Ad/Soyad = Selim Can Yazar
No = 152120191023

Pattern Recognition Hw2

```
[1]  from keras.datasets import cifar10

     (x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

1.İlk satır (from keras.datasets import cifar10) Keras kütüphanesinin veri setleri bölümünden CIFAR-10 veri setini içe aktarmak için yazılmıştır.

2.İkinci satırda ((x_train, y_train), (x_test, y_test) = cifar10.load_data()), cifar10.load_data() fonksiyonu çağrılıyor. Bu fonksiyon, CIFAR-10 veri setini indiriyor ve veriyi dört parçaya ayırıyor:

2.1) x_train: Eğitim için kullanılacak görüntülerin kendileridir. Bu, bir NumPy dizisidir ve her bir öğe, bir görüntüyü temsil eder.

2.2) y_train: Eğitim görüntülerinin etiketleridir. Yani, her bir x_train görüntüsünün hangi kategoriye (örneğin cat,horse,truck,airplane vb.) ait olduğunu gösteren bir NumPy dizisidir.

2.3) x_test: Test için kullanılacak görüntülerin kendileridir. Modelimizi eğittikten sonra performansını bu görüntüler üzerinde test edeceğiz.

2.4) y_test: Test görüntülerinin etiketleridir, yani her bir x_test görüntümüzün hangi kategoriye ait olduğunu gösterir.

```
x_train = x_train.reshape(-1, 3072)  # 32x32x3 = 3072
x_test = x_test.reshape(-1, 3072)
```

x_train = x_train.reshape(-1, 3072) ve x_test = x_test.reshape(-1, 3072): Resehape fonksiyonu sayesinde train veri setindeki görüntüler yeniden şekillendirilir. CIFAR-10 dataseti görüntüleri 32x32 piksel boyutundadır ve her bir piksel 3 renk kanalına (kırmızı, yeşil, mavi) sahiptir. Yani her bir görüntü 32x32x3 = 3072 sayısına eşit bir sayıda değere sahiptir.Her bir görüntüyü tek bir sıra haline getirerek 3072 uzunluğunda bir vektöre dönüştürürüz. Bu işlem sayesinde görüntüyü bir matristen bir vektöre yani tek boyuta dönüştürürüz, böylece makine öğrenmesi algoritmaları bu verilerle daha kolay çalışabilir. -1 kullanımı NumPy kütüphanesinin otomatik olarak doğru boyutu belirlemesini yardımcı olur. Burada –1 diğer boyutları koruyarak kalan tüm öğeleri bu boyuta yerleştirilmesini sağlar.

```python
import numpy as np

def custom_similarity(x, y):
    # Vektörlerin nokta çarpımını hesapla
    dot_product = np.dot(x, y)
    # Vektörlerin öklid normlarını hesapla
    norm_x = np.linalg.norm(x)
    norm_y = np.linalg.norm(y)
    if norm_x == 0 or norm_y == 0:
        return 0  # Bölen sıfır olmamalı, bu yüzden herhangi bir norm sıfırsa, benzerlik 0 olarak kabul edilir.
    similarity = dot_product / (norm_x * norm_y)
    return similarity
```

def custom_similarity(x, y): Bu fonksiyon, x ve y adında iki parametre alır. Bu parametreler, benzerliklerini hesaplamak istediğimiz vektörlerdir.

dot_product = np.dot(x, y): x ve y vektörlerinin nokta çarpımını hesaplar. Nokta çarpım, vektörlerin ilgili elemanlarının çarpımlarının toplamıdır ve vektörlerin birbiriyle olan ilişkisini ölçen bir değer verir.

norm_x = np.linalg.norm(x) ve norm_y = np.linalg.norm(y): Bu satır, x vektörünün öklid normunu hesaplar. Öklid normu, vektörün başlangıç noktasından bitiş noktasına olan uzaklığını verir ve vektörün büyüklüğünü temsil eder.

if norm_x == 0 or norm_y == 0:: Bu ifade, herhangi bir vektörün normu sıfır ise kontrol eder. Bir vektörün normu sıfır ise, bu vektörün tüm elemanları sıfırdır. Bu durumda, benzerlik tanımsız olacağından fonksiyon 0 değerini döndürür.

similarity = dot_product / (norm_x * norm_y): Benzerlik iki vektörün nokta çarpımının, vektörlerin normlarının çarpımına bölünmesiyle elde edilir. Bu değer, vektörlerin birbirine olan yön benzerliğini ifade eder.

```python
def knn_custom_similarity(x_train, y_train, sample_test, k):
    similarities = []
    for i in range(len(x_train)):
        sim = custom_similarity(sample_test, x_train[i])
        similarities.append((sim, y_train[i]))
    # Benzerliklere göre sıralama ve en yüksek 'k' değerini seçme
    sorted_similarities = sorted(similarities, key=lambda x: x[0], reverse=True)
    top_k = sorted_similarities[:k]
    # En sık rastlanan sınıfı bulma
    classes = [item[1] for item in top_k]
    prediction = max(classes, key=classes.count)
    # Eğer prediction bir dizi ise, ilk öğeyi alarak bir tamsayıya dönüştür
    if isinstance(prediction, np.ndarray):
        prediction = prediction.item()

    return prediction
```

1. def knn_custom_similarity(x_train, y_train, sample_test, k): Bu fonksiyon knn_custom_similarity adında bir fonksiyon tanımlar. Bu fonksiyon, x_train (eğitim verileri),

y_train (eğitim verilerinin etiketleri), sample_test (sınıflandırılmak üzere olan tek bir test örneği) ve k (en yakın kaç komşunun dikkate alınacağı) parametrelerini alır.

2. similarities = []: Boş bir liste oluşturur. Bu liste, eğitim setindeki her örneğin test örneği ile olan benzerliklerini ve ilgili sınıf etiketlerini saklayacaktır.

3. for i in range(len(x_train)): Bu döngü, eğitim veri setindeki her bir örneği gezer.

4. sim = custom_similarity(sample_test, x_train[i]): Her döngü adımında, test örneği ile eğitim setindeki i'nci örnek arasındaki benzerliği hesaplar.

5. similarities.append((sim, y_train[i])): Hesaplanan benzerlik değeri ve ilgili sınıf etiketi similarities listesine bir çift olarak eklenir.

6. sorted_similarities = sorted(similarities, key=lambda x: x[0], reverse=True): Benzerlikler benzerlik değerine göre azalan sırada sıralanır.

7. top_k = sorted_similarities[:k]: Sıralı listeden en yüksek benzerliğe sahip ilk k öğe seçilir.

8. classes = [item[1] for item in top_k]: Seçilen k öğenin sınıf etiketleri bir listeye eklenir.

9. prediction = max(classes, key=classes.count): En çok tekrar eden sınıf etiketi, prediction olarak belirlenir.

10. if isinstance(prediction, np.ndarray):Eğer prediction bir NumPy dizisiyse (çok boyutlu bir çıktıysa), bu dizi tek bir değere indirgenir.

11. prediction = prediction.item(): Prediction NumPy dizisinden bir Python skalerine dönüştürülür.

```python
import numpy as np

# Rastgele bir index seç
random_index = np.random.randint(0,len(x_test))  # 0 ile 9999 arası bir sayı üretir

# Rastgele seçilen indekse göre test vektörünü al
sample_test = x_test[random_index]
k = 5

# Sınıflandırma fonksiyonunu çağırarak tahmini sınıfı al
predicted_class = knn_custom_similarity(x_train, y_train, sample_test, k)
print(f"Predicted class for index {random_index}: {predicted_class}")

# Sınıf isimlerinin listesi
class_names = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

# Tahmin edilen sınıf etiketini sınıfın ismine çevir
predicted_class_name = class_names[predicted_class]

# Sonucu yazdır
print(f"Predicted class name for index {random_index}: {predicted_class_name}")
```

1.random_index = np.random.randint(0, len(x_test)): Burada, x_test dizisinin boyutuna bağlı olarak rastgele bir indeks numarası seçilir. Bu indeks, test setinden seçilecek rastgele bir örneği belirler.

2.sample_test = x_test[random_index]: Seçilen rastgele indekse karşılık gelen test örneği sample_test değişkenine atanır.

3.k = 5: Bu, k-NN algoritmasında dikkate alınacak en yakın komşu sayısını belirler.

4.predicted_class = knn_custom_similarity(x_train, y_train, sample_test, k): knn_custom_similarity fonksiyonu çağrılarak, sample_test için bir sınıf tahmini yapılır. Bu fonksiyon, eğitim seti x_train, eğitim etiketleri y_train, test örneği sample_test ve komşu sayısı k ile çalışır.

5.print(f"Predicted class for index {random_index}: {predicted_class}"): Tahmin edilen sınıf etiketi ekrana yazdırılır.

6.class_names = []: CIFAR-10 veri setindeki sınıfların isimlerini içeren liste oluşturuldu.

7.predicted_class_name = class_names[predicted_class]: Tahmin edilen sınıf etiketi karşılık gelen sınıf ismine dönüştürülür.

8.print(f"Predicted class name for index {random_index}: {predicted_class_name}"): Tahmin edilen sınıfın ismi ekrana yazdırılır.

Örnek Çıktı Görselleri

```
Predicted class for index 1580: 2
Predicted class name for index 1580: bird
```

```
Predicted class for index 6223: 8
Predicted class name for index 6223: ship
```

```
Predicted class for index 2551: 3
Predicted class name for index 2551: cat
```

```
Predicted class for index 162: 0
Predicted class name for index 162: airplane
```

```
Predicted class for index 5278: 6
Predicted class name for index 5278: frog
```

```
 Predicted class for index 2116: 4
 Predicted class name for index 2116: deer
```