algoritma ini mendapat nilai akurasi 0 pada saat hidden neuron yang digunakan berjumlah 30.

    Selain itu, disini saya juga memperhatikan waktu eksekusi dari algoritma ini dan saya dapatkan bahwa tidak terlalu banyak perbedaan waktu eksekusi saat jumlah hidden neuronnya ditambah. Perbedaan waktu eksekusi yang saya dapati tidak konsisten dan saya pikir itu lebih disebabkan keadaan hardware/sistem saat komputasi terjadi. Hal ini bisa terjadi karena dalam algoritma pelatihan yang ditulis menggunakan operasi matriks daripada looping atau semacamnya.







1) Pengujian algoritma ELM dengan jumlah hidden neuron yang berbeda-beda

```
iris = datasets.load_iris()
X = minmax_scale(iris.data)
Y = iris.target
Y += 1

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=.3)


list_hidden_neuron = [3, 5, 7, 10, 30]
list_result = []

for num_hidden in list_hidden_neuron:
  W, b, mape, execution = elm_fit(X_train, y_train, num_hidden)
  print('MAPE:', mape)

  output = elm_predict(X_test, W, b, round_output=True)
  accuracy = accuracy_score(output, y_test)

  print(f'Jumlah hidden neuron : {num_hidden}')
  print('Output:', output)
  print('True :', y_test)
  print('Accuracy:', accuracy)
  print()

  result = [mape, execution, accuracy]
  list_result.append(result)
```

*Untitled - Notepad
File Edit View
Dimas Tri Mustakim
20515020011049
Ln 2, Col 16 100% Windows (CRLF) UTF-8

[66] ✓ 0.4s

```
Output exceeds the size limit. Open the full output data in a text editor
Waktu eksekusi: 0.0 detik
MAPE: 10.061582154862672
Jumlah hidden neuron : 3
Output: [1, 2, 2, 2, 3, 3, 3, 3, 1, 3, 1, 2, 3, 1, 3, 2, 3, 1, 2, 2, 2, 2, 2, 3, 1, 3, 2, 3, 2, 1, 2, 3, 2, 1, 1, 1, 3, 2, 3, 1, 1, 2, 2]
True : [1 3 2 2 3 3 3 3 3 1 3 1 2 3 1 3 2 3 1 2 3 2 2 2 2 3 1 2 2 3 2 1 2 3 3 1 1
 1 3 3 3 1 1 2 2]
Accuracy: 0.8888888888888888

Waktu eksekusi: 0.0 detik
MAPE: 8.522732586632287
Jumlah hidden neuron : 5
Output: [1, 3, 2, 2, 3, 3, 3, 3, 1, 3, 1, 2, 3, 1, 3, 2, 3, 1, 2, 2, 2, 2, 2, 3, 1, 2, 2, 3, 2, 1, 2, 3, 3, 1, 1, 1, 3, 3, 3, 1, 1, 2, 2]
```

2. (a) Lakukan klasifikasi menggunakan dataset Iris seperti pada contoh di atas dengan menggunakan metode Backpropagation dengan parameter berikut:
   - Rasio data latih: 70% dan data uji: 30%
   - Hidden neuron = 3
   - Max epoch = 100
   - Learning rate = 0.1
   - Max error = 0.5

   Catat hasil klasifikasi dengan menggunakan metode Backpropagation.

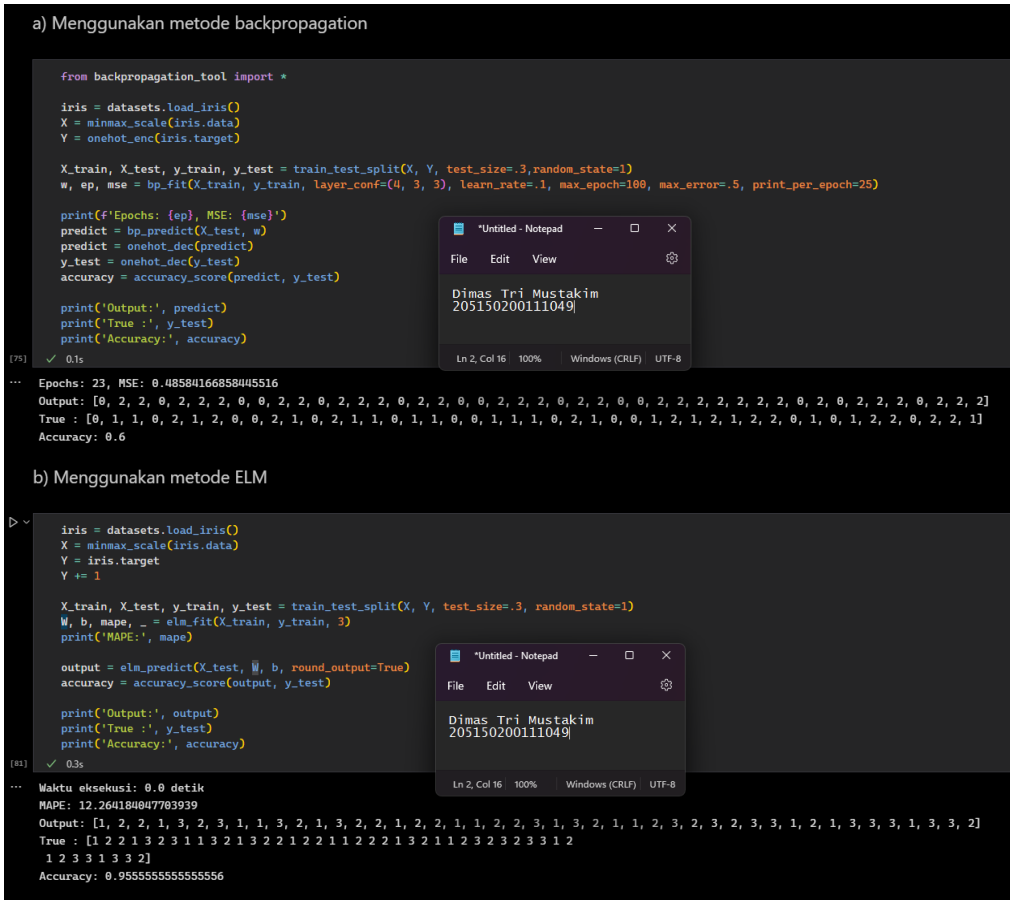
(b) Lakukanlah klasifikasi menggunakan dataset Iris seperti pada contoh diatas dengan menggunakan metode ELM dengan parameter berikut:

- Rasio data latih: 70% dan data uji: 30%
- Hidden neuron = 3

Catat hasil klasifikasi dengan menggunakan metode ELM.
Lakukan analisa dari perbandingan kedua penerapan klasifikasi tersebut dari segi akurasi dan identifikasi waktu komputasi pada saat proses training. Metode manakah yang terbaik dilihat dari segi akurasi dan waktu komputasi? Analisa hasil tersebut.

Jawab:

Dari percobaan membandingkan algoritma backpropagation dan ELM pada nomor ini, saya dapatkan hasil bahwa algoritma ELM menghasilkan nilai akurasi yang lebih tinggi dibandingkan backprogation ketika dikonfigurasi dengan parameter seperti soal diatas. Pengaturan tersebut membuat algoritma backpropagation bisa berjalan lebih cepat namun tidak dapat mencapai akurasi maksimalnya. Sedangkan ELM bisa mendapat akurasi yang baik karena memiliki kelebihan yaitu proses trainingnya berjalan dengan cepat. Jadi, algoritma ELM memiliki proses training yang cepat dan dapat melakukan generalisasi yang baik, serta tidak kalah dari algoritma backpropagation.



D. Kesimpulan

Single-hidden-layer Feedforward Neural Networks (SLFNs) merupakan jenis arsitektur jaringan saraf tiruan yang memiliki struktur berupa input layer, sebuah hidden layer, dan Output layer. Layer-layer tersebut terhubung satu dengan yang lainnya. Data yang masuk dari input layer akan diteruskan ke hidden layer, kemudian ke output layer untuk mendapatkan keluaran yang berupa klasifikasi atau regresi. Extreme Learning Machine merupakan salah satu algoritma pelatihan yang dapat digunakan untuk arsitektur SLFNs. Penyesuaian bobot pada algoritma ELM berbeda dengan algoritma lain yang mana menggunakan operasi pseudo-inverse Moore-Penrose dengan menggunakan pendekatan metode Least Square. Algoritma ini memiliki kelebihan dimana proses trainingnya jauh lebih cepat dan dapat melakukan generalisasi dengan baik.

Algoritma ELM berbeda dengan backpropagation pada cara mereka mengupdate bobot pada proses pelatihan. ELM tidak membutuhkan proses backpropagation untuk mengupdate bobot dan menggantinya dengan operasi pseudo-inverse Moore-Penrose. ELM juga bekerja pada arsitektur SLFNs sedangkan backpropagation dapat bekerja di arsitektur dengan beberapa hidden layer. ELM juga hanya melakukan proses pelatihannya sekali dan tidak melakukan perulangan sehingga algoritmanya juga jauh lebih cepat daripada algoritma backpropagation.

Semua algoritma terdapat kasus penggunaannya sendiri-sendiri dan baik pada kasus dimana mereka dibuat. ELM paling bagus digunakan untuk jumlah dataset yang relatif kecil dan menghasilkan hasil yang lebih baik daripada menggunakan backpropagation. ELM juga menurut saya cocok digunakan ketika waktu dan resource komputasi sangat terbatas karena lebih cepat dalam proses trainingnya. Namun, pada kasus dimana datasetnya lebih kompleks seperti data gambar, backpropagation jauh lebih baik daripada ELM.