T.C.
HARRAN ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

ELEKTRİK – ELEKTRONİK MÜHENDİSLİĞİ PROJESİ

RADYAN TABANLI FONKSİYONLAR

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

HAZIRLAYAN ÖĞRENCİNİN ADI-SOYADI VE NUMARASI
SABRİ BEREKET - 190507003

DANIŞMAN
Araş. Gör. Dr. BUKET SONBAŞ

2022

ŞANLIURFA

İçindekiler

1.GİRİŞ

Radyal Tabanlı Fonksiyonlar, 1980'lerin sonlarında yapay sinir ağının bir çeşidi olarak ortaya çıkmıştır. Ortaya çıkışından sonrasında temelleri, potansiyel fonksiyonlar, kümeleme, fonksiyonel yaklaşım ve karışım modelleri gibi çok daha eski örüntü tanıma tekniklerine yerleşmiştir. Radyal tabanlı fonksiyonlar (RBF), her bir gizli birimin radyal etkinleştirilmiş bir işlevi uyguladığı iki katmanlı bir sinir ağına gömülüdür. Bir RBF ağında çıktı doğrusalken girdi doğrusal değildir. Doğrusal olmayan yaklaşım özellikleri nedeniyle, RBF ağları, algılayıcı sinir ağlarının yalnızca çoklu ara katmanlar aracılığıyla modelleyebildiği karmaşık eşleşmeleri modelleyebilir. RBF ağı kullanmak için, gizli birim aktivasyon fonksiyonunu, işlem birimlerinin sayısını, belirli bir görevi modellemek için bir kriteri ve ağın parametrelerini bulmak için bir eğitim algoritmasını belirtmemiz gerekir. RBF ağırlıklarını bulmaya ağ eğitimi denir. Elimizde eğitim seti adı verilen bir dizi girdi çıktı çifti varsa, ağ çıktılarını verilen girdilere uydurmak için ağ parametrelerini optimize ederiz. Uyum, genellikle ortalama kare hatası olduğu varsayılan bir maliyet fonksiyonu aracılığıyla değerlendirilir. Burada en üst başlıktan başlayıp kademe kademe parçalar halinde en derine inerek radyal tabanlı fonksiyonları inceleyeceğiz kodlar üzerinde denemeler yapacağız ve çıkan sonuçları tartışacağız.

2.MAKİNE ÖĞRENİMİ

2.1 Makine Öğrenmesi Tarihçesi

Makine öğrenimi başlangıç olarak 1959 yılında bilgisayar biliminin yapay zekada sayısal öğrenme ve model tanıma çalışmalarından geliştirilen bir alt dalıdır. Makine öğrenimi ilk olarak sinir ağlarının matematiksel modellenmesiyle ortaya çıktı.

2.2 Makine Öğrenimi Nedir?

Makine öğrenimi, veri bilimi alanının önemli parçalarından birisidir. Makine öğrenimi, sistemlerin açıkça programlanmadan deneyimlerden öğrenmesini ve geliştirmesini sağlayan bir yapay zekâ uygulamasıdır. Makine öğrenimi, verilere erişebilen ve eriştikleri verileri kendileri için öğrenmek amacıyla kullanabilen bilgisayar programları geliştirmeye odaklanır. Veri madenciliği projelerindeki temel bilgileri ortaya çıkarır. Bir disiplin olarak makine öğrenimi, verilerden öğrenebilen ve veriler üzerinde tahminler yapabilen algoritmaların analizini ve oluşturulmasını araştırır.

2.3 Makine Öğreniminin Kullanıldığı Bazı Etkileyici Noktalar

Veri: Veri güvenliği açıklarını önceden belirleyebilir. Geçmiş deneylere bakarak gelecekteki ihlal edilebilecek noktaları tahmin edebilir.

Finans: Bankalar ticareti hızlandırmak ve otomatikleştirmek amacı ile makine öğrenimi algoritmalarını kullanır. Canlı destek üzerindeki sohbet robotlarını örnek alabiliriz.

Sağlık Hizmetleri: Hastalıkların sebeplerini ve sorunların çözümlerin bulunmasını hızlandırmak için sağlık verilerini analiz etmek için kullanılır.

2.4 Makine Öğrenmesi Nasıl Çalışır?

UC Berkeley okulu bu alanda yapmış olduğu araştırmada [1] öğrenme sistemini üç ana bölüme ayırıyor.

- Karar Süreci: ML (Machine Learning = Makine Öğrenimi) algoritmaları genel olarak tahmin ya da sınıflandırma yapmak için kullanılır.

- Hata İşlevi: Modelin tahminini değerlendirmek için kullanılır. Kıyaslama yaparak model doğruluğunu inceler
- Model Optimizasyonu Süreci: Modelimiz, eğitim kümesindeki veri ile daha iyi uyum sağlıyorsa, bu durumda gerçek ile model tahmini arasındaki çelişkiyi azaltmak için düzenlenir. Algoritma tekrar kontrol edip kendini geliştirerek günceller.

3.YAPAY SİNİR AĞLARI

3.1 Yapay Sinir Ağları Tarihçesi

Yapay sinir ağları adı altında yapılan ilk çalışma 1943 yılında öne sürüldüğü kabul edilir. McCullogh ve Pitts ilk olarak yapay sinir tanımını yaparlar. Donald Hebb, The Organisation of Behavior adlı kitabında bu fikri daha da ileri götürerek, sinirsel yolların, özellikle de nöronlar arasında, birbirini izleyen her kullanımda güçlendiğini ve böylece karmaşık süreçleri özelleştirmeye yönelik uzun bir yolculuğa başladığını öne sürdü.

3.2 Yapay Sinir Ağları Nedir?

Yapay sinir ağları insan beyninin öğrenme becerisini bilgisayarla birlikte gerçekleşmesini sağlayan sistemlerdir. Öğrenme işlemi örnekler ve edinilen tecrübeler ile oluşur. Yapay sinir ağları verilen girdilere karşı çıktı üretir ve bunu kaydeder. Yapay sinir ağları uyumlu, eksik verilerle çalışabilen, belirsizlikler üzerinde tahmin ile karar verebilen, yaptığı her işlemi tecrübe olarak kendine ekleyen belirsiz bir türdür.

3.3 Yapay Sinir Ağları Çalışma Şekli

Genel olarak yapay sinir ağları, sınıflandırma, sinyal şeffaflığı, veri işlemleri ve optimizasyon çalışmalarında en güçlü tekniklerden biri olarak kabul edilebilir. Yapay sinir ağları kullanımı esnasında beklenen değerler ile yapay sinir ağlarının sonuçları arasında karşılaştırma yapılır. Bu yapılan karşılaştırma sistem zekasında sonuçları hata miktarlarına göre eğitilerek tecrübe edinilir. Burada eklenilen tecrübe ile kendisini yeniler ve yineler.

3.4 Yapay Sinir Ağları Şebekeleri

Yapay sinir ağları şebekesi, yapısal ve bu yapının işlemesi olarak iki parça şeklinde incelenebilir. Mimari olarak adlandırdığımız kısmı giriş, ara, çıkış kısımlarındaki sinir hücreleri ile bağlantılarından meydana gelir. İşleyiş kısmına geldiğimizde hücreler içinde tetikleme

işlemcisi bulunur. Yapının işlenmesini iç ve dış işleyiş olarak iki matematiksel işleyiş olarak düşünebiliriz. İç işleyiş gizli kısımdaki işlemciler yardımıyla tamamlanır. Dış işleyiş sırayla gelen kısımların hücreleri arasındaki bağlantı değerlerini atamak, sonrasında çıktı sonuçlarının geri beslenme işlemleriyle yenilenmesi sonucunda oluşur.

4.VERİ BİLİMİ

4.1 Geçmişten Günümüze Veri Bilimi

Geçmiş zamanlardan günümüze kadar gelen büyük miktarda veriyi elektronik olarak tanıma, toplama ve saklama becerilerimizi geliştirdik. Günümüzde elimizde oldukça fazla veri ve bilgi sürümü var. Bunun da ekstrası olarak bilmeye aç olduğumuzu belirtmeliyiz. Baktığımız her yerde veriler bizi çevreliyor. Veri setleriyle her modellemeyi deneyebilir, her şeyi öğrenebilir ve yapabiliriz. Verilerin alt kümelerini elektronik olarak yakalıyor ve saklıyoruz. Çevremizdeki verileri hesaplamak ve analiz etmek için numaralandırarak kullanıyoruz. Hesaplama ve analiz işlemlerini kolaylaştırmak ve erişilebilir yapmak için elektronik cihazlarımıza geçiş yapıyoruz. Veriler günümüzde çok hızlı bir şekilde elektronik ortama geçiş yapıyor. Şu anda yakalanan çok fazla veri var ve her zaman yeni veriler geliyor. Bunların çoğu analiz edilmeyi ve tam olarak kullanılmayı bekliyor.

4.2 Veri Bilimi Nedir?

Veri bilimi, veriyi bilgiye dönüştürme çabasını yakalayan geniş bir etikettir. Veri bilimcileri, verinin işleyişini ve temiz bilgi edinmek için karmaşık sorunları daha küçük görevlere ayrıştırmak için matematik, statik, makine öğrenimi ve yapay zekadan sürekli değişen ve geniş bir teknik ve teknoloji birikimini uygular. Daha küçük görevlerden elde edilen bilgi birikimi, gelecek zamanda bir bütünün anlayışını oluşturmak ve bilgisayar tabanlı uygulamaların gelişimini yönlendirmek için sentezlenebilir. Veri bilimi, bilim-iş faydasından, endüstri, hükümet, çevre ve genel olarak gerçek fayda sağlar. Bugün etrafımıza baktığımızda her kuruluşun veri odaklı olduğunu görebiliriz.

4.3 Veri Bilimcisi Görevleri

Veri bilimcileri, belirsiz olanı bilinene dönüştürür. Bir veri bilimcisi, çeşitli araçlar kullanarak bir göreve derin bir hesaplama becerileri koleksiyonu getirir. Verileri keşfeder, görselleştirir, analiz eder ve modellerler, ardından bütüne ilişkin bilgimizi oluşturmak için bir araya gelen yeni iç görüleri sentezlerler.

5.PYTHON PROGRAMLAMA DİLİ İLE VERİ BİLİMİ

5.1 Python Tarihçesi

Yapılmaya 1990 yılında Guido van Rossum tarafından Amsterdam'da başlanmıştır. Günümüzde Python Yazılım Vakfı çevresinde toplanan gönüllülerin çabalarıyla geliştirilmeye devam etmektedir. Python 1 sürümüne 1994 Ocak ayında ulaşmıştır. Python 2 16 Ekim 2000 tarihinde yayınlanmıştır. 3 Aralık 2008'de Python 3 sürümleri yayınlanmaya başlamıştır.

5.2 Python Nedir?

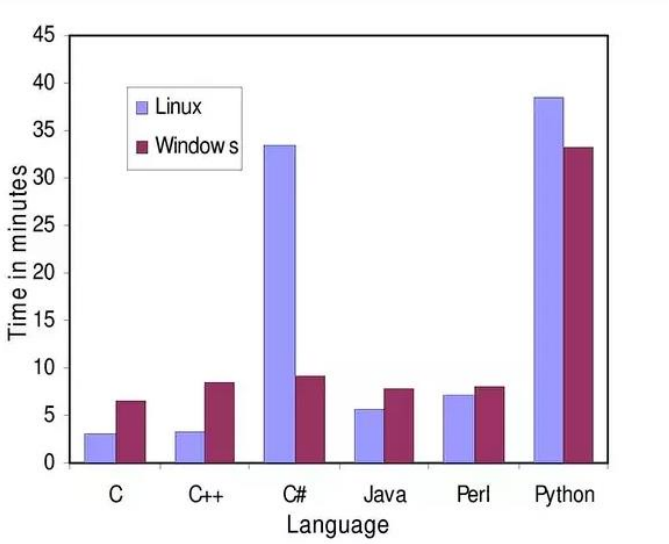
Python bir yazılım dilidir. Python programlama dili veri bilimi, makine öğrenimi, otomasyonlar, web ve API geliştirmek için temel yapıdır. Anlaşılması kolay ve okuması eğlencelidir. Python yeni başlayan programcılar için hem de İngilizceye yeni başlayan kullanıcılar için oldukça uygundur. Sizi temelden kapsamlı ve çeşitli konulara götürür. Son yıllarda Python yeni nesil yazılım geliştirme, veri analizinde birinci sınıf bir programlama dili olarak parlamıştır.

5.3 Python Avantajları

Python'un diğer yazılım dillerinden ayıran en önemli avantajı bu programla dilini kullanmak oldukça kolaydır. Diğer diller karmaşık yapısından dolayı öğrenmek zaman alır. Kullanım alanlarının fazla olması dili daha da karmaşık duruma getirdiğinden dolayı öğrenmesi zaman alıcı ve yorucudur. Python sözdizimi okunması oldukça rahat ve geliştirilebilirdir. Github projelerinin birçoğu python diliyle yürütülmektedir. Kullanım alanı en küçük işletim sisteminden en büyük işletim sistemine kadar destek görmektedir. İnternetteki büyük kütüphanelerin ve API'lerin çoğunlukla Python için uyumlu versiyonları vardır. Veri analizleri günümüzde bilişim teknolojileri için en önemli konular haline gelmiştir. Python bu durumlar için en uygun programlama dilidir. Kütüphanelerin kaliteli komutları ve yapıları makine öğreniminin ve diğer algoritma kütüphanelerinin sürekli gelişmesine yardımcı olmuştur.

5.4 Python Dezavantajları

Python her ne kadar kaliteli uygulamalar yazmak için uygun olsa da bazı eksiklikleri vardır. Sistem düzeyinde programlama için uygun değildir. Platformlar arası bağımsız binary dosyalar için çağrı yapmak gerektiğinde Python yetersiz kalacaktır. Masaüstü için oluşturulan uygulamaların görkemli olunmadığı gözle görünür.



(Şekil 5.1: Yapılan bir araştırmada programlama dillerinin derleme sürelerinin grafiği verilmiştir.)

6.RADYAL TABANLI FONKSİYON AĞI

6.1 Radyal Tabanlı Fonksiyonlar

Radyal tabanlı fonksiyonlar, tek değişkenli fonksiyona dayalı terimlerin lineer kombinasyonları ile çok değişkenli fonksiyonlara yaklaşmanın araçlarıdır. Bu, birden fazla boyutta kullanılabilmesi için radikal edilmiştir. Bunlar genellikle sadece sınırlı sayıda noktası bilinen yaklaşık fonksiyonlara veya verilere uygulanır böylelikle yaklaşma fonksiyonunun değerlendirmeleri sık ve verimli bir şekilde gerçekleşebilir.

6.2 Radyal Tabanlı Fonksiyon Ağları Nedir?

Radyal tabanlı fonksiyon ağı (RBFN), yapay sinir ağlarında girdilerin toplamından oluşan skaler değerleri olarak radyal tabanlı fonksiyonları kullanan bir yapay sinir ağıdır. Ağın çıkışı, girişlerin ve nöron değerlerinin radyal tabanlı fonksiyonlarının doğrusal birleşimidir. Radyal tabanlı fonksiyon ağlarının, denklem yaklaşımı, zaman serisi tahmini, kümeleştirme ve sistem kontrolü olmak üzere çok fazla kullanım alanı vardır.

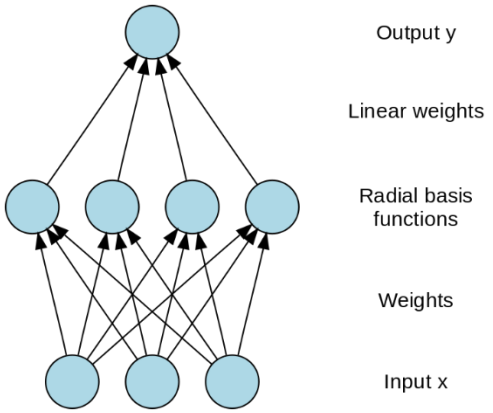
6.3 Ağ Mimarisi

Ağ mimarisini inceleyecek olursak RBF ağları üç katmana sahiptir.

- Giriş katmanı
- Doğrusal olmayan gizli katmanı

- Doğrusal çıkış katmanı

Giriş, gerçek sayıların bir vektörü olarak modellenebilir. Ağın çıkışı daha sonra giriş vektörünün skaler fonksiyonudur.



(Şekil 6.1 Radyal tabanlı bir fonksiyon ağının mimarisi.)

$X \in R^n \qquad \varphi: R^n \to R$

$$\varphi(x) = \sum_{i=1}^{N} a_i p(||x - c_i)$$

N gizli katmandaki nöron sayısıdır. c_i nöronun merkez vektörüdür.

i ve a_i nöronun ağırlığıdır.

Sadece bir merkez vektörden uzaklığa bağlı olan fonksiyonlar, bu vektöre radyal olarak simetriktir bundan dolayı radyal temel fonksiyon denir. Temel düzlemde, tüm girişlerin her biri farklı gizli nörona bağlıdır.

$$\rho(|x - c_i|) = \exp\left[-\beta_i ||x - c_i||^2\right]$$

Gauss temel fonksiyonları, merkez vektöre yereldir.

$$\lim_{||x|| \to \infty} \rho\left(||x - c_i||\right) = 0$$

Nöronun değişkenlerini değiştirmek, o nöronun merkezinden çok uzakta olan giriş değerleri için çok küçük bir etkiye sahiptir.

Aktivasyon fonksiyonun şekli üzerinde belirli hafif koşullar verildiğinde, radyal tabanlı fonksiyon ağı, alt kümede evrensel tahmincilerdir. Bunun anlamı yeterliliğini tamamlamış radyal tabanlı fonksiyon ağı kapalı, sınırlandırılmış bir küme üzerindeki herhangi bir sürekli fonksiyona isteğine bağlı olarak bir hassasiyet ile yaklaşabileceği anlamına gelir.

6.3.1 Normalleştirilmiş Mimari

Normalleştirilmiş mimariye bakacak olursak normalleştirilmemiş mimariye ek olarak, RBF ağları normalleştirilebilir. Haritalandıracak olursak,

$$\varphi(x) =^{def} \frac{\sum_{i=1}^{N} a_i \rho(||x - c_i||)}{\sum_{i=1}^{N} \rho(||x - c_i||)} = \sum_{i=1}^{N} a_i u(||x - c_i||)$$

$$u(||x - c_i||) =^{def} \frac{\rho(||x - c_i||)}{\sum_{j=1}^{N} \rho(||x - c_i||)}$$

Normalleştirilmiş RBF olarak bilinir.

6.3.2 Yerel Doğrusal Modeller

Bazen mimariyi yerel doğrusal modelleri içererek genişleterek uygundur.

$$\varphi(x) = \sum_{i=1}^{N} (a_i + b_i \,.\, (x - c_i)) \rho(||x - c_i||)$$

ve

$$\varphi(x) = \sum_{i=1}^{N} (a_i + b_i \,.\, (x - c_i)) u(||x - c_i||)$$

Sırasıyla normalleştirilmemiş ve normalleştirilmiş durumlar verilmiştir. Burada b_i belirlenecek ağırlıklardır. Daha yüksek mertebeden lineer terimler de mümkündür.

Bu şekilde:

$\varphi(x) = \sum_{i=1}^{2N} \sum_{j=1}^{N} e_{ij} v_{ij} (x - c_i)$

Üzerine tablodan e_{ij} 'yi alabiliriz. $e_{ij} \begin{cases} a_i, & eğer\ i \in [1, N] \\ b_{ij}, & eğer\ i \in [N+1, 2N] \end{cases}$ bu durumda normalleştirilmemiş durumda,

$$v_{ij}(x - c_i) =^{def} \begin{cases} \delta_{ij} u(||x - c_i||), & eğer\ i\ \epsilon[1, N] \\ (x_{ij} - c_{ij})\rho(||x - c_i||), & eğer\ i\ \epsilon[N + 1, 2N] \end{cases}$$

Ve normalleştirilmiş durumda ise

$$v_{ij}(x - c_i) =^{def} \begin{cases} \delta_{ij} u(||x - c_i||), & eğer\ i\ \epsilon[1, N] \\ (x_{ij} - c_{ij})u(||x - c_i||), & eğer\ i\ \epsilon[N + 1, 2N] \end{cases}$$

Bu şekildedir.

Burada $\delta_{ij} \begin{cases} 1, eğer\ i = j \\ 0, eğer\ i \neq j \end{cases}$ olarak tanımlanan bir Kroncker delta işlevidir.

6.4 Radyal Tabanlı Fonksiyonlarda Eğitim

RBF ağları genel olarak iki aşamalı algoritma ile $x(t), y(t), t = 1, \ldots, T$ gibi girdi ve hedef değer çiftlerinden eğitilir.

İlk aşamada, merkez vektörler gizli tabakadaki RBF' ten seçilir. Bu adım denetimsizdir. Birkaç şekilde gerçekleştirilebilir; Merkezi vektörler, bazı örneklerden örneklenebilir veya k-means kümeleme kullanılabilir.

İkinci aşamada ise amacı olan fonksiyonlara göre gizli katmanın çıktılarına w{i} katsayıları olan bir lineer model uydurur. En azından regresyon/fonksiyon tahmini için ortak bir amacı olan fonksiyona örnek vermek gerekirse, En küçük kareler fonksiyonudur:

$$K(w) =^{def} \sum_{t=1}^{T} K_t(w)$$

$K_t(w)$ 'yi bulmak için;

$$K_t(w) =^{def} [y(t) - \varphi(xt, w)]^2$$

Uygun değer ağırlık seçimi ile en küçük kareler amaç fonksiyonunun minimizasyonu, uyum doğruluğunu optimize eder. Düzgünlük ve doğruluk gibi birden çok hedefin optimize edilmesi gereken durumlar vardır.

6.4.1 Enterpolasyon

RBF ağları, o fonksiyonun değerleri sonlu sayıda noktası biliniyorsa. Bir $y: R^n \rightarrow R$ fonksiyonunu enterpolasyon yapmak için kullanılabilir. $y(x_i) = b_i, i = 1, \dots N$. Bilinen x_i noktalarını radyal tabanlı fonksiyonların merkezleri olarak alarak ve aynı noktalarda temel fonksiyonların değerlerini hesaplayarak g_{ij} ağırlıklar denklemden çözülebilir.

$$\begin{bmatrix} \boldsymbol{g_{11}} & \boldsymbol{g_{12}} & \cdots & \boldsymbol{g_{1N}} \\ \boldsymbol{g_{21}} & \boldsymbol{g_{22}} & & \boldsymbol{g_{2N}} \\ \vdots & & \ddots & \vdots \\ \boldsymbol{g_{N1}} & \boldsymbol{g_{N2}} & \cdots & \boldsymbol{g_{NN}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega_1} \\ \boldsymbol{\omega_2} \\ \vdots \\ \boldsymbol{\omega_N} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b_1} \\ \boldsymbol{b_2} \\ \vdots \\ \boldsymbol{b_N} \end{bmatrix}$$

x_i noktaları farklı ise, yukarıdaki denklemdeki enterpolasyon matrisinin tekil olmadığı gösterilebilir ve böylelikle ω ağırlıkları basit yöntemlerle çözülebilir.

$\omega = G^{-1}b$ bu şekilde $G = (g_{ij})$ yazabiliriz.

6.4.2 Fonksiyon Yaklaşımı

Amaç katı enterpolasyon yapmak değil, bunun yerine daha genel fonksiyon yaklaşımı veya sınıflandırması yapmaksa, merkezler için belirgin bir seçim olmadığı için optimizasyon biraz daha karmaşıktır. Eğitim tipik olarak önce genişlik ve merkezler ve ardından ağırlıklar sabitlenerek iki aşamada yapılır. Bu, doğrusal olmayan gizli nöronların doğrusal çıkış nöronuna karşı farklı doğası dikkate alınarak haklı çıkarılabilir. Fonksiyon yaklaşımını 4 başlıkta ele alabiliriz.

- Temel fonksiyon merkezlerinin eğitimi
- Doğrusal ağırlıklar için sözde ters çözüm
- Doğrusal ağırlıkların gradyan iniş eğitimi
- Doğrusal ağırlıkların projeksiyon operatörü eğitimi

6.5 Radyal Tabanlı Fonksiyon (RBF) ile Çok Katmanlı Bir Algılayıcı (MLP) Arasındaki Benzerlikler

- MLP'ler, verilerin temel karakteristik özelliği, örneğin görüntü tanıma vb. gibi çok yüksek boyutlu kümelerin derinliklerine gömülü olduğunda RBF'lerden daha iyidir.
- RBF'ler, MLP'lere kıyasla çok daha hızlı yakınsama oranına sahiptir, çünkü RBF'lerin yalnızca bir gizli katmanı vardır.
- RBF ağı, derin öznitelik çıkarımının gerekli olmadığı ve sonuçların doğrudan girdi vektörünün bileşenleriyle ilişkilendirildiği düşük boyutlu veriler için MLP'lere göre genellikle tercih edilir.
- Çoğu makine öğrenimi modelinin aksine RBF, sağlam bir öğrenme modelidir.
- Sınıflandırma, RBF'de MLP'den daha fazla zaman alır.
- Her ikisi de ileriye dönük
- Her ikisi de evrensel yaklaşımcılardır.
- Her ikisi de benzer uygulama alanlarında kullanılmaktadır.

6.6 Radyal Tabanlı Fonksiyon (RBF) ile Çok Katmanlı Bir Algılayıcı (MLP) Arasındaki Farklar

RBF	MLF
Aktivasyon fonksiyonu, girdi vektörünün ve belirli bir vektörün öklid mesafesinin bir fonksiyonudur.	Aktivasyon fonksiyonu,amaca hizmet edebilecek herhangi bir lineer olmayan fonksiyon olabilir
RBF'nin son katmanı aktivasyon fonksiyonunu kullanmaz, önceki nöronun çıktısını lineer olarak birleştirir.	MLP'deki son katman, doğrusal olarak birleştirmeden önce aktivasyon işlevini de kullanır.
RBF'de yalnızca bir gizli katman ve bir çıktı katmanı vardır.	MLP'de birden fazla gizli katman olabilir.
Eğitim seti, RBF'e göre kıyasla daha yavaştır.	Eğitim seti, MLP'den az da olsa daha hızlıdır.
Eğitimden sonra MLP, RBF'den çok daha hızlıdır.	Eğitimden sonra RBF, MLP'den çok daha yavaştır.
Son katmanda sadece bir nöron bulunur.	

7.GITHUB ve GIT

7.1 GIT

2005'ten bu zamana piyasadaki en popüler versiyon kontrol sistemlerden biri haline gelen açık kaynaklı bir projedir. Git sayesinde yapacağınız projelerin her versiyonunun kopyalarını alarak daha sonra ihtiyaç duyduğunuzda kopyalara kolayca erişim sağlayabilirsiniz.

7.2 GITHUB

Bir topluluğun sürüm kontrol sisteminde bir yazılım geliştirirken kullanabilecekleri internet tabanlı bir depolama servisidir. Günümüzde geliştiriciler projelerinde sürekli kodlarına yeni eklentiler yapmaktadır. Geliştiriciler projelerini yayınlayabilmesinin ardından yeni özellikler ekleme, hata düzenleme ve güncelleme gibi işlemler ile yaptıkları projeyi ayakta tutmaya çalışırlar. Github yazılım sektöründe kullanılan neredeyse rakipsiz bir kontrol sistemi olmaktadır. Github'un yararlarına değinecek olursak:

- Grup veya bireysel olarak yazılım geliştirmeye veya yazılım yönetmeye imkân sağlar
- Uygulanacak veya hazırlanan projelerin görev dağılımının netleştirilmesini sağlamaktadır.
- Projenizi özel veya genel olarak görünürlüğünü ayarlayabilirsiniz. Bu şekilde kodlar açık kaynak haline getirilebilir. Özel seçeneği sadece istenen kişiler ile kod paylaşımı yapmak için kullanılır.
- Ücretsiz kaynak kodlarının depolanmasını sağlamaktadır.
- Projeyi geliştirmek için dahil olan bireylerin yaptıkları herhangi bir değişikliğin takip edilmesine olanak sağlar ve grup içinde senkronizasyonu sağlamaktadır.

8 ANACONDA3 NEDİR?

Anaconda ücretsiz ve açık kaynaklı olması yanında, Python ve R programlama dillerinin bilimsel hesaplama ve analiz kullanımında paket yönetimini kolaylaştırmayı hedefleyen bir açık kaynaklı dağıtımdır. Paket sürümleri conda ile yönetilir.

9. UYGULAMA SÜRECİ

9.1 Anaconda3 ve Github Desktop Bilgisayara Yüklenmesi

İlk başta bilgisayarıma Anaconda3 ve GitHub Desktop uygulamalarını kendi websiteleri üzerinden indirip kurulumunu sağladım. Sonrasında inen uygulamalar üzerinde Jupyter Notebook'ta birkaç python denemesi yaparak uygulamanın doğru olarak kurulup kurulmadığını test ettim.

9.2 Veri Seti Araştırılması

Kaggle adlı website üzerinde bu projeme uygun veri setleri aramaya başladım. Araştırmalarım sonucunda University of California'nın 2012 yılında paylaştığı Banka Pazarlama veri setini kullanmayı belirledim.[2]

9.3 Veri Setinin Özeti

Özet olarak bu veriler, bir Portekiz bankacılık kurumunun doğrudan pazarlama kampanyaları (telefon görüşmeleri) ile ilgilidir. Sınıflandırma hedefi, müşterinin bir vadeli mevduata abone olup olmayacağını tahmin etmektedir.

Veriler, bir Portekiz bankacılık kurumunun doğrudan pazarlama kampanyalarıyla ilgilidir. Pazarlama kampanyaları telefon görüşmelerine dayanıyordu. Çoğu zaman, ürüne (banka vadeli mevduat) abone olup olmayacağına ('evet') veya abone olmayacağına ('hayır') erişmek için aynı müşteriyle birden fazla iletişim gerekliydi.

Sınıflandırma hedefi, müşterinin bir vadeli mevduata (y değişkeni) abone olup olmayacağını (evet/hayır) tahmin etmektir.

9.4 Veri Setinin Özellikleri

9.4.1 Girdi Değişkenleri

Banka müşteri verileri:

- Yaş (sayısal olarak):
- İş: İş türü (kategorik: 'yönetici', 'mavi yakalı', 'girişimci', 'hizmetçi', 'yönetim', 'emekli', 'serbest meslek', 'hizmet', 'öğrenci', 'teknisyen', 'işsiz', 'bilinmiyor')
- Medeni Hali: medeni durum (kategorik: 'boşanmış', 'evli', 'bekar', 'bilinmiyor') '; not: 'boşanmış' boşanmış veya dul anlamına gelir)
- Eğitim: (kategorik: 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'okuma yazma bilmeyen', 'profesyonel kurs ', 'university.degree ', 'bilinmiyor')
- Temerrüt: temerrütte kredi var mı? (Kategori: 'hayır', 'evet', 'bilinmiyor')
- Konut: konut kredisi var mı? (Kategori: 'hayır','evet','bilinmeyen')
- Kredi: kişisel kredisi var mı? (Kategori: 'hayır', 'evet', 'bilinmeyen')

Mevcut kampanyanın son temasıyla ilgili:

- Kişi: iletişim türü (kategorik: 'cep telefonu', 'telefon')
- Ay: yılın son iletişim ayı (kategorik: 'ocak', 'şubat', ' mar', ..., 'kasım', 'aralık')
- Haftanın günü: haftanın son iletişim günü (kategorik: 'pte','sal','çrş','per','cum')
- Süre: saniye cinsinden son iletişim süresi (sayısal). Önemli not: bu öznitelik, çıktı hedefini büyük ölçüde etkiler (örneğin, süre=0 ise y='hayır'). Ancak arama yapılmadan önce ne kadar süreceği bilinmiyor. Ayrıca, çağrının bitiminden sonra y açıkça bilinir. Böylece, bu girdi yalnızca kıyaslama amacıyla dahil edilmelidir ve eğer amaç gerçekçi bir tahmine dayalı modele sahip olmaksa atılmalıdır.

Diğer Nitelikler

- Kampanya: bu kampanya sırasında ve bu müşteri için gerçekleştirilen iletişim sayısı (sayısal, son iletişim dahil)
- Pdays: müşteriyle önceki bir kampanyadan en son iletişim kurulduktan sonra geçen gün sayısı (sayısal; 999, müşteriyle iletişim kurulmadığı anlamına gelir) 14- önceki: bu kampanyadan önce ve bu müşteri için gerçekleştirilen temasların sayısı (sayısal)
- Poutcome: önceki pazarlama kampanyasının sonucu (kategorik: 'başarısızlık', 'yok', 'başarı')

9.4.2 Çıktı Değişkeni (İstenen Hedef)

- Y: müşteri bir vadeli mevduat aboneliği yaptı mı? (İkili: 'evet', 'hayır')

10. KODLAMA İŞLEMLERİ

- Gerekli paketlerimizi içeri aktarıyoruz (Math, Pandas, Sklearn, Numpy). [Kod 11.1]
- Dosyadan verilerimizi alıyoruz ve verilerimizi gösteriyoruz. [Kod 11.2]
- LabelEncoder sınıfını kullanarak etiketleri kodluyoruz. [Kod 11.4 ve Kod 11.5]
- Verileri eğitim ve test setine ayırıyoruz. [Kod 11.9 ve Kod 11.10]
- StandardScaler sınıfını kullanarak verileri ölçeklendiriyoruz [Kod 11.10]
- Kmeans kullanarak nöronların merkezlerini belirliyoruz. [Kod 11.11]
- Sigma değerini belirliyoruz. [Kod 11.11]
- G matrisini buluyoruz. [Kod 11.12]
- Ağı eğitmek için W ağırlık matrisini buluyoruz. [Kod 11.12]
- Test seti için G matrisini kuruyoruz. [Kod 11.12 ve Kod 11.13]
- Test setinde tahminin doğruluğunu analiz ediyoruz. [Kod 11.14]

11. KODLAR

Kod 11.1:

```
1. import pandas as pd
2. import numpy as np
3. from sklearn.preprocessing import LabelEncoder
4. from sklearn.model_selection import train_test_split
5. from sklearn.preprocessing import StandardScaler
6. from sklearn.cluster import KMeans
7. import math
8. from sklearn.metrics import accuracy_score
```

Kod 11.2:

```
1. dataset = pd.read_table("bank-full.csv", sep= None, engine=
   "python")
2. dataset
```

Kod 11.2 Çıktısı:

```
In [35]: dataset
```

Out[35]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	-1	0	unknown	yes
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	-1	0	unknown	yes
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	3	success	yes
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	-1	0	unknown	no
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	11	other	no

45211 rows × 17 columns

Kod 11.3

```
1. dataset.head()
```

Kod 11.3 Çıktısı:

```
In [13]: dataset.head()
```

Out[13]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

Kod 11.4:

```
1. col_to_use =
   ['age','balance','day','duration','campaign','pdays','previous'
   ]
2. data = dataset.drop(col_to_use,axis=1)
3. data.head()
```

Kod 11.4 Çıktısı:

```
In [14]: col_to_use = ['age','balance','day','duration','campaign','pdays','previous']
         data = dataset.drop(col_to_use,axis=1)
```

```
In [15]: data.head()
```

Out[15]:

	job	marital	education	default	housing	loan	contact	month	poutcome	y
0	management	married	tertiary	no	yes	no	unknown	may	unknown	no
1	technician	single	secondary	no	yes	no	unknown	may	unknown	no
2	entrepreneur	married	secondary	no	yes	yes	unknown	may	unknown	no
3	blue-collar	married	unknown	no	yes	no	unknown	may	unknown	no
4	unknown	single	unknown	no	no	no	unknown	may	unknown	no

Kod 11.5

```
1. data = data.apply(LabelEncoder().fit_transform)
2. data.head()
```

Kod 11.5 Çıktısı:

```
In [16]: data = data.apply(LabelEncoder().fit_transform)
```

```
In [17]: data.head()
```

Out[17]:

	job	marital	education	default	housing	loan	contact	month	poutcome	y
0	4	1	2	0	1	0	2	8	3	0
1	9	2	1	0	1	0	2	8	3	0
2	2	1	1	0	1	1	2	8	3	0
3	1	1	3	0	1	0	2	8	3	0
4	11	2	3	0	0	0	2	8	3	0

Kod 11.6:

```
1. data['job'].unique()
```

Kod 11.6 Çıktısı:

```
array([ 4,  9,  2,  1, 11,  5,  0,  7,  6, 10,  3,  8])
```

Kod 11.7:

```
1. dataset['job'].unique()
```

Kod 11.7 Çıktısı:

```
array(['management', 'technician', 'entrepreneur', 'blue-collar',
       'unknown', 'retired', 'admin.', 'services', 'self-employed',
       'unemployed', 'housemaid', 'student'], dtype=object)
```

Kod 11.8:

```
1. data_rest = dataset[col_to_use]
2. data_rest.head()
```

Kod 11.8 Çıktısı:

```
In [20]: data_rest = dataset[col_to_use]
         data_rest.head()
```

Out[20]:

	age	balance	day	duration	campaign	pdays	previous
0	58	2143	5	261	1	-1	0
1	44	29	5	151	1	-1	0
2	33	2	5	76	1	-1	0
3	47	1506	5	92	1	-1	0
4	33	1	5	198	1	-1	0

Kod 11.9:

```
1. dataset2 = pd.concat([data_rest,data],axis=1)
2. dataset2.head()
```

Kod 11.9 Çıktısı:

```
In [21]: dataset2 = pd.concat([data_rest,data],axis=1)
```

```
In [22]: dataset2.head()
```

Out[22]:

	age	balance	day	duration	campaign	pdays	previous	job	marital	education	default	housing	loan	contact	month	poutcome	y
0	58	2143	5	261	1	-1	0	4	1	2	0	1	0	2	8	3	0
1	44	29	5	151	1	-1	0	9	2	1	0	1	0	2	8	3	0
2	33	2	5	76	1	-1	0	2	1	1	0	1	1	2	8	3	0
3	47	1506	5	92	1	-1	0	1	1	3	0	1	0	2	8	3	0
4	33	1	5	198	1	-1	0	11	2	3	0	0	0	2	8	3	0

Kod 11.10:

```
1. X= dataset2.drop('y',axis=1)
2. y= dataset2['y']
3.
4. X_train, X_test, y_train, y_test= train_test_split(X,y,
   test_size= 0.33, random_state= 4)
5.
6. scaler = StandardScaler()
7. X_train = scaler.fit_transform(X_train)
8. X_test = scaler.transform(X_test)
9. X_train
```

Kod 11.10 Çıktısı:

```
array([[ 3.39736083, -0.45852779,  0.14087303, ..., -0.71332324,
        -1.50513854,  0.44672656],
       [-0.27605947,  2.43941798,  0.50067953, ...,  0.40010942,
        -1.83733874, -2.58685457],
       [ 0.66584317, -0.13326522, -0.93854645, ..., -0.71332324,
         1.48466323, -0.56446715],
       ...,
       [-0.74701079, -0.22619739,  0.62061503, ..., -0.71332324,
         1.15246303,  0.44672656],
       [ 1.60774581, -0.42275894, -1.53822394, ...,  1.51354209,
         0.15586244,  0.44672656],
       [ 0.28908211, -0.13627382,  1.58009901, ..., -0.71332324,
        -0.17633776,  0.44672656]])
```

Kod 11.11:

```
K_cent= 8
km= KMeans(n_clusters= K_cent, max_iter= 100)
km.fit(X_train)
cent= km.cluster_centers_

max=0
for i in range(K_cent):
    for j in range(K_cent):
        d= np.linalg.norm(cent[i]-cent[j])
        if(d> max):
            max= d
d= max

sigma= d/math.sqrt(2*K_cent)
```

Kod 11.12:

```
shape= X_train.shape
row= shape[0]
column= K_cent
G= np.empty((row,column), dtype= float)
for i in range(row):
  for j in range(column):
    dist= np.linalg.norm(X_train[i]-cent[j])
    G[i][j]= math.exp(-math.pow(dist,2)/math.pow(2*sigma,2))
```

Kod 11.13

```
GTG= np.dot(G.T,G)
GTG_inv= np.linalg.inv(GTG)
fac= np.dot(GTG_inv,G.T)
W= np.dot(fac,y_train)
row= X_test.shape[0]
column= K_cent
G_test= np.empty((row,column), dtype= float)
for i in range(row):
    for j in range(column):
            dist= np.linalg.norm(X_test[i]-cent[j])
            G_test[i][j]= math.exp(-
math.pow(dist,2)/math.pow(2*sigma,2))
```

Kod 11.14:

```
prediction= np.dot(G_test,W)
prediction= 0.5*(np.sign(prediction-0.5)+1)

score= accuracy_score(prediction,y_test)
print(score.mean())
```

Kod 11.14 Çıktısı:

0.8904155495978552

12.SONUÇ ve TARTIŞMA

RBF ağı ile %89 tahmin doğruluğu elde ettik. Bu kod çalışmamızı ele alacak olursak RBF ağında elde ettiğimiz sonuç ile çok katmanlı algılayıcı (MLP) sonuç aynı çıkacaktır. RBF ağları ile MLP kıyaslanmaya çalışıldığında, MLP eğitiminin hesaplama maliyeti RBF'den daha yüksektir. Bu nedenle MLP yerine RBF ağını kullanmamız daha uygundur. Rapor sonucunda, makine öğrenimi, yapay sinir ağları, veri bilimi, RBF, RBFN'in ne derecede önemli olduğunu, günümüzde ne şekillerde kullanıldığını, hangi parçalar ile bağlı olduğunu, eğitim maliyeti durumunu, alternatif seçenekler arasında neden seçilmesi gerektiğini öğrenip örnek üzerinde açıklamaya çalıştık.

KAYNAKÇA

International Business Machines(IBM), “Makine Öğrenmesi “, https://www.ibm.com/tr-tr/cloud/learn/machine-learning

Papatya Yayincilik, Istanbul, 2003- papatyabilim.com.tr Yapay Sinir Ağları-http://papatyabilim.com.tr/PDF/yapay_sinir_aglari.pdf

Z. Şen, YAPAY SİNİR AĞLARI İLKELERİ, Su Vakfı, Istanbul 2004

G. J. Williams, The Essentials of Data Science: Knowledge Discovery Using R, Ocak 2017

B. Lubanovic, Introducing Python: Modern Computing in Simple Packages, Kasım 2014

Wikipedia, Radial Basis Functions Networks, https://en.wikipedia.org/wiki/Radial_basis_function_network

[Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

[1] BERKELEY, “What Is Machine Learning (ML)?”, https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/

[2] University of California Bank Marketing Data Set, http://archive.ics.uci.edu/ml/datasets/Bank+Marketing

[Şekil 5.1] R.Achal, Quara, https://www.quora.com/Can-C-do-everything-that-Python-can 2017

GITHUB Kaynak Dosyaları, https://github.com/SzBereket/Radial-Basis-Function-Networks