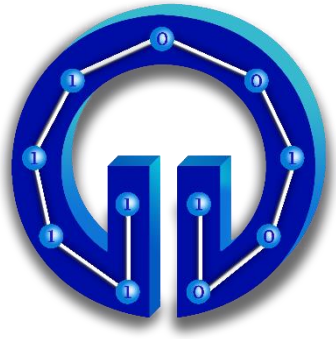
KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



YÜZ TANIMA SİSTEMİ

MÜHENDİSLİK TASARIMI

Adı SOYADI
EKREM AĞCA
FEYZANUR MEMİŞ
KADER NUR TEKİN
MERVE AYDIN
DANIŞMAN: DR. ÖĞR.ÜYESİ MURAT AYKUT

2021-2022 GÜZ DÖNEMİ

KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YÜZ TANIMA SİSTEMİ

MÜHENDİSLİK TASARIMI

Adı SOYADI
EKREM AĞCA
FEYZANUR MEMİŞ
KADER NUR TEKİN
MERVE AYDIN
DANIŞMAN: DR. ÖĞR.ÜYESİ MURAT AYKUT

2021-2022 GÜZ DÖNEMİ

IEEE Etik Kuralları
IEEE Code of Ethics

Mesleğime karşı şahsi sorumluluğumu kabul ederek, hizmet ettiğim toplumlara ve üyelerine en yüksek etik ve mesleki davranışta bulunmaya söz verdiğimi ve aşağıdaki etik kurallarını kabul ettiğimi ifade ederim:

1. Kamu güvenliği, sağlığı ve refahı ile uyumlu kararlar vermenin sorumluluğunu kabul etmek ve kamu veya çevreyi tehdit edebilecek faktörleri derhal açıklamak;
2. Mümkün olabilecek çıkar çatışması, ister gerçekten var olması isterse sadece algı olması, durumlarından kaçınmak. Çıkar çatışması olması durumunda, etkilenen taraflara durumu bildirmek;
3. Mevcut verilere dayalı tahminlerde ve fikir beyan etmelerde gerçekçi ve dürüst olmak;
4. Her türlü rüşveti reddetmek;
5. Mütenasip uygulamalarını ve muhtemel sonuçlarını gözeterek teknoloji anlayışını geliştirmek;
6. Teknik yeterliliklerimizi sürdürmek ve geliştirmek, yeterli eğitim veya tecrübe olması veya işin zorluk sınırları ifade edilmesi durumunda ancak başkaları için teknolojik sorumlulukları üstlenmek;
7. Teknik bir çalışma hakkında yansız bir eleştiri için uğraşmak, eleştiriyi kabul etmek ve eleştiriyi yapmak; hatları kabul etmek ve düzeltmek; diğer katkı sunanların emeklerini ifade etmek;
8. Bütün kişilere adilane davranmak; ırk, din, cinsiyet, yaş, milliyet, cinsi tercih, cinsiyet kimliği, veya cinsiyet ifadesi üzerinden ayırımcılık yapma durumuna girişmemek;
9. Yanlış veya kötü amaçlı eylemler sonucu kimsenin yaralanması, mülklerinin zarar görmesi, itibarlarının veya istihdamlarının zedelenmesi durumlarının oluşmasından kaçınmak;
10. Meslektaşlara ve yardımcı personele mesleki gelişimlerinde yardımcı olmak ve onları desteklemek.

IEEE Yönetim Kurulu tarafından Ağustos 1990'da onaylanmıştır.

ÖNSÖZ

Proje konusu olarak yüz tanımlama seçilmesindeki amaç makine öğrenmesi konusunda kendimizi geliştirmemizi sağlamaktı. Projede Yale veri setindeki yüzleri çeşitli makine öğrenmesi yöntemleriyle tanımlamaya çalıştık. Proje, LLE (locally linear embedding), LPP (locality preserving projections), Laplacian Eigenmaps gibi feature extraction yöntemleri ve ONPP (orthogonal Neighborhood Preserving Projections) gibi local feature extraction yöntemini çalışma mantığını öğrenmemizin yanı sıra veri ön hazırlık süreçleri ve classification kısmında temel k-NN hakkında bilgi edinmemizi sağlayarak, feature engineering yapmamıza olanak sağladı. Bu çalışmamızda bizi dinleyip yönlendiren ve zamanını ayıran değerli Danışman Hocamız Murat AYKUT 'a teşekkürlerimizi sunarız.

Adı SOYADI
EKREM AĞCA
FEYZANUR MEMİŞ
KADER NUR TEKİN
MERVE AYDIN
Trabzon 2022

İÇİNDEKİLER

ÖZET

YALE VERİ SETİNİN ÇEŞİTLİ BOYUT İNDİRGEME YÖNTEMLERİYLE SINIFLANDIRMA ÇALIŞMASI

Karadeniz Teknik Üniversitesi
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Dr. Öğr. Üyesi Murat AYKUT
2022, 34 Sayfa

Boyut Azaltma (Dimensionality Reduction) veri bilimi için oldukça önemli bir yöntem. Gerçek hayattaki veriler çok fazla boyuta (özniteliğe) sahip oluyor ve boyut büyüdükçe veri temizlemeden model kurmaya bütün süreçlerde harcamamız gereken zaman ve kaynaklar artıyor. Ne kadar çok boyuta sahip olursak görselleştirme de o kadar zor oluyor. 3 boyuttan sonrasını hayal etmek zor ama görsel olarak da yaptığımız çalışmanın bir karşılığı olsun insanlara anlatalım istiyoruz. Hemen her veri setinde bazı öznitelikler arasında yüksek korelasyon oluyor ve bu bizim gereksiz bilgiye sahip olmamıza ve modelimizde overfitting problemine sebep olabiliyor. Bu sebeplerden ötürü boyut indirgeme yöntemlerinin Yale veri seti üzerindeki etkisi incelenmiştir. Boyut küçültmenin en kolay yolu verimizi en iyi tanımlayan öznitelikleri bulup diğerlerini atmaktır (öznitelik seçimi — feature selection). Örneğin 100 boyuttan 10 tanesinin önemli olduğunu belirleyip kalan 90 özniteliği atmak ama bu tahmin edeceğiniz gibi bilgi kaybına sebep oluyor. Bizim uğraşmamız gereken şey ise öznitelik çıkarımı (feature extraction) yapmak -en az bilgi kaybıyla boyut küçültmek-. Bunu yapmak için verideki dağılımın maksimum varyansını-bilgisini tutan minimum sayıda değişken oluşturuyoruz. Eğer bir değişken her örnek için aynı değere sahip ise gereksiz bir değişkendir. Biz en yüksek varyansa sahip olan değişkenleri bulmalıyız.

Tasarım 3 aşamadan oluşmaktadır. Birinci aşamada veri ön işleme yapılmaktadır. Daha sonraki aşamalarda ise veri seti üzerinde LPP, LLE, LE, ONPP boyut indirgeme yöntemleri kullanılıp K-NN sınıflandırılması yapılarak Yale veri seti üzerindeki etkisi incelenmiştir.

1. GENEL BİLGİLER

1.1. Giriş

Bilgi güvenliğini sağlamak için günümüzde çeşitli biyometrik doğrulama sistemleri kullanılmaktadır. Bunlardan biri de son zamanlarda popüler olan yüz tanıma biyometrisidir.

Örüntü tanımlamada yüz görüntülerinin ön işleme tabi tutulmasıyla başlar. Ön işlemenin amacı, boyut azaltmak olarak ifade edebiliriz: boyut olarak sıkıştırmak ve değişkenlerinin kompakt temsillerinin keşfetmektir diyebiliriz.

Bu yazıda LPP (Locality Preserving Projections), LLE (locally linear embedding), Laplacian Eigenmaps gibi feature extraction yöntemleri ve ONPP (Orthogonal Neighborhood Preserving Projections) local feature extraction boyutluluk azaltma yöntemlerini inceliyoruz.

1. 2 Veri Seti

Veri seti, belirli bir konunun sayılar veya değerler koleksiyonu olarak toplanıp saklanmasıyla oluşturulan dosyalara denir. Yapay Zeka, insanların beyin çalışma mantığını makineye öğreterek, insanların düşünce ve temel becerilerini kazandırmak için yazılmış algoritmalardır. Bu algoritmalar insanların öğrenme tekniği gibi dış dünyadaki olayları, cisimleri görme ve tanıma ihtiyacı duyarlar. Fakat insanlar bu olayı uzun süre boyunca yapar ve zihinlerinde biriken bilgileri, nesneleri ancak bu süreç sonunda kullanılabilir hale dönüştürebilir fakat bu durum yapay zekada söz konusu değildir. Yapay zekaya önceden toplanmış verileri anlık sisteme vermekle sonuç elde etmek mümkündür. Bu toplanmış verilere “Veri Seti” deriz.

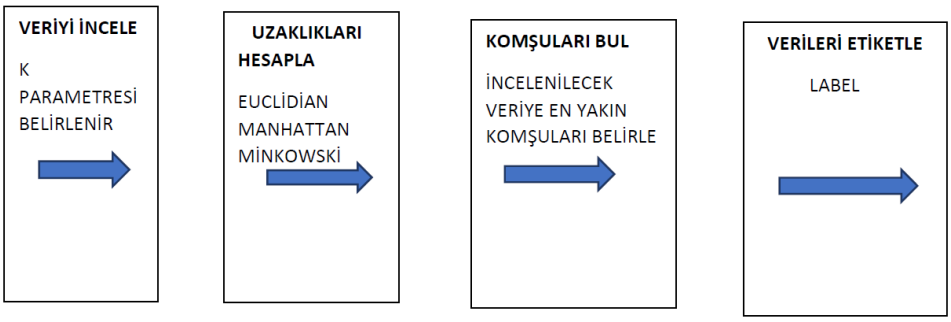
Yale Yüz Veri tabanı, 165 gri tonlamalı 15 kişilik GIF formatında görüntü içerir. Görüntüler gözlüklü, gözlüksüz, üzgün, uykulu, şaşırmış vb. farklı yüz ifadesi içermektedir.

Veri kümesinin boyutu 6.4 MB'dir ve her biri 576 görüntüleme koşulunda görülen 10 nesnenin 5760 tek ışık kaynağı görüntüsünü içerir.

Şekil 1: Veri setinden örnek görüntüler

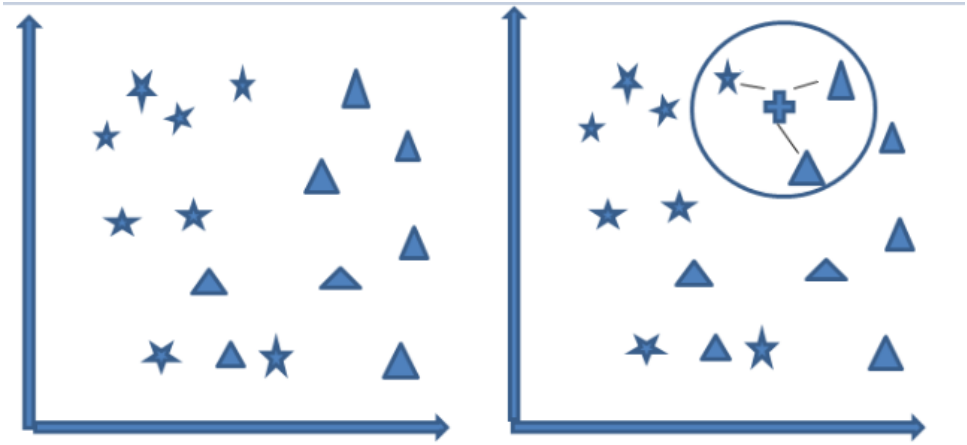
1.3 K-NN Algoritması

Sınıflandırma yöntemlerinden en basit olan danışmanlı bir yöntem diyebiliriz. Danışmanlı (supervised) Öğrenme: sınıf sayısı veya örneklerin (bazıları) hangi sınıfa dahil olduğu bilgisinin önceden sisteme verilmiş olan bir öğrenme şeklidir.



Şekil 2: En yakın komşuluk algoritma adımları

Algoritmanın adımlarını özetlersek:
• K değerinin belirlenmesi,
•Veri seti üzerindeki komşuların belirlenmesi,
•En yakın k komşunun etiket değerlerinin belirlenmesi,
• K komşu arasından çoğunluğun etiketine göre, yeni verinin etiketinin belirlenmesi.
K sayısı, genellikle üzerinde çalışılan probleme ve veri setine göre, kendine özgü olarak belirlenmektedir.
Örnek verirsek, aşağıdaki şekilde sol taraftaki şekil, 16 tane veri noktasının 2 boyutlu halini göstermektedir. 8 veri yıldız şekli 8 tanesi ise üçgen şekli olacak şekilde etiketlenir. Sağ taraftaki şekilde k değeri 3 (k=3) verildiğinde en yakın komşuluk algoritmasıyla artı şekliyle işaretlenmiş noktayı nasıl sınıflandıracağımızı göstermektedir.
K=3 olduğundan en yakın 3 komşuyu buluyoruz ve bu komşuların hangi şekil sınıfından olduğuna bakarız. Böylelikle 3 noktadan 2 ‘si üçgendir, bu nedenle artı şekli de üçgen olarak etiketlenir.

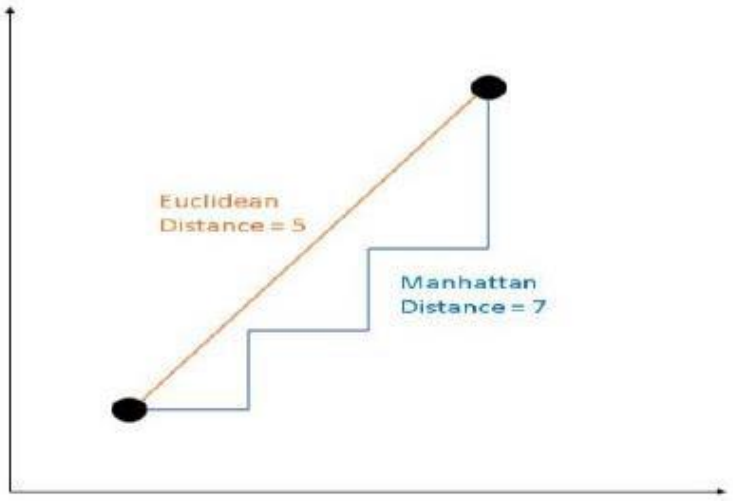


Şekil 3: Örnek verinin en yakın komşuluk ile sınıflandırma örnek grafiği

Noktalar arasındaki uzaklığı aşağıdaki uzaklık fonksiyonu ile hesaplarız.

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} \quad (1)$$

Bu denklemde X ve Y veri noktalarını n boyutların sayısını ve p güç parametresini temsil eder. Burada p=1 ise Manhattan mesafe, p=2 Öklid mesafesi olarak bilinir. Aşağıdaki görselle Manhattan ve Öklid mesafelerinin soyut hali gösterilmektedir.

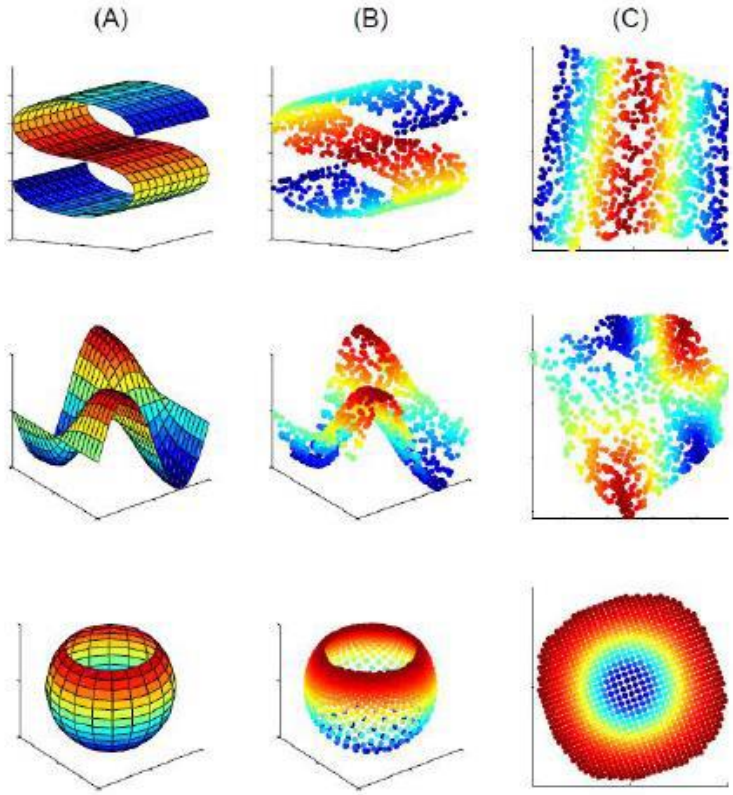


Şekil 4: Manhattan ve Öklid mesafeleri

Bizler projemizde Öklid mesafe ölçümünü kullanmaktayız.
Euclidean (Öklidyen) Uzaklık Hesaplaması; Öklid uzayında iki nokta arasındaki mesafeyi hesaplamak olarak tanımlanır.

1.4 LLE

İlk olarak boyut küçülmeden bahsedecek olursak, veri bilimi için önemli bir kavramdır. Gerçek hayattaki verilerimiz fazla boyutta oluyorlar. Bu nedenle boyut büyüdükçe veri temizleme, model kurmak için süreç aşamalarındaki harcamamız gereken zaman ve kaynaklarda artmaktadır. Boyutun fazla olması görselleştirmenin de zor bir hal almasına neden oluyor. Veri setlerinin çoğunda öz nitelikler arasında yüksek bir korelasyon oluyor ve bu durumda gereksiz bilgilere sahip olmamıza neden oluyor bu sebeplerden de anlaşıldığı üzere boyut küçültme veri bilimi için oldukça büyük bir öneme sahiptir.
Bu bölümde düşük boyutlu verilerin yüksek boyutlu verilerin komşuluklarını koruyarak yerleştirmelerini hesaplayan denetimsiz öğrenmenin nasıl gerçekleştiğini anlatacağız.
Bir manifolddan örneklendiği varsayılan veriler, daha düşük boyutlu küresel bir sisteme eşlenir. Verilerin konumu, yerel olarak doğrusal yeniden yapılandırılmaların simetrilerinden türetilir ve gömmenin doğru hesaplanması, seyrek bir öz değer problemine indirgenir.
LLE'nin uygulanması kolaydır ve yerel minimumları içermez fakat doğrusal olmayan sonuçlar üretebilir.

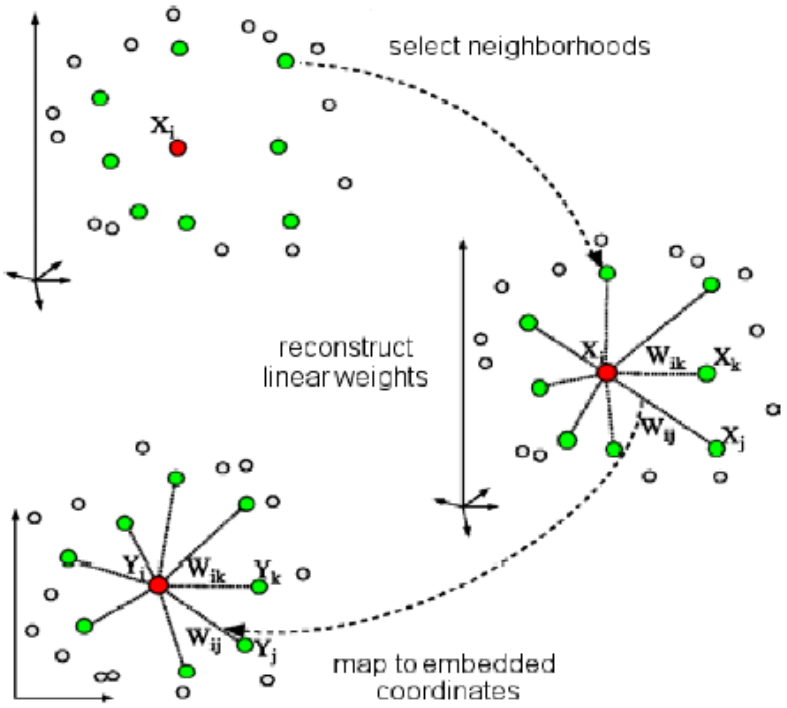


Şekil 5: Yerel doğrusal gömme ile boyutlandırılmış manifoldlar

Şekil 5 üzerinden açıklayacak olursak, (A) da ki şekiller iki boyutlu manifoldlar (B) ise bu manifoldlardan örneklenen üç boyutlu verilerdir. (C) de ise bu manifoldlardan aldığımız örnekleri doğrusal olmayan boyut indirgeme yöntemlerinden LLE uygulanmış halidir. (C) 'de gösterilen komşuluğu koruyan eşlemeleri ifade eder, kısacası verilerin iki boyutta nasıl gömülü olduğunu ortaya koyar.

LLE algoritması birbirine uzak veri noktaları arasındaki mesafe ve ilişkilere bakmaksızın çalışır. Verilerimizin her birinin D boyutlu N gerçek değerli vektörlerden (X vektörü) oluştuğunu ve temel bir manifolddan örneklendiğini varsayalım. Her veri noktasının ve komşularının manifoldun yerel olarak doğrusal bir parçası üzerinde veya yakınında olmasını bekleriz. LLE algoritması, adını yeniden yapılandırmaların doğasından alır: her yeniden yapılandırmaya yalnızca komşuların katkıda bulunması anlamında yereldir ve yeniden yapılandırmaların doğrusal alt uzaylarla sınırlı olması anlamında doğrusaldır.

Öncelikle şekil 6 ile algoritmanın temel mantığını anlamaya çalışalım.



Şekil 6: Yerel doğrusal gömme algoritmasının uygulanma adımları

•Her veri noktası başına k en yakın komşu tanımlanır.

$$E(W) = \sum_i \left| \vec{X}_i - \sum_j W_{ij} \vec{X}_j \right|^2 \qquad (2)$$

•Her veri noktasını Xi komşularından yeniden yapılandıran Wij ağırlıklarını hesaplarız. Yeniden yapılandırma hatalarını denklem (2)'deki maliyet fonksiyonu ile ölçülür. Ağırlıkları hesaplamak için denklem (2)'deki fonksiyonu minimize etmemiz gereklidir. İki şekilde mümkündür bunlar seyreklik kısıtlaması ve değişmezlik kısıtlamasıdır. Seyreklik kısıtlaması her veri noktasının Xi yalnızca komşularından yeniden yapılandırılmasıdır. Xj bu komşuluklardan değilse Wij=0' a zorlanır. Değişmezlik kısıtlanması ise ağırlık matrisinin satırlarının bire eşit olmasıdır ΣjWij = 1. Bu kısıtlamalar sonucu bulunan optimal ağırlıklar Wij, bir dizi en küçük kareler problemi çözülerek bulunur.
Verilerimizin d<<D boyutlu bir manifold üzerinde veya yakınında olduğunu düşünelim. Her bir komşuluğun yüksek boyutlu koordinatlarını manifold üzerindeki küresel iç koordinatlara eşleyen bir yeniden ölçeklendirme, döndürme ve ötelemeden oluşan doğrusal bir eşlemenin varlığını düşünelim. Tasarım gereği, yeniden yapılandırma ağırlıkları Wij, bu tür dönüşümlerde değişmez olan verilerin geometrik özelliklerini yansıtır. Bu nedenle giriş uzayımızdaki geometrinin karakterizasyonunun manifold üzerindeki yerel parçalar içinde aynı olmasını bekleriz. D boyutlarındaki Xi girişini yeniden yapılandıran aynı ağırlıklar Wij d boyutlarında gömülü manifold koordinatlarını da yeniden oluşturmalıdır.
LLE bir mahalle koruma eşlemesi oluşturur. Algoritmanın bu adımında her bir yüksek boyutlu Xi manifold üzerindeki koordinatlarını temsil eden düşük boyutlu bir çıktıya Yi 'ye eşlenir. Bu gömme maliyet fonksiyonunu en aza indirgeyebilmek için her çıktının Yi boyutlu koordinatları seçilerek yapılır.

Bu maliyet fonksiyonu (denklem 2) yerel olarak doğrusal yeniden yapılandırma hatalarına dayanmaktadır, ancak burada çıktıları Yi optimize ederken ağırlıkları Wij sabitliyoruz. Gömme doğrudan Wij ağırlık matrisinden hesaplanır. Orijinal girdiler Xi, algoritmanın bu adımında yer almaz. Böylece, gömme tamamen ağırlıklar Wij tarafından kodlanan geometrik bilgiler tarafından belirlenir. Amacımız, yüksek boyutlu girdiler Xi ile aynı Wij ağırlıkları ile yeniden oluşturulan düşük boyutlu çıktıları Yi 'leri bulmaktır.

$$\Phi(Y) = \sum_i \left| \vec{Y}_i - \sum_j W_{ij} \vec{Y}_j \right|^2 \quad (3)$$

Denklem (3)' teki ikinci dereceden formu en alt sıfır olmayan öz vektörleriyle en aza indirerek, en iyi Wij ağırlıkları tarafından yeniden oluşturulan Yi'leri hesaplarız.
Her veri noktaları için yeniden oluşturma ağırlıkları, diğer veri noktalarından bağımsız şekilde yerel komşuluklardan hesaplanırken, gömme koordinatlarının, tüm veri noktalarını birleştiren ve küresel bir işlem olan NxN öz çözücüler tarafından hesaplanır. Algoritma örtüşen yerel mahallelerden gelen bilgileri entegre ederek küresel yapıyı bu şekilde ortaya çıkarmış olur.

1.5 LAPLACIAN EIGENMAPS

Laplacian Eigenmaps de başka bir doğrusal olmayan boyut azaltma algoritmasıdır. Asıl amaç komşu örnekler arasındaki mesafeyi en aza indirmek ve komşu olmayan örnekler arasındaki mesafeyi en üst düzeye çıkarmaktır. En yakın komşuların grafiğini kullanarak sonucu modellemesi ile en yakın komşuların grafiği oluşturularak elde edilir. Bu tekniği diğer tekniklerden ayıran özellik ise Laplacian matrisini kullanmasıdır.
Birbirine yakın komşuların grafiği oluşturulur. Bunu için iki seçenek vardır.
1) Yakınlık matrisi: G = (V, E)
Burada V, x gözlemlerini içeren düğüm kümesidir ve E ise komşu düğümler arasındaki bağlantı kümesidir. Laplacian Eigenmaps tekniğinde bu grafiği oluşturmak için iki çok popüler teknik vardır;
a) G komşuluk Grafiği: Bu teknikte, her gözlem sırasında komşularla bağlantı kurulur ve Öklid mesafesine göre daha yakın olan, komşu kabul edilir.
Avantajları: Veri dağıldığında yerel geometriyi korur.
Dezavantajları: Birçok kez birkaç ilgili bileşen içeren bir grafik üretir. Bu sorunu çözmenin bir yolu, bağlı bir grafik oluşturan minimum ε değerini seçmektir.

$$N_{Eps}(x_i) = \{x_j \in D : d(x_j, x_i) \leq \epsilon\} \quad (4)$$

b) K- En yakın Komşular: Daha önce olduğu gibi, her gözlem komşularıyla bağlantılıdır ve Öklid mesafesine göre daha yakın olan komşu olarak kabul edilir.

$$N(x_i) = \{x_j \in D : d(x_j, x_i) \leq d(x_L, x_i)\} \quad (5)$$

Burada XL, xi' ye en yakın gözlem k' dır.
Avantajları: Genellikle ilgili grafikler üretir.

Dezavantajları: Veriler dağıldığında yerel geometri genellikle kaybolur.

2) Ağırlık Matrisi: Gözlemler veya düğümler arasındaki ilişkiler G'nin ağırlıklarına yansır, bunlar iki gözlem arasındaki benzerliği gösterir
a) HeatKernel(orLaplacianKernel): Komşu gözlemleri xı ve xj' yi birbirine bağlayan bağlantının ağırlığı wij olsun. Formül şekilde incelenirse eğer; xi, xj ile bağlantılı ise üstteki değeri alır. Başka bir değerlendirme sonucuna varılırsa sıfır değerini alır.

$$w_{ij} = \begin{cases} e^{-\frac{\|x_i - x_j\|^2}{\sigma}} \\ 0 \end{cases}$$

(6)

b) Basit Fikirle;

$$w_{ij} = \begin{cases} 1 \\ 0 \end{cases}$$

(7)

Yine xi ve xj komşu örnekleri birbirine bağlayan bir bağlantının ağırlığı wij olursa; xi ve xj'nin bağlantılı bulunması haline bir, diğer durumlar için sıfır değerini alır.
Bu kısımdan sonra G grafiğinin oluştuğunu ve bağlantılı olduğunu varsayacağız. Tek boyutlu uzayda(doğrusal) xi örneklerinin kümesini bulma problemini ele alacağız. y = (y1, y2, … ,yk) kümesini bulmak için ilk adım aşağıdaki fonksiyonu minimize etmektir.

$$\phi(y) = \sum_{i,j} w_{ij}(y_i - y_j)^2$$

(8)

ϕ (y)'nin ayrıntılarını incelersek; yi ve yj örnekleri komşu değilse, wij 0'a eşit sayılacak. Başka bir durumda, yi ve yj komşu değerler ise, wij yüksek bir ağırlık değerini temsil edecek ve bu yüzden minimize olacak ifade (yi − yj)'nin karesi olacaktır, sonuç olarak her iki örnek arasındaki Öklid mesafesinin karesi. Özetle, ϕ (y)'yi en aza indirerek, komşu örneklerin m boyutlu uzayda yakın görünmesini sağlıyoruz.
Burada laplacian matrisini şu şekilde tanımlayabiliriz;

L=D-W (9)

Burada W, n × n boyutlarının grafiğinin ağırlık matrisidir. D, n × n boyutlarında bir köşegen matristir. Bunun köşegen elemanları şu şekilde tanımlanır;

$$d_i = \sum_{j}^{J} w_{ij}$$

(10)

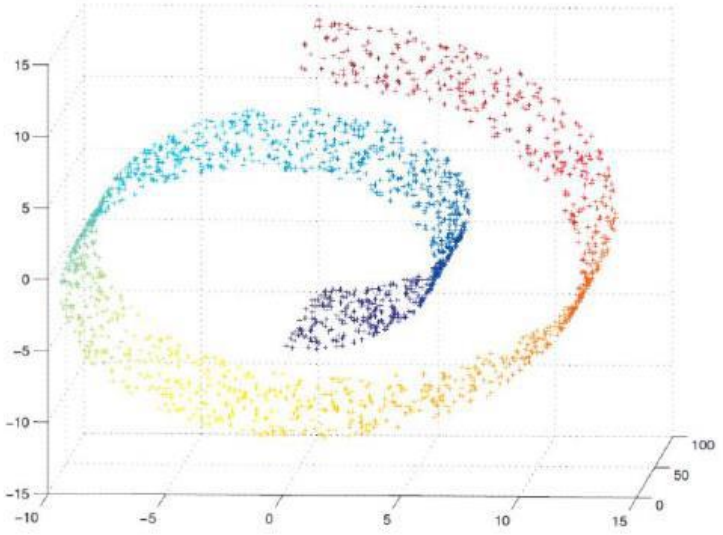
Yani, i köşesinin tüm bağlarının ağırlıklarının toplamı.
Özellikler
1. L pozitif yarı tanımlı simetrik matristir
2. En düşük özdeğer L 0'dır ve ilişkili özvektör bir vektördür
3. L gerçek ve negatif olmayan bir özdeğer vardır ve bunlar artan bir şekilde ayarlanır;
$0=\lambda 1 \leq \lambda 2 \leq \cdots \leq \lambda n$.
4. G pozitif ağırlıklara sahip yönlendirilmemiş bir grafiktir. Öz değerin çokluğu L grafiğin ilgili bileşenlerinin sayısına eşittir. Yani, grafiğimizin iki ilgili bileşeni varsa, iki eşit özdeğer 0'a eşittir.
5. Herhangi bir vektör için $f \in Rn$ bu yerine getirilir;

$$f^t L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \tag{11}$$

Optimizasyon için ϕ (Y)' yi minimize etmeye çalışırsak, Bu optimizasyon probleminin çözümü, formun genelleştirilmiş özdeğer probleminin en düşük özdeğerleri ile ilişkili özvektörlerden elde edilir;

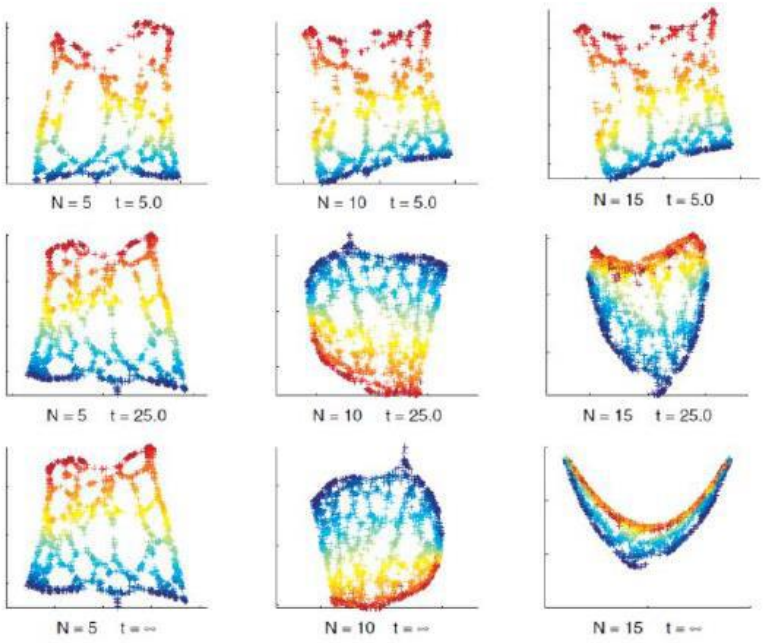
$$Ly = \lambda Dy \rightarrow (D - W)y = \lambda Dy \tag{12}$$

Karakteristik polinomdan özdeğerler elde edildiğinde; det (L -λD) ve özdeğerler sistem çözülerek elde edilir; (L − λD) y = 0 her bir özdeğeri çin.Bu sonuca, m = 1 ile yaptığımız gibi Lagrange çarpanları çözülerek ulaşılır.
Doğrusal olmayan bir boyut küçültme algoritmasının etkinliğini göstermek için kullanılan klasik veri kümelerinden biri, SwissRoll'dür.



Şekil 7: İsviçre Rulosu kümesinin boyutunun küçültülmesi

Şekil 8 k-n komşu algoritmasının grafiğindeki komşu sayısı olan farklı t ve N değerleri için yukarıdaki Swiss-Roll örneğini göstermektedir.



Şekil 8: Farklı komşulardan oluşan İsviçre Rulosu örnek grafikleri

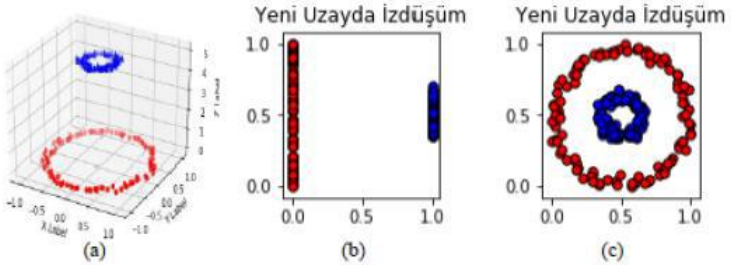
Ağırlıklar ikili değerler aldığında t = ∞ durumuna gelir. Daha küçük N değerleri için grafik t'den bağımsız olacak şekilde çok benzer gösterimlerin elde edildiği fark edilmiştir. Bununla birlikte, N arttıkça, t'nin küçük değerleri daha iyi bir şekil çıkarır.

1.6 LPP (Locality Preserving Projections)

Yerelliği Koruyan Projeksiyonları (LPP) algoritması veri setimiz üzerinde denediğimiz boyut indirgeme yöntemlerinden birisidir. Gerçek hayatta karşımıza çıkan çok boyutlu veri setlerinin hemen hepsinde değişkenler arasında yüksek korelasyon, anlamsız veya eksik değerler olması model oluşturma aşamasında modelimizin veri setimizi ezberlemesi (aşırı öğrenme- overfitting) sorunu ile bizi karşı karşıya getirir. Bu sorunları en aza indirgemek ve performansı arttırmak için boyut indirgeme işlemleri yaparız. Boyut indirgeme işlemlerinin temelde en uygun yaklaşımı veri setimizi en iyi anlatan, veri setimizin içerisinde en çok ağırlığa sahip olan özellikleri belirleyip diğerlerini elimine etmektir. Bu işleme feature selection yani özellik seçimi denir. feature selection işlemi yaparken özellikle dikkat edilmesi gereken durum bize gerekmediğini düşündüğümüz özellikleri atarken geri dönülemez bilgi kayıplarına neden olmamaktır. Feature extraction özellik çıkarımı işlemi ile en az sayıda bilgi kaybıyla boyut küçültme işlemi yapılarak bilgi kaybı sorununun üstesinden gelinmeye çalışılır. Feature extraction işlemi için verimizdeki dağılımın maksimum varyansını -bilgisini tutan minimum sayıda değişken tutuyoruz. En yüksek varyansa sahip değişkenleri bulmayı amaçlıyoruz. Bu bağlamda her örnek veri için aynı değere sahip bir değişken varsa bu bizim için anlamsız kullanışsız bir değişkendir. Biz projemizde YALE veri seti üzerinden bu işlemleri yapmış bulunmaktayız.Bu gibi yüksek boyutlu verilerin analizinde lineer boyutluluk indigeme yöntemlerinden olan yerelliği koruyan projeksiyon (lpp-locality preserving projections) yaklaşımı kullanılmıştır, Yerellik koruyucu izdüşüm olarak da adlandırılır.

PCA yönteminin bir alternatifi olan ve PCA yöntemini iyileştiren LPP yöntemi doğrusal(lineer) bir boyut indirgeme yöntemidir. Boyutu azaltılacak veri içerisinde verinin komşuluk ilişkisini koruyacak şekilde izdüşüm yapması ile PCA yönteminden ayrılır.

Farklılıklarının daha net ortaya konabilmesi için aşağıdaki görseller incelenebilir: (a) görselinde verilen 3 boyutlu ve 2 sınıfa sahip veri setin üzerinde ayrı ayrı PCA yaklaşımı ve LPP yaklaşımı uygulandığında sırasıyla (b) ve (c) görselleri ile belirtilen izdüşümler elde edilmektedir. PCA yaklaşımı ile elde edilen izdüşümde verinin uzaydaki asıl geometrik yapısının korunmadığını, LPP yaklaşımı ile elde edilen izdüşümde ise verinin uzaydaki asıl geometrik yapısının korunduğunu görebiliyoruz.



Şekil 9: (a) Üç boyutlu veri seti, (b) TBA izdüşümü ve (c) yerellik koruyucu izdüşümü.

Aykırı değerlerin izdüşüm üzerinde sahip oldukları etki de LPP ve PCA arasındaki bir diğer farktır. PCA varyans tabanlı bir yöntem iken, LPP verinin komşuluk ilişkisini korumaya odaklanır bu sebeple aykırı değerler PCA üzerinde büyük etkilere sebep olurken LPP aykırı değerlerden o kadar etkilenmemektedir. LPP yönteminin analizi yapılırken öncelikle denklem (13) minimize edilmeye çalışılır.

$$\min_{p} \sum_{i,j=1}^{n} ||y_i - y_j||^2 S(i,j), \qquad S(i,j) = e^{\frac{-||x_i - x_j||^2}{t}} \tag{13}$$

Denklem 13 ‘te **y**i ve **y**j, **x**i ve **x**j’nin yeni uzaya izdüşürülmüş biçimlerini ifade ederken **S** (i,j) ise verinin komşuluk ilişkisini ifade etmektedir. Komşuluk matrisi, LPP yönteminin kullanım şekline göre farklılık gösterebilir. Etiket bilgisi kullanıldığında, **x**i ve **x**j aynı sınıfta ise **S** (i,j) yukarıdaki formüldeki gibi hesaplanır. Eğer aynı sınıfta değiller ise **S** (i,j) 0’a setlenir. Etiket bilgisi kullanılmadığında ise, **x**i **x**j’ye en yakın k komşudan biriyse **S** (i,j) yine formüldeki gibi bulunur. Aksi durumda ise **S** (i,j) 0’a setlenir. Minimize edilecek denklem daha da basitleştirilerek denklem (14) elde edilir.

$$\begin{aligned}
&\tfrac{1}{2}\textstyle\sum_{i,j=1}^{n} \|w^T x_i - w^T x_j\|^2 S(i,j) \\
&= \tfrac{1}{2}\left(\textstyle\sum_{i,j=1}^{n} w^T x_i S_{ij} x_i^T w - 2\sum_{i,j=1}^{n} w^T x_i S_{ij} x_j^T w + \sum_{i,j=1}^{n} w^T x_j S_{ij} x_j^T w\right) \\
&= \textstyle\sum_{i,j=1}^{n} w^T x_i S_{ij} x_i^T w - \sum_{i,j=1}^{n} w^T x_i S_{ij} x_j^T w \\
&= \textstyle\sum_{i} w^T x_i D_{ii} x_i^T w - w^T X S X^T w \\
&= w^T X D X^T w - w^T X S X^T w \\
&= w^T X (D - S) X^T w
\end{aligned} \tag{14}$$

Eşitlik (14)’deki D matrisi diagonal bir matristir ve Dii = Σ**S** (i,j)j eşitliğiyle hesaplanır.

Minimizasyon problemi $\mathbf{w}^T\mathbf{XDX}^T\mathbf{w}=1$ kısıt koşulu altında eşitlik (15)'deki gibi genelleştirilmiş özdeğer özvektör problemine dönüştürülür. En düşük özdeğerlere karşılık gelen özvektörler aranan sonuçlardır.

$$\mathbf{X(D-S)X}^T\mathbf{w} = \lambda\mathbf{XDX}^T\mathbf{w} \tag{15}$$

Eşitlik (15)'de, en küçük özdeğerlere karşılık düşen özvektörler, eşitlik (16)'daki en büyük özdeğerlere karşılık düşen özvektörlerle aynıdır

.

$$\mathbf{XSX}^T\mathbf{w} = \lambda\mathbf{XDX}^T\mathbf{w} \tag{16}$$

LPP yöntemi komşuluk matrisinin hesaplanma türüne göre danışmanlı ya da danışmansız olabilir. LPP yöntemi danışmansız olarak kullanıldığında PCA'dan daha iyi sonuç vermektedir. Bunun nedeni LPP'nin verinin komşuluk ilişkisini korumaya çalışmasından kaynaklanmaktadır.

```
options['NeighborMode'] = 'Supervised' #danışmanlı
options['NeighborMode'] = 'KNN'#danışmansız
```

Orijinali danışmansız olduğu için danışmansız türü kullanılmıştır.
LPP yöntemi danışmanlı olarak kullanıldığında ise sınıf içi geometrik yapı korunmaya çalışılır. Danışmanlı yöntemde bir örnek sadece aynı sınıftaki örneklerle eşitlik (13)'den geçirilir.

1.7 Orthogonal Neighborhood Preserving Projections (ONPP)

Boyutsallığı azaltma sorunu, veri madenciliği, makine öğrenimi dahil olmak üzere birçok alanda ortaya çıkıyor. Bu ihtiyacın çözümlerinden biri olan ONPP doğrusal bir boyutsallık azaltma tekniğidir. Veri örneklerinin içsel komşuluk geometrisini ve küresel geometriyi korumaya çalışır. ONPP ile önerilen teknik Locally Linear Embedding (LLE)'e benzer bir şekilde ağırlıkların veriye dayalı bir şekilde oluşturulduğu, ağırlıklı bir veri grafiği oluşturur. Ancak LLE' nin aksine daha anlaşılır bir doğrusal dönüşüm kullanır. ONPP, Locality Preserving Projections (LPP) 'nin bazı özelliklerini paylaşır. Hem ONPP hem de LPP, veri topolojisini yakalamak için bir k-en yakın komşu grafiğine dayanır. Boyut azaltma matrisi V, indirgenmiş uzayda içsel komşuluk geometrilerinin tutarsızlığını yakalayan bir amaç fonksiyonunun minimize edilmesiyle elde edilir. Bu konu genellikle gözden kaçırılsada, yöntemin performansı için çok önemlidir ve Gauss ağırlıklarını kullanırken ciddi bir handikap olmaya devam etmektedir. Bir matrisin sütunlarıyla temsil edilen bir veri kümesini düşünün, X = [x_1, x_2, . . ., x_n] ∈ R^{mxn}. ONPP'nin ilk kısmı, LLE ile aynıdır ve her komşulukta bazı optimal ağırlıkların hesaplanmasından oluşur. Temel varsayım, her veri örneğinin en yakın k komşusu ile yerel olarak lineer bir manifold üzerinde yer almasıdır. Dolayısıyla, her veri örneği xi, onun doğrusal bir kombinasyonu ile yeniden oluşturulur. Yeniden yapılandırma hataları, amaç fonksiyonunu en aza indirerek ölçülür.

$$\mathcal{E}(W) = \sum_i \|x_i - \sum_j W_{ij} x_j\|_2^2. \quad (17)$$

Ağırlık W_{ij}, x_i örneğini komşularından $\{x_j\}$ yeniden oluşturmak için doğrusal katsayıyı temsil eder. Ağırlıklar için uygulanan kısıtlamalar aşağıdaki gibidir:

1. $W_{ij} = 0$, eğer x_j k en yakın komşulardan biri değilse,
2. $\Sigma_j W_{ij} = 1$, yani x_i, komşularının dışbükey bir kombinasyonu ile yaklaşık olarak bulunur.

ONPP Algoritması:

Giriş: Veri Kümesi X $\in R^{mxn}$ ve d: azaltılmış alan boyutu.
Çıkış: Embedding (Gömme) vektörleri Y $\in R^{dxn}$

1. $x_1, \cdots, x_n$ veri noktasının en yakın k komşusunu hesaplanmalıdır.
2. Her xi veri noktasının komşuları tarafından en iyi lineer yeniden oluşturulmasını veren W_{ij} ağırlıklarını hesaplanmalıdır.
3. Yansıtılan vektörler, $y_i = V^T x_i$,i=1,…,n hesaplanmalıdır. Burada V en küçük özdeğerlerle ilişkili aşağıda verilen (18) numaralı denklemin d + 1 özvektörlerinin hesaplanmasıyla belirlenir.

$$\tilde{M} = X(I - W^\top)(I - W)X^\top \quad (18)$$

X → Y'den açık bir doğrusal eşleme uygular. $y_i = V^T x_i$,i=1,…,n uygun şekilde belirlenmiş bir matris V $\in R^{mxd}$. V, ONPP matrisini belirlemek için indirgenmiş durumdaki her bir veri örneğinin y i kısıtlamasını uygular. Uzay, k komşusundan tam olarak giriş uzayındaki ile aynı ağırlıklar. Bu çözüme yol açar aşağıdaki optimizasyon probleminden

$$\text{M} = (I - W^\top)(I - W) \text{ and } \tilde{M} = XMX^\top$$

$$\begin{aligned}
\min_Y \Phi(Y) &= \min_Y \sum_i \|y_i - \sum_j W_{ij} y_j\|_2^2 \\
&= \min_{V \in R^{m \times d}} \sum_i \|V^\top x_i - \sum_j W_{ij} V^\top x_j\|_2^2 \\
&= \min_{V \in R^{m \times d}} \|V^\top X(I - W^\top)\|_F^2 \\
&= \min_{V \in R^{m \times d}} \text{tr}(V^\top X M X^\top V) \\
&= \min_{V \in R^{m \times d}} \text{tr}(V^\top \tilde{M} V) \, .
\end{aligned}$$

(19)

Yeni veri noktaları. Şimdi yeni bir test verisi örneği x_t düşünün bunun projelendirilmesi gerekiyor. Test örneği üzerine yansıtılarak uzay $y_t = V^T x_t$ boyutluluk azaltmayı kullanarak matris V. Bu nedenle, yeni projeksiyonun hesaplanması bir matris vektör ürününü basitleştirir. Hesaplamalı maliyet. ONPP' nin ilk bölümü şunlardan oluşur: k-NN grafiğini oluşturur. Bu, O(n^2) olarak ölçeklenir. Onun ikinci kısmı M'nin en küçük özvektörlerden birkaçının hesaplanmasını gerektirir. Pratikte bu matris hesaplanmaz. Bunun yerine, hesaplamak için yinelemeli teknikler kullanılır. $X(I - W)^T$ matrisinin karşılık gelen en küçük tekil vektörleri. Bu tekniklerin iç hesaplama çekirdeği ile ikinci dereceden ölçeklenen matris- vektör çarpımı eldeki matrisin boyutları.

Kernel ONPP

ONPP'nin çekirdeklendirilmiş bir versiyonunu formüle etmek mümkündür. Çekirdekler, Destek Vektör makineleri (SVMs) bağlamında yaygın olarak kullanılmaktadır. Esasen, doğrusal olmayan bir haritalama kullanırız Φ: $R^m \rightarrow$ H. Özellik uzayındaki dönüştürülmüş veri kümesi H, Φ(X) = [$\Phi_{(x1)}$, $\Phi_{(x2)}$ . . ., $\Phi_{(xn)}$ ile gösterilir. Ayrıca, özellik alanı ile ilişkili çekirdek k (x, y) tarafından indüklenen Gram matrisi K_{ij} = k (x_i,x_j) = $\Phi_{(xi)}$, $\Phi_{(xj)}$ ile de belirtilmektedir. d = 1 durumu düşünüldüğünde, veri örnekleri bir $y_i = V^T x_i$,doğrusu üzerinde yansıtılır. Ardından, özellik uzayındaki optimizasyon probleminin aşağıda bulunan (20) numaralı denklemdeki gibi formüle edildiği gözlemlenmektedir.

$$\min_v v^\top \Phi(X) M \Phi(X)^\top v \tag{20}$$

Burada v= $\Sigma_{i=1}^{n} a_i \Phi_{(xi)}$ = Φ(X)α eşitliği söz konusudur. Daha sonra yukarıdaki optimizasyon problemi aşağıda bulunan (21) numaralı denklemdeki şekilde yeniden yazılır.

$$\min_\alpha \alpha^\top \Phi(X)^\top \Phi(X) M \Phi(X)^\top \Phi(X) \alpha = \min_\alpha \alpha^\top K M K \alpha. \tag{21}$$

Bu nedenle, α, en küçük özdeğeriyle ilişkili KMK'nın özvektörü olmalıdır. İndirgenmiş uzayın keyfi bir d boyutunun genel durumu için, en küçük özdeğerleriyle ilişkili KMK'nın $a^{(1)}$, $a^{(2)}$,. . ., $a^{(d)}$ özvektörlerini hesaplıyoruz. Ardından, yeni bir xt noktasının izdüşümü, bileşenleri v(i)'ler boyunca hesaplanarak hesaplanır. Özellikle, x_t'nin $v^{(i)}$ ile nokta çarpımı şu şekilde hesaplanır:

$$\langle v^{(i)}, \Phi(x_t) \rangle = \langle \sum_{j=1}^{n} \alpha_j^{(i)} \Phi(x_j), \Phi(x_t) \rangle = \sum_{j=1}^{n} \alpha_j^{(i)} k(x_j, x_t). \tag{22}$$

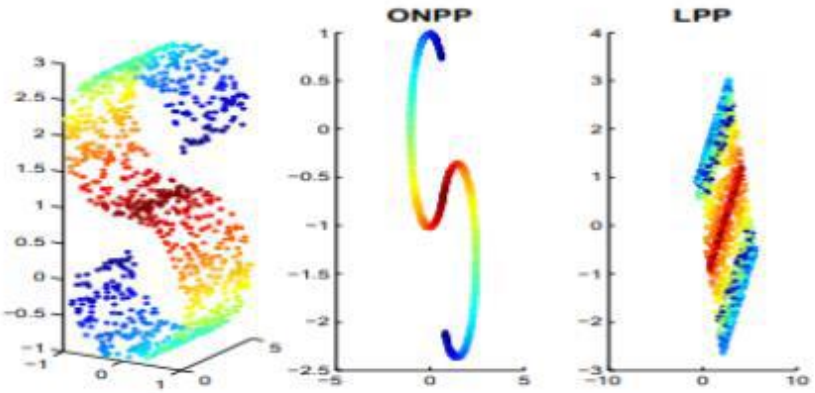
Bu aşamada basitlik için d = 1 durumunu tekrar ele almak gereklidir. Eğitim noktalarıyla ilgili olarak, v özvektörü boyunca izdüşümlerin y=Ka ile verildiği gözlemlenmektedir. Ardından, optimizasyon probleminin (5) tam olarak LLE tarafından çözülen özdeğer

problemi olan $min_y y^T$My olarak yeniden yazılmaktadır. Bu nedenle, LLE (doğrusal olmayan), özellik alanı H'de örtük olarak ONPP (doğrusal) gerçekleştiriyor olarak görülebilir.

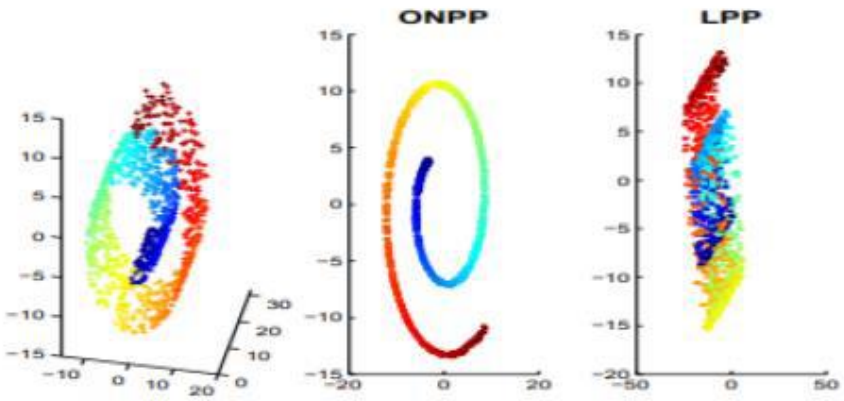
Deneysel Sonuç

Sentetik Data

Üç sentetik veri kümesi kullanılmaktadır. Bunlar: S-eğrisi, Swissroll ve Delinmiş küredir. Şekil 10 ve Şekil 11, s-eğrisi ile swissroll veri kümesindeki ONPP ve LPP yöntemleri ile elde edilen iki boyutlu projeksiyonları göstermektedir. Her iki algoritmada da k = 12 kullanılmaktadır. Her iki yöntemin de yerelliği koruduğu ve bunun renk gölgelemesiyle gösterildiği görülmektedir. Bununla birlikte, ONPP, manifoldun yüksek boyutlu uzayda nasıl katlandığı hakkında bilgi ileten doğru bir izdüşüm sağlayarak yerel ve küresel geometrik özellikleri de korumaktadır.

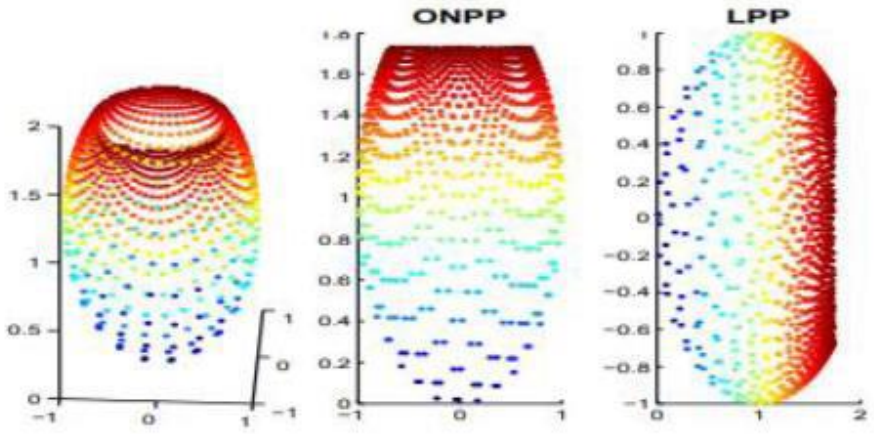


Şekil 10: S-curve'e uygulanan Onpp ve Lpp



Şekil 11: Swissroll'a uygulanan Onpp ve Lpp.

Şekil 12’de delinmiş küre üzerindeki her iki algoritmanın sonucu görülmektedir. Bu veri kümesi, LPP'nin yerelliği korumasına rağmen mahallelerin yerel geometrilerini bozduğu doğrulanmaktadır. Öte yandan, ONPP sadece yöreye değil, aynı zamanda yerel geometrilere de saygı duymaktadır. ONPP’'nin bu özelliği veri görselleştirme amaçları için çok önemlidir.

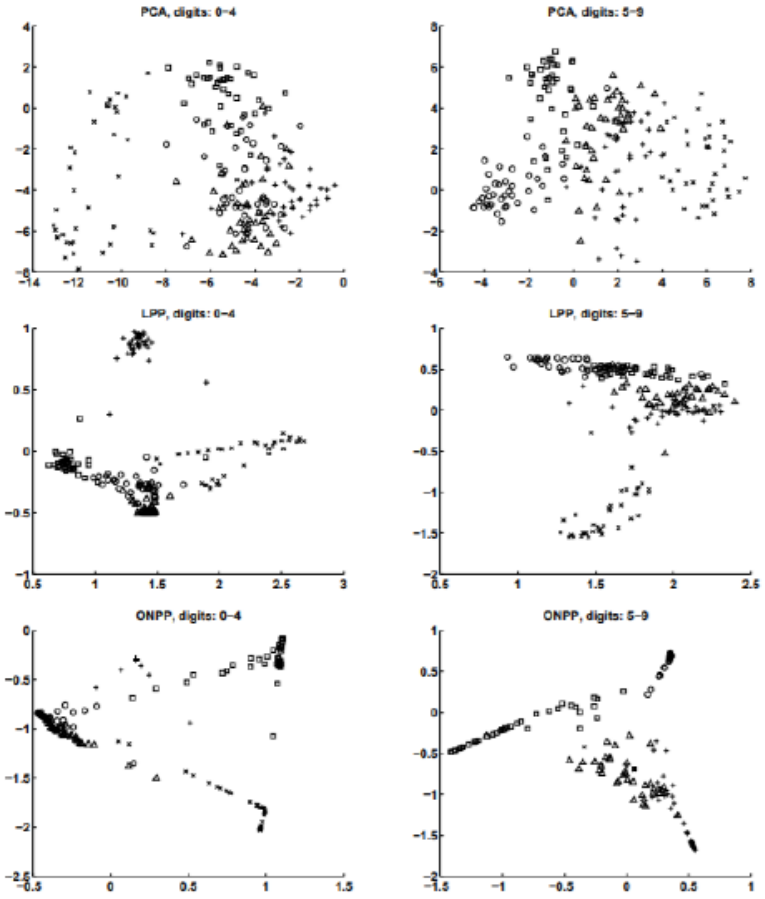


Şekil 12: Onpp ve Lpp’ nin Delinmiş Küre üzerinde uygulanması

Sayısal Görselleştirme

Veri seti içerisinde ki her sınıftan alınan görüntü örneği, sözlük bilimsel olarak 320 uzunluğunda yüksek boyutlu bir vektör olarak temsil edilir. Veri seti iki boyutlu uzayda yansıtılmaktadır ve sonuçlar Şekil 13‘te gösterilmektedir. Hem ONPP hem de LPP için k = 6 kullanılmaktadır. Sol paneller '0'-'4' rakamlarına ve sağ paneller '5'- '9' rakamlarına karşılık gelir.

PCA varyansı en üst düzeye çıkarmayı amaçladığından, PCA'nın projeksiyonlarının yayıldığını gözlemlenir. Ayrıca, farklı basamak sınıfları büyük ölçüde örtüşüyor gibi görünmektedir. Bu, PCA'nın veriler arasında ayrım yapmada başarılı olmadığı anlamına gelmektedir. Öte yandan, aynı sınıfa ait örnekler birbirine yakın haritalandığından, ONPP ve LPP'nin daha anlamlı projeksiyonlar verdiği gözlenmiştir. Çünkü bu yöntemler yerelliği korumayı amaçlamaktadır.



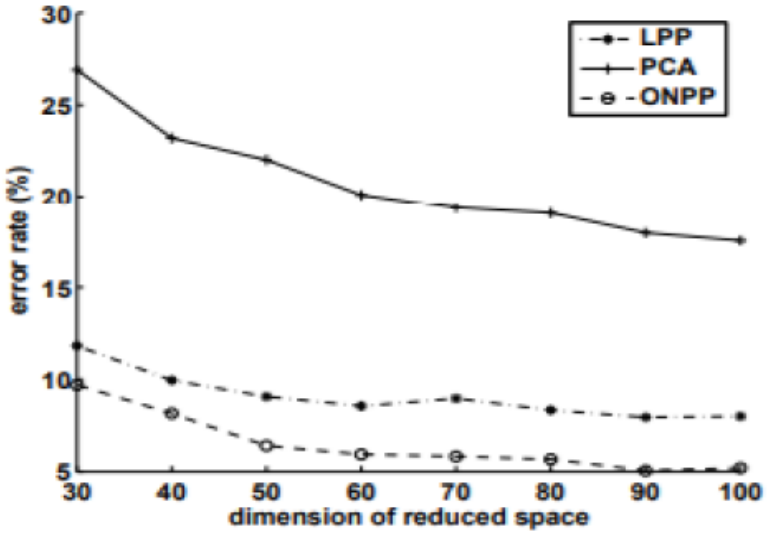
Şekil 13: Rakamların iki boyutlu izdüşümleri

Yüz Tanıma

Yüz tanıma için, denetimli bir ortamda ONPP ve LPP algoritmaları çalıştırılmaktadır. Gauss ağırlıklı bir LPP kullanılmaktadır. Gauss zarfının σ genişliği, ilk önce 1000 noktanın rastgele örneklenmesi ve daha sonra aralarındaki ikili mesafenin hesaplanmasıyla belirlenmektedir. Daha sonra σ, bu ikili mesafelerin ortancasının yarısına eşit olarak ayarlanmaktadır. Bu, σ değeri için iyi ve makul bir tahmin vermektedir.

Her bir birey için 8 farklı yüz ifadesi ve değişken aydınlatma koşulları altında 126 özne içeren AR yüz veritabanının [8] bir alt kümesini kullanılmaktadır. Eğitim setini denek başına 4 farklı yüz ifadesi/pozdan oluşan rastgele bir alt küme ile oluşturulmakta ve kalan 4'ü test seti olarak kullanılmaktadır. Test setinin 20 rastgele denemesi boyunca ortalama hata oranı çizilmektedir.

Sonuçlar Şekil 14'te gösterilmiştir. ONPP'nin diğer yöntemlerden daha iyi performans gösterdiğini gözlemlenmektedir. Ayrıca Tablo 1, her bir yöntemle elde edilen en iyi hata oranını ve azaltılmış alanın karşılık gelen d boyutunu bildirmektedir. PCA'nın (denetimsiz), denetlenen diğer yöntemlere göre daha düşük performansa sahip olduğu görülmektedir. ONPP'nin test edilen tüm yöntemlerin en iyisi olduğu görülmektedir.



Şekil 14: Küçültülmüş d boyutuna göre hata oranı

	dim	Error (%)
PCA	100	17.53
LPP	90	7.96
ONPP	90	**5.05**

Tablo 1: Veri Set üzerinde tüm yöntemlerle elde edilen en iyi hata oranı

Bu yazıda tanıtılan Ortogonal Komşuluk Koruma İzdüşümleri (ONPP), yüksek boyutlu veri örneklerinin sadece lokalitesini (Yerelliği) değil aynı zamanda yerel ve global geometrisini de koruma eğiliminde olacak bir lineer boyutluluk azaltma tekniğidir. Denetimli bir yönteme genişletilebilir ve ayrıca çekirdek teknikleriyle birleştirilebilir. Sonuç olarak, ONPP 'nin veri görselleştirme için çok etkili olabileceği gözlemlendi.

2.PROJE TASARIMI

2.1 Gereksinim Analizi

Proje Fikrinin Hedef Kitlesi
(Kullanıcılar) Bankalar, Elektronik para transferlerinde kullanılan web siteler, mobil uygulamalar (Kripto para borsası uygulamaları), Havalimanlarında, sınır kapılarında, Futbol sahası girişleri, Binalara, tesislere ve ofislere, Hastanelerde ve sigorta kuruluşlarında, Devlet dairelerinde, Tatil köyü, otel, fabrika, depo şantiye büyük mağaza ve alışveriş merkezi gibi alanlar bizim kullanıcılarımızdır.

Problem ve Sorun Tanımlama
Günümüzde güvenlik sistemleri ne kadar hızlı gelişiyor olsa da aynı hızda kırılabilecek bir düzene de sahip. Güvenlik sistemlerinin kişileştirilmesi ya da daha fazla bireysel nitelikler kazanması güvenlik sisteminin işlevini artırmakta büyük etkendir. Kişiselleştirilmiş sistemler güvenlik açısından çok gereklidir.

Çözüm
Telefonların kilidini açma bu teknoloji, kişisel verileri korumak için güçlü bir yöntem sunar ve telefonun çalınması durumunda hassas verilere erişilmesini engeller. Apple, rastgele bir yüzün sizin telefonunuzun kilidini açma olasılığının milyonda bir olduğunu beyan etmiştir. Havaalanları ve sınır denetimi Yüz tanıma teknolojisi, dünyanın dört bir yanındaki havaalanlarında görmeye alıştığımız bir manzara haline geldi. Giderek daha fazla sayıda yolcu biometrik pasaport kullanıyor ve kapıya daha hızlı ulaşmak için genelde uzun olan pasaport kontrolü sırasını atlayıp bunun yerine otomatik e-pasaport denetiminden geçiyor. Yüz tanıma, bekleme sürelerini azaltmanın yanı sıra havaalanlarının güvenliği artırmasına da olanak tanıyor. ABD İç Güvenlik Bakanlığı, yüz tanımanın 2023 yılı itibarıyla yolcuların %97'sinde kullanılacağını tahmin ediyor. Havaalanları ve sınır geçişlerinin yanı sıra teknoloji, Olimpiyatlar gibi büyük ölçekli etkinliklerde güvenliği artırmak için de kullanılıyor. Kayıp kişilerin bulunması Yüz tanıma, kayıp kişileri ve insan kaçakçılığı kurbanlarını bulmak için de kullanılabilir. Kayıp kişilerin veri tabanına eklendiğini varsayalım. Bu durumda bu kişiler ister hava alanı, ister mağaza, ister başka bir kamuya açık alanda olsun, yüz tanıma tarafından tanındığı anda emniyet güçlerine haber verilebilir. Mağaza suçlarının azaltılması Yüz tanıma; bilinen dükkan hırsızları, organize mağaza suçluları veya dolandırıcılık geçmişine sahip insanlar mağazalara girdiğinde onları tanımak için kullanılır. Potansiyel olarak tehdit oluşturan müşteriler mağazaya girdiğinde, kayıp önleme ve mağaza güvenliği yetkililerinin haberdar edilebilmesi için kişilerin görüntüleri geniş suçlu veri tabanlarıyla eşleştirilir. Bankacılık Biyometrik çevrimiçi bankacılık, yüz tanımanın diğer bir faydasıdır. Müşteriler, tek seferlik parolalar kullanmak yerine akıllı telefon ya da bilgisayarlarına bakarak işlemleri onaylayabilir. Yüz tanıma sayesinde korsanların ele geçirebileceği bir parola olmaz. Korsanların fotoğraf arşivinizi ele geçirmesi durumunda, biyometrik örnek kaynağının canlı bir insan mı yoksa sahte bir temsil mi olduğunu belirlemek için kullanılan "canlılık" algılama tekniği, fotoğraflarınızın sizi taklit etme amacıyla kullanılmasını (teoride) engeller. Yüz tanıma teknolojisi, banka kartlarını ve imzaları mazide bırakabilir. Öğrenci ve çalışan devamsızlığını izleme Öğrencilerin dersleri asmadığından emin olmak için Çin'deki bazı eğitim kurumlarında yüz tanıma kullanılıyor. Tabletlerle öğrencilerin yüzleri taranıp veritabanındaki görüntülerle göre FG eşleştiriliyor ve kimlikleri doğrulanıyor. Daha geniş çaplı olarak yüz tanıma teknolojisi, işverenlerin çalışan

devamsızlığını takip edebilmesi için çalışanların iş yerine giriş ve çıkışında kullanılabilir. Sürücüleri tanıma bu tüketici raporuna göre otomobil şirketleri, yüz tanıma teknolojisinin araba anahtarlarının yerini alması için çalışmalarını sürdürüyor. Bu durumda yüz tanıma teknolojisi, araca erişmek ve aracı çalıştırmak için anahtar yerine kullanılacak ve sürücülerin koltuk ve ayna konumu ayarları ile radyo istasyonları ön ayarlarını hatırlayacak.

Kullanılacak Teknolojiler
Programlama Dili: PYTHON
Programlama Ortamı: Anaconda Spyder
Veri Seti: Yale Yüz Veritabanı
Yöntemler: LLE (locally linear embedding)
LPP (locality preserving projections)
Laplacian Eigenmaps
ONPP (Orthogonal Neighborhood Preserving Projections)

Classification kısmında temel K-NN kullanılacaktır.
Önümüzdeki dönem için ise Deeplearning tabanlı mimarilerden Konvolüsyon, CNN, RNN kullanılacaktır.

Kodlama Standartları
- Genel isimlendirme şekilleri
- Sekmeleme, hizalama
- Yorum satırları ve açıklamalar
- İstisnai durum ve hata yönetimi
- Dizi/koleksiyon kullanımı

Riskler
• En büyük dezavantajı insan yüzünün zaman içerisindeki değişimidir.
• Bir insanın yüzü yaşın ilerlemesiyle biyolojik değişime uğrar ve yüz ölçülerindeki değişim sistemin başarısızlığını arttırır.
• Diğer bir etken, bir kaza veya herhangi bir olay sonucunda yüzde gelebilecek gözle görülür bir değişim sonucunda bu yine sistemin başarısızlığını arttırır.
• Günümüz teknolojisindeki bilgi güvenliğinin önemini düşünürsek eğer, yüz tanıma kişi bazlı önemli derecede güvenlik zafiyetine sebebiyet verir.

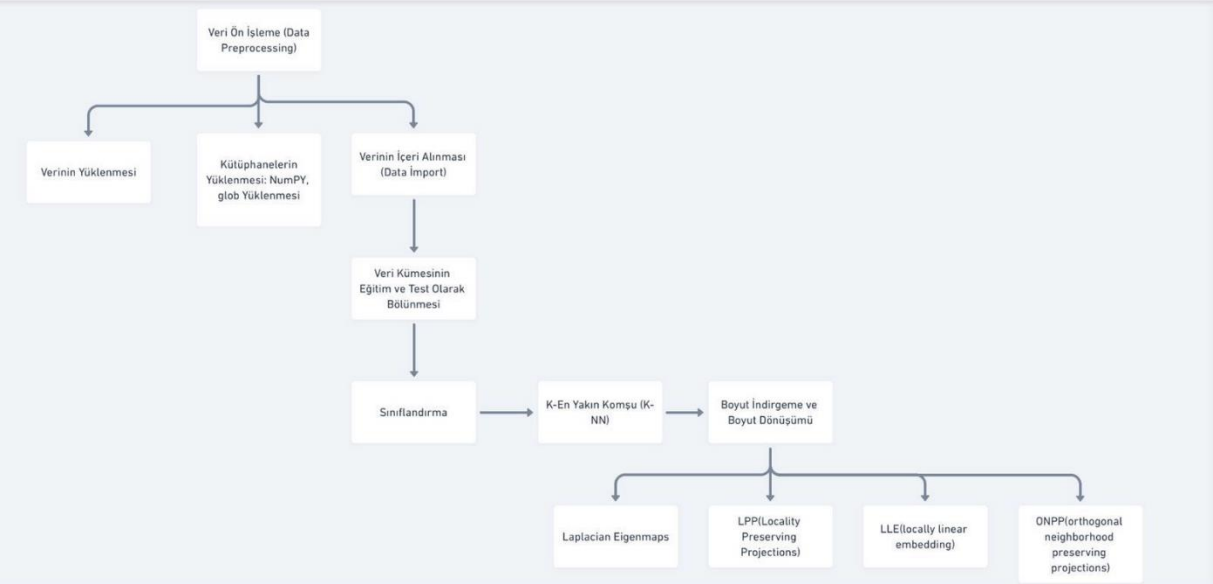
Proje Beyanı
“Tasarım Projesi” /” Bitirme Projesi” kapsamında “Mühendislik standartları ve gerçekçi koşullar”, “Proje yönetimi”, “risk yönetimi” ve “değişiklik yönetimi” gibi iş hayatındaki uygulamalar ile “Girişimcilik, yenilikçilik, Sürdürülebilir kalkınma” konularında verilen “seminer videosunu izledim”. Bu proje planını seminerde verilen bilgiler doğrultusunda hazırladığımı beyan ederim.

Proje ekibi iş paylaşımı
400633 Ekrem Ağca ONPP (Orthogonal Neighborhood Preserving Projections)
365295 Feyzanur Memiş Laplacian Eigenmaps
365299 Kader Nur Tekin LLE (Locally Linear Embedding),
376943 Merve Aydın LPP (Locality Preserving Projections),

2.2 Mimari tasarımı

Şekil 16: UML diyagramı içerisindeki adımlar sistemin nasıl modellenebileceğini belirleyen ve açıklayan yöntemlerdir. Tasarım mimarimiz nesne modelinde verilen adımlardan oluşmaktadır.

2.3.UML Nesne Modeli



Şekil 15: UML diyagramı

2.4 Yapılan Çalışmalar

Şimdi öğrendiklerimizi makine öğrenmesinin en temel veri setlerinden biri olan Yale üzerinde uygulayalım. İlk olarak çalışmamızda kullanacağımız modülleri import (yükleme-dahil etme) ediyoruz.

```
from PIL import Image
import numpy as np
import glob
```

İkinci aşama olan veri setimizin ölçeklendirme ve normalize işlemlerini gerçekleştirelim. Veri ölçeklendirmemizin amacı girdilerimizin birbirleriyle aynı aralıktaki sayılardan oluşturmaktır.

```
def build_dataset():
org_dataset = []
labels = []

      for i in range(1, 16):
filelist = glob.glob('./data/subject'+str(i).zfill(2)+"*")
for fname in filelist:
img = Image.open(fname)
                          img = np.array(img.resize((32, 32), Image.ANTIALIAS))
img = img.reshape(img.shape[0] * img.shape[1])
```

```
org_dataset.append(img)
labels.append(i)
return np.array(org_dataset), np.array(labels)

data, labels = build_dataset()

data = data/255
print(len(data))
```

Veri ölçeklendirme ve normalize işlemlerinden sonra verilerimizi eğitim ve test verisi olarak ayırdık. Verimizin %70 eğitim %30 test olarak ayrıldı.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data,labels, test_size =
0.33, shuffle=True, random_state=42, stratify=labels)
```

Sınıflandırma

```
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=1)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print('Accuracy = ' + str(accuracy_score(y_test, y_pred)))
```

Sınıflandırma kısmında hem kendi yazdığımız kod hem de hazır kod kullanarak sonuçlar elde etmeye çalıştık.

```
from sklearn.model_selection import train_test_split
import math

def dist(x1, x2):
    total = 0
    for i, j in zip(x1, x2):
        total += math.pow(i - j, 2)

    return math.sqrt(total)

def get_nearest_neighbors(row0, x_train, k):
    distances = []
    neighbors = []
    for i, row1 in enumerate(x_train):
        c = dist(row0, row1)
        distances.append([c, i])

    distances.sort(key = lambda x: x[0])
    for j in range(k):
          neighbors.append(distances[j])

    return neighbors



def KNN(K, X_test, X_train, y_train):

    Y_predict = []
```

```
    for x_test in X_test:
        neighbors = get_nearest_neighbors(x_test, X_train, K)
        targets = []
        for n in neighbors:
            index = n[1]
            targets.append(y_train[index])

        Y_predict.append(max(targets, key = targets.count))

    return Y_predict


y_pred = KNN(1, X_test, X_train, y_train)


from sklearn.metrics import r2_score
print(r2_score(y_pred, y_test))
```

Hazır kütüphane kullanarak elde ettiğimiz sonuç: Accuracy = 0.7454545454545455
Kendi yazdığımız Öklid uzaklık mesafesini kullanarak elde ettiğimiz sonuç= 0.6532924439901184

İlk olarak iki veri arasındaki benzerliği hesaplayan bir fonksiyon yazdık daha sonra bu uzaklıkları kullanarak en yakın komşuları bulduk. Bu verilerle sınıflandırmayı gerçekleştirdik.
Sınıflandırma işlemlerinden sonra gerçekleyebildiğimiz boyut indirgeme yöntemlerini kullanmaya çalıştık.

LLE

```
import numpy


USE_SVD = True


def LLE(M, k, m, quiet=False):
    M = numpy.matrix(M)
    # print(M)
    d,N = M.shape
    assert k<N


    #build the weight matrix
    W = numpy.zeros((N,N))
    m_estimate = []
    var_total = 0.0

    for row in range(N):
        M_Mi = numpy.array(M-M[:,row])

        vec = (M_Mi**2).sum(0)
        nbrs = numpy.argsort(vec)[1:k+1]
```

```
        M_Mi = numpy.matrix(M_Mi[:,nbrs])
        Q = M_Mi.T * M_Mi
        sig2 = (numpy.linalg.svd(M_Mi,compute_uv=0))**2
        v=0.9
        sig2 /= sig2.sum()
        S = sig2.cumsum()
        m_est = S.searchsorted(v)
        if m_est>0:
            m_est += ( (v-S[m_est-1])/sig2[m_est] )
        else:
            m_est = v/sig2[m_est]
        m_estimate.append(m_est)
        r = numpy.sum(sig2[m:])
        var_total += r
        Q.flat[::k+1] += r
        w = numpy.linalg.solve(Q,numpy.ones(Q.shape[0]))
        w /= numpy.sum(w)
        W[row,nbrs] = w


    I = numpy.identity(W.shape[0])
    U,sig,VT = numpy.linalg.svd(W-I,full_matrices=0)
    indices = numpy.argsort(sig)[1:m+1]

    return numpy.array(VT[indices,:])
```

LLE Uygulanışı

```
vt = LLE(X_train, 7, 80) ## donusum matrisini verir
X_trainn =  np.dot(X_train, np.transpose(vt))
X_testn =  np.dot(X_test, np.transpose(vt))
y_pred = KNN(1, X_testn, X_trainn, y_train)
from sklearn.metrics import r2_score
print(r2_score(y_test, y_pred))
print('Accuracy  With LE = '  + str(accuracy_score(y_test, y_pred)))
```

LLE Sonuç

0.7071731779264696
Accuracy = 0.7454545454545455
0.6399021512339018
Accuracy With LE = 0.7454545454545455

LE

```
from sklearn.metrics import pairwise_distances

import numpy as np
from scipy.linalg import eigh
import warnings
import networkx as nx


class LE:
```

```
    def __init__(self, X:np.ndarray, dim:int, k:int = 2, eps = None, graph:str =
'k-nearest', weights:str = 'heat kernel',
                 sigma:float = 0.1, laplacian:str = 'unnormalized',
opt_eps_jumps:float = 1.5):

        self.X = X
        self.dim = dim
        self.k = k
        self.eps = eps

        if graph not in ['k-nearest', 'eps']:
            raise ValueError("graph is expected to be a graph name; 'eps' or 'k-
nearest', got {} instead".format(graph))

        self.graph = graph

        if weights not in ['simple', 'heat kernel', 'rbf']:
            raise ValueError("weights is expected to be a weight name; 'simple'
or 'heat kernel', got {} instead".format(weights))

        self.weights = weights
        self.sigma = sigma
        self.n = self.X.shape[0]

        if laplacian not in ['unnormalized', 'random', 'symmetrized']:
            raise ValueError("laplacian is expected to be a laplacian name;
'unnormalized', 'random' or 'symmetrized', got {} instead".format(laplacian))

        self.laplacian = laplacian
        self.opt_eps_jumps = opt_eps_jumps

        if self.eps is None and self.graph == 'eps':
            self.__optimum_epsilon()

    def __optimum_epsilon(self):

        dist_matrix = pairwise_distances(self.X)


        self.eps = min(dist_matrix[0,1:])
        con = False
        while not con:
            self.eps = self.opt_eps_jumps * self.eps
            self.__construct_nearest_graph()
            con = self.cc == 1
            print('[INFO] Epsilon: {}'.format(self.eps))
        self.eps = np.round(self.eps, 3)

    def __heat_kernel(self, dist):
            """
            k(x, y) = exp(- ||x-y|| / sigma )
            """
            return np.exp(- (dist*dist)/self.sigma)

    def __rbf(self, dist):
        """
        k(x, y) = exp(- (1/2*sigma^2) * ||x-y||^2)
```

```
        """
        return np.exp(- dist**2/ (2* (self.sigma**2) ) )

    def __simple(self, *args):
        return 1

    """
     Step 1: kNN graph and adjacency matrix
    """

    def __construct_nearest_graph(self):
        """
        Compute weighted graph G
        """
        similarities_dic = {'heat kernel': self.__heat_kernel,
                            'simple':self.__simple,
                            'rbf':self.__rbf}

        dist_matrix = pairwise_distances(self.X)
        if self.graph == 'k-nearest':
            nn_matrix = np.argsort(dist_matrix, axis = 1)[:, 1 : self.k + 1]
        elif self.graph == 'eps':
            nn_matrix = np.array([ [index for index, d in
enumerate(dist_matrix[i,:]) if d < self.eps and index != i] for i in
range(self.n) ])
        # Weight matrix
        self._W = []
        for i in range(self.n):
            w_aux = np.zeros((1, self.n))
            similarities = np.array([
similarities_dic[self.weights](dist_matrix[i,v]) for v in nn_matrix[i]] )
            np.put(w_aux, nn_matrix[i], similarities)
            self._W.append(w_aux[0])
        self._W = np.array(self._W)
        # D matrix
        self._D = np.diag(self._W.sum(axis=1))
        # Check for connectivity
        self._G = self._W.copy() # Adjacency matrix
        self._G[self._G > 0] = 1
        G = nx.from_numpy_matrix(self._G)
        self.cc = nx.number_connected_components(G) # Multiplicity of lambda = 0
        if self.cc != 1:
            warnings.warn("Graph is not fully connected, Laplacian Eigenmaps may
not work as expected")
    """
    Step 2: The Laplacian
    """
    def __compute_unnormalized_laplacian(self):
            self.__construct_nearest_graph()
            #L=D-W
            self._L = self._D - self._W
            return self._L

    def __compute_normalized_random_laplacian(self):
            self.__construct_nearest_graph()
            self._Lr = np.eye(*self._W.shape) -
(np.diag(1/self._D.diagonal())@self._W)
            return self._Lr
```

```
    def __compute_normalized_symmetrized_laplacian(self):
            self.__construct_nearest_graph()
            self.__compute_unnormalized_laplacian()
            d_tilde = np.diag(1/np.sqrt(self._D.diagonal()))
            self._Ls = d_tilde @ ( self._L @ d_tilde )
            return self._Ls




    """
    Step 3: Eigenvector Decomposition and Embedding
    """

    def transform(self):
            """
            Compute embedding
            """

            m_options = {
                'unnormalized':self.__compute_unnormalized_laplacian,
                'random':self.__compute_normalized_random_laplacian,
                'symmetrized':self.__compute_normalized_symmetrized_laplacian
            }

            L = m_options[self.laplacian]()

            if self.laplacian == 'unnormalized':
                eigval, eigvec = eigh(L, self._D) # Generalized eigenvalue
problem Ly = λDy
            else:
                eigval, eigvec = np.linalg.eig(L)

            order = np.argsort(eigval)
            self.Y = eigvec[:, order[self.cc:self.cc+self.dim + 1]]

            return self.Y
```

LPP

```
import numpy as np
from scipy import linalg

from sklearn.neighbors import kneighbors_graph, NearestNeighbors
from sklearn.utils import check_array
from sklearn.base import BaseEstimator, TransformerMixin


class LocalityPreservingProjection(BaseEstimator, TransformerMixin):

    def __init__(self, n_components=2, n_neighbors=5,
                 weight='adjacency', weight_width=1.0,
                 neighbors_algorithm='auto'):
        self.n_components = n_components
        self.n_neighbors = n_neighbors
        self.weight = weight
```

```
        self.weight_width = weight_width
        self.neighbors_algorithm = neighbors_algorithm

    def fit(self, X, y=None):
        X = check_array(X)
        W = self._compute_weights(X)
        self.projection_ = self._compute_projection(X, W)
        return self

    def transform(self, X):
        X = check_array(X)
        return np.dot(X, self.projection_)

    def _compute_projection(self, X, W):
        X = check_array(X)

        D = np.diag(W.sum(1))
        L = D - W
        evals, evecs = eigh_robust(np.dot(X.T, np.dot(L, X)),
                                   np.dot(X.T, np.dot(D, X)),
                                   eigvals=(0, self.n_components - 1))
        return evecs

    def _compute_weights(self, X):
        X = check_array(X)
        self.nbrs_ = NearestNeighbors(n_neighbors=self.n_neighbors,
                                      algorithm=self.neighbors_algorithm)
        self.nbrs_.fit(X)

        if self.weight == 'adjacency':
            W = kneighbors_graph(self.nbrs_, self.n_neighbors,
                                 mode='connectivity', include_self=True)
        elif self.weight == 'heat':
            W = kneighbors_graph(self.nbrs_, self.n_neighbors,
                                 mode='distance', include_self=True)
            W.data = np.exp(-W.data ** 2 / self.weight_width ** 2)
        else:
            raise ValueError("Unrecognized Weight")
        W = W.toarray()
        W = np.maximum(W, W.T)
        return W


def eigh_robust(a, b=None, eigvals=None, eigvals_only=False,
                overwrite_a=False, overwrite_b=False,
                turbo=True, check_finite=True):
    kwargs = dict(eigvals=eigvals, eigvals_only=eigvals_only,
                  turbo=turbo, check_finite=check_finite,
                  overwrite_a=overwrite_a, overwrite_b=overwrite_b)

    if b is None:
        return linalg.eigh(a, **kwargs)

    kwargs_b = dict(turbo=turbo, check_finite=check_finite,
                    overwrite_a=overwrite_b)  # b is a for this operation
    S, U = linalg.eigh(b, **kwargs_b)

    S[S <= 0] = np.inf
    Sinv = 1. / np.sqrt(S)
```

```
    W = Sinv[:, None] * np.dot(U.T, np.dot(a, U)) * Sinv
    output = linalg.eigh(W, **kwargs)

    if eigvals_only:
        return output
    else:
        evals, evecs = output
        return evals, np.dot(U, Sinv[:, None] * evecs)
```

LPP Uygulanışı

```
import lpproj
lppModel = lpproj.LocalityPreservingProjection(n_components = 80, n_neighbors =
1)
selfObject = lppModel.fit(X_train)
trainKlpp = np.dot(X_train, selfObject.projection_)
testKlpp = np.dot(X_test, selfObject.projection_)

## classification
# default=2 Minkowski metric
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=1)
model.fit(trainKlpp, y_train)
y_pred = model.predict(testKlpp)
print('Accuracy with LPP = '  + str(accuracy_score(y_test, y_pred)))
print('Accuracy with Knn LPP = '  + str(r2_score(y_pred, y_test)))
```

LPP Sonuç

Accuracy with LPP = 0.8545454545454545

ONPP

```
clear;
close all;
addpath('../utils');


jpegFiles = dir('*.jpg');
numfiles = length(jpegFiles);
mydata = cell(1, numfiles);

for k = 1:numfiles
  mydata{k} = imread(jpegFiles(k).name);
  B{k} = rgb2gray(mydata{k});
  F{k} = imresize(B{k},[32 32],'bilinear');
end
cd('../ELGE');


% r =20 is the number of iterations and q=50 is the number of individuals, while
12 is the number of training samples for each person
for r=1:20
```

```
for q=1:50
random_selection_numbers(q,:,r) = randsample(15,12);
end
end

for r=1:20
for q=1:50
for j=1:12
zzz(q,j,r) = 15*(q-1)+ random_selection_numbers(q,j,r);
end
end
end

X_index=reshape(zzz,[600 20]);
X_index_sorted = sort(X_index, 1);

all_random_numbers = randsample(750,750);

for i=1:20
test_numbers(:,i) = setdiff(all_random_numbers,X_index_sorted(:,i));
end

for vv = 1:20
X_faces_matrices(vv,:) = F(X_index_sorted(:,vv));
test_faces_matrices(vv,:) = F(test_numbers(:,vv));
end

for i=1:20
for j=1:600
X_faces_vectors{i,j}= X_faces_matrices{i,j}(:);
end
end

for i=1:20
for j=1:150
test_faces_vectors{i,j}= test_faces_matrices{i,j}(:);
end
end

for fff = 1:20
Q = X_faces_vectors(fff,:);
X_cell{fff}= {cat(2, Q{:})};
R = test_faces_vectors(fff,:);
test_cell{fff}= {cat(2, R{:})};
clear Q
clear R
end

for fff = 1:20
X_matrix(:,:,fff) = cell2mat(X_cell{fff});
test_matrix(:,:,fff) = cell2mat(test_cell{fff});
end

n=12;
p=3;
x=(1:50)';
X_labels=repmat(x,1,n)';
test_labels = repmat(x,1,p)';
X_labels=X_labels(:);
```

```
test_labels = test_labels(:);

clearvars -except X_matrix test_matrix X_labels test_labels
for iteration = 1:20

X = X_matrix(:,:,iteration);
X = X';
X = im2double(X);

test_iter = im2double(test_matrix(:,:,iteration));

k = 5;

% Get dimensionality and number of dimensions
    [n, d] = size(X);
    mapping.mean = mean(X, 1);

% Compute pairwise distances and find nearest neighbours
    disp('Finding nearest neighbors...');
    [distance, neighborhood] = find_nn(X, k);
    max_k = size(neighborhood, 2);
    mapping.nbhd = distance;
    X = X';
    neighborhood = neighborhood';

% Find reconstruction weights for all points by solving the MSE problem of
reconstructing a point from each neighbours. A used constraint is that the sum
of the reconstruction weights for a point should be 1.
    disp('Compute reconstruction weights...');
    if k > d
        tol = 1e-5;
    else
        tol = 0;
    end

    % Construct reconstruction weight matrix
    W = zeros(max_k, n);
    for i=1:n
        nbhd = neighborhood(:,i);
        if ischar(k)
           nbhd = nbhd(nbhd ~= 0);
        end
        kt = numel(nbhd);
        z = X(:,nbhd) - repmat(X(:,i), 1, kt);                  % Shift point to
origin
        C = z' * z;
             % Compute local covariance
        C = C + eye(kt, kt) * tol * trace(C);                          %
Regularization of covariance (if K > D)
        wi = C \ ones(kt, 1);                                   % Solve linear
system
        wi = wi / sum(wi);                                      % Make sure that
sum is 1
        W(:,i) = [wi; nan(max_k - kt, 1)];
    end

    % Now that we have the reconstruction weights matrix, we define the sparse
cost matrix M = (I-W)'*(I-W).
    M = sparse(1:n, 1:n, ones(1, n), n, n, 4 * max_k * n);
```

```
    for i=1:n
       w = W(:,i);
       ww(~isnan(w)) = 0;
       j = neighborhood(:,i);
       w = w(j ~= 0);
       j = j(j ~= 0);
       M(i, j) = M(i, j) - w';
       M(j, i) = M(j, i) - w;
       M(j, j) = M(j, j) + w * w';
    end

% For sparse datasets, we might end up with NaNs or Infs in M.
M(isnan(M)) = 0;
M(isinf(M)) = 0;

W_LGE = eye(n)-full(M);
D_LGE = eye(n)/(X'*X);

options.ReducedDim = 80;

X = X';

[eigvector, eigvalue] = LGE(W_LGE, D_LGE, options, X);
[eigvalue, ind] = sort(diag(eigvalue), 'descend');
eigvector_final = eigvector(:,ind(1:options.ReducedDim));

A_final= eigvector_final;
Y_train= A_final'*im2double(X');
Y_test = A_final'*im2double(test_iter);


% If you have newer versions of Matlab, comment the following line and
c1=knnclassify (Y_test', Y_train', X_labels, 1);
% uncomment the following lines
% mdl = fitcknn(Y_train', X_labels, 'Distance', 'euclidean', 'NumNeighbors', 1);
% c1 = predict(mdl, Y_test');


diff1 = ne(c1, test_labels);
sumdiff1=sum (diff1);
accuracy_k1(iteration) = (1- (sumdiff1/length(test_labels)))*100;

clearvars -except X_matrix test_matrix X_labels test_labels eigvalue accuracy_k1

end

average_accuracy = mean(accuracy_k1)
standard_deviation_accuracy = std(accuracy_k1)
```

ONPP SONUÇ

Average accuracy =81.5333
Accuracy With Onpp=0.79348

3. KAYNAKLAR

Makaleler

1. By christoper J.C Burges, Dimension Reduction: A Guided Tour, 2010, 276-360.
2. Dick de Ridder ve Robert P.W Duin, Locally linear embedding for classification, 2002, 1-15.
3. E. Kokiopoulou ve Y. Saad,Orthogonal Neighborhood Preserving Projections: A projection-based dimensionality reduction technique, March 21, 2006,1-20
4. Jing Chen ve Yang Liu, Locally linear embedding: a survey, 2011, 1-20.
5. Lawrance Cayton, Algorithms for manifold learning ,2005, 1-17.
6. Lawrance K. Saul ve Sam T. Rowesis, An Introduction to Locally Linear Embedding 1-13.
7. Orhan SİVAZ, Dokunmatik ekranlarda kaydırma biyometrisine dayalı kimlik doğrulama çalışmaları Yüksek Lisans Tezi 2021.
8. Xiaofei He Locality Preserving Projections a dıssetation submitted to the faculty of the divisiın of the physical sciences in canadidadcy for the dedree of doctor of philosophy Aralık 2005
9. Alexandre L.M. Leyada LeLecture notes in dimensionality reduction for unsupervised metric learning Aralık 17,2020
10. S. Roweis and L. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science, 290:2323–2326, 2000.
11. Xiaofei He ve Partha Niyogi, Locality Preserving Projections, 1-8.
12. https://bit.ly/3t6gLPm
13. https://bit.ly/3q5fkih
14. https://bit.ly/3q2shJt
15. https://bit.ly/3sZUin0
16. https://bit.ly/3f0j1zk

STANDARTLAR ve KISITLAR FORMU

Projenin hazırlanmasında uyulan standart ve kısıtlarla ilgili olarak, aşağıdaki soruları cevaplayınız.

1. Projenizin tasarım boyutu nedir? (Yeni bir proje midir? Var olan bir projenin tekrarı mıdır? Bir projenin parçası mıdır? Sizin tasarımınız proje toplamının yüzde olarak ne kadarını oluşturmaktadır?)

Yüz tanıma sistemlerinde kullanılan ve PCA dan sonra kullanılan en bilinen boyut indirgeme yöntemleridir yeni bir proje değildir ve bir projenin devamı değildir öncesinde kodlanmış yöntemleri kendimiz kodlamaya ve boyut indirgeme sonuçlarını incelenmesi amaçlanmıştır.

2. Projenizde bir mühendislik problemini kendiniz formüle edip, çözdünüz mü? Açıklayınız.

Var olan matematiksel yöntemleri anlayıp kodlamasını gerçekleştirdik.

3. Önceki derslerde edindiğiniz hangi bilgi ve becerileri kullandınız?

Bilgisayar Mühendisliğine giriş dersi, Algoritmalar, Yapay ProgeZeka ve Yazılım Mühendisliğine giriş derslerinde edindiğimiz programlama bilgilerini, mantığını, algoritma geliştirme becerilerini, veri ön işleme, sınıflandırma ve dökümantasyon oluşturma becerilerimizi kullandık.

4. Kullandığınız veya dikkate aldığınız mühendislik standartları nelerdir? (Proje konunuzla ilgili olarak kullandığınız ve kullanılması gereken standartları burada kod ve isimleri ile sıralayınız).

IEEE/IE 12207 Standardı ile belirtilen yazılım yaşam döngüsü süreçleri dikkate alınmıştır.

5. Kullandığınız veya dikkate aldığınız gerçekçi kısıtlar nelerdir? Lütfen boşlukları uygun yanıtlarla doldurunuz.

a) Ekonomi

Proje geliştirme süreçlerinde kullanılan Anaconda Spyder gibi geliştirme ortamının community versiyonu kullanıldığı için bir ücretlendirme yapılmamıştır. Kaggle üzerinden ücretsiz olan paylaşılan Starter: Yale Face Database c5f3978b-5 kullanılmıştır ve herhangi bir ücret ödenmemiştir.

b) Çevre sorunları:

Projemiz çevre için herhangi bir sorun teşkil etmemektedir.

c) Sürdürülebilirlik:

Projeyi başarılı bir şekilde tamamlayabilirsek kullanıcı feedbackleri ile sürdürülebililirliği sağlamayı amaçlıyoruz.

d) Üretilebilirlik:

Projenin sonunda elde ettiğimiz başarı oranına göre üretkenliği amaçlamayı hedeflemekteyiz.

e) Etik:

İnsani değere zarar verecek etik açıdan herhangi bir içeriğe sahip değildir.

f) Sağlık:

Projemiz sağlık açısından herhangi bir sorun teşkil etmemektedir.

g) Güvenlik:

Projede kullanıcının herhangi bir verisine izinsiz erişim olmadığından dolayı güvenlik için bir sorun teşkil etmeyecektir.

h) Sosyal ve politik sorunlar:

Projemizin sosyal açıdan sorununu henüz tespit edememekle beraber politik açıdan bir sorun teşkil etmemektedir.