



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

BAB : SELF ORGANIZING MAPS

NAMA : RAIKHAN GEZA ALBURAMA
NIM : 225150207111040
TANGGAL : 11/11/2024
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 self-organizing maps

```
import numpy as np
import matplotlib.pyplot as plt

def som(X, lrate, decay_rate, max_epoch, n_cluster):
    centroids = np.random.uniform(size=(n_cluster,
len(X[0])))
    print('Centroid/Weight awal:\n', centroids)

    epoch = 0
    labels = []

    while epoch < max_epoch:
        for x in X:
            distances = [np.sum((w - x) ** 2) for w in
centroids]
            closest_idx = np.argmin(distances)
            centroids[closest_idx] += lrate * (x -
centroids[closest_idx])

            lrate *= decay_rate
            epoch += 1

        for x in X:
            distances = [np.sum((w - x) ** 2) for w in
centroids]
            closest_idx = np.argmin(distances)
            labels.append(closest_idx)

    return centroids, labels

def draw(X, labels, centroids):
```

```

        colors = 'rgbcmk'
        for x, label in zip(X, labels):
            plt.plot(x[0], x[1], colors[label %
len(colors)] + '.')
        plt.plot(centroids[:, 0], centroids[:, 1], 'kx')
        plt.show()

```

51

b. Klasterisasi

```

from sklearn.datasets import make_blobs
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt

X, target = make_blobs(n_samples=30, n_features=2,
centers=3, random_state=3)

centroids, labels = som(X, lrate=0.5, decay_rate=0.5,
max_epoch=100, n_cluster=3)

silhouette = silhouette_score(X, labels)
print('Silhouette score:', silhouette)

draw(X, labels, centroids)

```

B. Screenshot

a. Fungsi self-organizing maps

▼ a) Fungsi *self-organizing maps*

Tulis kode ke dalam *cell* di bawah ini:

```
[2] import numpy as np
import matplotlib.pyplot as plt

def som(X, lr_rate, decay_rate, max_epoch, n_cluster):
    centroids = np.random.uniform(size=(n_cluster, len(X[0])))
    print('Centroid/Weight awal:\n', centroids)

    epoch = 0
    labels = []

    while epoch < max_epoch:
        for x in X:
            distances = [np.sum((w - x) ** 2) for w in centroids]
            closest_idx = np.argmin(distances)
            centroids[closest_idx] += lr_rate * (x - centroids[closest_idx])

            lr_rate *= decay_rate
            epoch += 1

        for x in X:
            distances = [np.sum((w - x) ** 2) for w in centroids]
            closest_idx = np.argmin(distances)
            labels.append(closest_idx)

    return centroids, labels

def draw(X, labels, centroids):
    colors = 'rgbcmyk'
    for x, label in zip(X, labels):
        plt.plot(x[0], x[1], colors[label % len(colors)] + '.')
    plt.plot(centroids[:, 0], centroids[:, 1], 'kx')
    plt.show()
```

Completed at 17:53:04

b. Klasterisasi

▼ b) Klasterisasi

Tulis kode ke dalam *cell* di bawah ini:

```
from sklearn.datasets import make_blobs
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt

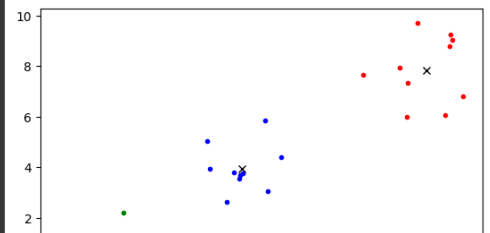
X, target = make_blobs(n_samples=30, n_features=2, centers=3, random_state=3)

centroids, labels = som(X, lr_rate=0.5, decay_rate=0.5, max_epoch=100, n_cluster=3)

silhouette = silhouette_score(X, labels)
print('Silhouette score:', silhouette)

draw(X, labels, centroids)
```

Centroid/Weight awal:
[[0.64992752 0.63179637]
[0.02007265 0.04931825]
[0.14825353 0.46691936]]
Silhouette score: 0.7215682462170806



The scatter plot displays 30 data points in a 2D space. The points are colored based on their cluster assignment: red, blue, and green. Three centroids are marked with black 'x' symbols. The x-axis ranges from approximately 0 to 10, and the y-axis ranges from approximately 2 to 10. The clusters are well-separated, indicating good clustering performance.

C. Analisis

1. Ubah parameter pada kode b menjadi learning rate = 0,05 dan epoch maksimum = 3 lalu jalankan program. Amati gambar hasil klasterisasi dan nilai silhouette yang didapatkan.

Jawaban:

Pada soal tersebut setelah mengubah learning rate menjadi 0.05 dan epoch max menjadi 3, hasil silhouette score-nya yang semula 0.72 berubah menjadi lebih rendah atau turun dari hasil sebelum diubah (pada screenshot dibawah berubah menjadi 0.35) Hal tersebut terjadi karena perubahan learning rate dan epoch membuat keakurasian hasil menurun dan penyebaran centroid (tanda x) tidak merata.

Analisis

1. Ubah parameter pada kode b menjadi learning rate = 0,05 dan epoch maksimum = 3 lalu jalankan program. Amati gambar hasil klasterisasi dan nilai silhouette yang didapatkan.

```
from sklearn.datasets import make_blobs
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt

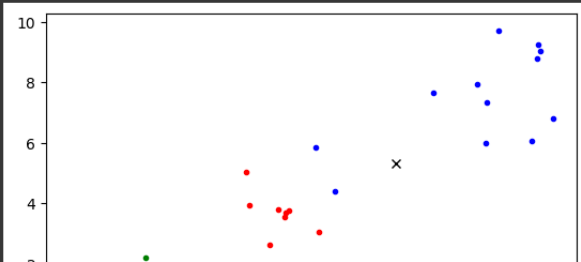
X, target = make_blobs(n_samples=30, n_features=2, centers=3, random_state=3)

centroids, labels = som(X, lrate=0.05, decay_rate=0.5, max_epoch=3, n_cluster=3)

silhouette = silhouette_score(X, labels)
print('Silhouette score:', silhouette)

draw(X, labels, centroids)
```

```
Centroid/Weight awal:
[[0.93427982 0.55274692]
 [0.35747694 0.0034162 ]
 [0.97933164 0.94112942]]
Silhouette score: 0.6157837764356677
```



D. Kesimpulan

Self-organizing maps (SOM), atau yang disebut juga dengan topology-preserving maps adalah sebuah algoritma klasterisasi (clustering) sehingga termasuk pada algoritma unsupervised learning. Centroid pada setiap cluster di SOM ini didapat dari nilai-nilai bobot. Banyaknya centroid harus ditentukan lebih dulu sebelum proses clustering dimulai. Selama proses clustering berjalan, centroid dengan posisi terdekat dengan suatu data input akan dipilih sebagai “pemenang”. Centroid tersebut, beserta centroid di sekitarnya yang berada pada radius R , kemudian akan memperbarui posisinya untuk lebih mendekat ke input tersebut.

Alur kerja kodingan pada poin Fungsi Self Organizing-Maps yaitu yang pertama melakukan import numpy. Selanjutnya mengidentifikasi parameter SOM

yaitu data, learning rate, beta, max epoch, dan jumlah clusternya. Kemudian menginisiasi nilai bobot yaitu centroid menggunakan uniform. selanjutnya inisiasi epoch = 0 dan labelnya. Kemudian proses selanjutnya yaitu menghitung jarak antar centroid data. dan dilanjut menentukan neuron pemenang. Setelah itu dilakukan update centroid dengan jarak terkecil dan proses tersebut nantinya akan diulang sebanyak jumlah datanya. Baru kemudian learning ratenya diupdate. Selanjutnya yaitu menentukan labeling atau menentuka masuk kedalam cluster mana. Baru kemudian centroid dan labelnya direturn. Proses yang terakhir yaitu untuk melihat hasil akhir yaitu posisi centroid dimana dan datanya berupa apa