BLM6103 – Olasılık, Rastgele Değişkenler ve Stokastik Süreçler

2023-2024 Güz Yarıyılı

Ödev – 2

- "Olasılık, R.D. ve Stokastik Prosesler" isimli Google Classroom grubundan güncel duyuruları takip etmek için ackaraca@yildiz.edu.tr mail adresine mail atabilirsiniz.
- Bu ödev, toplam 6 adımdan oluşmaktadır. Soru çözümlerini bu kapak sayfasının arkasına ekleyerek pdf haline dönüştürünüz. Tek bir pdf dosyası halinde mail yukarıdaki mail adresine gönderiniz.
- Dönem boyunca 3 ödev, 1 arasınav ve 1 final yapılacaktır. Bu sebeple, ödevler oldukça önemlidir. Eksiksiz ve kopyasız bir şekilde yapmanız önerilmektedir.
- Birbirinin tamamen aynısı olan çözümlerde hak edilen puan kişi sayısına bölünerek hesaplanacaktır. Bu sebeple, kendi çözümlerinizi paylaşmayınız.
- Bilgisayar üzerinden çözülecek sorular için kod ve ekran çıktıları paylaşılmalıdır. Derste MATLAB üzerinden kodlar paylaşılsa da siz ödevleri farklı programlama dillerinde yapabilirsiniz.
- Soruların tamamı diğer sayfada paylaşılmıştır.
- SON GÖNDERİM TARİHİ 15 ARALIK 2023 OLUP ÖDEVLER GOOGLE CLASSROOM ÜZERİNDEN YÜKLENECEKTİR.

NORMAL DAĞILIMLI RASTGELE VEKTÖR ÖZELLİKLERİYLE SINIFLANDIRMA

İris veri kümesi makine öğrenmesi yöntemlerinin anlaşılması ve geliştirilmesi tarafında faydalı bir kaynak olarak ortaya çıkmaktadır. Veri kümesi farklı çiçeklerin petal ve sepal yapraklarının genişlik ve yükseklikleri olmak üzere her çiçek için 4 özniteliğe karşılık hangi sınıfa ait olduğunu göstermektedir.

Bu ödevde Mahalanobis uzaklığı kullanan bir sınıflandırıcı geliştirilecektir. Beklenen adımlar aşağıda verilmektedir:

1. Veri içerisindeki her sınıftaki örneklerin %80'i eğitim ve geri kalanlar test fazına ayırılır.
2. Eğitim verilerini kullanarak özniteliklerin farklı kombinasyonları ile scatterplot çizilir ve ilinti durumu hesaplanmadan yorumlanır.
3. Eğitim verilerini kullanarak her bir sınıf için ortalama vektörü ve kovaryans matrisi hesaplanır. Bu hesaplar ödev sahibi tarafından kendi kodu ile yapılmalıdır.
4. Test verisindeki her bir örnek elde edilen ortalama vektörleri ve kovaryans matrislerini kullanarak Mahalanobis uzaklığı hesaplanır. En düşük uzaklığa sahip olan sınıf tahmin edilen sınıf olarak belirlenir.
5. 4'teki adım Öklid uzaklığı ile hesaplanır ve performans sınıflandırma başarım metrikleriyle (doğruluk, kesinlik vb.) karşılaştırılır.
6. Doğru ve hatalı örnekler çizim üzerinde görselleştirilerek sonuçlar karşılaştırılır.

Yukarıdaki isterler rapor için minimum çıktılardır. Ayrıntılı içerik paylaşmanın herhangi bir sakıncası yoktur.

Başarılar dilerim.

Dr. Öğr. Üyesi Ali Can KARACA

Normal Dağılımlı Rastgele Vektör Özellikleriyle Sınıflandırma

BLM6103 – Olasılık, Rastgele Değişkenler ve Stokastik Süreçler

Recep Furkan Koçyiğit

22501048

furkan.kocyigit@std.yildiz.edu.tr

Bilgisayar Mühendisliği Bölümü

Elektrik Elektronik Fakültesi, Yıldız Teknik Üniversitesi

Özet

Bu ödevde, İris veri kümesinde makine öğrenmesi yöntemlerinin anlaşılması ve geliştirilmesi için Mahalanobis ve Öklid uzaklığını kullanan çiçeklerin petal ve sepal yapraklarının genişlik ve yükseklikleri olmak üzere, her çiçek için 4 özniteliğe karşılık hangi sınıfa ait olduğunu göstermektedir. Sistemin geçekleştirilmesi için programlama dili olarak Python kullanılmıştır.

Giriş

Ödev kapsamında Iris veri kümesi kullanılmıştır. Bu veri kümesindeki verilerin %80' i eğitim ve %20' test için ayrılmıştır. Eğitim verilerinin öz niteliklerinin farklı kombinasyonları ile nokta grafikler çizdirilmiş ve ilintilerine bakılmıştır. Daha sonra, eğitim verilerini kullanarak her bir sınıf için ortalama vektörleri ve kovaryans matrisleri hesaplanmıştır. Test verisindeki her bir örnek için Mahalanobis ve Öklid uzaklıkları kullanılarak en düşük uzaklık bulunmuştur. Bulunan bu değer tahmin edilen sınıfı göstermektedir. Tüm test verisi için doğru ve yanlış tahmin edilen değerler görselleştirilmiştir.

Yöntem

Bu sistem aşağıdaki ana modüllerden oluşmaktadır:

1. **Veri Kümesinin Yüklenmesi:** Iris veri kümesinin sisteme yüklenmesi.
2. **Verinin Kümesinin Analizi:** Veride 5 adet özellik bulunmaktadır. Bunlardan 1 adeti çiçeğin sınıfını göstermektedir. Veri kümesinin histogramı çıkarılmıştır ve veriye baktığımızda sınıfların eşit dağıldığı görülmektedir.
3. **Veri Kümesinin Ön İşlenmesi:** Sınıf bilgisine encode işlemi ve kopya veri varsa silme işlemi yapılmıştır.
4. **Eğitim Verisinin Özelliklerinin Nokta Grafiğinin Çizdirilmesi:** Eğitim veri kümesi için özelliklerin kendi aralarındaki ilintisine bakmak için nokta grafikleri çıkarılmıştır.
5. **Ortalama Vektörün Hesaplanması:** Veri kümesindeki her sınıf için özelliklerin ortalama vektörleri çıkarılmıştır. E
6. **Kovaryans Matrisinin Hesaplanması:** Veri kümesindeki her sınıf için kovaryans matris oluşturulmuştur. Şu formül kullanılmıştır:

$$S = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})'$$

7. **Mahalanobis Uzaklığının Bulunması**: Test kümesindeki her bir örnek için Mahalanobis uzaklığı şu formülle bulunmuştur:

$$d_M(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}$$

Burada x test verisindeki satırdaki özelliklere ait değerleri, y ise ortalama vektörünü, S ise kovaryans matrisini temsil etmektedir.

8. **Öklid Uzaklığının Bulunması**: Test kümesindeki her bir örnek için Öklid uzaklığı şu formülle bulunmuştur:

$$d_{L2}(x,y) = \sqrt{(x-y)^T(x-y)}$$

Burada x test verisindeki satırdaki özelliklere ait değerleri, y ise ortalama vektörünü temsil etmektedir.

9. **Mahalanobis ile Tahmin:** Test verisindeki her bir örnek için minimum uzaklıklar Mahalanobis ile hesaplanır ve gerçek değerleriyle karşılaştırılır.
10. **Öklid ile Tahmin:** Test verisindeki her bir örnek için minimum uzaklıklar Öklid ile hesaplanır ve gerçek değerleriyle karşılaştırılır.

Uygulama

Iris verisindeki bilgilere baktığımızda şu bilgileri görmekteyiz:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal.length  150 non-null    float64
 1   sepal.width   150 non-null    float64
 2   petal.length  150 non-null    float64
 3   petal.width   150 non-null    float64
 4   variety       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

150 adet veri olduğu ve veri özellikleri olarak 5 adet özellik olduğu görülmektedir. Bu özellikler şu şekildedir:

1. **Sepal length:** Sepal yaprağının uzunluğu
2. **Sepal width:** Sepal yaprağının uzunluğu
3. **Petal length**: Petal yaprağının uzunluğu
4. **Petal width:** Petal yaprağının uzunluğu
5. **Variety:** Tür

Veri kümesine baktığımızda herhangi bir kopya değer olmadığı görülmektedir. Aşağıda veri kümesine ait örnek bir kısım gösterilmiştir:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

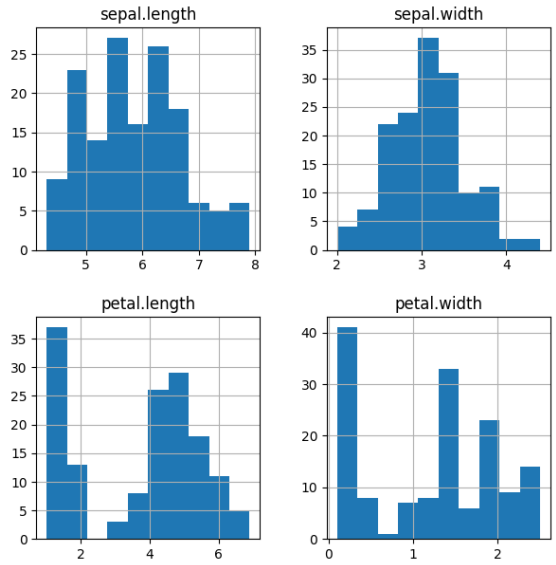
Veri kümesindeki tür dağılımlarına baktığımızda her sınıf için eşit bir dağılım olduğu görülmektedir.

```
variety
Setosa        50
Versicolor    50
Virginica     50
Name: count, dtype: int64
```

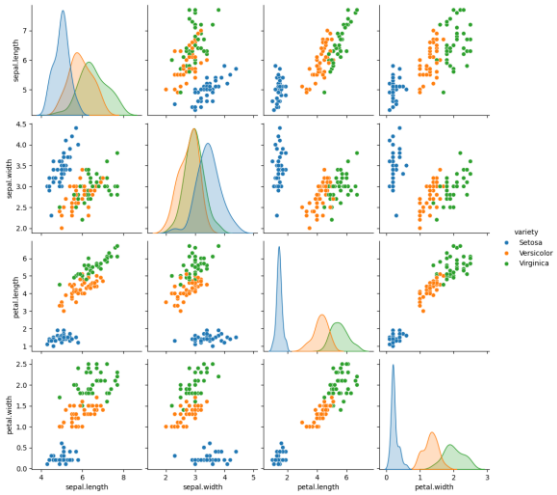
Veri kümesinin sayısal özellikleri ise şu şekildedir:

	count	mean	std	min	25%	50%	75%	max
sepal.length	150.0	5.843333	0.828066	4.3	5.1	5.80	6.4	7.9
sepal.width	150.0	3.057333	0.435866	2.0	2.8	3.00	3.3	4.4
petal.length	150.0	3.758000	1.765298	1.0	1.6	4.35	5.1	6.9
petal.width	150.0	1.199333	0.762238	0.1	0.3	1.30	1.8	2.5

Veri kümesindeki özelliklerin histogramı ise şu şekildedir:



Sepal yapraklarındaki genişlik ve uzunluk dağılımlarının ortalamada biriktiği alt ve üst değerlerde daha düşük değerler aldığı görülmektedir. Eğitim kümesindeki özelliklerin nokta grafikleri şu şekildedir:



Bu grafiğe bakarak Sepal ve Petal yaprak uzunluklarının Versicolor ve Virgina türlerinde ilintili olduğu söylenebilir. Ayrıca Petal yaprak genişlik ve uzunluklarının da ilintili olduğu söylenebilir. Sepal yapraklarının genişliği ve Petal yapraklarının genişlik ve uzunlukları arasında da Versicolor ve Virgina türleri için ilinti oladuğu söylenebilir.

Her bir sınıf için ortalama vektörleri şöyle bulunmuştur:

Setosa = [4.99, 3.45, 1.45, 0.25]

Versicolor = [5.92, 2.77, 4.24, 1.32]

Virginia = [6.53, 2.97, 5.52, 2.00]

Kovaryans matrisleri ise şu şekildedir:

Setosa =
[[0.12, 0.11, 0.02, 0.01],

[0.11, 0.16, 0.01, 0.01],

[0.02, 0.01, 0.03 , 0.01],

[0.01, 0.01, 0.01, 0.01]]

Versicolor =

[[0.29, 0.10, 0.19, 0.06],

[0.10, 0.10, 0.09, 0.05],

[0.19, 0.09, 0.23, 0.08],

[0.06, 0.05, 0.08, 0.04]]

Virginia =

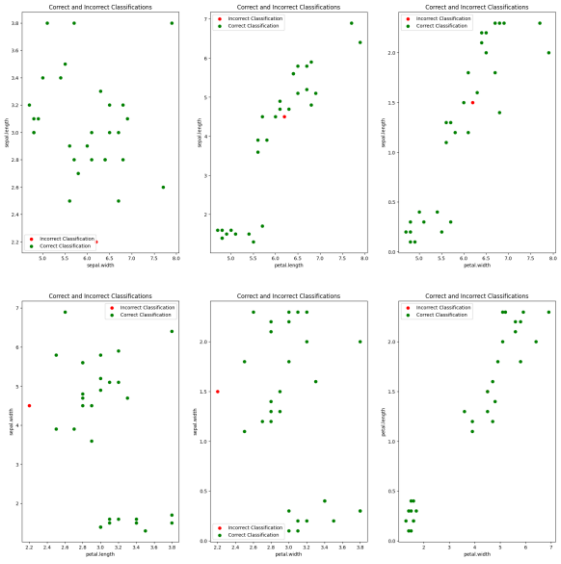
[[0.42 , 0.10, 0.31, 0.05],

[0.10, 0.10 , 0.09, 0.06],

[0.31, 0.09, 0.29, 0.051],

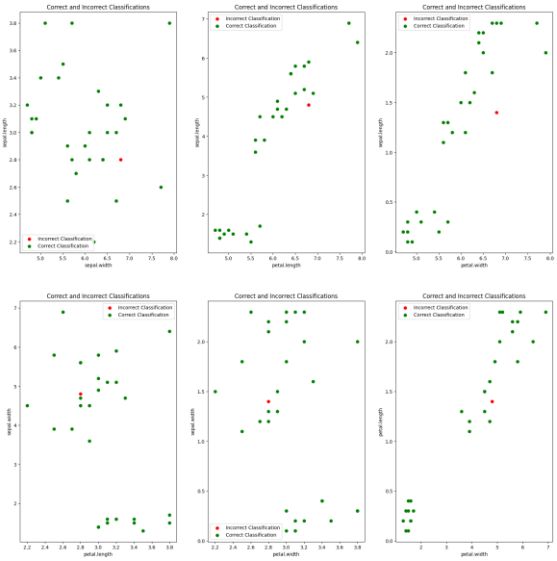
[0.05, 0.06, 0.051, 0.08]]

Mahalanobis ile yapılan sınıflandırmada bir örnek hatalı bulunmuştur. Özelliklere göre doğru tahmin edilenler yeşil yanlış tahmin edilenler kırmızı ile işaretlenmiştir.



Yanlış tahmin edilen örnekler için özellikler şu şekildedir: [6.2, 2.2, 4.5, 1.5, Versicolor]. Başarı oranı ise %96,67 çıkmıştır.

Öklid ile yapılan tahmin için de bir örnek başarısız tahmin edilmiştir.



Öklid için de başarı oranı %96,67'dir. Yanlış tahmin edilen örnekler için özellikler şu şekildedir: [6.8, 2.8, 4.8, 1.4, Versicolor]

Her iki uzaklık için de yanlış örneği Virginia olarak tahmin ettiği görülmektedir. Bunun sebebi her iki sınıfa ait özelliklere ait değerlerin yakın olmasıdır.

Sonuç

Bu çalışmada Mahalanobis ve Öklid uzaklıkları kullanılarak bir sınıflandırma probleminin çözümünün sağlanmasıdır. Bu işlem için her sınıfa ait ortalama vektörleri ve kovaryans matrisleri kullanılmıştır. Her iki uzaklıkla yapılan tahminler için de tek bir örnekte sınıfın yanlış tahmin edildiği görülmektedir.

Kaynak Kod

```
# %%

import pandas as pd

import seaborn as sn

import matplotlib.pyplot as plt

from sklearn.model_selection import
train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy_score
```

```
import numpy as np

# %%
df = pd.read_csv("iris.csv")

# %%
df.info()

# %%
df.drop_duplicates()

# %%
df.head()

# %%
df.variety.value_counts()

# %%
df.describe().T

# %%
df.hist(figsize=(7, 7))
plt.show()

# %%
le = LabelEncoder()
train, test = train_test_split(df, test_size=0.2,
random_state=42)

X_train = train.iloc[:, :-1]
y_train = le.fit_transform(train.variety)
X_test = test.iloc[:, :-1]
y_test = le.fit_transform(test.variety)

# %%
sn.pairplot(train, hue='variety')
plt.show()

# %%
def get_mean_vector(df):
    columns = df.columns
    mean_vector = []

    for i in range(len(columns)):
        mean_vector.append(sum(df[columns[i]]) /
len(df[columns[i]]))

    return np.array(mean_vector)

# %%
def get_covariance_matrix(df):
    columns = df.columns
    covariance_matrix = []
    mean_vector = get_mean_vector(df)

    for i in range(len(columns)):
        covariance_vector = []
        for j in range(len(columns)):
            Ex = df[columns[i]] - mean_vector[i]
            Ey = df[columns[j]] - mean_vector[j]
            covariance_vector.append(sum(Ex * Ey) /
len(df[columns[i]]))
        covariance_matrix.append(covariance_vector)

    return np.array(covariance_matrix)

# %%
```

```
def get_mahalanobis_distance(X, u, covariance_matrix):
    X_u = X - u
    return np.sqrt(np.dot(np.dot(X_u, np.linalg.inv(covariance_matrix)), X_u.T))


# %%
def get_euclidean_distance(X, u):
    X_u = X - u
    return np.sqrt(np.dot(X_u, X_u.T))


# %%
def get_mean_vector_and_cov_matrices(X_train, y_train):
    mean_vectors = [None] * len(np.unique(y_train))
    cov_matrices = [None] * len(np.unique(y_train))

    for target in np.unique(y_train):
        target_data = X_train[y_train == target]
        mean_vectors[target] = get_mean_vector(target_data)
        cov_matrices[target] = get_covariance_matrix(target_data)

    return mean_vectors, cov_matrices


# %%
def predict_with_mahalanobis(mean_vectors, cov_matrices):
    y_pred = []

    for sample in X_test.values:
        distances = [get_mahalanobis_distance(sample, mean_vectors[i], cov_matrices[i]) for i in range(len(mean_vectors))]
        predicted_class = np.argmin(distances)
        y_pred.append(predicted_class)

    return y_pred


# %%
def predict_with_euclidean(mean_vectors):
    y_pred = []
    for sample in X_test.values:
        distances = [get_euclidean_distance(sample, mean_vectors[i]) for i in range(len(mean_vectors))]
        predicted_class = np.argmin(distances)
        y_pred.append(predicted_class)

    return y_pred


# %%
def plot_result(correct_samples, incorrect_samples, X_test):
    fig, axs = plt.subplots(2, 3, figsize=(20, 20))
    columns = X_test.columns
    k = 0
    m = 0

    for i in range(len(columns) - 1):
        for j in range(i + 1, len(columns)):
            axs[k, m].scatter(incorrect_samples.values[:, i], incorrect_samples.values[:, j], c='red', label='Incorrect Classification')
            axs[k, m].scatter(correct_samples.values[:, i], correct_samples.values[:, j], c='green', label='Correct Classification')

            axs[k, m].set(xlabel=columns[j], ylabel=columns[i], title='Correct and Incorrect Classifications')
```

```
        axs[k, m].legend()


        m += 1
        if m == 3:
            k += 1
            m = 0



# %%
def predict(mean_vectors, cov_matrices, X_test,
y_test, method):
    if(method == "mahalanobis"):
        y_pred =
predict_with_mahalanobis(mean_vectors,
cov_matrices)
    else:
        y_pred =
predict_with_euclidean(mean_vectors)


    correct_samples = X_test[y_pred == y_test]
    incorrect_samples = X_test[y_pred != y_test]
    print(test[y_pred != y_test])
    acc = round(accuracy_score(y_test, y_pred) *
100, 2)
    print("Accuracy is {} for {} distance
method".format(acc, method))
    plot_result(correct_samples, incorrect_samples,
X_test)



# %%
mean_vectors, cov_matrices =
get_mean_vector_and_cov_matrices(X_train,
y_train)


# %%
mean_vectors, cov_matrices


# %%
predict(mean_vectors, cov_matrices, X_test,
y_test, method="mahalanobis")


# %%
predict(mean_vectors, cov_matrices, X_test,
y_test, method="euclidean")
```