LABORATORIUM PEMBELAJARAN ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

BAB : BACKPROPAGATION (2)
NAMA : ARION SYEMAEL SIAHAAN
NIM : 225150207111060
TANGGAL : 30/10/2024
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* Backpropagation

```
def bp_fit(X, target, layer_conf, max_epoch, max_error=0.1,
learn_rate=0.1, print_per_epoch=100):
    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 L in range(len(dw)):
                w[L] += dw[L]

        mse /= len(X)

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

    return w, epoch, mse
```

b. Fungsi *Testing* Backpropagation

```
def bp_predict(X, w):
    n = [np.empty(len(i)) for i in w]
    nin = [np.empty(len(i[0])) for i in w]
    predict = []
    n.append(np.empty(len(w[-1][0])))

    for x in X:
        n[0][:-1] = x

        for L in range(0, len(w)):
            nin[L] = np.dot(n[L], w[L])
            n[L + 1][:len(nin[L])] = sig(nin[L])

        predict.append(n[-1].copy())

    return predict
```

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 = onehot_enc(iris.target)
X_train,   X_test,   y_train,   y_test   =   train_test_split(X,   Y,
test_size=0.3, random_state=1)

w,  ep,  mse  =  bp_fit(X_train,  y_train,  layer_conf=(4,  3,  3),
learn_rate=0.1, max_epoch=1000, max_error=0.1, print_per_epoch=25)
print(f'Epochs: {ep}, MSE: {mse}')

predict = bp_predict(X_test, w)
```

```
predict = onehot_dec(predict)
y_test = onehot_dec(y_test)

accuracy = accuracy_score(predict, y_test)
print('Output:', predict)
print('True :', y_test)
print('Accuracy:', accuracy)
```

B. Screenshot

a. Fungsi *Training* Backpropagation

```
[42] def bp_fit(X, target, layer_conf, max_epoch, max_error=0.1, learn_rate=0.1, print_per_epoch=100):
        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 L in range(len(dw)):
                    w[L] += dw[L]
            mse /= len(X)
            if print_per_epoch > -1 and epoch % print_per_epoch == 0:
                print(f'Epoch {epoch}, MSE: {mse}')
        return w, epoch, mse
```

ARION SYEMAEL SIAHAAN
225150207111060

b. Fungsi *Testing* Backpropagation

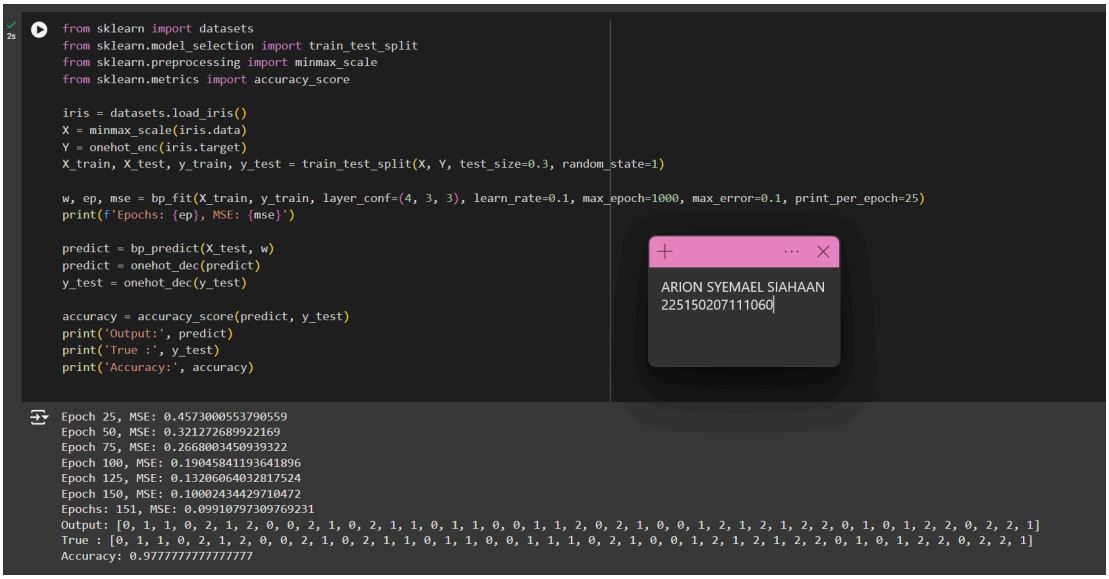
b) Fungsi *Testing* Backpropagation

Tulis kode ke dalam *cell* di bawah ini:

```
[35] def bp_predict(X, w):
        n = [np.empty(len(i)) for i in w]
        nin = [np.empty(len(i[0])) for i in w]
        predict = []
        n.append(np.empty(len(w[-1][0])))

        for x in X:
            n[0][:-1] = x

            for L in range(0, len(w)):
                nin[L] = np.dot(n[L], w[L])
                n[L + 1][:len(nin[L])] = sig(nin[L])

            predict.append(n[-1].copy())

        return predict
```

ARION SYEMAEL SIAHAAN
225150207111060

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 = onehot_enc(iris.target)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=1)

w, ep, mse = bp_fit(X_train, y_train, layer_conf=(4, 3, 3), learn_rate=0.1, max_epoch=1000, max_error=0.1, print_per_epoch=25)
print(f'Epochs: {ep}, MSE: {mse}')

predict = bp_predict(X_test, w)
predict = onehot_dec(predict)
y_test = onehot_dec(y_test)

accuracy = accuracy_score(predict, y_test)
print('Output:', predict)
print('True :', y_test)
print('Accuracy:', accuracy)
```

ARION SYEMAEL SIAHAAN
225150207111060

```
Epoch 25, MSE: 0.4573000553790559
Epoch 50, MSE: 0.321272689922169
Epoch 75, MSE: 0.2668003450939322
Epoch 100, MSE: 0.19045841193641896
Epoch 125, MSE: 0.13206064032817524
Epoch 150, MSE: 0.10002434429710472
Epochs: 151, MSE: 0.0991079730976923l
Output: [0, 1, 1, 0, 2, 1, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0, 1, 1, 2, 0, 2, 1, 0, 0, 1, 2, 1, 2, 1, 2, 2, 0, 1, 0, 1, 2, 2, 0, 2, 2, 1]
True : [0, 1, 1, 0, 2, 1, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 2, 1, 0, 0, 1, 2, 1, 2, 1, 2, 2, 0, 1, 0, 1, 2, 2, 0, 2, 2, 1]
Accuracy: 0.9777777777777777
```

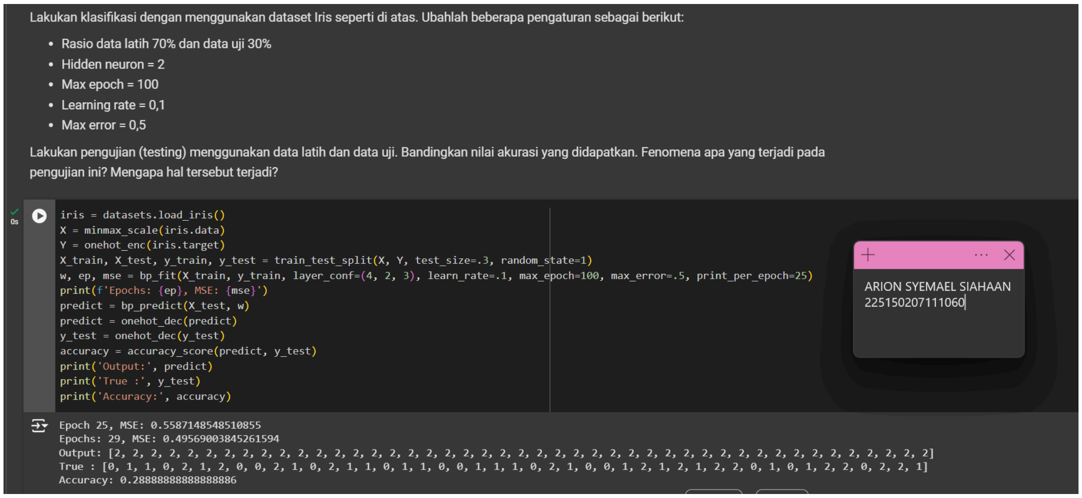
C. Analisis

1. Lakukan klasifikasi dengan menggunakan dataset Iris seperti di atas. Ubahlah beberapa pengaturan sebagai berikut:

- Rasio data latih 70% dan data uji 30%
- Hidden neuron = 2
- Max epoch = 100
- Learning rate = 0,1
- Max error = 0,5

Lakukan pengujian (testing) menggunakan data latih dan data uji. Bandingkan nilai akurasi yang didapatkan. Fenomena apa yang terjadi pada pengujian ini? Mengapa hal tersebut terjadi?

```
iris = datasets.load_iris()
X = minmax_scale(iris.data)
Y = onehot_enc(iris.target)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=.3, random_state=1)
w, ep, mse = bp_fit(X_train, y_train, layer_conf=(4, 2, 3), learn_rate=.1, max_epoch=100, max_error=.5, print_per_epoch=25)
print(f'Epochs: {ep}, MSE: {mse}')
predict = bp_predict(X_test, w)
predict = onehot_dec(predict)
y_test = onehot_dec(y_test)
accuracy = accuracy_score(predict, y_test)
print('Output:', predict)
print('True :', y_test)
print('Accuracy:', accuracy)
```



model hanya mencapai akurasi 0.28888888888888886. Hal ini terjadi karena model mengalami **underfitting** akibat jumlah neuron yang terlalu sedikit untuk menangkap pola data yang kompleks. Selain itu, konvergensi cepat mencapai error maksimum menunjukkan model belum terlatih dengan baik, dan parameter yang digunakan mungkin kurang sesuai, sehingga perlu dilakukan penyesuaian seperti menambah neuron tersembunyi atau mengurangi nilai max error.

2. Lakukan klasifikasi dengan menggunakan dataset Iris seperti di atas. Ubahlah beberapa pengaturan sebagai berikut:

- Rasio data latih 70% dan data uji 30%
- Hidden neuron = 25
- Max epoch = 10000
- Learning rate = 0,1
- Max error = 0,01

Lakukan pengujian (testing) menggunakan data latih dan data uji. Bandingkan nilai akurasi yang didapatkan. Fenomena apa yang terjadi pada pengujian ini? Mengapa hal tersebut terjadi?

```
iris = datasets.load_iris()
X = minmax_scale(iris.data)
Y = onehot_enc(iris.target)
X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=.3, random_state=1)
w, ep, mse = bp_fit(X_train, y_train, layer_conf=(4, 25, 3),
learn_rate=.1, max_epoch=10000, max_error=.01, print_per_epoch=25)
print(f'Epochs: {ep}, MSE: {mse}')
predict = bp_predict(X_test, w)
predict = onehot_dec(predict)
y_test = onehot_dec(y_test)
accuracy = accuracy_score(predict, y_test)
print('Output:', predict)
```

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

Lakukan klasifikasi dengan menggunakan dataset Iris seperti di atas. Ubahlah beberapa pengaturan sebagai berikut:

- Rasio data latih 70% dan data uji 30%
- Hidden neuron = 25
- Max epoch = 10000
- Learning rate = 0,1
- Max error = 0,01

Lakukan pengujian (testing) menggunakan data latih dan data uji. Bandingkan nilai akurasi yang didapatkan. Fenomena apa yang terjadi pada pengujian ini? Mengapa hal tersebut terjadi?

```
iris = datasets.load_iris()
X = minmax_scale(iris.data)
Y = onehot_enc(iris.target)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=.3, random_state=1)
w, ep, mse = bp_fit(X_train, y_train, layer_conf=(4, 25, 3), learn_rate=.1, max_epoch=10000, max_error=.01, print_per_epoch=25)
print(f'Epochs: {ep}, MSE: {mse}')
predict = bp_predict(X_test, w)
predict = onehot_dec(predict)
y_test = onehot_dec(y_test)
accuracy = accuracy_score(predict, y_test)
print('Output:', predict)
print('True :', y_test)
print('Accuracy:', accuracy)
```

```
Epoch 25, MSE: 1.9997475602526067
Epoch 50, MSE: 1.9996839107720699
Epoch 75, MSE: 1.9995729256252905
Epoch 100, MSE: 1.9993234166160632
Epoch 125, MSE: 1.9988004842894972
Epoch 150, MSE: 1.5034310711672474
Epoch 175, MSE: 0.9882734055300789
Epoch 200, MSE: 0.9358385629366138
Epoch 225, MSE: 0.8962403847767554
Epoch 250, MSE: 0.5720113088188044
Epoch 275, MSE: 0.17189641558830 48
Epoch 300, MSE: 0.13551945448567226
```

ARION SYEMAEL SIAHAAN
225150207111060

```
Epoch 9050, MSE: 0.020166707379553943
Epoch 9075, MSE: 0.020148333084190055
Epoch 9100, MSE: 0.020130228692938686
Epoch 9125, MSE: 0.020112388726943303
Epoch 9150, MSE: 0.020094807844390314
Epoch 9175, MSE: 0.020077480836615955
Epoch 9200, MSE: 0.020060402624327983
Epoch 9225, MSE: 0.020043568253939038
Epoch 9250, MSE: 0.020026972894008465
Epoch 9275, MSE: 0.020010611831790242
Epoch 9300, MSE: 0.019994480469883006
Epoch 9325, MSE: 0.01997857432298042
Epoch 9350, MSE: 0.01996288901471828
Epoch 9375, MSE: 0.019947420274615907
Epoch 9400, MSE: 0.019932163935108802
Epoch 9425, MSE: 0.019917115928670397
Epoch 9450, MSE: 0.01990227228501987
Epoch 9475, MSE: 0.01988762912841347
Epoch 9500, MSE: 0.019873182675017267
Epoch 9525, MSE: 0.019858929230358303
Epoch 9550, MSE: 0.01984486518685261
Epoch 9575, MSE: 0.019830987021406912
Epoch 9600, MSE: 0.019817291293092204
Epoch 9625, MSE: 0.01980377464088707
Epoch 9650, MSE: 0.019790433781488603
Epoch 9675, MSE: 0.01977726550718879
Epoch 9700, MSE: 0.019764266683813994
Epoch 9725, MSE: 0.01975143424872671
Epoch 9750, MSE: 0.019738765208886025
Epoch 9775, MSE: 0.019726256638966762
Epoch 9800, MSE: 0.019713905679533912
Epoch 9825, MSE: 0.019701709535271755
Epoch 9850, MSE: 0.019689665473265432
Epoch 9875, MSE: 0.019677770821333786
Epoch 9900, MSE: 0.01966602296641068
Epoch 9925, MSE: 0.019654419352975523
Epoch 9950, MSE: 0.019642957481529
Epoch 9975, MSE: 0.019631634907114 45
Epoch 10000, MSE: 0.019620449237882337
Epochs: 10000, MSE: 0.019620449237882337
Output: [0, 1, 1, 0, 2, 1, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0, 1, 1, 2, 0, 2, 1, 0, 0, 1, 2, 1, 2, 1, 2, 2, 0, 1, 0, 1, 2, 2, 0, 1, 2, 1]
True : [0, 1, 1, 0, 2, 1, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 2, 1, 0, 0, 1, 2, 1, 2, 1, 2, 2, 0, 1, 0, 1, 2, 2, 0, 2, 2, 1]
Accuracy: 0.9555555555555556
```

ARION SYEMAEL SIAHAAN
225150207111060

model ini mencapai akurasi yang sangat tinggi pada data uji 0.9555555555555556. Hal ini menunjukkan bahwa model berhasil mempelajari pola dalam data dengan baik tanpa overfitting, meskipun akurasi pada data latih mungkin tetap lebih tinggi. Fenomena ini terjadi karena peningkatan jumlah neuron memungkinkan model menangkap pola yang lebih kompleks, sementara batas maksimal error yang rendah (0,01) memastikan model berlatih hingga error minimal, yang akhirnya

meningkatkan performa pada data uji.

3. Ubahlah parameter berikut agar mendapatkan akurasi tertinggi saat melakukan testing menggunakan data uji :

- Hidden neuron
- Max epoch
- Max error

Berapakah nilai akurasi tertinggi yang dapat Anda peroleh? Berapakah nilai masing-masing parameter tersebut?

```
iris = datasets.load_iris()
X = minmax_scale(iris.data)
Y = onehot_enc(iris.target)
X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=.3, random_state=1)
w, ep, mse = bp_fit(X_train, y_train, layer_conf=(4, 10, 3),
learn_rate=.1, max_epoch=5000, max_error=.01, print_per_epoch=25)
print(f'Epochs: {ep}, MSE: {mse}')
predict = bp_predict(X_test, w)
predict = onehot_dec(predict)
y_test = onehot_dec(y_test)
accuracy = accuracy_score(predict, y_test)
print('Output:', predict)
print('True :', y_test)
print('Accuracy:', accuracy)
```

Ubahlah parameter berikut agar mendapatkan akurasi tertinggi saat melakukan testing menggunakan data uji :

- Hidden neuron
- Max epoch
- Max error

Berapakah nilai akurasi tertinggi yang dapat Anda peroleh? Berapakah nilai masing-masing parameter tersebut?

```
iris = datasets.load_iris()
X = minmax_scale(iris.data)
Y = onehot_enc(iris.target)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=.3, random_state=1)
w, ep, mse = bp_fit(X_train, y_train, layer_conf=(4, 10, 3), learn_rate=.1, max_epoch=5000, max_error=.01, print_per_epoch=25)
print(f'Epochs: {ep}, MSE: {mse}')
predict = bp_predict(X_test, w)
predict = onehot_dec(predict)
y_test = onehot_dec(y_test)
accuracy = accuracy_score(predict, y_test)
print('Output:', predict)
print('True :', y_test)
print('Accuracy:', accuracy)
```

```
Epoch 25, MSE: 0.383776096514961
Epoch 50, MSE: 0.2938290580342846
Epoch 75, MSE: 0.23051680773921684
Epoch 100, MSE: 0.1641746565691314
Epoch 125, MSE: 0.12054245285529827
Epoch 150, MSE: 0.09528092568898881
Epoch 175, MSE: 0.08037329840572681
Epoch 200, MSE: 0.07110175933017954
Epoch 225, MSE: 0.06496455009618983
Epoch 250, MSE: 0.060650291850340744
Epoch 275, MSE: 0.057455956978168486
Epoch 300, MSE: 0.05498915006560867
Epoch 325, MSE: 0.05302027463743097
Epoch 350, MSE: 0.051408353869327045
Epoch 375, MSE: 0.05006270656010097
Epoch 400, MSE: 0.04892241143572634
Epoch 425, MSE: 0.04794487414778065
Epoch 450, MSE: 0.04709922028747376
Epoch 475, MSE: 0.04636234110506702
```

ARION SYEMAEL SIAHAAN
225150207111060

```
Epoch 4025, MSE: 0.0386404924222064
Epoch 4050, MSE: 0.03863176668585225
Epoch 4075, MSE: 0.03862299960835085
Epoch 4100, MSE: 0.03861418964465683
Epoch 4125, MSE: 0.03860533535361555
Epoch 4150, MSE: 0.038596435401733666
Epoch 4175, MSE: 0.03858748856647637
Epoch 4200, MSE: 0.03857849373904515
Epoch 4225, MSE: 0.03856944992658601
Epoch 4250, MSE: 0.038560356253793705
Epoch 4275, MSE: 0.03855121196387516
Epoch 4300, MSE: 0.038542016418849724
Epoch 4325, MSE: 0.03853276909916297
Epoch 4350, MSE: 0.03852346960260977
Epoch 4375, MSE: 0.0385141176425609
Epoch 4400, MSE: 0.03850471304550395
Epoch 4425, MSE: 0.03849525574791253
Epoch 4450, MSE: 0.038485745792471576
Epoch 4475, MSE: 0.038476183323689704
Epoch 4500, MSE: 0.038466568582944366
Epoch 4525, MSE: 0.03845690190300361
Epoch 4550, MSE: 0.03844718370208474
Epoch 4575, MSE: 0.038437414477507946
Epoch 4600, MSE: 0.038427594799011144
Epoch 4625, MSE: 0.03841772530179378
Epoch 4650, MSE: 0.03840780667936292
Epoch 4675, MSE: 0.038397839676249546
Epoch 4700, MSE: 0.03838782508066817
Epoch 4725, MSE: 0.03837776371719042
Epoch 4750, MSE: 0.038367656439496044
Epoch 4775, MSE: 0.03835750412327031
Epoch 4800, MSE: 0.038347307659300306
Epoch 4825, MSE: 0.03833706794682804
Epoch 4850, MSE: 0.03832678588720755
Epoch 4875, MSE: 0.0383164623779047
Epoch 4900, MSE: 0.03830609830687769
Epoch 4925, MSE: 0.03829569454736284
Epoch 4950, MSE: 0.0382852519530897
Epoch 4975, MSE: 0.038274771353936386
Epoch 5000, MSE: 0.03826425355203584
Epochs: 5000, MSE: 0.03826425355203584
Output: [0, 1, 1, 0, 2, 1, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 2, 1, 0, 0, 1, 2, 1, 2, 1, 2, 2, 0, 1, 0, 1, 2, 2, 0, 2, 2, 1]
True : [0, 1, 1, 0, 2, 1, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 2, 1, 0, 0, 1, 2, 1, 2, 1, 2, 2, 0, 1, 0, 1, 2, 2, 0, 2, 2, 1]
Accuracy: 1.0
```

ARION SYEMAEL SIAHAAN
225150207111060

Nilai akurasi tertinggi yang diperoleh adalah **1.0**. Nilai masing-masing parameter yang digunakan adalah:

- **Hidden neuron**: 10
- **Max epoch**: 5000
- **Max error**: 0.01

D. Kesimpulan

Overfitting adalah kondisi ketika model terlalu "menghafal" data latih hingga performanya sangat baik di data latih namun buruk pada data baru. Hal ini terjadi karena model terlalu kompleks dan menangkap detail atau noise yang sebenarnya tidak relevan. Untuk mengatasi overfitting, kita bisa menerapkan regularisasi (seperti L1/Lasso Regression atau L2/Ridge Regression), menggunakan dropout pada jaringan saraf, menambah data latih agar model mempelajari pola yang lebih umum, atau mengurangi kompleksitas model dengan memilih parameter yang lebih sedikit atau mengurangi jumlah layer.

Underfitting terjadi saat model terlalu sederhana untuk menangkap pola yang ada dalam data, sehingga hasilnya buruk baik pada data latih maupun data uji. Hal ini umumnya disebabkan oleh model yang kurang kompleks atau data yang belum diolah dengan baik. Untuk mengatasi underfitting, kita dapat menambah kompleksitas model dengan menambah parameter atau layer, memperpanjang waktu pelatihan, melakukan feature engineering untuk menyoroti fitur penting dalam data, atau mencoba algoritma yang lebih sesuai untuk pola dalam data.