LABORATORIUM PEMBELAJARAN ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

BAB : LEARNING VECTOR QUANTIZATION
NAMA : RAIKHAN GEZA ALBURAMA
NIM : 225150207111040
TANGGAL : 11/12/2024
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 LVQ

```
import numpy as np

def lvq_fit(train, target, learning_rate, decay_rate,
max_epochs):
    labels, initial_indices = np.unique(target,
return_index=True)
    weights = train[initial_indices].astype(np.float64)
    remaining_data = [(x, y) for i, (x, y) in
enumerate(zip(train, target)) if i not in
initial_indices]
    train, target = np.array([x for x, _ in
remaining_data]), np.array([y for _, y in
remaining_data])

    epoch = 0
    while epoch < max_epochs:
        for i, x in enumerate(train):
            distances = [np.sum((w - x) ** 2) for w in
weights]
            closest_index = np.argmin(distances)
            sign = 1 if target[i] ==
labels[closest_index] else -1
            weights[closest_index] += sign *
learning_rate * (x - weights[closest_index])

        learning_rate *= decay_rate
        epoch += 1

    return weights, labels
```

b. Fungsi testing LVQ

```
def lvq_predict(X, model):
    center, label = model
    Y = []
    for x in X:
      d = [sum((c - x) ** 2) for c in center]#
      Y.append(label[np.argmin(d)])
    return Y
```

c. Fungsi hitung akurasi

```
def calc_accuracy(a, b):
  s = [1 if a[i] == b[i] else 0 for i in range(len(a))]

  return sum(s) / len(a)
```

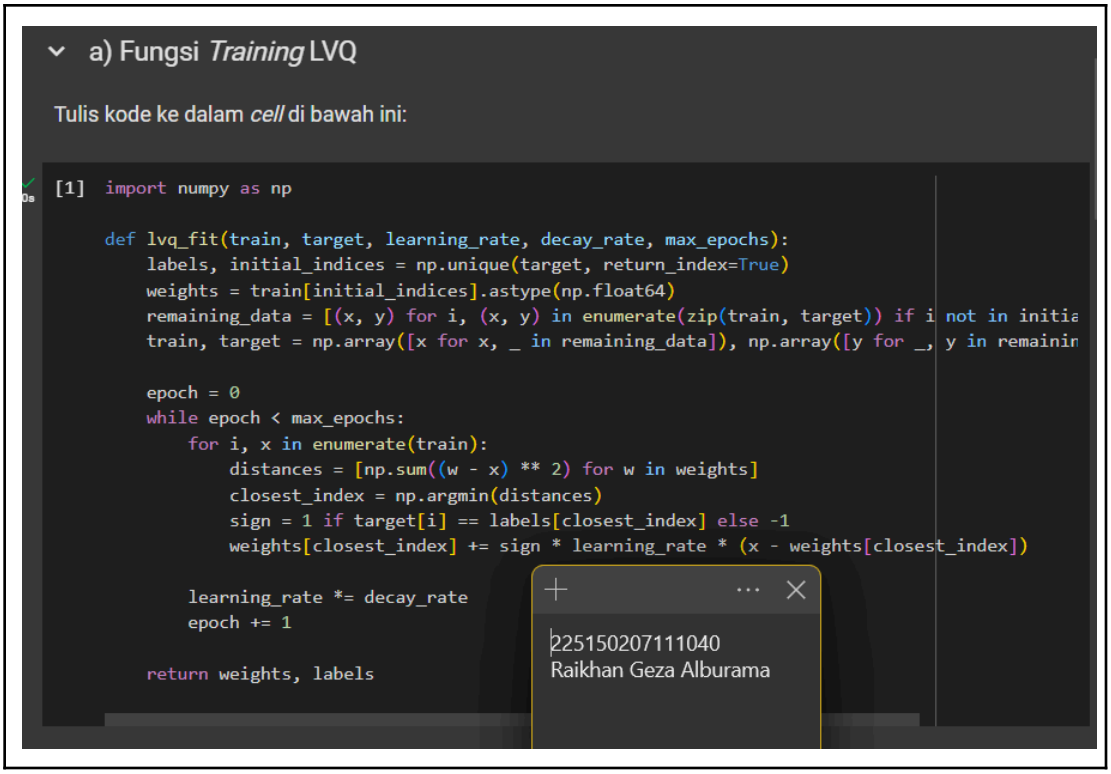
d. Percobaan data acak dengan visualisasi

```
from random import uniform
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
import numpy as np

X, y = make_classification(n_samples=31, n_features=2,
n_redundant=0, n_informative=2, n_classes=3,
n_clusters_per_class=1)
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.2)

model = lvq_fit(X_train, y_train, learning_rate=0.5,
decay_rate=0.8, max_epochs=50)
output = lvq_predict(X_test, model)
accuracy = calc_accuracy(output, y_test)
```

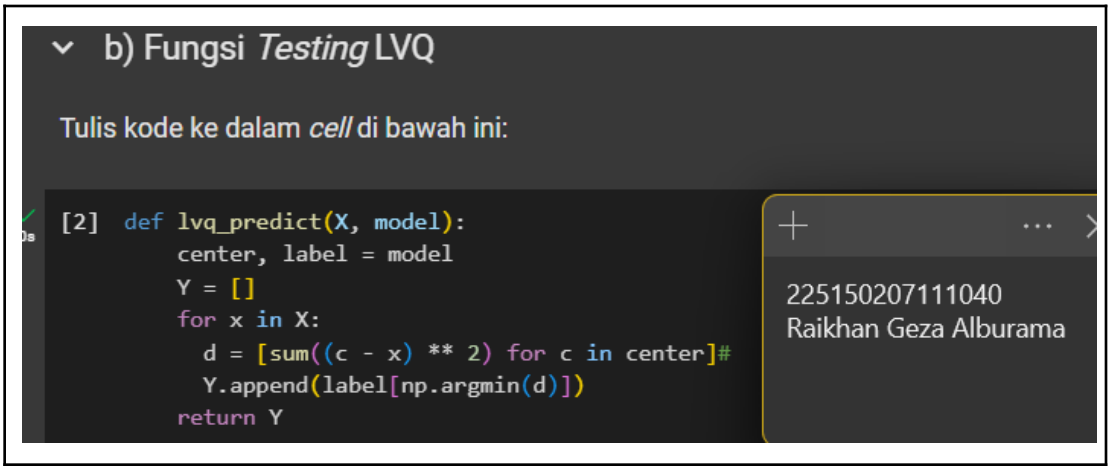
```
print('Accuracy:', accuracy)
colors = 'rgbcmyk'

for x, label in zip(X_train, y_train):
    plt.plot(x[0], x[1], colors[label] + '.')

for center, label in zip(model[0], model[1]):
    plt.plot(center[0], center[1], colors[label] + 'o')

for x, label in zip(X_test, output):
    plt.plot(x[0], x[1], colors[label] + 'x')

plt.show()
```

B. Screenshot

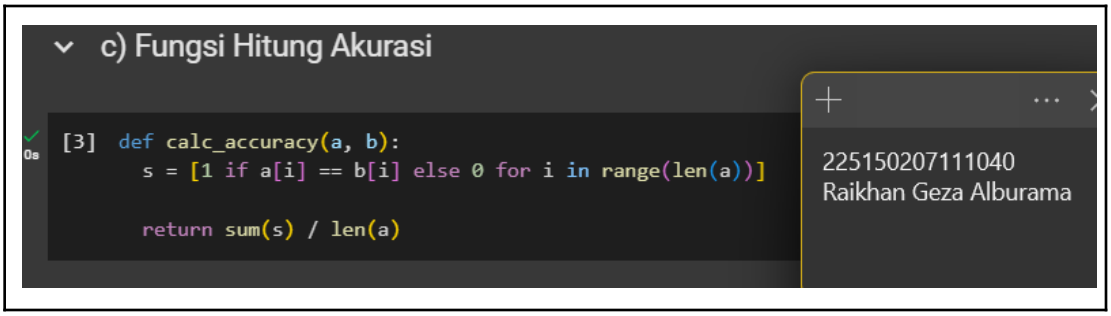
a. Fungsi training LVQ



b. Fungsi testing LVQ



c. Fungsi hitung akurasi



d. Percobaan data acak dengan visualisasi

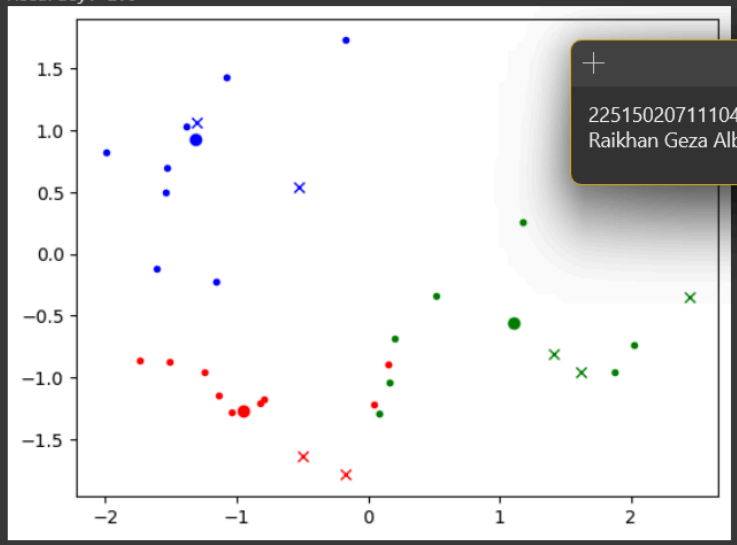
d) Percobaan LVQ

Tulis kode ke dalam *cell* di bawah ini:

```
from random import uniform
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
import numpy as np

X, y = make_classification(n_samples=31, n_features=2, n_redundant=0, n_informative=2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = lvq_fit(X_train, y_train, learning_rate=0.5, decay_rate=0.8, max_epochs=50)
output = lvq_predict(X_test, model)
accuracy = calc_accuracy(output, y_test)

print('Accuracy:', accuracy)
colors = 'rgbcmyk'

for x, label in zip(X_train, y_train):
    plt.plot(x[0], x[1], colors[label] + '.')

for center, label in zip(model[0], model[1]):
    plt.plot(center[0], center[1], colors[label] + 'o')

for x, label in zip(X_test, output):
    plt.plot(x[0], x[1], colors[label] + 'x')

plt.show()
```

225150207111040
Raikhan Geza Alburama

```
Accuracy: 1.0
```



225150207111040
Raikhan Geza Alburama

C. Analisis

Jalankan kode d beberapa kali hingga didapat akurasi dari 1. Amati dan analisis di mana terjadi error.

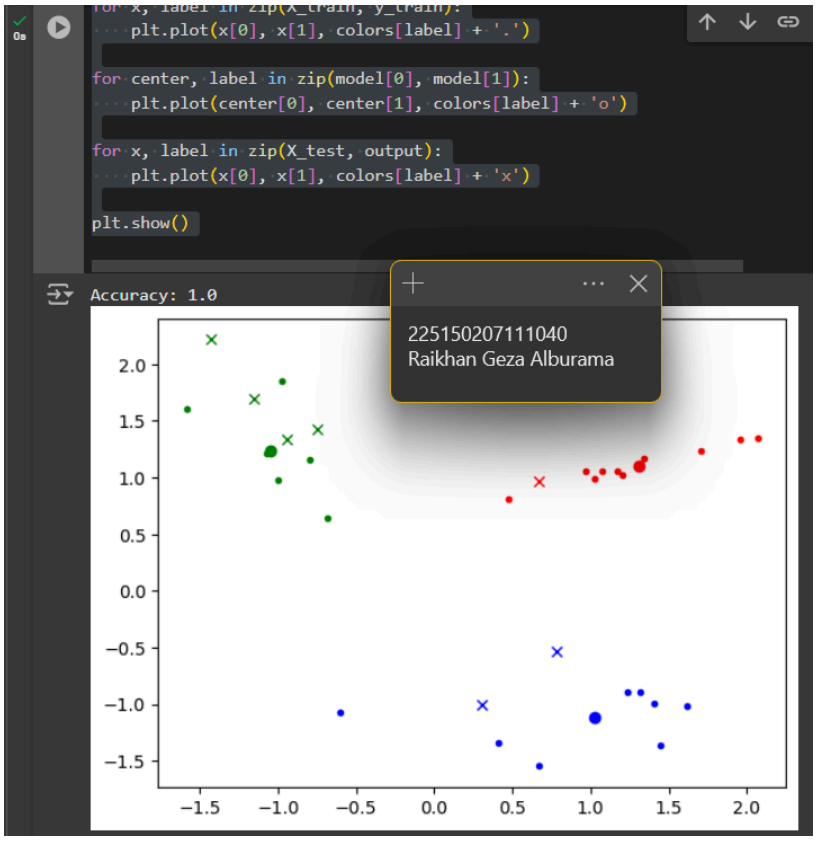
Jawaban :

Untuk mencapai akurasi 1 pada model LVQ memerlukan pengaturan parameter yang tepat, terutama `learning_rate`, `decay_rate`, dan jumlah epoch. `Learning_rate` yang terlalu tinggi atau `decay_rate` yang terlalu besar dapat menghambat konvergensi yang optimal. Jumlah epoch yang mencukupi juga membantu model belajar lebih efektif.

Selain itu, distribusi data antar kelas harus seimbang dan tidak terlalu berdekatan untuk memudahkan pemisahan kelas. Beberapa kali percobaan dengan penyesuaian parameter dapat mendekatkan akurasi ke 1. Jika tetap sulit, ini mungkin menunjukkan keterbatasan LVQ pada dataset tersebut atau kebutuhan model yang lebih kompleks.

```
from random import uniform
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
import numpy as np

X, y = make_classification(n_samples=31, n_features=2,
n_redundant=0, n_informative=2, n_classes=3,
n_clusters_per_class=1)
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.2)

model = lvq_fit(X_train, y_train, learning_rate=0.5,
decay_rate=0.8, max_epochs=50)
output = lvq_predict(X_test, model)
accuracy = calc_accuracy(output, y_test)

print('Accuracy:', accuracy)
colors = 'rgbcmyk'
```

```
for x, label in zip(X_train, y_train):
    plt.plot(x[0], x[1], colors[label] + '.')

for center, label in zip(model[0], model[1]):
    plt.plot(center[0], center[1], colors[label] + 'o')

for x, label in zip(X_test, output):
    plt.plot(x[0], x[1], colors[label] + 'x')

plt.show()
```



D. Kesimpulan

Learning Vector Quantization merupakan salah satu metode klasifikasi JST. LVQ memiliki arsitektur JST yang mirip dengan SOM. Namun, terdapat beberapa perbedaan dalam algoritmanya. Jika SOM bertujuan untuk melakukan klasterisasi, LVQ bertujuan untuk mengklasifikasi setiap unit output dan merepresentasikan ke dalam bentuk kelas. Terdapat perbedaan juga pada proses training yaitu LVD menghitung nilai output dengan jarak yang terdekat dengan nilai input dalam mengupdate bobot.

LVQ memiliki perbedaan dengan JST lainnya dimana LVQ dapat menyelesaikan masalah yang lebih kompleks contohnya data-data yang bersifat non-lieary separable. Dalam proses training LVQ juga terdapat perbedaan dengan JST lain dimana data training digunakan sebagai bobot dan nilai lainnya digunakan untuk testing.Selain itu proses update bobot oleh LVQ hanya dilakukan untuk neuron pemenangnya saja,apabila kelas dari input sama dengan kelas dari output neuron,maka bobot neuron pemenang diupdate sehingga mendekati input dan jika kelas dari input berbeda dengan kelas dari output neuron pemenang,bobot neuron pemenang di update sehingga menjauhi input.LVQ juga merupakan algoritma yang berbasis competitive learning atau memiliki competitive layer.