OnDokuz Mayıs Üniversitesi

Bilgisayar Mühendisliği Bölümü

Cihan DEMİR

19060428

Makine Öğrenimi Dönem Sonu Projesi

İÇİNDEKİLER

BÖLÜM: I

MATERYAL

1

VERI ÖNIŞLEME

```
import tensorflow as tf
import os
import numpy as np
```

Şekil 1: Veri Önişlemede Kullanılan Modüller

Resimdeki modüller veri önişleme kısmında kullanıldı.

```
dosya_yolu_tavuk = '/content/tavuk'
dosya_yolu_at    = '/content/at'
dosya_yolu_kedi  = '/content/kedi'
```

Şekil 2: Klasör Yolları

Her bir sınıfa ait olan resimler farklı klasörler içerisinde tutulmuştur ve değişkenlere atanmıştır.

```python
def tavuk_veri_seti(dosya_yolu_tavuk, verinin_sinifi):
  # Klasördeki resimler for dongüsü içerisnde okundu.
  for dosya in os.listdir(dosya_yolu_tavuk):
      # Resim okuma
      resim = tf.io.read_file(os.path.join(dosya_yolu_tavuk, dosya))
      # Resmi boyutlandır
      resim = tf.image.decode_jpeg(resim)
      resim = tf.image.resize(resim, (80, 80))
      # Renkli resmi renksiz hale getir
      resim = tf.image.rgb_to_grayscale(resim)
      # Resmi tek boyuta indir
      resim = tf.reshape(resim, (80*80,))
      resim = np.array(resim)
      #Veriyi Etiketle
      resim = np.append(resim,verinin_sinifi)
      # Resim verilerini csv dosyasına yaz
      csv_dosya = 'veri_seti.csv'
      with open(csv_dosya, 'a') as file:
        np.savetxt(file, resim.reshape(-1, resim.shape[-1]), delimiter=',')

tavuk_veri_seti(dosya_yolu_tavuk,0)
```

Şekil 3: Tavuk Veri Setinin Oluşturulması

os modülü kullanılarak dosyaya gidildi. Bir tane görüntü, resim değişkenine atandı. Daha sonra JPEG dosyası 80x80 piksel boyutuna indirgendi. Renkli resimli siyah beyaza dönüştürüldü. Bu dönüşüm her piksel değeri için RGB değerlerinin ortalaması alınarak bulundu. Görüntü, 1 boyuta dönüştürüldü ve numpy dizisine çevrildi. Veri etiketleme, her satırın sonuna ilgili görütünün sınıf numarası eklenerek yapıldı. CSV formatında dosya oluşturulup içine veriler yazdırıldı. Yapılan bu işlemler for dongüsü ile her bir resim için tekrarlandı. Son olarak oluşturulan metoda dosya yolu ve sınıf etiketi verilerek çağrıldı.

```python
def at_veri_seti(dosya_yolu_at, verinin_sinifi):
  # Klasördeki resimleri oku
  for dosya in os.listdir(dosya_yolu_at):
      # Resmi oku
      resim = tf.io.read_file(os.path.join(dosya_yolu_at, dosya))
      # Resmi boyutlandır
      resim = tf.image.decode_jpeg(resim)
      resim = tf.image.resize(resim, (80, 80))
      # Renkli resmi renksiz hale getir
      resim = tf.image.rgb_to_grayscale(resim)
      # Resmi tek boyuta indir
      resim = tf.reshape(resim, (80*80,))
      resim = np.array(resim)
      #Veriyi Etiketle
      resim = np.append(resim,verinin_sinifi)
      # Resim verilerini csv dosyasına yaz
      csv_dosya = 'veri_seti.csv'
      with open(csv_dosya, 'a') as file:
        np.savetxt(file, resim.reshape(-1, resim.shape[-1]), delimiter=',')

at_veri_seti(dosya_yolu_at,1)
```

Şekil 4: At Veri Setinin Oluşturulması

Tavuk veri setindeki işlemler, at veri seti için de tekrarlandı. Sadece dosyaya yazma kısmında dosya tekrar oluşturulmadı, var olan dosyanın sonuna eklendi.

```python
def kedi_veri_seti(dosya_yolu_kedi, verinin_sinifi):
  # Klasördeki resimleri oku
  for dosya in os.listdir(dosya_yolu_kedi):
      # Resmi oku
      resim = tf.io.read_file(os.path.join(dosya_yolu_kedi, dosya))
      # Resmi boyutlandır
      resim = tf.image.decode_jpeg(resim)
      resim = tf.image.resize(resim, (80, 80))
      # Renkli resmi renksiz hale getir
      resim = tf.image.rgb_to_grayscale(resim)
      # Resmi tek boyuta indir
      resim = tf.reshape(resim, (80*80,))
      resim = np.array(resim)
      #Veriyi Etiketle
      resim = np.append(resim,verinin_sinifi)
      # Resim verilerini csv dosyasına yaz
      csv_dosya = 'veri_seti.csv'
      with open(csv_dosya, 'a') as file:
        np.savetxt(file, resim.reshape(-1, resim.shape[-1]), delimiter=',')


kedi_veri_seti(dosya_yolu_kedi,2)
```

Şekil 5: Kedi Veri Setinin Oluşturulması

Tavuk veri setindeki işlemler, kedi veri seti için de tekrarlandı. Sadece dosyaya yazma kısmında dosya tekrar oluşturulmadı, var olan dosyanın sonuna eklendi.

```python
import csv
import numpy as np

# Veri setini oku
with open('veri_seti.csv', 'r') as f:
    oku = csv.reader(f)
    veri = list(oku)

# Veri setindeki her sütun için normalizasyon yap
veri = np.array(veri).astype(float)
for i in range(veri.shape[1]-1):
    sutun = veri[:, i]
    veri[:, i] = (sutun - sutun.min()) / (sutun.max() - sutun.min())

# Normalize edilmiş veriyi yeni bir CSV dosyasına yaz
with open('normalize_veri.csv', 'w', newline='') as f:
    yaz = csv.writer(f)
    for satir in veri:
        # Her hücreyi 4 basamaklı bir sayı olarak formatla
        yeni_satir = ["%.4f" % x for x in satir]
        yaz.writerow(yeni_satir)
```

Şekil 6: Normalizasyon

Veri seti okuma modunda açıldı, csv modülü ile okundu ve liste halinde veri değişkenine atandı. Daha sonra veri değişkenine atanan veriler numpy dizisine dönüştürüldü ve tipleri float olarak belirlendi. Csv dosyasından alınan verilerde sütun sütun gezdirildi ve her sütünün maksimum ve minimum değerleri bulunarak veri setine normalizasyon yapıldı. Daha sonra 'normalize_veri.csv' adındaki yeni dosyaya virgülden sonra 4 basamak olacak şekilde yazdırıldı. 4 basamak yaparak veri setini kullanan algoritmanın işlem yükünü hafifletmek amaçlanmıştır.

2

ANA DOSYA

Kullanılacak olan modülleri projemize ekliyoruz.

```
1  from random import seed
2  from random import randrange
3  from random import random
4  from csv import reader
5  from math import exp
```

Şekil 7: Kullanılan Modüller

CSV formatında bulunan veri setinden satır satır okuma yapıp veri_seti adındaki listeye ekliyoruz.

```
7  # CSV Dosyasını Yükle
8  def csv_yukle(dosya):
9      veri_seti = list()
10     with open(dosya, 'r') as file:
11         csv_oku = reader(file)
12         for row in csv_oku:
13             if not row:
14                 continue
15             veri_seti.append(row)
16     return veri_seti
```

Şekil 8: Veri Setini İçe Aktarma

Veri setinden string tipinde çekilen verileri float tipine dönüştürüyoruz.

```
19  # Veri Setindeki Değerleri Float Tipine Dönüştür
20  def stringi_floata_donustur(veri_seti, sutun):
21      for satir in veri_seti:
22          satir[sutun] = float(satir[sutun].strip())
```

Şekil 9: Tip Dönüşümü

Her verinin sonunda yer alan etiket değerini float tipinden integerr tipine dönüştürüyoruz.

```
25  # Veri Setindeki Etiket Değerlerini İnteger tipine Dönüştür
26  def stringi_inte_donustur(veri_seti, sutun):
27      sinif_degeri = [satir[sutun] for satir in veri_seti]
28      sinif = set(sinif_degeri)
29      tamsayi = dict()
30      for i, deger in enumerate(sinif):
31          tamsayi[deger] = i
32      for satir in veri_seti:
33          satir[sutun] = tamsayi[satir[sutun]]
34      return tamsayi
```

Şekil 10: Tip Dönüşümü

Bu metod ile csv formatındaki dosyadan alınan veriler ile eğitim seti ve test veri seti oluşturuluyor. Kullanıcı tarafından verilen sayıya bölünen veri setinden 1 tanesi test seti olarak ayrılıyor.

```
37  # Veri Setini Belirtilen Sayıya Böler ve 1 Tanesini Test Verisi Olarak Alır
38  def veriyi_bolme(veri_seti, parca_sayisi):
39      bolunmus_veri = list()
40      veri_kopyasi = list(veri_seti)
41      parca_boyutu = int(len(veri_seti) / parca_sayisi)
42      for i in range(parca_sayisi):
43          parca = list()
44          while len(parca) < parca_boyutu:
45              index = randrange(len(veri_kopyasi))
46              parca.append(veri_kopyasi.pop(index))
47          bolunmus_veri.append(parca)
48      return bolunmus_veri
```

Şekil 11: Veriyi Ayırma

Bu metodda tahmin edilen değer ile gerçek değerin karşılaştırılması yapılıp doğruluk yüzdesi hesaplanıyor.

```
51  # Doğruluk Yüzdesini Hesapla
52  def dogruluk_yuzdesi(gercek, tahmin):
53      dogruluk = 0
54      for i in range(len(gercek)):
55          if gercek[i] == tahmin[i]:
56              dogruluk += 1
57      return dogruluk / float(len(gercek)) * 100.0
```

Şekil 12: Doğruluk Yüzdesi

Şekil11'deki metodu kullanarak veri setini test ve eğitim seti olarak ayırır. Kullanılan backpropagation algoritması ile tahmin gerçekleştirir. Şekil12'deki metodu kullanarak doğruluk yüzdesini hesaplar. Başarı oranını geri döndürür.

```
61  def temel_metod(veri, algoritma, parca_sayisi, *args):
62      bolunmus_veri = veriyi_bolme(veri, parca_sayisi)
63      basari_orani = list()
64      egitim_verisi = list(bolunmus_veri)
65      egitim_verisi.remove(bolunmus_veri[3])
66      egitim_verisi = sum(egitim_verisi,[])
67      test_verisi = list()
68      for i in bolunmus_veri[3]:
69          i_copy = list(i)
70          test_verisi.append(i_copy)
71          i_copy[-1] = 1
72
73      tahmin = algoritma(egitim_verisi, test_verisi, *args)
74      gercek = [i[-1] for i in bolunmus_veri[3]]
75      basari = dogruluk_yuzdesi(gercek, tahmin)
76      basari_orani.append(basari)
77
78      return basari_orani
```

Şekil 13: Temel İşlemler

Kullanılan modelin hesaplanması işlemi bu metodda yapılır.

```
82    def activate(agirlik, girdi):
83        aktivasyon= agirlik[-1]
84        for i in range(len(agirlik) - 1):
85            aktivasyon += agirlik[i] * girdi[i]
86        return aktivasyon
```

Şekil 14: Modelin Hesaplanması

Yapay sinir ağlarında ileri yayılım işlevi bu metod ile yapılır. Girdiler alınır rastgele atanan ağırlıklarla çarpılır, aynı nörona bağlanan ağırlıklar ve girdiler çarpılıp toplanır. Son olarak elimizde bir çıktı değeri oluşur.

```
95    def ileri_yayilim(ag, satir):
96        girdi = satir
97        for katman in ag:
98            yeni_girdi = []
99            for noron in katman:
100               aktivasyon = activate(noron['weights'], girdi)
101               noron['output'] = sigmoid(aktivasyon)
102               yeni_girdi.append(noron['output'])
103           girdi = yeni_girdi
104       return girdi
```

Şekil 15: İleri yayılım

İleri yayılımda her nöronda çıktıları bulmak için kullanılan sigmoid fonksiyonu bu metod ile tanımlanmıştır.

```
89    # Sigmoif Fonksiyonu
90    def sigmoid(aktivasyon):
91        return 1.0 / (1.0 + exp(-aktivasyon))
92
```

Şekil 16: Sigmoid Fonksiyonu

Geri yaymada kullanılacak olan sigmoid fonksiyonunun türevi burada tanımlanmıştır.

```
107     def sigmoid_turev(cikti):
108         return cikti * (1.0 - cikti)
109
```

Şekil 17: Sigmoid Fonksiyonunun Türevi

Bu metodda her katmanda bulunan her nöronun hata bilgisi bulunarak hata_liste adındaki listeye atanıyor.

```
111 def hatayi_yay(ag, gercek_deger):
112     for i in reversed(range(len(ag))):
113         katman = ag[i]
114         hata_liste = list()
115         if i != len(ag)-1:
116             for j in range(len(katman)):
117                 hata = 0.0
118                 for noron in ag[i + 1]:
119                     hata += (noron['weights'][j] * noron['delta'])
120                 hata_liste.append(hata)
121
122         else:
123             for j in range(len(katman)):
124                 noron = katman[j]
125                 hata_liste.append(noron['output'] - gercek_deger[j])
126
127         for j in range(len(katman)):
128             noron = katman[j]
129             noron['delta'] = hata_liste[j] * sigmoid_turev(noron['output'])
```

Şekil 18: Hatayı Geriye Yayma

Şekil17'de kullanılan metod ile bulunan hatalar kullanılarak ağırlık değerlerinin güncellemesi işlemi bu metod ile yapılacaktır.

```
133 def agirliklari_guncelle(ag, satir, alfa):
134     for i in range(len(ag)):
135         girdi = satir[:-1]
136         if i != 0:
137             girdi = [noron['output'] for noron in ag[i - 1]]
138         for noron in ag[i]:
139             for j in range(len(girdi)):
140                 noron['weights'][j] -= alfa * noron['delta'] * girdi[j]
141             noron['weights'][-1] -= alfa * noron['delta']
```

Şekil 19: Ağırlıkları Güncelleme

Bu metod ile ağı eğitme işlevini gerçekleştirir. Verilen epoch sayısı kadar dönen bir for'un içinde bir for döngüsü daha kullanılarak eğitim setinden satır satır okuma işlemi yapılır. İçteki for'da ilk olarak ilk veri ile ileri yayılım yapılır. Hatayı geriye yayma işlemi yapılır ve en sonda ağırlıklar güncellenir. Ekrana da epoch sayısı adım adım yazdırılır.

```
144  def train_network(ag, train, alfa, n_epoch, n_outputs):
145      for epoch in range(n_epoch):
146          for satir in train:
147              outputs = ileri_yayilim(ag, satir)
148              expected = [0 for i in range(n_outputs)]
149              expected[satir[-1]] = 1
150              hatayi_yay(ag, expected)
151              agirliklari_guncelle(ag, satir, alfa)
152          print(epoch)
```

Şekil 20: Ağı Eğitme

Kullanılan ağ yapısı burada oluşturulur.ag adında liste oluşturulur. Kullanıcıdan alınan değer kadar gizli katmanda nöron oluşturulur. Çıktı katmanındaki nöron sayısı ise sınıflandırma sayısına eşittir.

```
154  def initialize_network(girdi, g_noron_sayisi, cikti_sayisi):
155      ag = list()
156      gizli_katman = [{'weights':[random() for i in range(girdi + 1)]} for i in range(g_noron_sayisi)]
157      ag.append(gizli_katman)
158      cikti_katmani = [{'weights':[random() for i in range(g_noron_sayisi + 1)]} for i in range(cikti_sayisi)]
159      ag.append(cikti_katmani)
160      return ag
```

Şekil 21: Ağı Oluşturma

İleri yayılım metodunu kullanarak oluşan çıktılarla sınıflandırma yapar.

```
168  def tahmin_et(ag, satir):
169      cikti = ileri_yayilim(ag, satir)
170      return cikti.index(max(cikti))
```

Şekil 22: Sınıflandırma Yapma

Girdi ve çıktı sayısı burada belirlenir. Ağ yapısı burada çağrılır ve oluşturulur. Ağ burada eğitilir. Tahmin metodu burada çağrılarak işlem yapılır.

```
173    def back_propagation(train, test, alfa, epoch, g_noron_sayisi):
174        girdi_sayisi = len(train[0]) - 1
175        cikti_sayisi = len(set([row[-1] for row in train]))
176        ag = initialize_network(girdi_sayisi, g_noron_sayisi, cikti_sayisi)
177        train_network(ag, train, alfa, epoch, cikti_sayisi)
178        tahmin_list = list()
179        for row in test:
180            tahmin = tahmin_et(ag, row)
181            tahmin_list.append(tahmin)
182        return(tahmin_list)
```

Şekil 23: Tahmin Oluşturma

Veri setinin ismi dosya değişkenine atanır. csv_yukle fonksiyonu çağrılarak veri seti uygulamaya yüklenir. For döngüsü içerisinde veri setindeki değerler stringden floata dönüştürülür. Daha sonra etiket değerleri stringden integerr e dönüştürülür. Veri setinin kaça ayrılacağı değeri verilir. Öğrenme değeri alfa değişkenine atanır. Epoch sayısı belirlenir. Gizli katmanda bulunan nöron sayısı verilir. İşlemler sonucunda başarı oranı ekrana yazdırılır.

```
185    dosya = 'normalize_veri.csv'
186    veri_seti = csv_yukle(dosya)
187    for i in range(len(veri_seti[0])-1):
188        stringi_floata_donustur(veri_seti, i)
189    stringi_inte_donustur(veri_seti, len(veri_seti[0])-1)
190    parca_sayisi = 5
191    alfa = 0.5
192    epoch = 1000
193    g_noron_sayisi = 2
194    basari_orani = temel_metod(veri_seti, back_propagation, parca_sayisi, alfa, epoch, g_noron_sayisi)
195    print('Başarı Oranı: %s' % basari_orani)
```

Şekil 24: Değerler

BÖLÜM: II

SONUÇ

3

SONUÇ

Model için at,tavuk ve kedi veri kümeleri kullanıldı. Her veri kümesinde 300 adet jpeg formatında görüntü bulunmaktadır.

Veri kümelerinde görüntüyü boyutlandırma, görüntüyü renksiz hale getirme ve etiketleme işlemleri yapılmıştır.

Model resim boyut kadar girdi, 2 nöronlu gizli katman ve 3 tane nöron bulanan çıkış katmanına sahiptir.

Sonuç olarak yapılan 6 denemede %45.8'lik bir doğruluk oranı sağlanmıştır.

Denemede kullanılan değerler ve sonuçları tabloda verilmiştir.

Görüntü Sınıflandırma					
Deneme Sayısı	Parça Sayısı	Alfa	Epoch	Gizli Nöron Sayısı	Başarı Oranı
1	5	0.5	2200	2	%50
2	4	0.5	2200	2	%45
3	4	0.5	1000	2	%42
4	4	0.5	4000	2	%45
5	4	0.5	3000	2	%47
6	4	0.5	2500	2	%46