python powered

print("Hello, world!")

5. Python ile Görüntü İşleme

Python ile Görüntü İşlemede Örnek Bir Uygulama

Öğr. Gör. Dr. Murat GEZER

5.1 Giriş

Günümüz büyük bir veri tufanının tam ortasındadır. 2012 rakamlarına göre dijital evrende 2.7 Zetabyte veri bulunmaktaydı ve her geçen gün 2.5 exabyte boyutunda veri bu evrene eklenmektedir [1]. Verinin bu denli büyük boyutta olması nedeniyle anlamlandırılması yani bilgi haline dönüştürülmesi çok önemlidir Bu konuda her geçen gün yeni çalışmalar yapılmaktadır. Bu çalışmalar için makine öğrenmesi teknikleri kullanılmaktadır. Görüntü işleme özellikle dijital teknolojilerin çok hızlı gelişim sağlaması nedeniyle hayatımızın birçok alanında yerini almaktadır. Böylelikle dijital evren için en önemli verinin bilgiye dönüştürme kaynaklarından biri haline gelmiştir. Günümüzde veri bilimciler görüntü sınıflandırma, video analizi gibi çalışmaları için makineyle öğrenmesi için GPU'ları (Grafik İşlemci Ünitesi) kullanmaktadır. Bu yöntemler ile çok büyük miktarlarda eğitim verisini kullanarak çok iyi sonuçlar üretmektedir. Bu çalışmada Raspberry Pi gibi düşük güç tüketimine sahip mini bilgisayarlar kullanılabilecek olan temel yöntemler kullanılmıştır. Bu yöntemler görüntülerin benzerlerinin bulunması için benzerlik ve uzaklık ölçülerinin kullanılması şeklindedir. Bu çalışmada tüm kodlama Python 3.6 kullanarak Anaconda dağıtımı üzerinde gerçekleştirilmiştir.

5.1.1 Temel Tanımlar

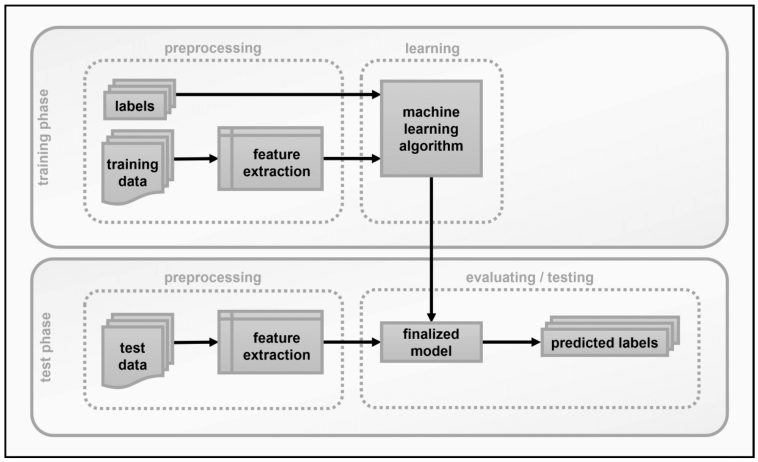
Görüntüler üzerinde gerçekleştirmeye başlamadan önce Makine Öğrenmesi, Görüntü İşleme ve Bilgisayarlı Görme tanımları üzerinde durmakta yarar vardır.

Makine Öğrenmesi:

Makine öğrenmesi bilgisayar bilimlerinin kuramsal alt dalıdır. Matematiksel ve istatistiksel yöntemler kullanarak mevcut verilerden çıkarımlar yapan, bu çıkarımlarla bilinmeyene dair tahminlerde bulunan yöntemler bütünü olarak adlandırabiliriz. Makine Öğrenmesinin öncülerinden olan Arthur

Samuel tanımı "bilgisayarın açık bir şekilde programlanmadan bir işi öğrenme yeteneği olarak" şeklinde vermiştir [2]. Üniversitelerde yaygın bir şekilde ders kitabı olarak kullanılan Introduction to Machine Learning adlı kitabında Alpaydın Makine öğrenmesinin temelini gözlenmiş bir örneklem kümesinden çıkarıp yapmak ve bu çıkarımlara uygun bir model oluşturmak için istatistik kullanılması olarak belirtmiştir [3]. Tom M. Mitchell tarafından verilen ve literatürde genellikle kullanılan tanım: "Bazı amaçları (T) gerçekleştirmek için P performansı ile çalışan bir program, deneyimleri sonucunda (E), P ile ölçülen performansını arttırarak amaçları gerçekleştiriyorsa öğrenebildiği söylenebilir." şeklindedir [4]. Makine öğrenmesi terimi, verilen tanımlardan anlaşılacağı gibi çok genel bir şekilde kullanılmaktadır. Ve genellikle büyük veri kümeleri içeresinden desen çıkarmak ya da mevcut verileri analiz ederek daha önce öğretilenleri temel alan yeni verilere ilişkin tahminler yapma kabiliyetine atıfta bulunmaktadır. Makine öğrenmesi ile istatistik bilimi, veri madenciliği, iş zekası, örüntü tanıma, yapay zekalı sistemler, doğal dil işleme, bilgisayarlı görme, biyoinformatik, robotik, görüntü işleme gibi birçok farklı alan arasında bir ilişki vardır Makine öğrenmesi algoritmaları genel olarak Danışmanlı Öğrenme (Supervised Learning), Danışmansız Öğrenme (Unsupervised Learning) , Yarı-Danışmanlı Öğrenme (Semi-Supervised Learning) ve Pekiştirmeli (Takviyeli) Öğrenme (Reinforcement Learning) olmak üzere dörde ayrılmaktadır. Bu çalışmada Danışmanlı Öğrenme ve Danışmanız Öğrenme algoritmaları ile işlem yapacağız. Danışmanlı Öğrenme algoritmaları sınıf verilerinin tanımlı olduğu durumlarda kullanılmaktadır. Danışmansız Öğrenme algoritmaları ise sınıf verilerinin tanımlı olmadığı durumlarda kullanılmaktadır.

Literatür incelendiğinde makine öğrenmesi süreçlerinde farklı yaklaşımlar bulunduğu görülmemektedir. Ancak en yaygın kullanılan makine öğrenmesi süreci Şekil 5.1 de görülmektedir.



Şekil 5.1: Örnek bir makine öğrenmesi akış şeması [5]

Burada görüldüğü üzere süreç iki aşamadan oluşmaktadır. Eğitim aşaması (training phase) ve test aşaması (test phase) şeklindedir. Makine öğrenmesindek bazı kavramlar şu şekildedir.

Gözlemler (Observations): öğrenmek ya da değerlendirmek için kullanılan her bir veri parçası. Örn: her bir e-posta bir gözlemdir.

Özellikler (Features): Bir gözlemi temsil eden (genelde sayısal) verilerdir. Örn: e-posta'nın uzunluğu, tarihi, bazı kelimelerin varlığı.

Etiketler (Labels): Gözlemlere atfedilen kategoriler. Örn: spam, spam-değil.

Eğitim Verisi (Training Data): Algoritmanın öğrenmesi için sunulan gözlemler dizisi. Algoritma bu veriye bakarak çıkarımlarda bulunur, kafasında model kurar. Örn: çok sayıda spam/spam-değil diye etiketlenmiş e-posta gözlemi.

Test Verisi (Test Data): Algoritmanın kafasında şekillendirdiği modelin ne kadar gerçeğe yakın olduğunu test etmek için kullanılan veri seti. Eğitim esnasında saklanır, eğitim bittikten sonra etiketsiz olarak algoritmaya verilerek algoritmanın (vermediğimiz etiketler hakkında) tahminlerde bulunması beklenir. Örn: spam olup olmadığı bilinen (ama gizlenen), eğitim verisindekilerden farklı çok sayıda e-posta gözlemi.

Görüntü İşleme:

Görüntü İşleme kavramına geçmeden önce sayısal görüntüyü tanımlamak gerekir. Sayısal görüntü genel olarak Görüntülerin sayısal ortam için uygun hale dönüştürülmüş şekilleri olarak tanımlanabilir. Görüntü, iki boyutlu m, n uzay koordinatlarında bir olarak tanımlanan ışık yoğunluk fonksiyonudur [6]. Burada x, y değerleri ve fonksiyon genlikleri sonlu ve tamsayı ise bu görüntü, sayısal görüntü olarak adlandırılır. Böylelikle görüntü işleme, görüntüyü sayısal hale getirmek ve bazı işlemleri gerçekleştirmek için geliştirilmiş, spesifik görüntü elde etmek veya ondan bazı yararlı bilgiler çıkarmak için kullanılan yöntemler olarak söyleyebiliriz. Görüntü işleme ile dijital görüntüler üzerinde çeşitli işlemler yapılarak yeni görüntülerin elde edilmesini amaçlanmaktadır. Bunun için ölçülmüş ya da kaydedilmiş olan dijital görüntü verilerine, bilgisayar da bulunan yazılımlar veya programla dilleri ile amaca uygun şekilde kullanılan matematiksel algoritmalar uygulanır. Genellikle Görüntü İşleme sistemi, önceden belirlenmiş sinyal işleme(Signal Processing)yöntemlerini uygularken görüntüleri iki boyutlu sinyaller olarak ele almaktadır.

Bilgisayar/Makine Görmesi:

Bilgisayar görmesi ya da Makine Görmesi, sayısal görüntülerden veya videolardan üst düzey bir anlam elde etmek için algoritmaların nasıl oluşturulabileceğini ele alan disiplinler arası bir alandır. Genel olarak insan görsel sisteminin yapabileceği görevleri otomatikleştirmeyi amaçlamaktadır. Gördüğünü anlayabilen akıllı bilgisayar sistemlerinin bilgisayar görmesinin ana amacıdır. Üzerinde çalışılan konular arasında, kamera görüntülerinden yüz tanıma, plaka tanıma, görüntüden 3-Boyutlu yüzey geometrisinin bulunması, ayrıt saptama bulunmaktadır.

Şekil 5.2: Görüntü işlemeden bilgisayarlı görmeye işlem zorluk seviyesi [7]

5.2 Python ile Görüntü İşleme

Python programlama dili 1980'lerin sonlarına doğru Guido van Rossum tarafından 1989 Aralık ayında geliştirildi. Adını bir yılandan değil Guido van Rossum'un çok sevdiği, Monty Python adlı altı kişilik bir İngiliz komedi grubunun Monty Python's Flying Circus adlı gösterisinden almıştır.

Günümüzde Python Yazılım Vakfı çevresinde toplanan Python topluluğu tarafından geliştirilmektedir.İlk kararlı sürümü olan Python 1.0 Ocak 1994 yayınlanmıştır. Son kararlı sürüm Eylül 2017 itibari ile 2.x serisinde Python 2.7.14 ve 3.x serisinde Python 3.6.2'dir. 3.x sürümü 3 Aralık 2008 yayınlanmaya başlamıştır; dikkat edilmesi gerek önemli bir nokta 3.x serisi 2.x serisi arasında tam bir geriye doğru uyumluluğun olmamasıdır [8]. Çalışma kapsamında kullandığımız Python 3.5 için her ne kadar doğrudan python.org adresine giderek en son python sürümünü indirmek mümkündür. Bu çalışmada kullanmak üzere https://www.anaconda.com/ adresinde bulunan Anaconda dağıtmını tercih edilmiştir. Anaconda piyasada bulunan bilimsel hesaplama için kullanılan ticari yazılımlara benzeyen açık kaynak kodlu bir platformdur. Python ile görüntü işleme ve makine öğrenmesi için bir çok modül bulunmaktadır. çalışmamızda OpenCV,sklearn, matplotlib ,pandas ve Numpy modülleri kullanılmıştır.

5.3 Benzerlik Algoritmaları ile Rakam Tanıma

5.3.1 Veri Kümesi

UCI Machine Learning Repository adresinde halka açık olan kalem tabanlı el yazısı karakterlerinin tanınması veri kümesi kullanılacaktır. Boğaziçi Üniversitesinden Ethem Alpaydın ve Fevzi Alimoğlu tarafından Wacom PL-100V tablet tarafından toplanmış olan bu veri kümesi 44 farklı kişinin yazmış olduğu 250 farklı rakamdan oluşmaktadır [9]. Toplam 5620 örnekten oluşmaktadır. Veri kümemiz sklearn kütüphanesinde hazır olarak gelmektedir.Veri kümesini yüklemek için öncelikle sklearn kütüphanesini içerisinde bulunan datasets nesnesini çağırdık. Bu işlem için **from sklearn import datasets** komutu kullanılır. datasets nesnesi içerisinde bulunan **load_digits()** metodu ile el yazısı veri kümesi otomatik olarak çevrimiçi olarak indirilmiştir.

5.3.2 Benzerlik ve uzaklık Ölçüleri

Makine öğrenmesinde kullanılan algoritmaların büyük kısmında nesnelerin birbirlerine olan benzerlikleri yada uzaklıkları özellikleri öznitelikleri değerleri vasıtası ile bulunmaktadır. Bu ölçütler problemin yapısı ve verinin türününe göre çeşitlidir .Tablo 5.1 da görüleceği gibi çeşitli Uzaklık ölçüleri bulunmaktadır ayrıca çeşitli benzerlik ölçüleride bulunmaktadır. Bu konuda detaylı bilgi istenirse türkçe kaynak olarak Haldun Akpınar tarafından yazılmış olan Data adlı kitabı gösterilebilir [10].

Aralık Ölçek	**Frekans**	**İkil**
Euclid	chi-square	Euclid
Kareli Euclid	Phi-square	Kareli Euclid
Minkowski		Büyüklük Farkı
Chebyskev		Örüntü Farkı
Manhattan		Varyans
Mahalobonis		Biçim
		Lance & Williams

Tablo 5.1: Uzaklık Ölçüleri [10]

Çalışmada Manhattan uzaklık metriği , Eucklid uzaklık metriği ve Kosinüs benzerliği ölçüleri kullanılmıştır. Vektörün arasındaki açı farkı kullanıldığında Kosinüs Benzerlik öçütü kullanılmış

Aralık Ölçek	**İkil**
Pearson Korelasyonu	Russell
Kosinüs benzerliği	Jackard
	Zar

Tablo 5.2: En Bilindik Benzerlik Ölçüleri [10]

olmaktadır (denklem 5.1).

$$cosinesimilarity = cos\theta = \frac{\vec{a} \,.\, \vec{b}}{\| \vec{a} \| \| \vec{b} \|} \tag{5.1}$$

Kosinüş benzerlik ölçütü aradaki açıya bağlı olarak -1 ile 1 sonuçunu döndürmektedir. Sonuç 1'e yaklaştıkça görüntü benzerliği artmaktadır. Manhattan uzaklık ölçüsünde , gözlemler arasındaki mutlak uzaklıkların toplamı alınarak hesap yapılmaktadır [11].

$$manhattandistance = d(i,j) = \sum_{k=1}^{p} (|x_{ik} - x_{jk}|) i, j = 1,2,...,n; k = 1,2,...,p \tag{5.2}$$

Öklit uzaklığı ise iki nokta arasındaki doğrusal uzaklık olarak tanımlanabilir.

$$eucliddistance = d(i,j) = \sqrt{\sum_{k=1}^{p} (x_{ik} - x_{jk})^2)} i, j = 1,2,...,n; k = 1,2,...,p \tag{5.3}$$

Uzaklık ve benzerlik ölçüleri için fonksiyonlar **sklearn.metrics.pairwise** nesnesinde bulunur ve bu konuda detaylı bilgi `http://scikit-learn.org` adresinde bulunmaktadır.

5.3.3 Uygulama

Veri kümemisin yüklenmesi *Spyder* editöründe

```
from sklearn import datasets
digits = datasets.load\_digits()
```

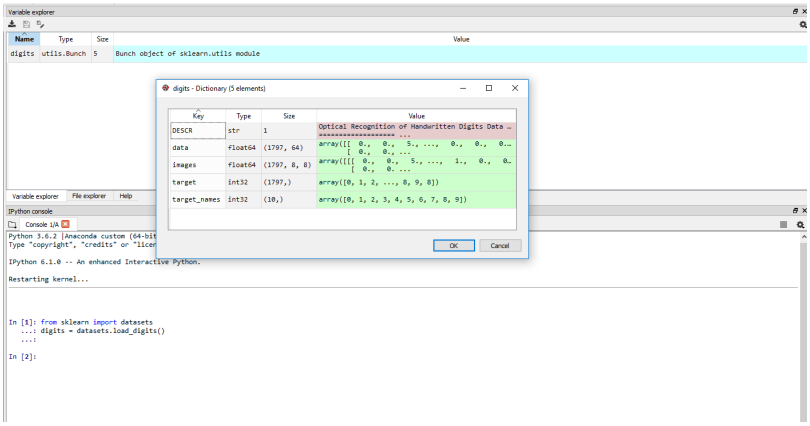
şeklinde girilerek sağlanmıştır. Şekil 5.3 'de veri kümesi yüklendikten spyder içersinde bulunan değişken tablosunda (variable explorer) 5 sözlük (dictionary) barındıran bir **digits** demeti (bunc) bulunmaktadır.

Digits veri kümesi 1797 elemandan oluşmaktadır (Tablo 5.3). Her bir eleman için data, images, target nitelikleri bulunmaktadır.

nitelik	**Açıklama**	**Tür**
images	Veri kümesindeki her bir elemanın matris halindeki 8x8 görüntüsü	float64
data	Veri kümesindeki her bir elemanın vektör haline getirilmiş hali	float64
target	her bir elemanın sınıfı	int32

Tablo 5.3: Digits veri kümesinin nitelikleri

Elemanları görüntü olarak göstermek için



Şekil 5.3: digits demeti

```
goruntuNo=1
import matplotlib.pyplot as plt
plt.imshow(digits.images[goruntuNo], cmap=plt.cm.gray_r,interpolation='nearest')
```

komutları kullanılmaktadır (Şekil 5.4).

Veri kümesi içersindeki matrix formundaki görüntülerin benzerliklerini hesaplamak için her bir elemanı için vektör haline dönüştürülmüş halini X dizisine (array)

```
X = digits.data
```

komutu ile aktarılır. Benzerini bulmak istediğimiz görüntünün vektor halini alıp işleme hazır hale getirmek için

```
goruntu=4
Y=X[goruntu].reshape(1,-1)
```

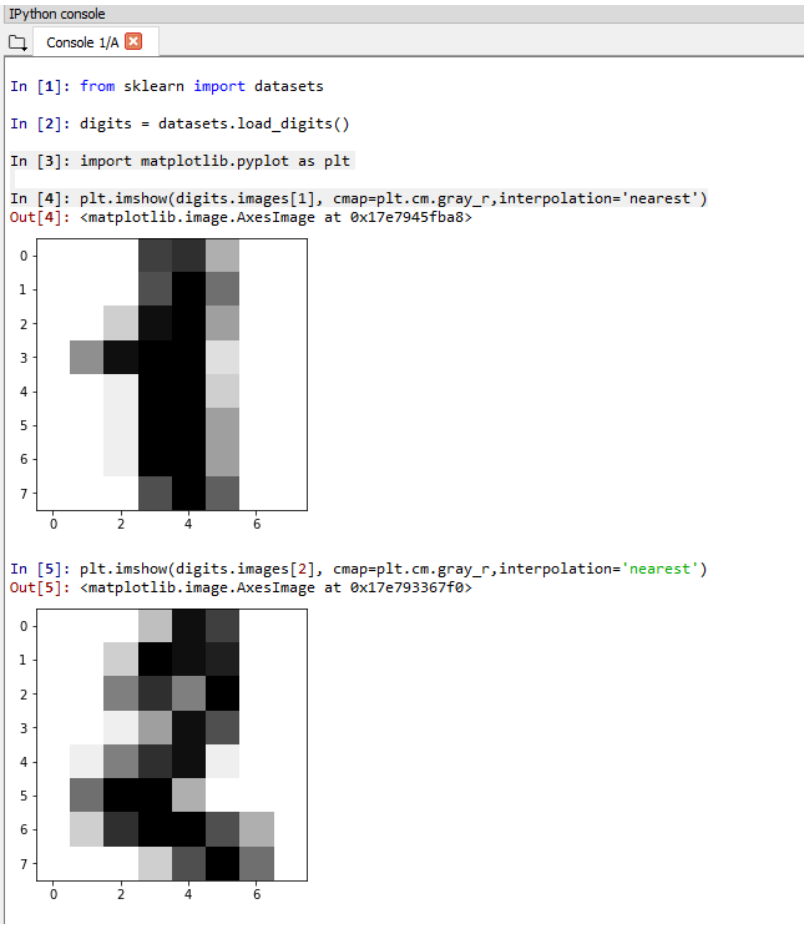
komutunu kullanılmaktadır. Örnek uygulamada 4 numaralı görüntü için işlem yapılmıştır. X dizisi bütün veri kümesinde bulunan elemanları vektör halinde bulundurmakta olup Y benzerlerini bulmak istediğimiz rakamı vektör halinde bulundurmaktadır. Kosinüs benzerlik denklemi 5.1 için

```
from sklearn.metrics.pairwise import cosine_similarity
coSim = cosine_similarity(Y, X)
```

kodu yazılabilir. Bu işlemin sonucunda elimizde benzerini bulmak istediğimiz görüntünün diğer görüntülere olana benzerlik değeri 0-1 arası değer olarak hesaplanacak ve *coSim* dizisine kaydedilmiştir. Benzerlik değerleri yüksekten düşüğe sıralamak ve ekranda göstermek için

```
cosf = pd.DataFrame(coSim).T
cosf.columns = ['similarity']
sirali=cosf.sort_values('similarity', ascending=False)
sirali=sirali.reset_index()
```

komutları kulanılmaktadır. Şekil 5.5'den görüleceği üzere sıralanmış olan dizinin ilk elemanı görüntünün kendisi olup benzerliği (similarity) 1'e eşittir. Örnekte benzerliği bulunması istenen 4 numaralı



Şekil 5.4: Digits veri kümesindeki görüntüler

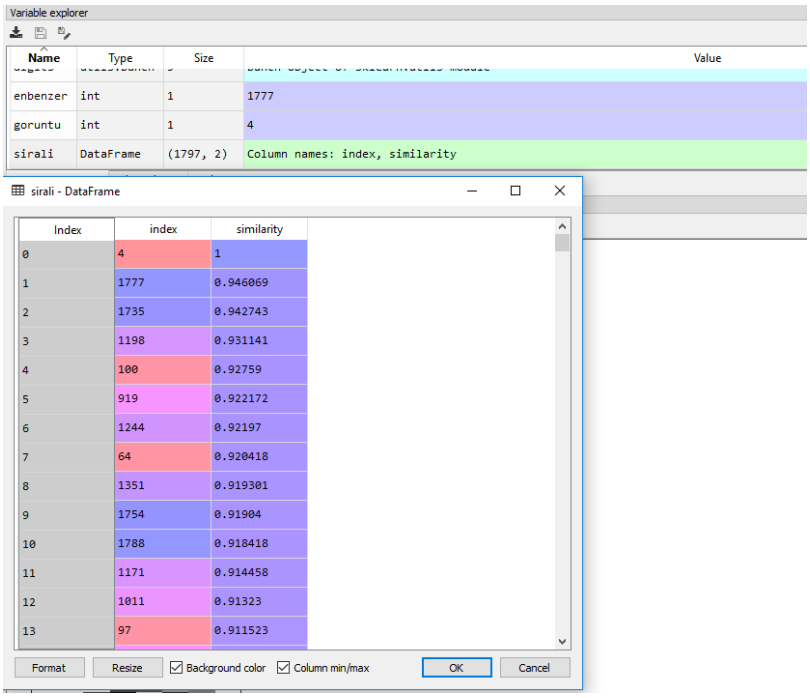
elemana en yakın örneğin benzerliği 0.946069 ile 1777 numaralı eleman olduğu görülmektedir.

Aynı şekilde uzaklık metrikleri için işlemler aynı şekilde yapılmaktadır. Öklit uzaklık metrği için kodumuz

```
from sklearn.metrics.pairwise import euclidean_distances
eucDis = euclidean_distances(Y, X)
eucDisf = pd.DataFrame(eucDism).T
eucDisf.columns = ['distance']
eucDissirali=eucDisf.sort_values('distance', ascending=True)
eucDissirali=sirali.reset_index()
```

Manhattan uzaklık metriği için kodumuz

```
from sklearn.metrics.pairwise import manhattan_distances
manDis = euclidean_distances(Y, X)
manDisf = pd.DataFrame(manDis).T
manDisf.columns = ['distance']
manDissirali=eucDisf.sort_values('distance', ascending=True)
manDissirali=sirali.reset_index()
```



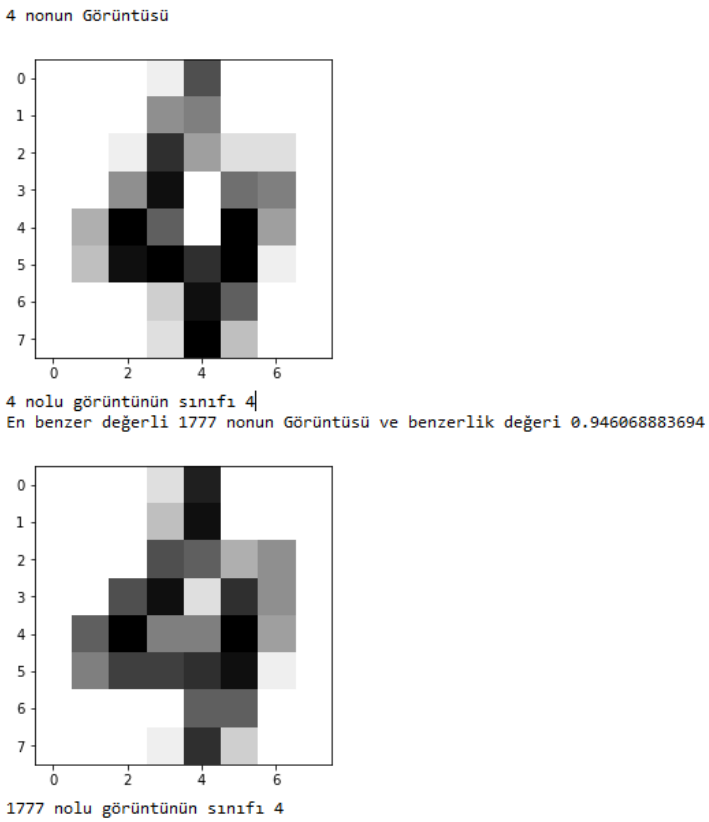
Şekil 5.5: Her bir görüntü indisine göre benzerlik değerlerinden bir kısım

şeklinde olmaktadır. Burada dikkat edilmesi gerek benzerlik ölçülerinde sıralama büyükten küçüğe olurken. Uzaklık ölçülerinde sıralama küçükten büyük olmasıdır. Bu bölümde anlatılan tüm kaynak kodlar ve fazlası yazarın github hesabından `https://github.com/mgezer` alınabilir.

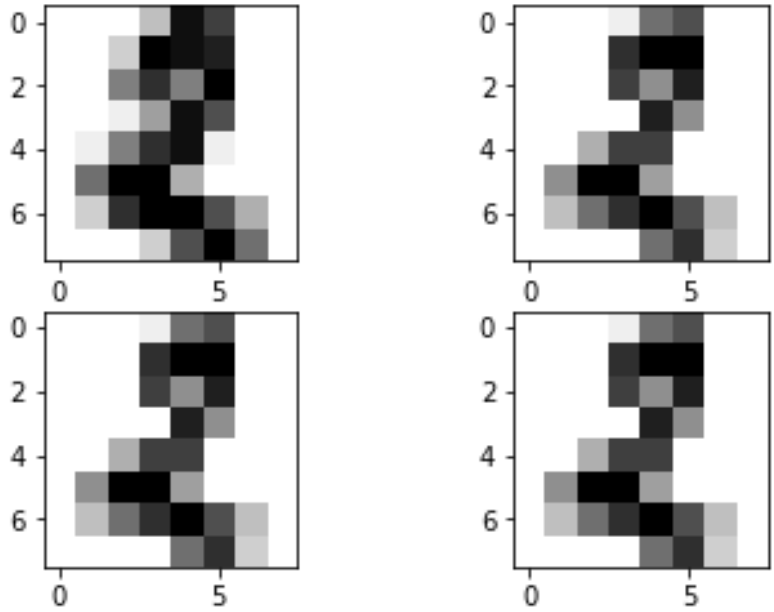
5.4 Sonuç

Bu bölüm kapsamında digit veri kümesi içersinde bulunan görüntülerdeki rakamların tanınması için benzerlik ve uzaklık ölçüleri kullanılmıştır. Şekil 5.7 de 2 nolu örnek için değişik metriklerde en iyi bulunan görüntülerin görünmektedir. şekil 5.8, şekil 5.9 ve şekil5.10 sırasıyla 5 rakam sınıfına ait olan 15 nolu görüntünün en iyi , 10. sıradaki ve 100. sıradaki benzerliklerine ait sonuçları göstermektedir. Bu şekillerden görülmektedir ki yapmış benzerlik ve uzaklık metrikleri ile yapılan sonuçlar ile uygun ve hızlı şekilde iyi sonuçlar alınmaktadır.

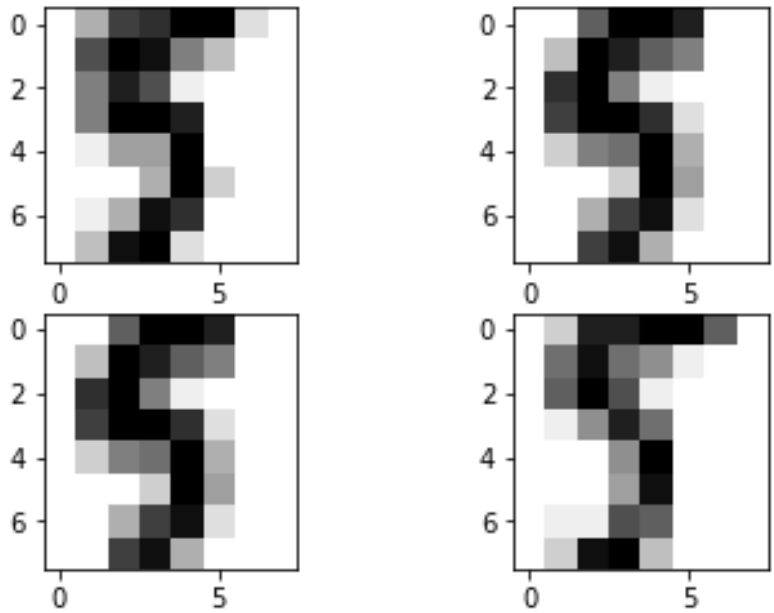
Okuyucunun kendisi geliştirmesi için kaynak kodlar sunulmuştur. Günümüzde Derin Öğrenme yöntemleri ile tanınma işlemleri çok boyutlu ve hacmi büyük veriler üzerinde çok daha etkin şekilde yapılabilmektedir.



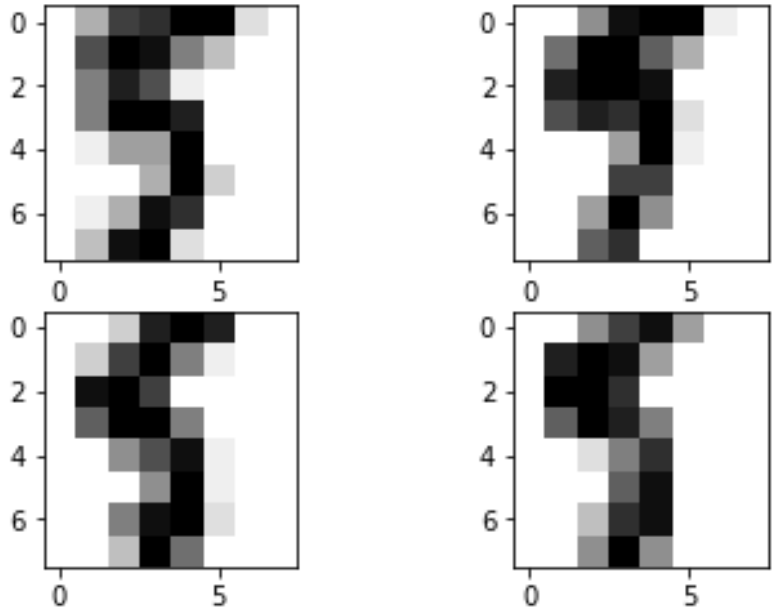
Şekil 5.6: Kosinüs benzerlik ölçütüne göre 4 numaralı görüntüye en yakın benzer görüntü



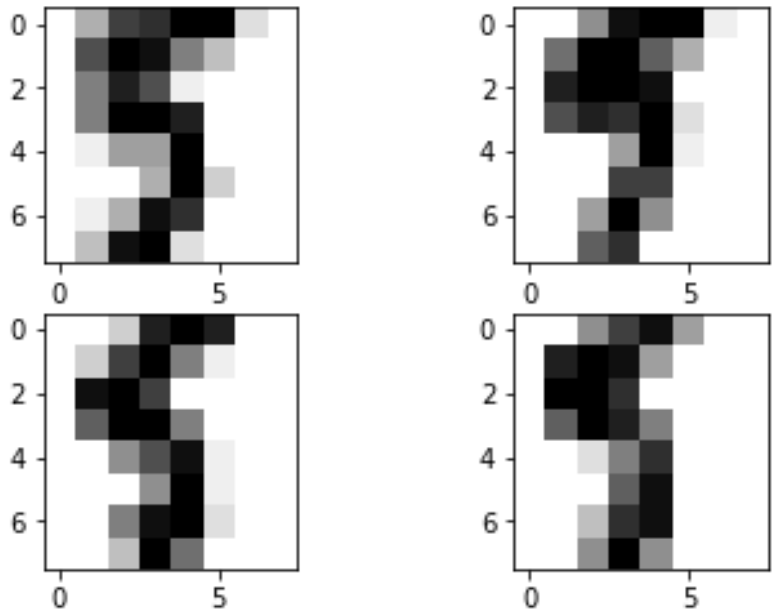
Şekil 5.7: Yukardan soldan sağa doğru 2 nolu görüntü,Kosinüs benzerliğine göre bulunmuş olan en yakın görüntü (nolu eleman), Aşağıdan soldan sağa doğru öklid uzaklığına göre görüntü (57 nolu eleman), manhattan uzaklık metriğine göre (57 nolu eleman)



Şekil 5.8: Yukardan soldan sağa doğru 15 nolu görüntü,Kosinüs benzerliğine göre bulunmuş olan en yakın görüntü (1568 nolu eleman), Aşağıdan soldan sağa doğru öklid uzaklığına göre görüntü (1568 nolu eleman), manhattan uzaklık metriğine göre (1192 nolu eleman)



Şekil 5.9: 5 rakamı olan yukardan soldan sağa doğru 15 nolu görüntü,Kosinüs benzerliğine göre bulunmuş olan 10. sıradaki görüntü (1575 nolu eleman), Aşağıdan soldan sağa doğru öklid uzaklığına göre görüntü (1659 nolu eleman), manhattan uzaklık metriğine göre (1643 nolu eleman)



Şekil 5.10: 5 rakamı olan yukardan soldan sağa doğru 15 nolu görüntü,Kosinüs benzerliğine göre bulunmuş olan 100. sıradaki görüntü (1185 nolu eleman), Aşağıdan soldan sağa doğru öklid uzaklığına göre görüntü (204 nolu eleman), manhattan uzaklık metriğine göre (1333 nolu eleman)

5.5 Kaynakça

[1] G. Oleś, "Revolution or evolution of traditional Business Intelligence concept," 2013.

[2] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," BM J. Res. Dev., vol. 3, no. 3, pp. 210–229, 1959.

[3] E. Alpaydın, Introduction to Machine Learning. Mit Press, 2014.

[4] T. M. Mitchell, Machine Learning. McGraw Hills, 1997.

[5] M. Beyeler, Machine Learning for OpenCV. Published by Packt Publishing Ltd., 2017.

[6] B. Jahne, Digital Image Processing: Concepts, Algorithms, and Scientific Applications. Springer, 1995.

[7] E. T. G. Oguz Güngör, "Dijital Görüntü İşleme." 2016.

[8] Python, "Python," Python, 2017. [Online]. Available: http://bit.ly/2x2rOcn. [Accessed: 17-Oct-2017].

[9] F. Alimoglu, "Combining Multiple Classifiers for Pen-Based Handwritten Digit Recognition," Institute of Graduate Studies in Science and Engineering Bogazici University, 1996.

[10] H. Akpınar, Data, Veri Madenciliği Veri Analizi. İstanbul: Papatya Yayıncılık, 2014.

[11] Ç. S. E. Yalçın Özkan, Biyoenformatik DNA Mikrodizi Veri Madenciliği. İstanbul: Papatya Yayıncılık, 2017.

5.6 Ekler

Kosinüs Benzerlik için yazar tarafından hazırlanmış olan kaynak kod

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Sep  5 13:27:01 2017
@author: mgezer
"""
## Digits veritabani ile benzer goruntu bulma
from sklearn import datasets
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
# Veri Kumesini yukleyelim
digits = datasets.load_digits()
#display_img adinda goruntu gosterecek bir fonksiyon tanimlanmasi
def goruntuGoster(goruntuNo):
plt.imshow(digits.images[goruntuNo], cmap=plt.cm.gray_r,
interpolation='nearest')
plt.show()
#  bazi veri kumesi elemanlarini ekranda gostereli
print("")
goruntuGoster(0)
goruntuGoster(1)
goruntuGoster(111)
# Her bir 8x8 lik goruntunun vektor haline X degiskenine aktar
X = digits.data
# Benzerlik Analizi yapilacak gorntu olan vektor sekilendiriyoruz
#satir vektorden sutuna ceviriyoruz
goruntu=4
Y=X[goruntu].reshape(1,-1)
# Cosine Benzerlik metrigini uyguluyoruz
coSim = cosine_similarity(Y, X)
"""
#sonucu Pandas Veri cercevresinin icine aliyoruz
ve en benzerden itibaren siralama yapiyoruz
"""
cosf = pd.DataFrame(coSim).T
cosf.columns = ['similarity']
sirali=cosf.sort_values('similarity', ascending=False)
sirali=sirali.reset_index()
#ekrana bastiriyoruz
#print(sirali)
#enbenzerin indis degerini aliyoruz
print(goruntu,"nonun Goruntusu")
goruntuGoster(goruntu)
print(goruntu,"nolu goruntunun sinifi",digits.target[goruntu])
enbenzer=int(sirali.iloc[1]['index'])
bdegeri=sirali.iloc[1]['similarity']
print("En benzer degerli",enbenzer,"nonun Goruntusu ve benzerlik degeri",
bdegeri)
goruntuGoster(enbenzer)
print(enbenzer,"nolu goruntunun sinifi",digits.target[enbenzer])
```

Kaynak Kod: Değişik uzaklık ve benzerlik ölçülerine göre karşılaştırma kodu

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Sep  5 13:27:01 2017
@author: mgezer
"""
## Digits veritabani ile benzer goruntu bulma
from sklearn import datasets
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics.pairwise import euclidean_distances
from sklearn.metrics.pairwise import manhattan_distances
goruntu=1221
# Veri Kumesini yukleyelim
digits = datasets.load_digits()
X = digits.data
# Benzerlik Analizi yapilacak gorntu olan vektor sekilendiriyoruz
#satir vektorden sutuna ceviriyoruz
Y=X[goruntu].reshape(1,-1)
# Cosine Benzerlik metrigini uyguluyoruz
coSim = cosine_similarity(Y, X)
cosf = pd.DataFrame(coSim).T
cosf.columns = ['similarity']
cosfSirali=cosf.sort_values('similarity', ascending=False)
cosfSirali=cosfSirali.reset_index()
#ekrana bastiriyoruz
#print(sirali)
#enbenzerin indis degerini aliyoruz
enbenzer=int(cosfSirali.iloc[1]['index'])
# Oklid uzaklik metrigini uyguluyoruz
eucDis = euclidean_distances(Y, X)

eucf = pd.DataFrame(eucDis).T
eucf.columns = ['similarity']
eucfSirali=eucf.sort_values('similarity', ascending=True)
eucfSirali=eucfSirali.reset_index()
eucenbenzer=int(eucfSirali.iloc[1]['index'])
# Oklid uzaklik metrigini uyguluyoruz
manDis = manhattan_distances(Y, X)

manDisf = pd.DataFrame(manDis).T
manDisf.columns = ['similarity']
manDisfSirali=manDisf.sort_values('similarity', ascending=True)
manDisfSirali=manDisfSirali.reset_index()
manenbenzer=int(manDisfSirali.iloc[1]['index'])
plt.figure(1)
plt.subplot(221)
plt.imshow(digits.images[goruntu], cmap=plt.cm.gray_r,
interpolation='nearest')
plt.subplot(222)
plt.imshow(digits.images[enbenzer], cmap=plt.cm.gray_r,
interpolation='nearest')
plt.subplot(223)
plt.imshow(digits.images[eucenbenzer], cmap=plt.cm.gray_r,
interpolation='nearest')
plt.subplot(224)
plt.imshow(digits.images[manenbenzer], cmap=plt.cm.gray_r,
interpolation='nearest')
plt.show()
```