



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

BAB : EXTREME LEARNING MACHINE
NAMA : ARION SYEMAEL SIAHAAN
NIM : 225150207111060
TANGGAL : 20/11/2024
ASISTEN : ALIFAH KHAIRUNNISA
ANDHIKA IHSAN CENDEKIA



A. Praktikum

1. Buka Google Collaboratory 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
```

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. 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



```

a) Fungsi Training ELM

Tulis kode ke dalam cell di bawah ini:

[12] 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

```

b. Fungsi Testing ELM



```

b) Fungsi Testing ELM

Tulis kode ke dalam cell di bawah ini:

[13] 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. Klasifikasi Dataset Iris



The screenshot shows a Jupyter Notebook interface with a dark theme. At the top, there's a section titled 'c) Klasifikasi Dataset Iris' with a sub-header 'Iris Dataset'. Below it, a text prompt says 'Tulis kode ke dalam cell/di bawah ini:'. The main code cell contains Python code for loading the Iris dataset, scaling it, splitting it into training and testing sets, training an ELM model, and evaluating its performance. The output of the code is displayed at the bottom of the cell, showing the execution time, MAPE value, predicted and true labels, and the accuracy score. A watermark for 'ARION SYEMAE SIAHAAN' is visible on the right side of the code cell.

```
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)
```

Waktu eksekusi: 0.0002872943878173828 detik
MAPE: 10.402879515161738
Output: [2, 2, 2, 2, 3, 3, 1, 1, 1, 3, 3, 1, 1, 1, 3, 3, 2, 2, 1, 1, 2, 2, 2, 1, 3, 3, 3, 2, 2, 2, 1, 2, 2, 1, 3, 2, 2, 3, 2, 1, 1, 2, 1]
True : [2, 2, 2, 3, 3, 1, 1, 1, 3, 3, 1, 1, 1, 3, 3, 2, 2, 1, 1, 2, 2, 1, 3, 2, 2, 2, 2, 1, 2, 2, 1, 2, 1, 3, 2, 2, 3, 2, 1, 1, 2, 1]
Accuracy: 0.9777777777777777

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.

SOAL 1

Lakukan klasifikasi dengan menggunakan dataset Iris seperti pada contoh di atas. Ubahlah nilai pengaturan sebagai berikut:

1. Rasio data latih: 70% dan data uji: 30%
2. 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.

```
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)

hidden_neurons_list = [3, 5, 7, 10, 30]

for h_neurons in hidden_neurons_list:
    print(f"\nHidden layer: {h_neurons}")

    W, b, mape = elm_fit(X_train, y_train, h_neurons)
    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)
```

ARION SYEMAE SIAHAAN
225150207111060

```
Hidden layer: 3
Waktu eksekusi: 0.0002541542053222656 detik
MAPE: 7.983583373629873
Output: [3, 2, 1, 2, 3, 2, 1, 2, 3, 3, 3, 1, 2, 1, 3, 2, 1, 3, 2, 2, 2, 2, 1, 1, 1, 3, 3, 2, 1, 1, 1, 1, 2, 3, 2, 1, 2]
True : [3 2 1 2 3 2 1 3 3 3 1 2 1 3 2 1 3 2 2 2 2 1 1 1 3 3 2 1 1 1 1 2 3 2 1 2]
2 2 2 3 3 2 2]
Accuracy: 0.9555555555555556

Hidden layer: 5
Waktu eksekusi: 0.00027370452880859375 detik
MAPE: 7.638607860535032
Output: [3, 3, 1, 2, 3, 2, 1, 2, 3, 3, 3, 1, 2, 1, 3, 2, 1, 3, 2, 2, 2, 2, 1, 1, 1, 3, 3, 2, 1, 1, 1, 1, 2, 3, 2, 1, 2]
True : [3 2 1 2 3 2 1 3 3 3 1 2 1 3 2 1 3 2 2 2 2 1 1 1 3 3 2 1 1 1 1 2 3 2 1 2]
2 2 2 3 3 2 2]
Accuracy: 0.9555555555555556

Hidden layer: 7
Waktu eksekusi: 0.0002789497375488281 detik
MAPE: 7.400314741632559
Output: [3, 2, 1, 2, 4, 2, 1, 2, 3, 3, 3, 1, 2, 1, 3, 2, 1, 3, 2, 2, 2, 2, 2, 1, 1, 1, 3, 3, 2, 1, 1, 1, 1, 2, 3, 2, 1, 2]
True : [3 2 1 2 3 2 1 3 3 3 1 2 1 3 2 1 3 2 2 2 2 1 1 1 3 3 2 1 1 1 1 2 3 2 1 2]
2 2 2 3 3 2 2]
Accuracy: 0.9555555555555556
```

```

Hidden layer: 3
Waktu eksekusi: 0.0002541542053222656 detik
MAPE: 7.983583373629873
Output: [3, 2, 1, 2, 3, 2, 1, 2, 3, 3, 3, 1, 2, 1, 3, 2, 1, 3, 2, 2, 2, 1, 1, 1, 3, 3, 2, 1, 1, 1, 2, 3, 2, 2, 2]
True : [3 2 1 2 3 2 1 3 3 3 3 1 2 1 3 2 1 3 2 2 2 2 1 1 1 3 3 2 1 1 1 1 2 3 2 1 2
2 2 2 3 3 2 2]
Accuracy: 0.9555555555555556

Hidden layer: 5
Waktu eksekusi: 0.00027370452880859375 detik
MAPE: 7.638607860535032
Output: [3, 3, 1, 2, 3, 2, 1, 2, 3, 3, 3, 1, 2, 1, 3, 2, 1, 3, 2, 2, 2, 2, 1, 1, 1, 3, 3, 2, 1, 1, 1, 3, 3, 2, 1, 1]
True : [3 2 1 2 3 2 1 3 3 3 3 1 2 1 3 2 1 3 2 2 2 2 1 1 1 3 3 2 1 1 1 1 2 3 2 1 2
2 2 2 3 3 2 2]
Accuracy: 0.9555555555555556

Hidden layer: 7
Waktu eksekusi: 0.0002789497375488281 detik
MAPE: 7.490914741632559
Output: [3, 2, 1, 2, 4, 2, 1, 2, 3, 3, 3, 1, 2, 1, 3, 2, 1, 3, 2, 2, 2, 2, 1, 1, 1, 3, 3, 2, 1, 1, 1, 2, 3, 2, 1, 2]
True : [3 2 1 2 3 2 1 3 3 3 3 1 2 1 3 2 1 3 2 2 2 2 1 1 1 3 3 2 1 1 1 1 2 3 2 1 2
2 2 2 3 3 2 2]
Accuracy: 0.9555555555555556

Hidden layer: 10
Waktu eksekusi: 0.0002684593200683594 detik
MAPE: 6.7687618741327915
Output: [3, 2, 1, 2, 4, 2, 1, 2, 3, 3, 3, 1, 2, 1, 3, 2, 1, 3, 2, 2, 2, 2, 1, 1, 1, 3, 3, 2, 1, 1, 1, 2, 3, 2, 1, 2]
True : [3 2 1 2 3 2 1 3 3 3 3 1 2 1 3 2 1 3 2 2 2 2 1 1 1 3 3 2 1 1 1 1 2 3 2 1 2
2 2 2 3 3 2 2]
Accuracy: 0.9555555555555556

Hidden layer: 30
Waktu eksekusi: 0.008156061172485352 detik
MAPE: 250.95751825335714
Output: [-1, -2, -3, -2, -1, -2, -3, -2, -1, -1, -1, -3, -2, -3, -1, -2, -3, -1, -2, -2, -2, -3, -3, -3, -1, -1, -2, -3, -3, -3, -3, -2, -1, -2,
True : [3 2 1 2 3 2 1 3 3 3 3 1 2 1 3 2 1 3 2 2 2 2 1 1 1 3 3 2 1 1 1 1 2 3 2 1 2
2 2 2 3 3 2 2]
Accuracy: 0.0

```

Setelah melakukan pengujian,

- hidden neuron 3,7, dan 10 memiliki akurasi yang sama sebesar 0.95.
- hidden neuron 5 memiliki akurasi sebesar 1,
- hidden neuron 30 memiliki akurasi sebesar 0

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.

A. Metode Backpropagation

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import minmax_scale
from sklearn.metrics import accuracy_score
import time

def bp_fit(X, target, layer_conf, max_epoch, max_error=0.5, learn_rate=0.1, print_per_epoch=100):
    start_time = time.time()
    np.random.seed(1)
    nin = [np.empty(i) for i in layer_conf]
    n = [np.empty(j + 1) if i < len(layer_conf) - 1 else np.empty(j) for i, j in enumerate(layer_conf)]
    w = [np.random.rand(layer_conf[i] + 1, layer_conf[i + 1]) for i in range(len(layer_conf) - 1)]
    dw = [np.empty((layer_conf[i] + 1, layer_conf[i + 1])) for i in range(len(layer_conf) - 1)]
    d = [np.empty(s) for s in layer_conf[1:]]
    din = [np.empty(s) for s in layer_conf[1:-1]]
    epoch = 0
    mse = 1

    for i in range(0, len(n) - 1):
        n[i][-1] = 1

    while (max_epoch == -1 or epoch < max_epoch) and mse > max_error:
        epoch += 1
        mse = 0

        for r in range(len(X)):
            n[0][-1] = X[r]

            for L in range(1, len(layer_conf)):
                nin[L] = np.dot(n[L - 1], w[L - 1])
                n[L][:len(nin[L])] = sig(nin[L])

            e = target[r] - n[-1]
            mse += sum(e ** 2)
            d[-1] = e * sigd(nin[-1])
            dw[-1] = learn_rate * d[-1] * n[-2].reshape((-1, 1))

            for L in range(len(layer_conf) - 1, 1, -1):
                din[L - 2] = np.dot(d[L - 1], np.transpose(w[L - 1][::-1]))
                d[L - 2] = din[L - 2] * np.array(sigd(nin[L - 1]))
                dw[L - 2] = (learn_rate * d[L - 2]) * n[L - 2].reshape((-1, 1))

            for i in range(len(w)):
                w[i] += dw[i]

        mse /= len(X)

        if print_per_epoch > -1 and epoch % print_per_epoch == 0:
            print(f'Epoch {epoch}, MSE: {mse}')

    execution = time.time() - start_time
    print(f"Waktu eksekusi: {execution} detik")

    return w, epoch, mse

def bp_predict(X, w):
```

ARION SYEMAEL SIAHAAN
225150207111060

sedangkan backprop 0.6. Begitu juga dengan waktu eksekusi, diperoleh waktu sebesar 0.0008311271667480469 detik untuk ELM sedangkan 0.31127071380615234 detik untuk backpropagation.

D. Kesimpulan

SLFNs adalah jenis jaringan saraf tiruan dengan satu hidden layer. Dalam ELM, bobot hidden layer diinisialisasi secara acak dan tidak diubah selama pelatihan, membuat prosesnya sederhana dan cepat.

Extreme Learning Machine (ELM) dan Backpropagation memiliki perbedaan utama dalam cara pembelajaran bobot. Pada ELM, bobot hidden layer diinisialisasi secara acak dan tetap selama pelatihan, sementara bobot output dihitung langsung tanpa iterasi. Hal ini menjadikan proses pelatihan lebih efisien. Sebaliknya, Backpropagation menggunakan pembelajaran iteratif dengan penyesuaian bobot di setiap lapisan, yang memakan waktu lebih lama tetapi memungkinkan model mencapai akurasi yang lebih tinggi.

Metode terbaik bergantung pada kebutuhan kasus yang dihadapi. ELM lebih sesuai untuk kasus yang membutuhkan pelatihan cepat dan tidak terlalu menuntut akurasi tinggi. Di sisi lain, Backpropagation lebih cocok digunakan pada kasus yang memerlukan akurasi optimal, meskipun membutuhkan waktu komputasi yang lebih lama.