LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

BAB : LEARNING VECTOR QUANTIZATION
NAMA : ARION SYEMAEL SIAHAAN
NIM : 225150207111060
TANGGAL : 13/11/2024
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 LVQ

```
import numpy as np

def lvq_fit(train, target, lrate, b, max_epoch):
  label, train_idx = np.unique(target, return_index=True)
  weight = train[train_idx].astype(np.float64)

  noise = np.random.normal(0, 0.1, weight.shape)
  weight += noise

  train = np.delete(train, train_idx, axis=0)
  target = np.delete(target, train_idx, axis=0)

  epoch = 0
  while epoch < max_epoch:
    for i, x in enumerate(train):
      distance = [sum((w - x) ** 2) for w in weight]
      min_idx = np.argmin(distance)
      sign = 1 if target[i] == label[min_idx] else -1
      weight[min_idx] += sign * lrate * (x - weight[min_idx])
    lrate *= b
    epoch += 1

  return weight, label
```

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 LVQ

```
from random import uniform
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs, make_classification

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, lrate=0.01, b=0.8, max_epoch=100)
output = lvq_predict(X_test, model)
accuracy = calc_accuracy(output, y_test)
colors = 'rgbcmyk'
print('Accuracy:', accuracy)

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')
```

B. Screenshot

a. Fungsi Training LVQ

a) Fungsi *Training* LVQ

Tulis kode ke dalam *cell* di bawah ini:

```python
import numpy as np

def lvq_fit(train, target, lrate, b, max_epoch):
  label, train_idx = np.unique(target, return_index=True)
  weight = train[train_idx].astype(np.float64)

  noise = np.random.normal(0, 0.1, weight.shape)
  weight += noise

  train = np.delete(train, train_idx, axis=0)
  target = np.delete(target, train_idx, axis=0)

  epoch = 0
  while epoch < max_epoch:
    for i, x in enumerate(train):
      distance = [sum((w - x) ** 2) for w in weight]
      min_idx = np.argmin(distance)
      sign = 1 if target[i] == label[min_idx] else -1
      weight[min_idx] += sign * lrate * (x - weight[min_idx])
    lrate *= b
    epoch += 1

  return weight, label
```

ARION SYEMAEL SIAHAAN
225150207111060

b. Fungsi Testing LVQ

b) Fungsi *Testing* LVQ

Tulis kode ke dalam *cell* di bawah ini:

```python
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
```

ARION SYEMAEL SIAHAAN
225150207111060

c. Fungsi Hitung Akurasi

c) Fungsi Hitung Akurasi

```python
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)
```

ARION SYEMAEL SIAHAAN
225150207111060

d. Percobaan LVQ

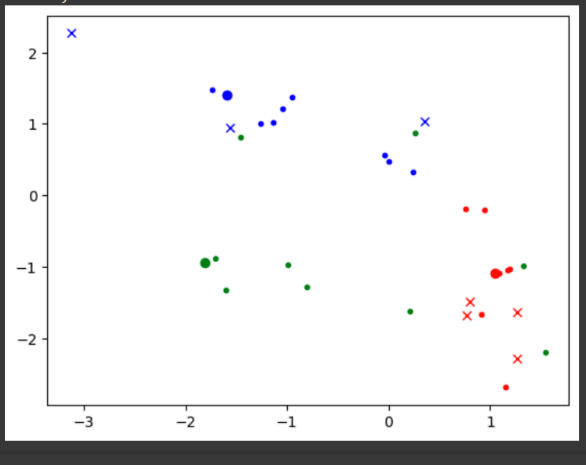
d) Percobaan LVQ

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_blobs, make_classification

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, lrate=0.01, b=0.8, max_epoch=100)
output = lvq_predict(X_test, model)
accuracy = calc_accuracy(output, y_test)
colors = 'rgbcmyk'
print('Accuracy:', accuracy)

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')
```

ARION SYEMAEL SIAHAAN
225150207111060

```
Accuracy: 0.8571428571428571
```



ARION SYEMAEL SIAHAAN
225150207111060

C. Analisis

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

Percobaan 1

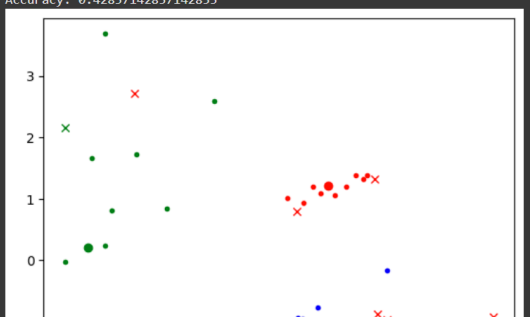
```
[28] from random import uniform
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
     from sklearn.datasets import make_blobs, make_classification

     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, lrate=0.2, b=0.8, max_epoch=100)
     output = lvq_predict(X_test, model)
     accuracy = calc_accuracy(output, y_test)
     colors = 'rgbcmyk'
     print('Accuracy:', accuracy)

     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')
```

```
Accuracy: 1.0
```

Percobaan 2

```
from random import uniform
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs, make_classification

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, lrate=0.05, b=0.08, max_epoch=100)
output = lvq_predict(X_test, model)
accuracy = calc_accuracy(output, y_test)
colors = 'rgbcmyk'
print('Accuracy:', accuracy)

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')
```

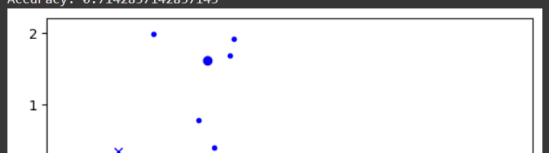
```
Accuracy: 0.42857142857142855
```



Percobaan 3

```
from random import uniform
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs, make_classification

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, lrate=0.02, b=0.8, max_epoch=100)
output = lvq_predict(X_test, model)
accuracy = calc_accuracy(output, y_test)
colors = 'rgbcmyk'
print('Accuracy:', accuracy)

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')
```

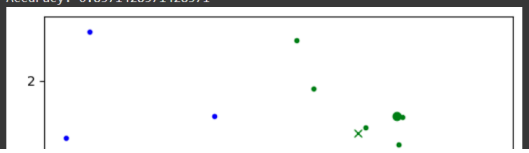
```
Accuracy: 0.7142857142857143
```



Percobaan 4

```
[31] from random import uniform
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs, make_classification

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, lrate=0.01, b=0.8, max_epoch=100)
output = lvq_predict(X_test, model)
accuracy = calc_accuracy(output, y_test)
colors = 'rgbcmyk'
print('Accuracy:', accuracy)

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')
```

```
Accuracy: 0.8571428571428571
```



Hasil variasi skor akurasi dalam percobaan dengan model *Learning Vector Quantization* (LVQ) dapat disimpulkan sebagai dampak dari beberapa faktor. Pertama, inisialisasi bobot acak pada model dan pembagian data yang bersifat acak dapat menyebabkan konvergensi yang berbeda pada setiap percobaan. Selain itu, sensitivitas model LVQ terhadap parameter, seperti *learning rate* dan faktor pemadaman, dapat memperbesar variasi hasil.

D. Kesimpulan

Learning Vector Quantization (LVQ) dan Self-Organizing Map (SOM) memiliki perbedaan utama dalam tujuan dan metode pembelajarannya. LVQ adalah metode klasifikasi berbasis pembelajaran terawasi, di mana data sudah memiliki label atau kelas tertentu. Bobot neuron pada LVQ disesuaikan untuk mendekati data sesuai kelas yang sudah ditentukan sebelumnya, membuatnya efektif untuk tugas klasifikasi yang spesifik. Sebaliknya, SOM adalah metode pembelajaran tak terawasi yang berfungsi untuk pemetaan data dan reduksi dimensi. Pada SOM, bobot neuron berkompetisi untuk menyesuaikan diri dengan data tanpa label, menghasilkan peta topologi yang mengelompokkan data secara otomatis berdasarkan kemiripannya, bukan berdasarkan kelas.

Dibandingkan dengan jenis Jaringan Syaraf Tiruan (JST) lain yang umum digunakan untuk klasifikasi seperti Multi-Layer Perceptron (MLP), LVQ berbeda dalam metode pembelajaran dan struktur komputasi. LVQ menggunakan pembelajaran kompetitif berbasis contoh, di mana neuron-neuron bertindak sebagai prototipe dari setiap kelas, dan bobotnya diperbarui berdasarkan kedekatan dengan titik data dari kelas tersebut, yang membuatnya lebih cepat dibandingkan JST lain yang menggunakan propagasi balik error. Sebaliknya, JST seperti MLP menggunakan pembelajaran berbasis error (misalnya, backpropagation), yang memerlukan perhitungan error pada setiap layer untuk mengoptimalkan bobot secara global melalui gradient descent. Selain itu, LVQ lebih cocok untuk pemisahan kelas yang linier, sedangkan MLP dengan hidden layer mampu menangani pemisahan kelas non-linier, memungkinkan klasifikasi yang lebih kompleks pada JST berbasis backpropagation.