T.C.

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ

MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

BİLGİSAYAR MÜHENDİSLİĞİ TEZLİ YÜKSEK LİSANS PROGRAMI

YAPAY SİNİR AĞLARI, KELİME VEKTÖRLERİ VE DERİN ÖĞRENME UYGULAMALARI

HAZIRLAYANLAR:

Sedrettin ÇALIŞKAN

Selahattin A.YAZICIOĞLU

Ulaş DEMİRCİ

Zeki KUŞ

DANIŞMAN:

Yrd. Doç. Dr. Ebubekir KOÇ

2018 - İSTANBUL

İçindekiler Tablosu

YAPAY SİNİR AĞLARI, KELİME VEKTÖRLERİ VE DERİN ÖĞRENME UYGULAMALARINA GİRİŞ

Zeki KUŞ

Mühendislik ve Fen Bilimleri Enstitüsü, Fatih Sultan Mehmet Vakıf Üniversitesi, Beyoğlu, İstanbul,
zeki.kus@stu.fsm.edu.tr

ÖZET

Bu çalışmada, yapay sinir ağları, kelime vektörleri ve derin öğrenme uygulamaları hakkında yapılan çalışmalardan bahsedilmiştir. Bu çalışmalarda kullanılan kütüphaneler, metodolojiler ve veri setleri açıklanmıştır. Yapay sinir ağları kullanılarak yapılan veri sınıflandırma örnekleri, yazar tanıma sistemi, kelime vektörleri kullanılarak verilen metin içerisinde geçen kelimelerin birbirleri ile olan ilişkilerinin görselleştirilmesi ve türkçe metinlerin anlam analizinin gerçekleştirilmesi çalışmaları anlatılmıştır. Yapılan farklı çalışmaların sonuçları, parametre analizi ve kaynak kodları makale içerisinde ve ekler kısmına konumlandırılmış bir şekilde paylaşılmıştır.

Anahtar kelimeler: Yapay sinir ağları, kelime vektörleri, sınıflandırma, derin öğrenme.

1. GİRİŞ

Yapay sinir ağları, makine öğrenmesi, derin öğrenme gibi konular son zamanlarda çok fazla konuşulmaya başlanan konulardır. Gelişen bilgisayar teknolojisi bu alanlarda yapılan çalışmaların tekrar bir ivme kazanmasını sağlamıştır.

Yapay sinir ağı ve derin öğrenme metodları günlük hayatta farkına varmasakta yoğun bir şekilde kullanılmaktadır. Bankaya kredi başvurusu için gittiğimizde bizim için bir kredi derecelendirmesi yapan sistem en temelinde veri analizi ve sınıflandırma tekniklerinden faydalanmaktadır. Bu sistemlerin eğitilmesinde yapay sinir ağları büyük önem taşımaktadır. Sistemlerin eğitimi aynı insanların öğrenme yöntemine benzer olarak ilerler. Bilgisayarlar örnekler üzerinden öğrenirler. Daha önce belirli özelliklere göre kredi notu derecelendirmesi yapılan veriler yani kişiler bilgisayarların öğrenmesi için verilir. Verilen örneklerle kurulan yapay sinir ağı modeli eğitilir. Yapay sinir ağının modellenmesi problemin tipine göre gerçekleştirilmektedir. Eğitilen yapay sinir ağı modeli test verileri ile denenerek başarı oranı belirlenir. İstenen başarı oranına erişen model artık bir nevi kredi derecelendirme uzmanı olarak nitelendirilir. Bu sistemlerin eğitilmesi, yeni bir insan uzmanın yetiştirilmesine kıyasla çok daha kolay ve hızlıdır. Aynı zamanda bilgisayar sistemlerinde geliştirilen uzmanların sayısının arttırılması sadece yazılımın çoğaltılması ile ilişkilidir.

Bir önceki paragrafta bahsettiğim yöntemler bilgisayarlar için sınırlı bir öğrenme şeklidir. Aslında derin öğrenme uygulamaları da yapay sinir ağlarının daha geliştirilmiş ve kompleks halidir. Fakat derin öğrenme yöntemleri kendi içinde de farklılaşmıştır. Görüntü sınıflandırma işlemleri için “Convolutional Neural Networks”, doğal dil işleme, metin analizi gibi yöntemler için “Long Short Term Memory (LSTM)”, “Recurrent Neural Networks” gibi farklı yapılar ve sinir ağları geliştirilmiştir. Bu gibi yapılarla bilgisayarların sınırlı öğrenme yapısı genişletilmeye çalışılmaktadır. Verilen sınırlara, kurallara bağlı kalmak yerine sistemin daha genel anlamda verilen farklı örneklerden yola çıkarak kendisinin yeni kurallar üretmesi için çalışmalar yapılmaktadır. Gözetimli öğrenme yerine kendi başına öğrenen, tanımlanmamış durumlarda mantıklı kararlar alabilen sistemler geliştirmek önemli bir uygulama alanı haline gelmiştir.

Yapay sinir ağları ve derin öğrenme kavramları dışında kelime vektörü kavramı da sık karşılaşılmaya başlanmıştır. Kelime vektörleri en basit şekilde kelimelerin birbirleri ile olan ilişkilerine odaklanmaktadır. Bu kelimelerin ilişkilerinden yola çıkarak anlamsal analizler yapılmaktadır. Bu ilişkilerden yola çıkarak daha ileri seviyede “chat bot”, “sanal asistan” gibi uygulamalar geliştirilmektedir. Bir cümle söylendiğinde aslında anlatılmak istenenin ne olduğu, kelimeler arasında bulunan sıklık ilişkileri ele alınarak geliştirilen öneri

sistemleri gibi uygulamalar kelime vektörleri ve derin öğrenme teknikleri ile yapılabilecek çalışmalara en güzel örneklerdir.

Bu çalışmada bahsedilen Türkçe yazar tanıma sistemi gibi uygulamalarla ilgili birkaç Türkçe çalışma yapılmıştır. Aynı zamanda yapay sinir ağları kullanılarak yapılan sınıflandırma örnekleri, derin öğrenme teknikleri ile türkçe metinlerin anlamsal analizi gibi uygulamalar da daha önce gerçekleştirilmiştir. Daha önce yapılmış olan uygulamaların bu çalışmada hangi yöntemler kullanılarak ele alındığı ve gerçekleştirildiği, aynı zamanda farklı olarak ele alınan özellik ve uygulamalar anlatılmıştır. Elde edilen sonuçlar son bölümde ve herbir uygulamanın kaynak koduna açıklama satırları ile beraber ekler bölümünde yer verilmiştir.

2. KULLANILAN YÖNTEMLER

2.1 Yapay Sinir Ağı

Yapay sinir ağları insan beyninin öğrenme şekli esas alınarak geliştirilmiştir. Öğrenme işleminin gerçekleştirildiği yapı insan vücudundaki sinir ağlarına benzediği için bu şekilde adlandırılmıştır. Belirli karmaşık problemlerin çözümü için sinir ağlarında olduğu gibi düğümler ve bu düğümler arasında bilgi akışı ve öğrenmeyi sağlayan bağlantılar bulunur. Yapay sinir ağlarında öğrenme işlemi örnekler ile gerçekleştirilir.[1] Aynı insan beyninde olduğu gibi yapay sinir ağlarında da sistem öncelikle örnekler ile eğitilir. Eğitilen sistem belli test değerleri ile çalıştırılır ve sistemin doğruluk oranı belirlenir.

Yapay sinir ağlarında öğrenme işlemi ağırlıklara verilen değerler ile gerçekleştirilir. Öğrenilen bilgi bütün ağa yayılmış olan ağırlıklarda saklıdır. [1] Yapay sinir ağları çok fazla girdinin, özelliğinin bulunduğu sistemlerde anlaşılması zor olan bağlantıları, ilişkileri ortaya çıkartabilir.[3] Yapay sinir ağları sınıflandırma, tahmin, örüntü tanıma vb. gibi birçok uygulamada başarılı olarak kullanılmaktadır. Problem tipleri farklılık gösterdiğinden dolayı ortak bir yapay sinir ağı modeli bulunmamaktadır. Problemin girdileri, girdilerin tipleri (tam sayı, metin, reel sayı vb.), beklenen çıktılar, çıktıların şekli (kategorik, ikili) gibi özellikler problemin zorluğunu ve aynı zamanda yapay sinir ağı modelinin yapısını değiştirmektedir. Oluşturulan yapı belirsizliklere, hatalara ve daha önce karşılaşılmamış olan örneklere, durumlara karşı toleranslı olmalıdır. [1]

Yapay sinir ağının tarihçesine bakıldığında ilk hesaplama modeli W.S. McCulloch ve W.A. Pitts'in, 1943 yılında yayınladığı makale ile ortaya çıkmıştır. Bu çalışmaları takiben 1954 yılında B.G. Farley ve W.A. Clark tarafından uyarılara tepki verebilen bir model oluşturulmuştur. 1970 yılına kadar XOR probleminin sinir ağları ile çözülememesi bu konuda yapılan çalışmaları sekteye uğratmıştır. XOR probleminin çözümü bu alanda yapılan çalışmaları tekrar hızlandırmış ve birbirinden farklı 30 farklı modelin daha geliştirilmesini sağlamıştır.[1] Günümüze kadar yapay sinir ağı çalışmaları belli aralıklarla devam etmiş fakat istenilen sonuçlara ulaşılamamıştır. Fakat son 10 yıl içerisinde gelişen bilgisayar teknolojisi ve donanımlar bu alanda yapılan çalışmaların tekrar hız kazanmasını sağlamış ve olumlu sonuçlar elde edilmiştir.

Yapay sinir ağları günümüzde finans, görüntü ve ses tanıma, metin analizi, tıp, haberleşme vb. gibi farklı alanlarda etkin olarak kullanılmaktadır.

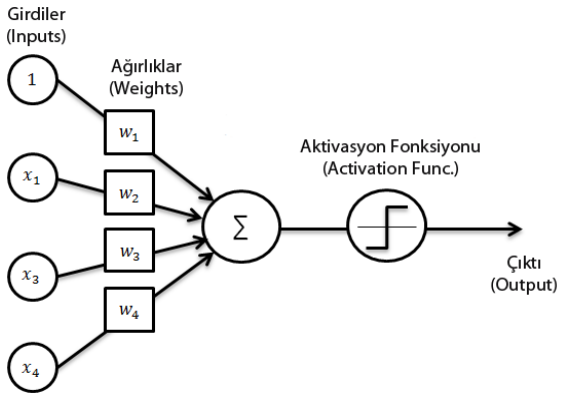
2.1.1 Yapay Sinir Ağı Yapısı

Yapay sinir ağı, biyolojik sistemde beyinde bulunan nöronlardan esinlenilerek oluşturulmuş basit işlemcilerden oluşur. Nöronlar, sinyalleri bir nörondan diğer nörona doğru geçiren ağırlıklandırılmış bağlantılar ile birbirine bağlıdır. Sinir hücreleri arasındaki iletişim synapsler yardımıyla gerçekleşir. Yapay sinir ağı, sinir hücrelerine gelen bilgileri toplayıp, belirlenen aktivasyon fonksiyonundan geçirerek çıktıyı üretir. Üretilen çıktı bağlantılar üzerinden diğer hücrelere gönderilir.[4]

Tablo 1. Biyolojik Sinir Sistemi ve Yapay Sinir Ağı Karşılaştırması

Biyolojik Sinir Sistemi	Yapay Sinir Ağı
Soma	Nöron
Dendrite	Girdi
Axon	Çıktı
Synapse	Ağırlık

Yapay sinir ağlarında biyolojik sistemde olduğu gibi nöron yapısı bulunmaktadır. Nöron yapısına bakıldığında girdilerin tutulduğu düğümler, bu girdilere ait ağırlıklar, bütün girdi ve ağırlıkların çarpılıp toplanmasıyla elde edilen toplama fonksiyonu, toplama fonksiyonundan gelen değerin belirli bir aralığa normalize edildiği aktivasyon fonksiyonu ve çıktıdan oluşur.



Şekil 1. Perceptron Yapısı

Çok katmanlı yapay sinir ağlarına bakıldığında temel olarak 3 katmandan meydana gelir. Bu katmanlar sıralı bir yapay sinir ağı modelini oluşturur. Bunlar;

- Girdi Katmanı (Input Layer)
- Gizli, Ara Katmanlar (Hidden Layers)
- Çıktı Katmanı (Output Layer)

Girdi Katmanı (Input Layer): Yapay sinir ağında gelen bilgiler girdi katmanında temsil edilir. Veri setine göre değişen özelliklerin her biri farklı bir düğüm olarak girdi katmanında temsil edilir. Her bir girdinin (özelliğin) ağırlık(weight) değeri vardır. Bu girdiler gizli katmandaki düğümler ile bu ağırlıklar aracılığı ile bağlıdır. Girdiler için belirlenen ağırlık değerleri yapay sinir ağında o özelliğin önemini, ağırlığını belirtmektedir.

Gizli Katmanlar (Hidden Layers): Gizli katmanlar, girdi katmanından gelen bilgilerin işlenip bir çıktıya dönüştürüldüğü katmanlardır. Çıktıya dönüştürme işlemi yapay sinir ağının ağırlık değerleri kullanılarak gerçekleştirilir. Gizli katman sayısı problemin zorluğuna göre çeşitlilik gösterebilmektedir.

Çıktı Katmanı (Output Layer): Gizli katmandan gelen sonuç bilgisi hesaplanan çıktı değeri olarak belirlenir. Çıktı değeri veya değerlerinin tutulduğu katmandır. Verilen girdi değerine ait çıktıyı barındırır.

Hesaplanan çıktı değeri ile beklenen çıktı değeri belirlenen "hata hesaplama" fonksiyonuna göre hesaplanır. Hesaplanan hata değeri beklenen sonuçtan ne kadar uzak olduğumuzu belirler. Hesaplanan bu değere göre seçilen "optimizasyon fonksiyonu" kullanarak ağırlıklar güncellenir. Bu işlem aslında ağın öğrenmesi demektir. Yapay sinir ağı ağırlıklarını güncelleyerek öğrenmeye başlar. Hata optimize edilmeye başlanır. Beklenen sonuçları veren ağırlıkların bulunması sistemin eğitilmesi anlamına gelir.[1] Ağırlıkların güncellenmesi için farklı yöntemler kullanılabilir. Bazı farklı yöntemler;

İleri Beslemeli Ağlar: Akışın girdi katmanından çıktı katmanına doğru gerçekleştiği ağ yapısıdır. Ağırlıkların güncellenmesi sadece ileriye doğru gerçekleşmektedir. Bir katmandan gelen nöronların çıktıları, diğer katmandaki nöronlara ağırlıklar aracılığıyla girdi olarak verilir.[5]

Geri Beslemeli Ağlar: Akışın yalnızca ileriye doğru değil aynı zamanda geriye doğru da olabildiği ağ yapısıdır. Ağırlıkların güncellenmesi geriye yada ileriye doğru gerçekleştirilebilir.[5]

Yapay sinir ağlarının başarısını doğrudan etkileyen birçok parametre bulunmaktadır. Bu parametrelerin optimize edilmesi yapay sinir ağının başarısını olumlu bir şekilde arttıracaktır. Bazı parametreler: [1]

- Örnek Sayısı
- Verilerin normalize edilmesi
- Katman sayısı
- Girdi ve çıktı verilerinin uygun bir şekilde temsil edilmesi
- Aktivasyon fonksiyonlarının doğru seçilmesi
- Öğrenme oranı
- Hata hesaplama ve hata optimizasyonu fonksiyonlarının doğru seçilmesi

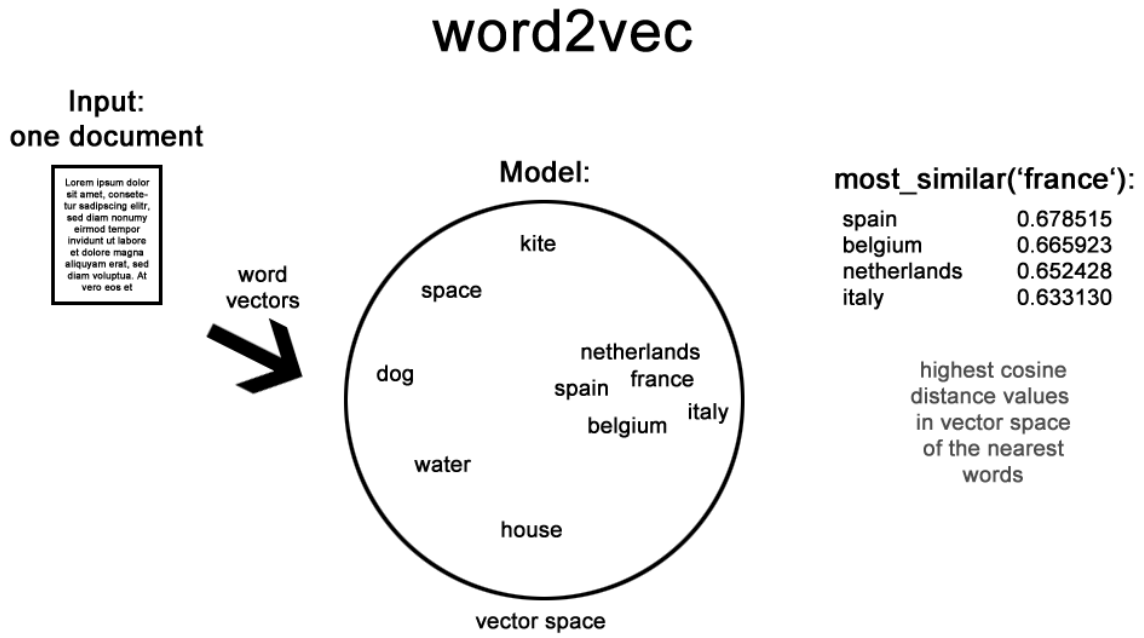
gibi parametreler yapay sinir ağının başarısını doğrudan etkilemektedir. Bu parametreler farklı problemler için farklılık gösterebilmektedir. Yapay sinir ağını oluştururken veri seti doğru bir şekilde incelenmelidir. Veri setinin nasıl temsil edileceği (İkili veri, kategorik veri, reel sayılardan oluşan veri), hangi yöntemlerin kullanılacağı, problemin temelde hangi ağ yapısı kullanılarak oluşturulabileceği (CNN, RNN, ANN vb.) araştırılarak yapay sinir ağı oluşturulmalıdır.

Yapay sinir ağları birçok gerçek dünya probleminin çözümünde iyi sonuçlar vermektedir. Sınıflandırma, eğri uydurma, tahmin etme, tanımlama vb.

2.2 Kelime Vektörleri (Word2Vec)

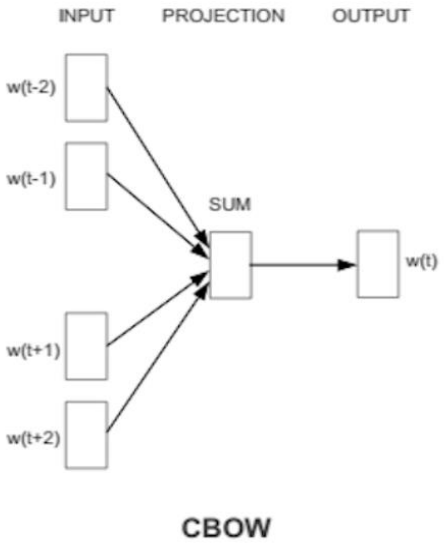
Word2Vec kelimeler arasındaki ilişkileri ortaya çıkarmamızı sağlayan bir çeşit algoritma aracıdır. Analiz edilen metinlerde geçen kelimelerin birbirleri ile olan uzaklık ve yakınlık ilişkilerini vektörel olarak hesaplanabilinmesini sağlar. Hesaplanan bu ilişkiler kolay bir şekilde görselleştirilebilir. Kullanılan bir kelimeye en yakın kelimeleri bularak öneri sistemleri oluşturabilirsiniz.

Word2Vec yapısı Google'da çalışan Tomas Mikolov tarafından yönetilen bir ekibin araştırma çalışmaları ile oluşturulmuştur. Geliştirilen yapı daha sonra diğer araştırmacılar tarafından analiz edilip açıklanmıştır.



Şekil 2. Word2Vec Örnek Çalışma Akışı [6]

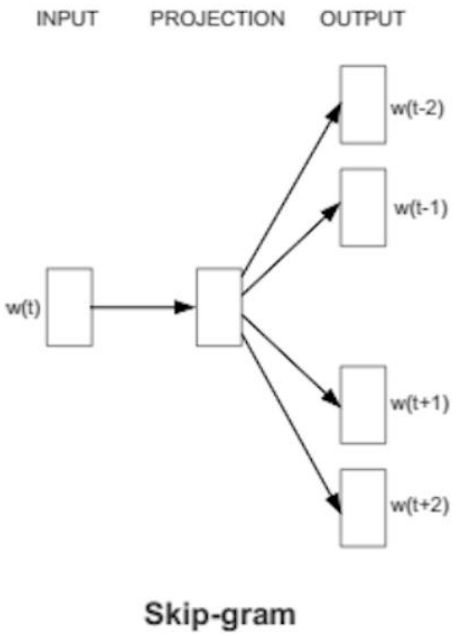
Word2Vec kelimelerin vektörel temsili için iki farklı model mimariden birini kullanabilir: CBOW(Continuous Bag Of Words) ve skip-gram.



Şekil 3. CBOW Yapısı [1]

Yukarıdaki şekilde gösterilen mimari sözcük çiftleri arasındaki ilişkileri öğrenmek içindir. Bu yaklaşım her bir kelimenin bir belgede kaç defa göründüğünü içerir. Fakat sözcük sıralaması dikkate alınmaz. Bu yaklaşım ile belge içerisinde en fazla geçen kelimelere odaklanılır. Az kullanılan kelimeler için

gereksiz bellek ayrılmasına gerek kalınmaz. Yaygın olarak basit belge sınıflandırmalarında kullanılır. Örneğin bir e-postanın spam yada spam değil olarak sınıflandırılması için başarılı sonuçlar elde edilebilir. Fakat anlam analizi, makine çevirisi gibi konularda başarılı sonuçlar elde etmek çok daha zordur. Bu yapıda kelimelerin sırası önemsiz olduğundan "bir adam köpek ısırdı" cümlesi ile "bir köpek adam ısırdı" cümlesi aynı anlama gelmektedir.



Şekil 4. Skip-gram Yapısı [1]

Skip-gram yapısı bir önceki yapıda olduğu gibi yine kelimelerin birbirleri ile olan ilişkilerinin öğrenilmesi, ortaya çıkarılması için kullanılan bir yapıdır. Bu yapıda verilen kelimelerin sağındaki ve solundaki kelimeler ile olan ilişki sıklığına bakılarak istenilen sayıda sonuçlar üretilir. "Window-size" olarak belirtilen parametre kelimenin ne kadar sağına ve soluna bakılacağını belirler. Bakılan bu kelimeler için ilişki sıklıkları belirlenir. Belirlenen sonuçlara göre verilen kelime ile en fazla ilişkili olan kelimeler çıktı olarak verilir. Bu yapı aslında tek bir gizli katmandan oluşan yapay sinir ağı yapısına benzemektedir. Burdaki amaç ağırlıkları öğrenmektedir. Öğrenile ağırlıklar ise bize sonuç olarak ilişkilendirmek istediğimiz kelimeleri verir. Örneğin "kalem" kelimesi verildiğinde "yüzmek" kelimesinin gelme olasılığı "kağıt, yazmak" gibi kelimelerin gelme olasılığına göre çok daha düşüktür.

İki modeli bir örnekle incelemek gerekirse:

CBOW: Ankara Türkiye'nin _____. Cümlesinde boşluk kısmına "başkentidir" kelimesi gelirken,

Skip-gram: _____ _____ başkentidir. Cümlesinde boşluklara "Ankara Türkiye'nin" kelimelerinin gelmesi beklenir.

3. Uygulamalar

3.1 Yapay Sinir Ağları Kullanılarak Iris Verisinin Sınıflandırılması Uygulaması

Bu uygulamada yapay sinir ağları kullanılarak verilen özelliklere göre çiçeğin hangi sınıfa ait olduğu bulunacaktır. Yapılan çalışmanın yazılan uygulama kodu ekler bölümünde bulunmaktadır.(Ek- A)

Veri Seti Özellikleri:

Veri setinde çiçeklere ait 4 özellik ve çiçeğin hangi sınıfa ait olduğu bilgisi verilmiştir. Veri seti 150 örnekten oluşmaktadır.

1. Çanak Yaprak Uzunluğu (Sepal Length)
2. Çanak Yaprak Genişliği (Sepal Width)
3. Taç Yaprak Uzunluğu (Petal Length)
4. Taç Yaprak Genişliği (Petal Width)
5. Sınıf (Çıktı):
    a. Iris Setosa
    b. Iris Versicolour
    c. Iris Virginica

Yapay Sinir Ağı Parametreleri:

1. Girdi Katmanı (Input Layer) :
    a. Düğüm Sayısı (Input Units): 4
2. Gizli Katman 1 (Hidden Layer 1) :
    a. Düğüm Sayısı (Hidden Units): 32
    b. Aktivasyon Fonksiyonu: Rectifier
3. Gizli Katman 2 (Hidden Layer 2):
    a. Düğüm Sayısı (Hidden Units): 32
    b. Aktivasyon Fonksiyonu: Rectifier
4. Çıktı Katmanı (Output Layer):
    a. Düğüm Sayısı (Output Units): 3
    b. Aktivasyon Fonksiyonu: Sigmoid
    c. Hata Optimizasyonu (Optimizasyon Fonksiyonu): Adam
    d. Hata Hesaplama (Loss Fonksiyonu): Categorical Crossentropy
    e. Değerlendirme Kriteri (Metric): Accuracy
5. Diğer:
    a. İterasyon Sayısı(Number epoch): 100
    b. Eğitim verileri yüzdesi: %80
    c. Test verileri yüzdesi: %20

Algoritma Akışı:

1. "Iris.data" dosyasından veri seti okunur.
2. Veri seti girdi ve çıktı olarak ayrıştırılıp değişkenlere atanır.

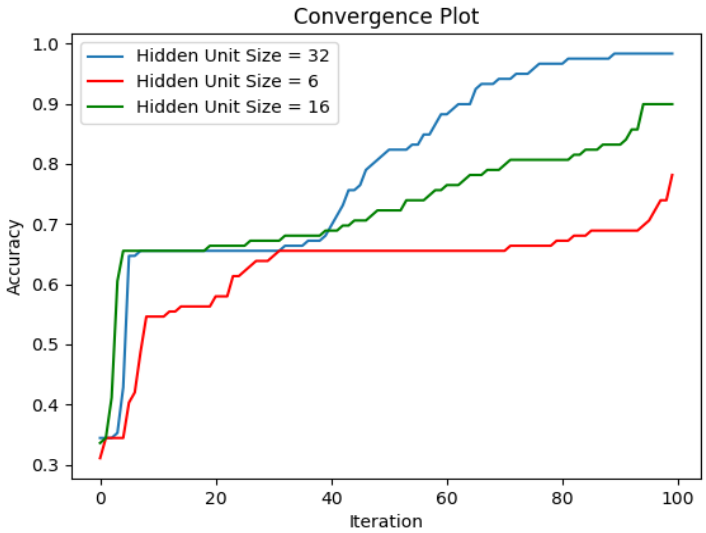
3. Veri seti çıktı olarak 3 sınıftan oluştuğundan çıktı kategorik veri olarak adlandırılır. Sınıflar "string" tipinde olduğundan sinir ağının işleyebileceği "numerik" tipine dönüştürülmesi gerekir. Bu adımda sınıflar sayısal verilere dönüştürülür.
4. Veri seti üzerinde "z-score normalizasyon" işlemi gerçekleştirilir. Normalizasyon işlemi yapılarak girdi verileri arasında bazı girdilerin baskın gelmesi engellenir. Örneğin: 4 özellikten oluşan bu veri seti için gelen örneğin şu şekilde olduğunu varsayalım; 5.4, 3.2, 2.2, 99.2. Böyle bir veri setinde en sondaki özellik diğerlerinden çok daha büyük olduğu için ağın yanlış eğitilmesine sebep olabilir. Ağ son özelliğin çok daha önemli olduğuna karar verebilir. Bu gibi durumlar ağın istenmeyen çıktıların oluşmasına neden olur. Bu nedenle normalizasyon işlemi yapılarak bütün girdiler belirli bir aralığa indirgenir.
5. Veri setinin %80'i eğitim ve %20'si test olarak ayrılır.
6. Bu bölümden sonra yapay sinir ağı oluşturulmaya başlanır. Sıralı model kurulur. Yani giriş katmanı – gizli katmanlar – çıktı katmanı bölümlerinden oluşan yapı kurulur.
7. Yapay sinir ağının parametreleri "Yapay Sinir Ağı Parametreleri" bölümünde belirtilen değerlere göre güncellenir.
8. Oluşturulan yapay sinir ağı modeli 100 iterasyon boyunca eğitim verileri ile eğitilir.
9. Eğitim sonunda doğruluk ve hata değerleri hesaplanır.
10. Oluşturulan model test verileri için denenir. Doğruluk ve hata değerleri hesaplanır.
11. İstenirse elle verilen girdilerin hangi sınıfa ait olduğunu belirlemek üzere tahmin (predict) fonksiyonu kullanılabilir.
12. Son olarak oluşturulan model farklı ortamlarda kullanılabilmek veya tekrar tekrar eğitilmesinin getireceği zaman kaybının önüne geçebilmek için kaydedilebilir. Kaydedilen model daha sonra istenilen şekilde çağrılabilir.

Sonuçlar:

Tablo 2. bulunan değerler modelin eğitimi ve test edilmesi sırasında elde edilen doğruluk değerlerini göstermektedir.

Tablo 2. Farklı "Unit Size" değerleri için doğruluk değerleri

	Hidden Layer Unit Size		
	6	**16**	**32**
Eğitim Doğruluk Oranı	0.78	0.89	0.98
Test Doğruluk Oranı	0.69	0.93	0.96



Şekil 5. Eğitim doğruluk oranlarının yakınsama grafiği

3.2 Yapay Sinir Ağları Kullanılarak Wisconsin Kanser Verisinin Sınıflandırılması Uygulaması

Bu uygulamada yapay sinir ağları kullanılarak verilen özelliklere göre hücrenin hangi sınıfa ait olduğu bulunacaktır. Yapılan çalışmanın yazılan uygulama kodu ekler bölümünde bulunmaktadır.(Ek- A)

Veri Seti Özellikleri:

Veri setinde hücreye ait 10 özellik ve hücrenin hangi sınıfa ait olduğu bilgisi verilmiştir. Veri seti 699 örnekten oluşmaktadır.

1. Örnek id numarası (id number)
2. Clump Kalınlığı (Clump Thickness : 1 - 10)
3. Hücre Boyutunun Eşbiçimliliği (Uniformity of Cell Size : 1 - 10)
4. Hücre Şekil Düzensizliği (Uniformity of Cell Shape)
5. Marjinal Bağlanma (Marginal Adhesion: 1 - 10)
6. Tek Epitelyal Hücre Boyutu (Single Epithelial Cell Size: 1 - 10)
7. Çıplak Çekirdekler (Bare Nuclei: 1 - 10)
8. İyi Huylu Kromatin (Bland Chromatin: 1 - 10)
9. Normal Nükleoliler (Normal Nucleoli: 1 - 10)
10. Mitozlar (Mitoses: 1 - 10)
11. Sınıf (Çıktı):
    a. İyi Huylu (benign)
    b. Kötü Huylu (malignant)

Yapay Sinir Ağı Parametreleri:

1. Girdi Katmanı (Input Layer) :
    a. Düğüm Sayısı (Input Units): 9
2. Gizli Katman 1 (Hidden Layer 1) :
    a. Düğüm Sayısı (Hidden Units): 20
    b. Aktivasyon Fonksiyonu: Rectifier
    c. Dropout: 0.1
3. Gizli Katman 2 (Hidden Layer 2):
    a. Düğüm Sayısı (Hidden Units): 20
    b. Aktivasyon Fonksiyonu: Rectifier
    c. Dropout: 0.1
4. Çıktı Katmanı (Output Layer):
    a. Düğüm Sayısı (Output Units): 5
    b. Aktivasyon Fonksiyonu: Sigmoid
    c. Hata Optimizasyonu (Optimizasyon Fonksiyonu): Adam
    d. Hata Hesaplama (Loss Fonksiyonu): Categorical Crossentropy
    e. Değerlendirme Kriteri (Metric): Accuracy
5. Diğer:
    a. İterasyon Sayısı(Number epoch): 100
    b. Eğitim verileri yüzdesi: %80
    c. Test verileri yüzdesi: %20

Algoritma Akışı:

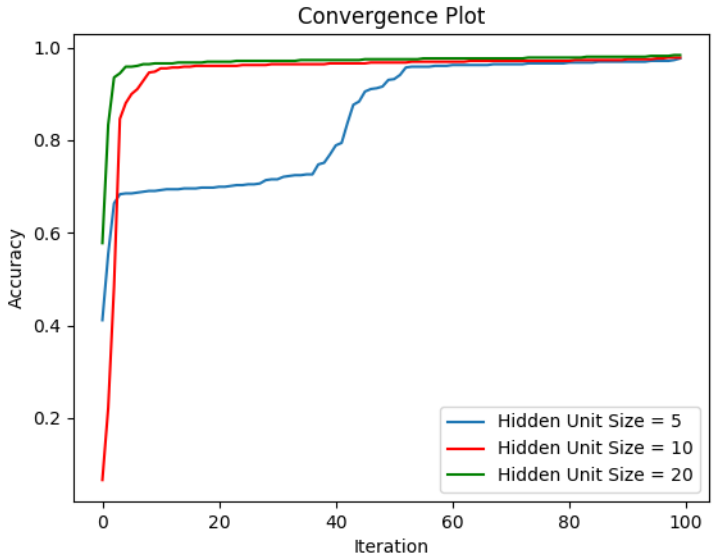
1. “Cancer.data” dosyasından veri seti okunur.
2. Bu veri seti eksik veriler içermektedir. Bu yüzden ilk olarak eksik veriler o hücrenin bulunduğu sütunun ortalaması ile doldurulur. Eksik veriler ait olduğu özellik sınıfına ait verilerin ortalaması, standart sapması veya medyanı gibi bilgilerler doldurulabilir.
3. Veri seti girdi ve çıktı olarak ayrıştırılıp değişkenlere atanır.
4. Veri seti çıktı olarak 2 sınıftan oluştuğundan çıktı kategorik veri olarak adlandırılır. Sınıflar “string” tipinde olduğundan sinir ağının işleyebileceği “numerik” tipine dönüştürülmesi gerekir. Bu adımda sınıflar sayısal verilere dönüştürülür.
5. Veri seti üzerinde “z-score normalizasyon” işlemi gerçekleştirilir.
6. Veri setinin %80’i eğitim ve %20’si test olarak ayrılır.
7. Bu bölümden sonra yapay sinir ağı oluşturulmaya başlanır. Sıralı model kurulur. Yani giriş katmanı – gizli katmanlar – çıktı katmanı bölümlerinden oluşan yapı kurulur.
8. Yapay sinir ağının parametreleri “Yapay Sinir Ağı Parametreleri” bölümünde belirtilen değerlere göre güncellenir.
9. Oluşturulan yapay sinir ağı modeli 100 iterasyon boyunca eğitim verileri ile eğitilir.
10. Eğitim sonunda doğruluk ve hata değerleri hesaplanır.
11. Oluşturulan model test verileri için denenir. Doğruluk ve hata değerleri hesaplanır.
12. İstenirse elle verilen girdilerin hangi sınıfa ait olduğunu belirlemek üzere tahmin (predict) fonksiyonu kullanılabilir.
13. Bu uygulamada diğer uygulamalardan farklı olarak cross validation yöntemi uygulanmıştır.Bu yöntem kullanılarak oluşturulan modelin kesin olarak doğruluk değeri belirlenmiş olur. Cross validation yöntemi uygulanmadığı takdirde algoritma akışınauygun bir şekilde sürekli aynı eğitim ve test verilerini seçip ağı ona göre eğitir. Seçilen bu veriler eğitilmesi kolay veriler olabilir. Bu durum bizi yanıltabilir. Çıkan sonuçlar modelin çok başarılı olduğuna işaret edebilir. Fakat veri seti tekrar karıştırılıp eğitildiğinde karşımıza çıkan doğruluk değeri çok daha düşük olabilir. Bu durumların önüne geçebilmek ve modelimizin gerçek doğruluk değerini hesaplamak için cross validation yöntemi kullanılır. Bu yöntem veri setinin kaça parçalanacağı parametresini kullanıcıdan ister. Örneğin parametrenin 10 verildiğini kabul edelim. Veri seti 10 parçaya ayrılır. İlk iterasyonda ayrılan parçalardan ilki test için geriye kalan 9 parça eğitim için kullanılır ve doğruluk,hata değerleri hesaplanır. Daha sonra ikinci adıma geçilir. İkinci adımda bu sefer 2. Parça test , geriye kalan 9 parça eğitim için seçilir. Yine doğruluk ve hata değerleri hesaplanır. Bu işlemler 10 adım boyunca devam eder. Her adımda elde edilen doğruluk değerlerinin ortalaması bize modelin gerçek doğruluk oranını verir. Bu yöntem ile veri setinin her bir parçası test için kullanılmış olur.

Sonuçlar:

Tablo 3. bulunan değerler modelin eğitimi ve test edilmesi sırasında elde edilen doğruluk değerlerini göstermektedir.

Tablo 3. Farklı "Unit Size" değerleri için doğruluk değerleri

	Hidden Layer Unit Size		
	5	**10**	**20**
Eğitim Doğruluk Oranı	0.9767	0.9788	0.9838
Test Doğruluk Oranı	0.9715	0.9785	0.9821

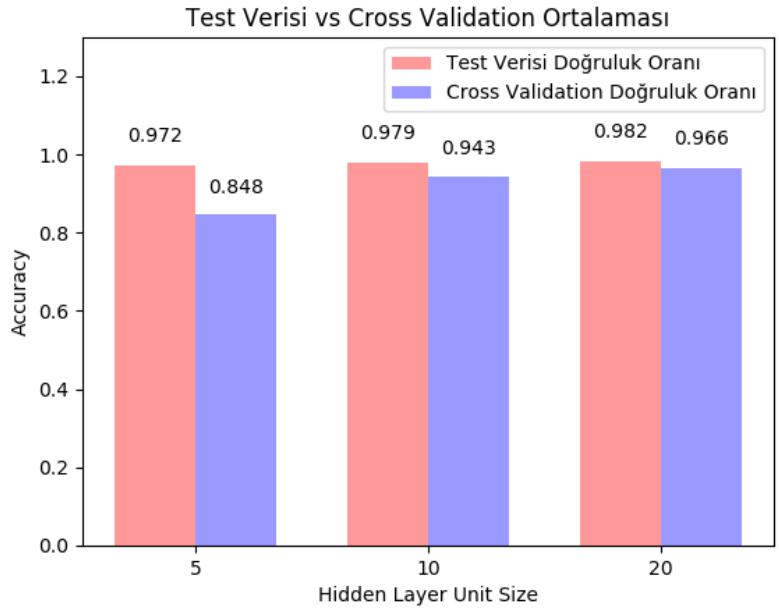


Şekil 6. Eğitim doğruluk oranlarının yakınsama grafiği

Tablo 4. bulunan değerler modelin "Cross Validation" yöntemi ile test edilmesi sonucu bulunan değerlerdir.

Tablo 4. Farklı "Unit Size" değerleri için Cross Validation doğruluk değerleri

	Hidden Layer Unit Size		
	5	**10**	**20**
Doğruluk Oranı	0.8479	0.9428	0.9661



3.3 Yapay Sinir Ağları Kullanılarak E-postaların Sınıflandırılması Uygulaması

Bu uygulamada yapay sinir ağları kullanılarak verilen özelliklere göre bir e-postanın “spam” yada “spam değil” olarak sınıflandırılması yapılmıştır. Yapılan çalışmanın yazılan uygulama kodu ekler bölümünde bulunmaktadır.(Ek- A)

Veri Seti Özellikleri:

Veri setinde e-postaya ait 57 özellik ve e-postanın hangi sınıfa ait olduğu bilgisi verilmiştir. Veri seti 4601 örnekten oluşmaktadır.

1. 48 adet kelimenin e-posta içerisinde geçme sıklıkları verilmiştir. Kelimelere ait bilgiler “ekler” bölümünde açıklanmıştır.
2. “ ; ”, “ (“ ,” [”, “ ! ”, “ $ ”, “ # ” işaretlerinin e-posta içerisinde geçme sıklıkları verilmiştir.
3. Büyük harfle yazılan kelimelerin e-postada geçen bütün kelimelere oranı verilmiştir.
4. Büyük harfle yazılan en uzun kelimenin uzunluğu verilmiştir.
5. Büyük harfle yazılan kelimelerin uzunlukları toplamı verilmiştir.
6. Sınıf (Çıktı):
    a. Zararlı (Spam) : 1
    b. Zararlı Değil (Non - Spam): 0

Yapay Sinir Ağı Parametreleri:

1. Girdi Katmanı (Input Layer) :
    a. Düğüm Sayısı (Input Units): 57
2. Gizli Katman 1 (Hidden Layer 1) :
    a. Düğüm Sayısı (Hidden Units): 128
    b. Aktivasyon Fonksiyonu: Rectifier
3. Gizli Katman 2 (Hidden Layer 2):
    a. Düğüm Sayısı (Hidden Units): 128
    b. Aktivasyon Fonksiyonu: Rectifier
4. Çıktı Katmanı (Output Layer):
    a. Düğüm Sayısı (Output Units): 1
    b. Aktivasyon Fonksiyonu: Sigmoid
    c. Hata Optimizasyonu (Optimizasyon Fonksiyonu): Adam
    d. Hata Hesaplama (Loss Fonksiyonu): Binary Crossentropy
    e. Değerlendirme Kriteri (Metric): Accuracy
5. Diğer:
    a. İterasyon Sayısı(Number epoch): 100
    b. Eğitim verileri yüzdesi: %80
    c. Test verileri yüzdesi: %20

Algoritma Akışı:

1. “Spambase.data” dosyasından veri seti okunur.
2. Veri seti girdi ve çıktı olarak ayrıştırılıp değişkenlere atanır.
3. Veri seti üzerinde “z-score normalizasyon” işlemi gerçekleştirilir.
4. Veri setinin %80’i eğitim ve %20’si test olarak ayrılır.

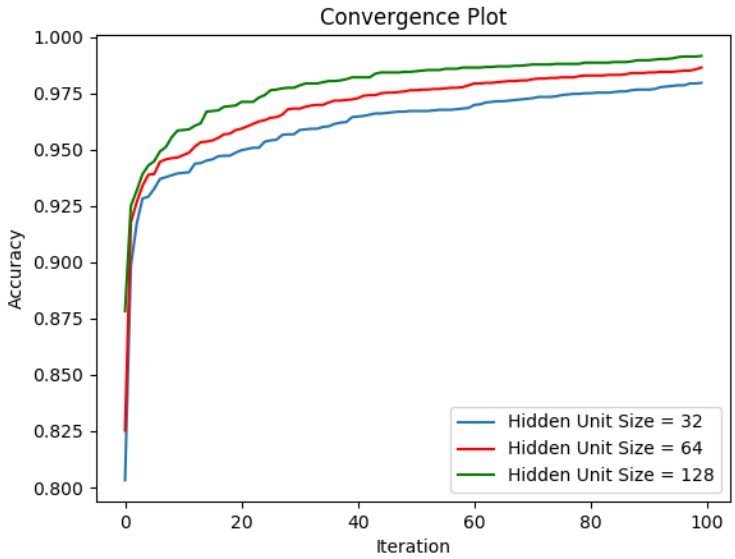
5. Bu bölümden sonra yapay sinir ağı oluşturulmaya başlanır. Sıralı model kurulur. Yani giriş katmanı – gizli katmanlar – çıktı katmanı bölümlerinden oluşan yapı kurulur.
6. Yapay sinir ağının parametreleri "Yapay Sinir Ağı Parametreleri" bölümünde belirtilen değerlere göre güncellenir.
7. Oluşturulan yapay sinir ağı modeli 100 iterasyon boyunca eğitim verileri ile eğitilir.
8. Eğitim sonunda doğruluk ve hata değerleri hesaplanır.
9. Oluşturulan model test verileri için denenir. Doğruluk ve hata değerleri hesaplanır.
10. İstenirse elle verilen girdilerin hangi sınıfa ait olduğunu belirlemek üzere tahmin (predict) fonksiyonu kullanılabilir.
11. Son olarak "Cross- Validation" uygulanarak elde edilen doğruluk değerleri karşılaştırılır.

Sonuçlar:

Tablo 5. bulunan değerler modelin eğitimi ve test edilmesi sırasında elde edilen doğruluk değerlerini göstermektedir.

Tablo 5. Farklı "Unit Size" değerleri için doğruluk değerleri

	Hidden Layer Unit Size		
	32	**64**	**128**
Eğitim Doğruluk Oranı	0.9796	0.9864	0.9915
Test Doğruluk Oranı	0.9402	0.9489	0.9499

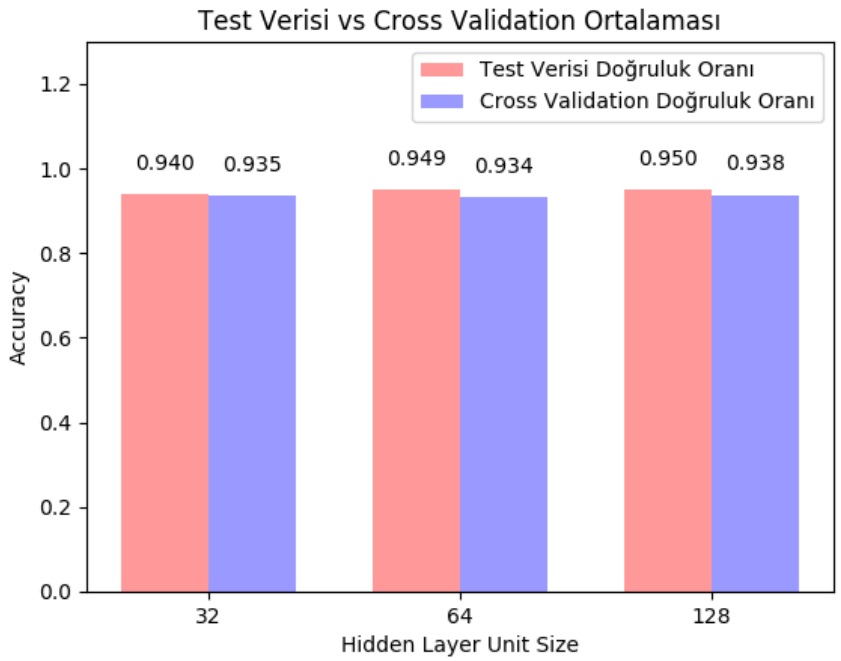


Şekil 7. Eğitim doğruluk oranlarının yakınsama grafiği

Tablo 6. bulunan değerler modelin “Cross Validation” yöntemi ile test edilmesi sonucu bulunan değerlerdir.

Tablo 6. Farklı “Unit Size” değerleri için Cross Validation doğruluk değerleri

	Hidden Layer Unit Size		
	32	**64**	**128**
Doğruluk Oranı	0.9347	0.9336	0.9377



Şekil 8. Test ve Cross Validatin doğruluk oranlarının karşılaştırılması

3.4 Yapay Sinir Ağları Kullanılarak Banka Müşterilerinin Sınıflandırılması Uygulaması

Bu uygulamada yapay sinir ağları kullanılarak verilen özelliklere göre banka müşterisinin bankadan ayrılıp ayrılmayacağına dair sınıf bulunacaktır.Uygulama kodu ekler bölümünde bulunmaktadır.(Ek- A)

Veri Seti Özellikleri:

Veri setinde müşteriye ait 13 özellik ve müşterinin hangi sınıfa ait olduğu bilgisi verilmiştir. Veri seti 10.000 örnekten oluşmaktadır.

a. Örnek id numarası (id number)
b. Müşteri Numarası (Customer Id)
c. Müşteri Soyadı (Surname)
d. Kredi Skoru (Credit Score)
e. Coğrafya (Geography)
f. Cinsiyet (Gender)
g. Yaş (Age)
h. Görev Süresi (Tenure)
i. Bakiye (Balance)
j. Ürün veya Hesap Sayısı (NumOfProducts)
k. Kredi Kartı Var mı? (HasCrCard)
l. Aktif Müşteri mi? (IsActiveMember)
m. Tahmini Maaş (EstimatedSalary)
n. Sınıf (Çıktı):
    c. Bankadan Ayrılır (Exit): 1
    d. Bankadan Ayrılmaz (Not Exit): 0

Yapay Sinir Ağı Parametreleri:

a. Girdi Katmanı (Input Layer) :
    a. Düğüm Sayısı (Input Units): 10
b. Gizli Katman 1 (Hidden Layer 1) :
    a. Düğüm Sayısı (Hidden Units): 32
    b. Aktivasyon Fonksiyonu: Rectifier
c. Gizli Katman 2 (Hidden Layer 2):
    a. Düğüm Sayısı (Hidden Units): 32
    b. Aktivasyon Fonksiyonu: Rectifier
d. Çıktı Katmanı (Output Layer):
    a. Düğüm Sayısı (Output Units): 1
    b. Aktivasyon Fonksiyonu: Sigmoid
    c. Hata Optimizasyonu (Optimizasyon Fonksiyonu): Adam
    d. Hata Hesaplama (Loss Fonksiyonu): Binary Crossentropy
    e. Değerlendirme Kriteri (Metric): Accuracy
e. Diğer:
    a. İterasyon Sayısı(Number epoch): 100
    d. Eğitim verileri yüzdesi: %80
    e. Test verileri yüzdesi: %20

Algoritma Akışı:

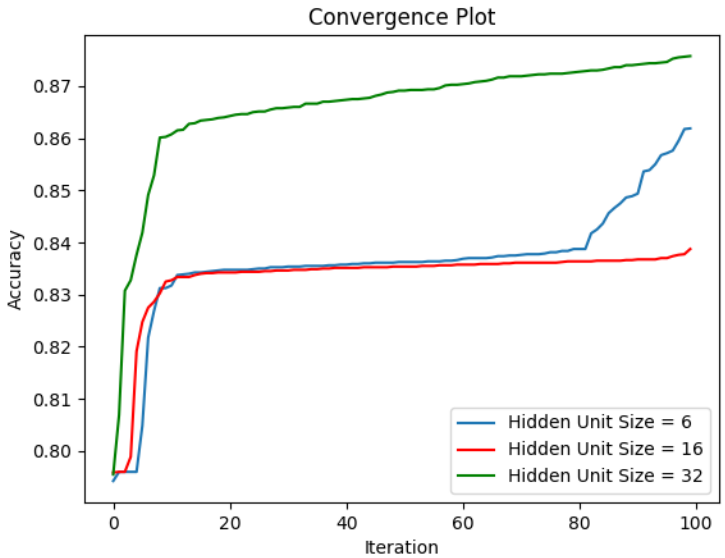
1. "Churn_Modelling.csv" dosyasından veri seti okunur.
2. Veri seti girdi ve çıktı olarak ayrıştırılıp değişkenlere atanır.
3. Veri seti özelliklerinden bazıları kategorik verilerdir. Coğrafya ve Cinsiyet kolonları kategorik, metinsel veri içermektedir. Bu yüzden bu iki özellik sayısal verilere dönüştürülmüştür. Örneğin: cinsiyet kolonu Erkek:1 , Kadın: 0 gibi. Aynı şekilde coğrafya kolonu da kategorik veriden sayısal veriye dönüştürülmüştür.
4. Veri seti üzerinde "z-score normalizasyon" işlemi gerçekleştirilir.
5. Veri setinin %80'i eğitim ve %20'si test olarak ayrılır.
6. Bu bölümden sonra yapay sinir ağı oluşturulmaya başlanır. Sıralı model kurulur. Yani giriş katmanı – gizli katmanlar – çıktı katmanı bölümlerinden oluşan yapı kurulur.
7. Yapay sinir ağının parametreleri "Yapay Sinir Ağı Parametreleri" bölümünde belirtilen değerlere göre güncellenir.
8. Oluşturulan yapay sinir ağı modeli 100 iterasyon boyunca eğitim verileri ile eğitilir.
9. Eğitim sonunda doğruluk ve hata değerleri hesaplanır.
10. Oluşturulan model test verileri için denenir. Doğruluk ve hata değerleri hesaplanır.
11. İstenirse elle verilen girdilerin hangi sınıfa ait olduğunu belirlemek üzere tahmin (predict) fonksiyonu kullanılabilir.
12. Son olarak "Cross- Validation" uygulanarak elde edilen doğruluk değerleri karşılaştırılır.

Sonuçlar:

Tablo 7. bulunan değerler modelin eğitimi ve test edilmesi sırasında elde edilen doğruluk değerlerini göstermektedir.

Tablo 7. Farklı "Unit Size" değerleri için doğruluk değerleri

	Hidden Layer Unit Size		
	6	**16**	**32**
Eğitim Doğruluk Oranı	0.8618	0.8387	0.8757
Test Doğruluk Oranı	0.8615	0.8449	0.8505

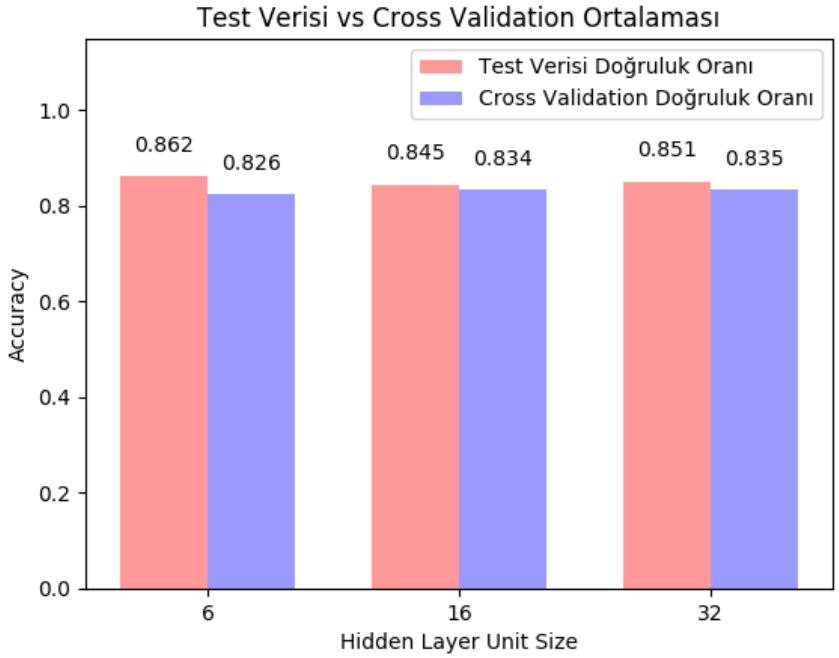


Şekil 9. Eğitim doğruluk oranlarının yakınsama grafiği

Tablo 8. bulunan değerler modelin “Cross Validation” yöntemi ile test edilmesi sonuca bulunan değerlerdir.

Tablo 8. Farklı “Unit Size” değerleri için Cross Validation doğruluk değerleri

	Hidden Layer Unit Size		
	6	**16**	**32**
Doğruluk Oranı	0.8262	0.8344	0.8353



Şekil 10. Test ve Cross Validatin doğruluk oranlarının karşılaştırılması

3.5 Yapay Sinir Ağları ile Türkçe Yazar Tanıma Uygulaması

Bu uygulamada yapay sinir ağları kullanılarak verilen özelliklere göre metnin hangi yazara ait olduğu bulunacaktır. Yapılan çalışmanın yazılan uygulama kodu ekler bölümünde bulunmaktadır.(Ek- C)

Veri Seti Özellikleri:

Veri setinde metine ait 20 özellik ve metnin hangi yazara ait olduğu bilgisi verilmiştir. Veri seti 247 örnekten oluşmaktadır.

a. Cümle sayısı
b. Kelime sayısı
c. Farklı kelime sayısı
d. Ünlem
e. Nokta Sayısı
f. Virgül sayısı
g. Soru İşareti
h. İki Nokta
i. İsim Sayısı
j. Fiil sayısı
k. Sıfat Sayısı
l. Zamir Sayısı
m. Zarf Sayısı
n. Bağlaç Sayısı
o. Edat Sayısı
p. ö. Zaman Belirten Kelime Sayısı
q. Sayı İçeren Kelime Sayısı
r. Özel Kelime sayısı
s. Kısaltma Sayısı
t. Soru Sayısı
u. Sınıf (Çıktı):
    a. 0 - Ali Ece
    b. 1 - Gila BenMayor
    c. 2 - Gülse Birsel
    d. 3 - İlber Ortaylı
    e. 4 - Mehmet Barlas
    f. 5 - Mehmet Yaşin

Yapay Sinir Ağı Parametreleri:

a. Girdi Katmanı (Input Layer) :
    a. Düğüm Sayısı (Input Units): 20
b. Gizli Katman 1 (Hidden Layer 1) :
    a. Düğüm Sayısı (Hidden Units): 512
    b. Aktivasyon Fonksiyonu: Rectifier

c. Gizli Katman 2 (Hidden Layer 2):
    a. Düğüm Sayısı (Hidden Units): 512
    b. Aktivasyon Fonksiyonu: Rectifier
d. Çıktı Katmanı (Output Layer):
    a. Düğüm Sayısı (Output Units): 6
    b. Aktivasyon Fonksiyonu: Sigmoid
    c. Hata Optimizasyonu (Optimizasyon Fonksiyonu): Adam
    d. Hata Hesaplama (Loss Fonksiyonu): Categorical Crossentropy
    e. Değerlendirme Kriteri (Metric): Accuracy
e. Diğer:
    a. İterasyon Sayısı(Number epoch): 300
    b. Eğitim verileri yüzdesi: %80
    c. Test verileri yüzdesi: %20

Ön Hazırlık:

Bu uygulamada en büyük iş veri toplamak ve toplanan metinsel verinin Türkçe analizini yapmaktır. Veri toplama işlemi için analiz etmek istediğim yazarlara ait köşe yazılarını yazmış olduğum "bot.py" scripti ile topladım. Bu script verilen linklerdeki HTML sayfalarını dolaşıp ilgili makalenin geçtiği metin kısmını toplar. Toplanan veri "makaleNo_makaleYazari.txt" formatında dosyaya otomatik olarak kaydedilir. Bu işlemleri gerçekleştirmek için Python'da bulunan çok faydalı bir kütüphane olan "BeautifulSoup" kütüphanesinden faydalandım.

Makale verisi toplandıktan sonra ikinci adımda metinsel verinin sayısal verilere dönüştürülmesi gerekiyordu. Yabancı dillerde yapılan çalışmalarda da genel olarak cümle sayısı, kelime sayısı, sıfat, fill sayısı vb. gibi özellikler yapay sinir ağına girdi olarak verilmekteydi. Kelimelerin morfolojik analizini gerçekleştiren yabancı dilde birçok kütüphane olmasına rağmen Türkçe bu alanda kısıtlı kütüphanelere sahip. Bu alanda Ahmet A. Akın'ın Java dilinde yazmış olduğu "Zemberek" kütüphanesi işimi çok kolaylaştırdı. Zemberek kütüphanesini kullanarak köşe yazılarının morfolojik analizlerini gerçekleştirdim. Gelen metin içerisinde kaç tane sıfat, isim, zarf, zamir vb. olduğunu "Zemberek" kütüphanesinin sunmuş olduğu yapı ile gerçekleştirdim. "Zemberek" Java ile yazılan bir kütüphane olduğundan bunu Python'da çalıştırmam gerekti. Bunun için araştırma yaparken "Jpype" kütüphanesini buldum. Bu kütüphane verilen jar dosyası içerisindeki bütün Java kodlarını çok rahat bir şekilde çalıştırdı. Bu adımı da tamamladıktan sonra yapay sinir ağına vereceğim veri seti hazır hale gelmiş oldu. Bu raporun yazıldığı sırada program içerisinde her bir yazara ait 40 adet makale bulunmaktadır. Toplam 6 yazar vardır. Yazarlara ait makale sayısının artması uygulamanın doğruluk değerini pozitif şekilde arttıracaktır.

Şekil 11. Metin Analizi Çalışma Yapısı

Algoritma Akışı:

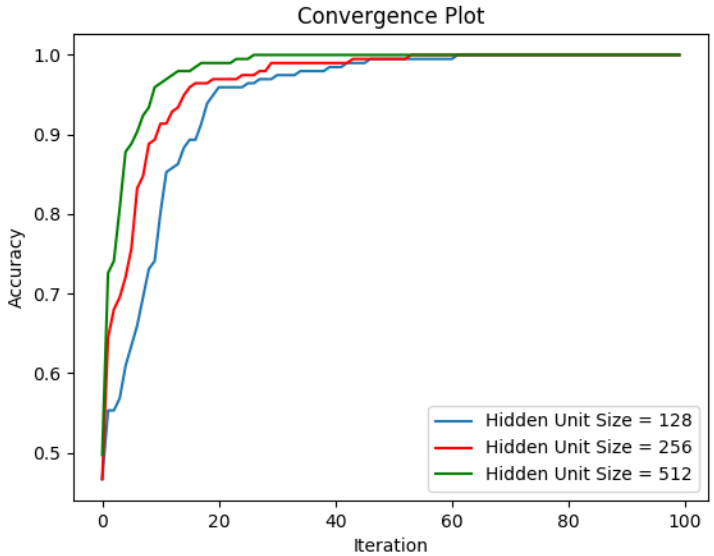
1. "Data.csv" dosyasından veri seti okunur.
2. Veri seti girdi ve çıktı olarak ayrıştırılıp değişkenlere atanır.
3. Veri seti sınıfları yani yazar isimleri kategorik verilerdir. Kategorik veriler numerik verilere dönüştürüldü. Örneğin: Ali Ece : 1 0 0 0 0 0, Gila BenMayor: 0 1 0 0 0 0 vb. şeklinde.
4. Veri seti üzerinde "z-score normalizasyon" işlemi gerçekleştirilir.
5. Veri setinin %80'i eğitim ve %20'si test olarak ayrılır.
6. Bu bölümden sonra yapay sinir ağı oluşturulmaya başlanır. Sıralı model kurulur. Yani giriş katmanı – gizli katmanlar – çıktı katmanı bölümlerinden oluşan yapı kurulur.
7. Yapay sinir ağının parametreleri "Yapay Sinir Ağı Parametreleri" bölümünde belirtilen değerlere göre güncellenir.
8. Oluşturulan yapay sinir ağı modeli 300 iterasyon boyunca eğitim verileri ile eğitilir.
9. Eğitim sonunda doğruluk ve hata değerleri hesaplanır.
10. Oluşturulan model test verileri için denenir. Doğruluk ve hata değerleri hesaplanır.
11. Aynı zamanda oluşturulan modelin gerçek doğruluk değerini bulmak için cross validation uygulanır.
12. İstenirse elle verilen girdilerin hangi sınıfa ait olduğunu belirlemek üzere tahmin (predict) fonksiyonu kullanılabilir.
13. Son olarak "Cross- Validation" uygulanarak elde edilen doğruluk değerleri karşılaştırılır.

Sonuçlar:

Tablo 9. bulunan değerler modelin eğitimi ve test edilmesi sırasında elde edilen doğruluk değerlerini göstermektedir.

Tablo 9. Farklı "Unit Size" değerleri için doğruluk değerleri

	Hidden Layer Unit Size		
	128	**256**	**512**
Eğitim Doğruluk Oranı	0.9948	0.9948	0.9949
Test Doğruluk Oranı	0.9200	0.9200	0.8800

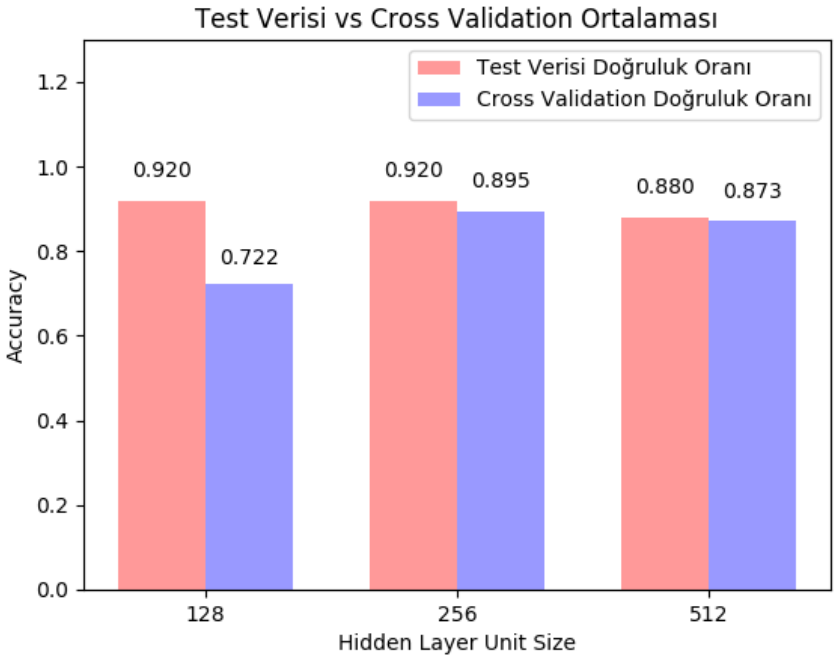


Şekil 12. Eğitim doğruluk oranlarının yakınsama grafiği

Tablo 10. bulunan değerler modelin “Cross Validation” yöntemi ile test edilmesi sonucu bulunan değerlerdir.

Tablo 10. Farklı “Unit Size” değerleri için Cross Valdation doğruluk değerleri

	Hidden Layer Unit Size		
	128	**256**	**512**
Doğruluk Oranı	0.7221	0.8947	0.8728



Şekil 13. Test ve Cross Validatin doğruluk oranlarının karşılaştırılması

Farklı Modellerin Karşılaştırılması:

Model 1: Optimizer: “*Adam*”, Loss Function: “*Categorical Crossentrpy*”, Metrics: “*Accuracy*”

	Unit Size	**Activation Func.**
Gizli Katman 1	[128,256,512]	Rectifier
Gizli Katman 2	[128,256,512]	Rectifier
Çıktı Katmanı	6	Sigmoid

Tablo 11. Model 1 Parametre Değerleri

Model 2: Optimizer: “*Adamax*”, Loss Function: “*kullback_leibler_divergence*”, Metrics: “*Accuracy*”

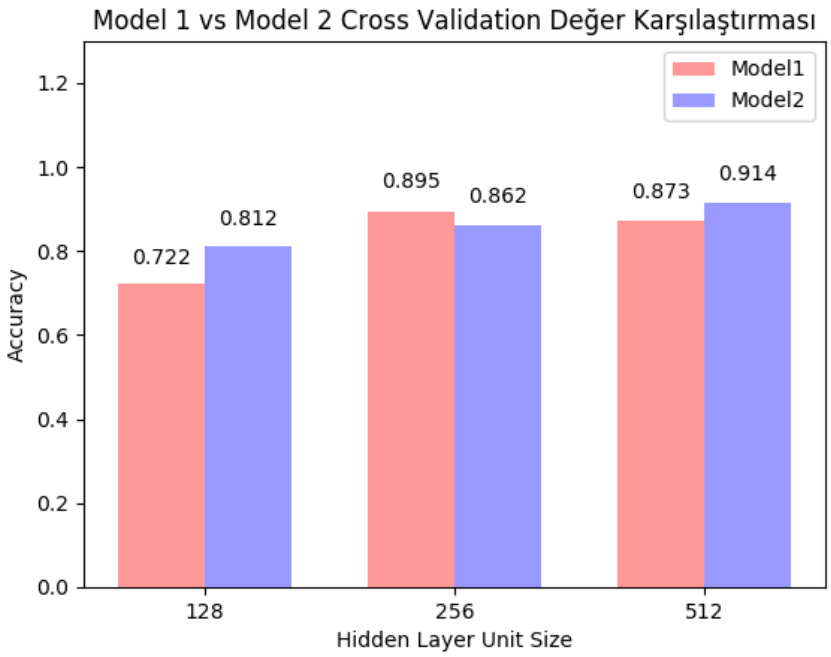
	Unit Size	**Activation Func.**
Gizli Katman 1	[128,256,512]	Rectifier
Gizli Katman 2	[128,256,512]	Rectifier
Çıktı Katmanı	6	Softmax

Tablo 12. Model 2 Parametre Değerleri

Tablo 13. bulunan değerler farklı modellerin “Cross Validation” yöntemi ile test edilmesi sonucu bulunan değerlerdir.

Tablo 13. Farklı modeller için Cross Validation doğruluk değerleri

	Hidden Layer Unit Size		
	128	**256**	**512**
Model 1	0.7221	0.8947	0.8728
Model 2	0.8118	0.8623	0.9136



Şekil 14. Model 1 ve Model 2’nin doğruluk oranlarının karşılaştırılması

3.6 Derin Öğrenme Uygulamalarına Giriş: Türkçe Cümlelerin Anlamsal Analizi Uygulaması

Bu uygulamada Türkçe cümlelerin anlamsal analizi yapılacaktır. Verilen cümlenin olumlu yada olumsuz bir cümle olduğu sınıflandırılacaktır. Yapılan çalışmanın yazılan uygulama kodu ekler bölümünde bulunmaktadır.(Ek- B)

Veri Seti Özellikleri:

Veri setinde kullanıcı yorumları ve kullanıcı yorumlarının hangi sınıfa ait olduğu bilgisi tutulmaktadır. Veri seti 3649 örnekten oluşmaktadır.

a. Örnek id numarası (id number)
b. Kullanıcı Yorumu (Review)
c. Sınıf (Çıktı):
    a. Olumlu: 1
    b. Olumsuz: 0

Model Parametreleri:

a. Girdi Katmanı (Embedding Layer) :
    a. Düğüm Sayısı (Embed Dimension): 128
    b. Dikkate alınacak Kelime Sayısı (Num Words): 25000
    c. Dropout Rate: 0.2
b. İkinci Katman: Long Short Term Memory (LSTM):
    a. Düğüm Sayısı (Lstm Out): 128
    b. Dropout Unit: 0.2
    c. Dropout Weight: 0.2
c. Çıktı Katmanı (Output Layer):
    a. Düğüm Sayısı (Output Units): 2
    b. Aktivasyon Fonksiyonu: Softmax
    c. Hata Optimizasyonu (Optimizasyon Fonksiyonu): Adam
    d. Hata Hesaplama (Loss Fonksiyonu): Binary Crossentropy
    e. Değerlendirme Kriteri (Metric): Accuracy
d. Diğer:
    a. İterasyon Sayısı(Number epoch): 3
    b. Eğitim verileri yüzdesi: %80
    c. Test verileri yüzdesi: %20

Parametre ve Model Açıklaması:

Anlam analizi uygulamalarında normal yapay sinir ağı modelinden farklı bir yapı kullanılır. Bu uygulamalarda girdi olarak tamamen metinsel veriler kullanılır. Fakat metinsel veriler modelin işleyebileceği formatta değildir. Bu verileri ağın anlayabileceği sayısal formata dönüştürülmesi gerekir. Bu noktada “Keras” kütüphanesi bize hazır fonksiyonlar sunmaktadır. “Keras” içerisinde bulunan “Tokenizer” sınıfı bize verilen cümleler içerisinde geçen kelimeleri ayıklar. Bu sınıf kullanılarak bütün kelimeler arasında en sık kullanılan “x” kadar kelimeye odaklanabiliriz. Bu işlem modelin eğitilmesi üzerindeki yükü de azaltır. Bu işlem “Tokenizer” sınıfına verilen “num_words” parametresi ile gerçekleştirilir. “Tokenizer” kelimesi verilen örnekler içerisinde geçen kelimelerin

sıklığını (frekansını) hesaplar. Daha sonra tüm cümleler tam sayı dizisine dönüştürülür. Bu işlemlerin gerçekleştirildiği fonksiyonları bir örnek üzerinde anlatmak daha faydalı olur.

Örnek metinler: [“Yapay zeka gelecek”, “Gelecek teknolojiler yapay zeka”,”Derin öğrenme ve yapay zeka”,”Metin analizi,öğrenme ve yapay zeka”,”Öğrenme ve yapay zeka”,”Bilgisayar ve gelecek”]

Kelime Sıklıklarının Belirlenmesi (fit_on_texts): [“yapay” : 5, “zeka” : 5, “gelecek”: 3, “teknolojiler” : 1, “derin”: 1, “öğrenme”: 3, “ve”: 4, “metin”: 1, “analizi”: 1, “bilgisayar”: 1] şeklinde verilen metinler taranarak, metinlerde geçen her bir kelimenin sıklığı(frekansı) hesaplanır. Hesaplanan bu kelimeler frekans değerlerine göre sıraya dizilir. En sık kullanılan kelime 1. Sıraya konulur. Daha sonra diğer kelimelerde frekans değerlerine göre sıraya dizilir. Örnek için sıralama şu şekilde olur:

>>> “*yapay:1, zeka: 2, ve:3, gelecek: 4, öğrenme: 5, teknolojiler: 6, derin: 7, metin: 8, analizi: 9, bilgisayar: 10”*

Kelimelerin Tam Sayı Dizilerine Dönüştürülmesi (texts_to_sequences): Kelimelerin sıralaması yapıldıktan sonra tam sayı dizisi haline dönüştürme işlemi gerçekleştirilir. Her bir metin soldan sağa doğru taranarak her bir kelimeye karşılık gelen sıra numarası ile tam sayı dizisi doldurulacaktır. Burda küçük bir nokta daha önemlidir. İstersek metin içerisinde sadece en çok geçen “x” sayıda kelimeye odaklanabiliriz. Örneğin devam eden örneğimiz için metinler içerisinde en sık geçen 5 (num_words) kelimeyi almak istiyoruz diğerleri önemsiz olsun.Sadece en sık kullanılan 5 kelimeyi bir önceki fonksiyonda yapıtğımız sıra değerleri ile dolduracağız. Devam eden örnek için çıktı:

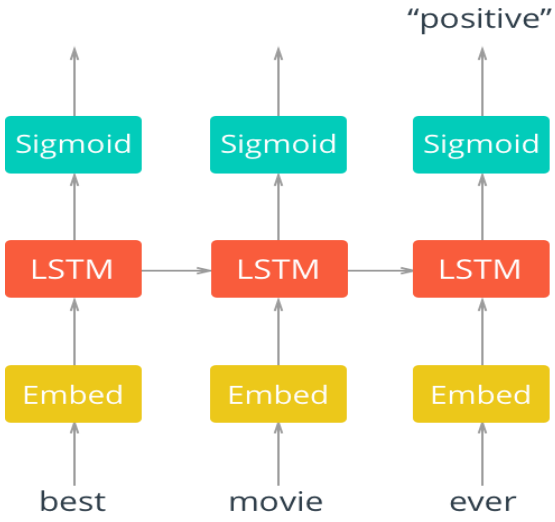
>>> [[1, 2, 4], [4, 1, 2], [3, 1, 2], [3, 1, 2], [3, 1, 2], [3, 4]]

Tam Sayı Dizisinin Son Haline Getirilmesi (pad_sequences): Kelimeler tam sayı dizisine dönüştürüldü fakat sadece en sık kullanılan 5 kelime numaralandırıldı. Bu yüzden oluşturulan tam sayı dizilerinin boyutları birbirinden çok farklı. Farklı olan bu dizi boyutlarını belirli bir boyutta sınırlamamız gerekir. Bu kısımda bütün dizileri tek bir boyutta toplamak için bir parametre (*maxlen*) belirtmemiz gerekir. Kullanacağım parametre dizi boyutlarını sınırlayacaktır. Bunu yaparken çok kısa olan cümleleri 0 lar ile dolduracak, aynı zamanda da çok uzun olan cümleleri kısıtlayacaktır. Bu parametre belirlenirken veri setinizde kullanılan cümlelerin uzunluklarını dikkate almanız gerekir. Çok uzun bir boyut seçerseniz çok fazla önemsiz kelimelerle uğraşırken, çok kısa bir boyut seçerseniz önemli kelimelerin sonuca olan etkisini kaçırabilirsiniz. Devam eden örnek için çıktı:

```
>>> array ([[1, 2, 4, 0, 0],
           [4, 0, 1, 2, 0],
           [0, 0, 3, 1, 2],
           [0, 0, 0, 3, 1],
           [0, 3, 1, 2, 0],
           [0, 3, 4, 0, 0]])
```

Embedding: Her bir kelimenin temsil edileceği vektör boyutunu belirler.

Long Short Term Memory (LSTM): Uzun vadeli bağımlılıkları öğrenebilen özel bir RNN (Recurrent Neural Network) türüdür. Çıkış noktası temel olarak "deep neural networkler" eğitilirken geri-yayılım(backpropagation) algoritmasının kullanımıyla ortaya çıkan "hataların katlanarak büyümesi" problemine çözüm üretmektir. Bu problemin temel sebebi aktivasyon fonksiyonun ürettiği değerlerin sürekli [-1, 1] aralığında olması nedeniyle bu değerlerin "backpropagation" algoritmasına verilmesiyle çarpıla çarpıla sıfıra yakınsamasıdır. Bu sorunun önüne geçebilmek ve karmaşık yapılar için daha iyi öğrenen algoritmalar tasarlamak amacıyla geliştirilen LSTM uzun vadeli bağımlıkların, uzun süreli bilgilerin hatırlanması gereken problemlerde iyi sonuçlar vermektedir. RNN hücresine bir de hafıza eşlik eder. Her adımda öğrenilen hücrelerden hangilerinin hafızada tutulacağı hangilerinin atılmasına gerektiğine, hangilerinin güncelleneceğine karar verilir. Nöral Makine Çevirisi(Neural Machine Translation) için Google tarafından başarılı bir şekilde kullanılan bir yapıdır. Bu çalışmada da kelimelerin daha önce kullanılan anlamlarını öğrenmek ve bu anlamlardan yola çıkarak tahmin üretmek amacıyla kullanılmıştır.



Şekil 15. RNN Kullanılarak Gerçekleştirilen Anlam Analizi Yapısı

Algoritma Akışı:

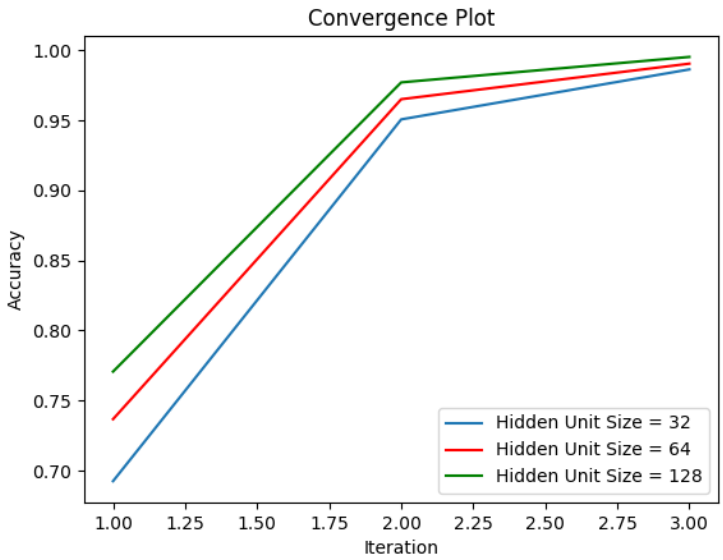
a. "Data.tsv" dosyasından veri seti okunur.
b. Veri seti girdi ve çıktı olarak ayrıştırılıp değişkenlere atanır.
c. Veri seti "*Parametre ve Model Açıklaması*" başlığında anlatıldığı şekilde sayısal veriye dönüştürülür.
d. Veri setinin %80'i eğitim ve %20'si test olarak ayrılır.
e. Bu bölümden sonra model oluşturulmaya başlanır. Embedding – LSTM – Output katmanı bölümlerinden oluşan yapı kurulur.
f. Modelin parametreleri "*Model Parametreleri*" bölümünde belirtilen değerlere göre güncellenir.
g. Oluşturulan yapay sinir ağı modeli 3 iterasyon boyunca eğitim verileri ile eğitilir.
h. Eğitim sonunda doğruluk ve hata değerleri hesaplanır.
i. Oluşturulan model test verileri için denenir. Doğruluk ve hata değerleri hesaplanır.
j. İstenirse elle verilen girdilerin hangi sınıfa ait olduğunu belirlemek üzere tahmin (predict) fonksiyonu kullanılabilir.

Sonuçlar:

Tablo 9. bulunan değerler modelin eğitimi ve test edilmesi sırasında elde edilen doğruluk değerlerini göstermektedir.

Tablo 14. Farklı "Unit Size" değerleri için doğruluk değerleri

	Hidden Layer Unit Size		
	32	**64**	**128**
Eğitim Doğruluk Oranı	0.9863	0.9904	0.9952
Test Doğruluk Oranı	0.8900	0.9000	0.9100



Şekil 16. Eğitim doğruluk oranlarının yakınsama grafiği

3.7 Kelime Vektörleri(Word2Vec) ile Metin Analizi Uygulaması

Bu uygulamada kelime vektörleri kullanılarak verilen metinlerin analizi gerçekleştirildi. Metin içerisinde geçen kelimelerin birbirleri ile olan ilişkileri hesaplanarak tahminlerde bulunuldu ve kelimeler arası uzaklık ve yakınlıklar grafik üzerinde görselleştirildi. Yapılan çalışmanın yazılan uygulama kodu ekler bölümünde bulunmaktadır.(Ek- D)

Veri Seti Özellikleri:

Veri seti olarak "Harry Potter" kitabının bütün serisi kullanıldı. Toplamda 7 adet Harry Potter kitabı analiz edilmek üzere programa verildi.

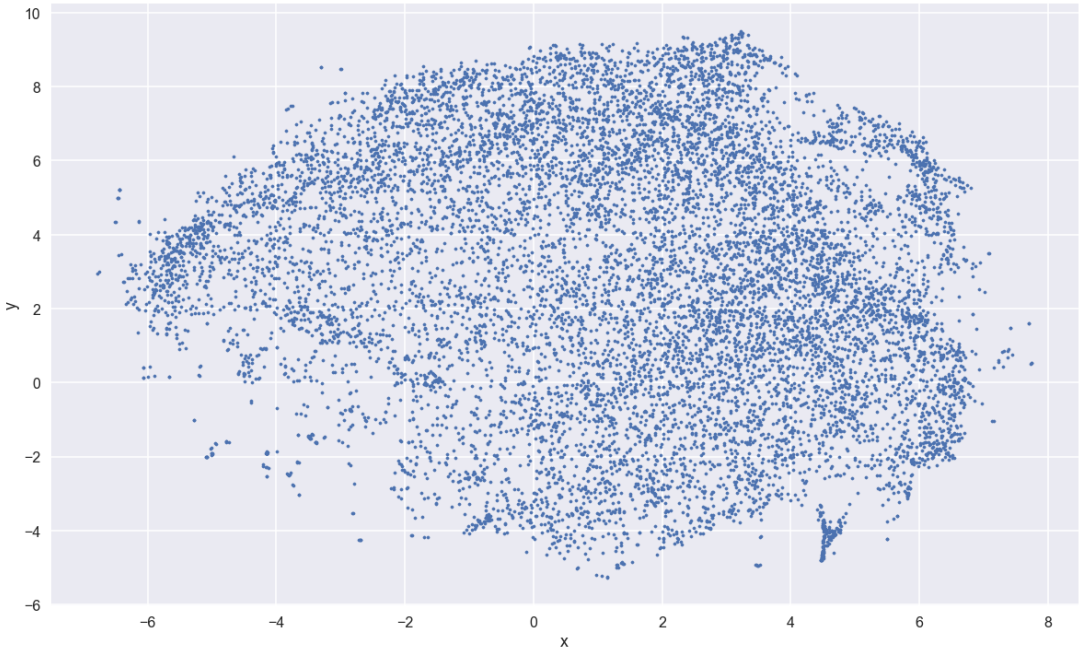
Word2Vec Parametreleri:

1. **Sg**: Eğitim algoritmasını tanımlar. Ön Tanımlı olarak "0" verilir. "0" ise "CBOW" yöntemi, "1" ise "skip-gram" yöntemi kullanılır. Bu çalışmada "skip-gram" kullanılmıştır.
2. **Seed**: Rastgele sayı üretici.
3. **Workers**: Modeli eğitmek için kullanılacak iş parçacıkları sayısı. Çok çekirdekli makineler için.
4. **Size**: Özellik vektörünün boyutudur. Kelimelerin temsil edileceği vektör boyutu
5. **Min_count**: Toplam frekanstan düşük olan bütün kelimeler yok sayılır. Örneğin parametre değeri olarak 5 verilirse metin içerisinde geçen kelimelerden frekansı 5'ten küçük olan bütün kelimeler yok sayılır.
6. **Window**: Cümle içerisinde geçerli ve tahmin edilen kelime arasındaki maksimum uzaklıktır. Yani seçilen kelime ile ilişki olan kelimeler aranırken, seçilen kelimenin sağındaki ve solundaki kelimelerden kaçar tanesinin inceleneceğini belirtir.

Algoritma Akışı:

1. "Data" klasöründen 7 adet "Harry Potter" kitabının olduğu veri seti okunur.
2. Veri seti istenmeyen karakterlerden temizlenir. Sadece harf ve rakamlar tutulur. Özel karakterler veri setinden temizlenir.
3. Veri seti içerisindeki cümleler analiz edilerek, cümlelerde geçen bütün kelimeler ayrıştırılır ve elde edilen kelimelerin frekans değerleri belirlenir.
4. Word2Vec eğitim modeli "Word2Vec Parametreleri" bölümünde belirtilen değerlere göre oluşturulur.
5. Oluşturulan model örnek veri seti ile eğitilir.
6. Model eğitildikten sonra veri setinde geçen bütün kelimelerin grafik üzerinde gösterilmek üzere "x" ve "y" noktaları belirlenir. Yani bütün kelimelerin birbirleri ile olan yakınlık ve uzaklık ilişkileri belirlenmiş olur
7. Kelimeler için belirlenen "x" ve "y" noktaları koordinat düzlemine yerleştirilir.
8. Kelimeler arasındaki ilişkiler grafik üzerinde görselleştirilir.
9. İstenirse elle verilen kelime ile benzerlik gösteren kelimeleri belirlemek üzere benzerlik (similarity) fonksiyonu kullanılabilir.

Sonuçlar:



Şekil 17. Veri setinde bulunan kelimelerin birbirleri ile olan ilişkilerinin genel görünümü

Bu çalışmada kullanılan kelimelerin birbirleri ile olan benzerlikleri, farklılıkları, verilen iki kelimenin birbirine ne kadar benzer oldukları belirlenebilir.

```
potter2vec.most_similar("Quidditch")
```

```
[('match', 0.8261326551437378),
 ('Cup', 0.8209196329116821),
 ('team', 0.8146774768829346),
 ('World', 0.8098978400230408),
 ('Seeker', 0.778986930847168),
 ('Captain', 0.7646517753601074),
 ('game', 0.7579631209373474),
 ('played', 0.7455272674560547),
 ('practice', 0.7369740009307861),
 ('training', 0.7350970506668091)]
```

Şekil 18. "Quidditch" kelimesi ile benzer olan kelimeler ve benzerlik oranları

Şekil 18'de bulunan sonuçlara bakıldığında verilen kelime ile "maç, kupa, takım ve oyun" kelimelerinin yüksek oranda benzer oldukları görülmektedir. Gerçek hikaye ile tutarlı sonuçlar elde edilmiştir.

```
potter2vec.most_similar("Lee")
```

```
[('Jordan', 0.9495911598205566),
 ('twins', 0.8789891004562378),
 ('Angelina', 0.8724427223205566),
 ('Charlie', 0.8442481756210327),
 ('Johnson', 0.8095115423202515),
 ('Bell', 0.808600664138794),
 ('Alicia', 0.8070407509803772),
 ('Demelza', 0.7998273372650146),
 ('Pygmy', 0.7990003824234009),
 ('bracingly', 0.7959675192832947)]
```

Şekil 19. "Lee" kelimesi ile benzer olan kelimeler ve benzerlik oranları

Şekil 19'da bulunan sonuçlara bakıldığında verilen kelime ile "Jordan, twins ve Angelina" kelimelerinin yüksek oranda benzer oldukları görülmektedir. Gerçek hikayeye bakıldığında "Lee" hikayede "twins" olarak adlandırılan "Weasley" ikiz kardeşleri ile "Gryffindors" ekibindedir. Aynı zamanda hikayede "Angelina" ile aynı ekipte olduklarından yakın ilişki içerisindedir.

```
nearest_similarity_cosmul("Severus", "Minerva", "Ron")
nearest_similarity_cosmul("Ron", "Potter", "Hermione")
nearest_similarity_cosmul("Dumbledore", "McGonagall", "Sirius")
```

```
Severus is related to Minerva, as Parvati is related to Ron
Ron is related to Potter, as Ginny is related to Hermione
Dumbledore is related to McGonagall, as Wormtail is related to Sirius
```

Şekil 20. Verilen kelimeler arasındaki pozitif – negatif ilişkisi

Bu çalışmada kullanılan "Gensim" kütüphanesi ile verilen kelimeler arasındaki pozitif – negatif ilişkisine bakılabilir. Şekil 20'de bulunan sonuçlara bakıldığında "King – man + woman = queen" ilişkisinde olduğu gibi bir çıkarım yapılmıştır. Örneğin "Severus ve Minerva karakterleri arasındaki ilişki Ron ve Parvati arasında da vardır veya Dumbledore ve McGonagall arasında bulunan ilişki Sirius Black ve Wormtail arasında bulunmaktadır gibi bir çıkarım yapılabilir.

KAYNAKLAR

1. Öztemel, E. (2003). *Yapay Sinir Ağlari*. PapatyaYayincilik, Istanbul.
2. FIRAT, M., & Güngör, M. (2004). Askı madde konsantrasyonu ve miktarının yapay sinir ağları ile belirlenmesi. Teknik Dergi, 15(73).
3. Hamzaçebi, C., & Kutay, F. (2004). Yapay sinir ağlari ile Türkiye elektrik enerjisi tüketiminin 2010 yilina kadar tahmini. *Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi*, *19*(3).
4. Yapay Sinir Ağları, Fatih Sultan Mehmet Vakıf Üniversitesi, Yapay Zeka Uygulamaları Ders Slaytları, Yrd. Doç. Dr. Ebubekir KOÇ
5. Asilkan, Ö., & IRMAK, A. G. S. (2009). İkinci el otomobillerin gelecekteki fiyatlarının yapay sinir ağları ile tahmin edilmesi. *Süleyman Demirel Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, *14*(2)
6. https://medium.com/@odayibasi/word2vec-nedir-ne-i%C5%9Fe-yarar-a314a37c45aa

Ekler:

Ek – A: https://github.com/zekikus/Yapay-Sinir-Agi-Uygulamalari

Ek – B: https://github.com/zekikus/Turkce-Anlam-Analizi

Ek – C: https://github.com/zekikus/Turkce-Yazar-Tanima

Ek – D: https://github.com/zekikus/Kelime-Vektorleri-ile-Kitap-Analizi

YAPAY SİNİR AĞLARI İLE DERİN ÖĞRENME WORD2VEC VE MAKALE ANALİZİ UYGULAMALARI

Ulaş DEMİRCİ
Mühendislik ve Fen Bilimleri Enstitüsü, Fatih Sultan Mehmet Vakıf Üniversitesi, Beyoğlu, İstanbul,
ulas.demirci@stu.fsm.edu.tr

ÖZET

Son dönemlerde teknolojinin gelişmesi ile internetin yaygınlaşmasıyla birlikte yapay zeka popülaritesi artmıştır. Endüstüri 4.0 ile makinelerin öğrenmesi önem kazanmıştır. Bu makalemde bende yapay sinir ağlarını anlatan diyabet hastalığının bulunması ile ilgil örnek vereceği. Ayrıca Word2vec uygulaması için tensor flow kullanarak kelime vektörleri hakkında görsel bir çıktı veren uygulama anlatacağım. Son olarak gensim kütüphanesi ile makalenin özeti ve anahtar kelimelerinin bulunmasından bahsedeceğim

Anahtar kelimeler: Yapay sinir ağları, kelime vektörleri, sınıflandırma, derin öğrenme, gensim,tenserflow

1. GİRİŞ

Yapay Zekâ nedir konusuna giriş yapmadan "zekâ nedir?" "zekâ sadece insanlarda mı" mı bulunur gibi konulardan bahsetmeliyiz Zekâ Arapça kökenli bir kelime olarak ruh bilimidir anlamına geliyor. TDK sözlüğün insanın düşünme, akıl yürütme, objektif gerçekleri algılama, yargılama ve sonuç çıkarma yeteneklerinin tamamı, anlak, dirayet, zeyreklik, feraset diye geçiyor. Kısaca insanın bir problem karşısında nasıl cevap vermesi gerektiği söyler. Burada doğru ve yanlış cevap vermesi bahsine girmemize gerek yoktur. Doğru ve yanlış kavramı kişiden kişiye değişir. Peki, İnsanlar karşılaştıkları sorunları nasıl cevap vermesini gerektiği zekâsı ile bulurken hayvanlar nasıl yapar? Doğuştan gelen bazı şeyleri kendileri yapar fakat ya sonradan karşılaştıkları olaylar? İşte burada hayvanların da bazı olaylara zaman içinde davranış geliştirdikleri görüyoruz buda bize hayvanlarda da bir zekâ olduğunun emaresidir. Zaman içinde geliştiriyor diyoruz çünkü bir öğrenme var demek ki buradan anlıyoruz ki zekâyı öğrenme yolu ile zamanla geliştirdiğini anlaya biliyoruz. Zekâ ile öğrenme aslında bir biri ile çok iç içe konulardır. Zeki varlıklar zamanla öğrendikleri yeni bilgileri işleyerek yeni yargı ve sonuçlar çıkartırlar bu da onları daha zeki hale getir.

Zekâ, öğrenme gibi kavramların şu ana kadar insan ve hayvanlarda olduğunu bahsettik. Zekâ sadece canlılara ait bir özellik midir? Peki, makinelerin belli problemlere göre tepki verebilir. İşte burada yapay zekâ devreye giriyor. Yapay zekâ canlıların(insanın) gösterdiği zeka emaresini taklit edilmesi şeklindedir Klasik programlamada "0,1" kavramları vardır yani bir şey ya siyah ya beyazdır. Ama işin içine zekâ girince ara renkler griler giriyor beyaz, çok beyaz vs gibi bulanıklaşıyor. Burada YZ giriyor yapay zekâyı ünlü bilim insanlarında bazılar "Bilgisayarları düşündürmeye... Kelimenin tam anlamıyla zihne sahip makineler yapmaya çalışan yeni ve heyecan verici bir çaba". (Haugeland,1985) "İnsanların zekâlarını kullanarak gerçekleştirdiği fonksiyonları gerçekleştiren makineleri yapma sanatı" .(Kurzweil, 1990) Bizde buradan bir çıkarım yaparsak Makinelerin karşılaştıkları bir problem karşısında canlı varlıklar gibi tepki vermesi ve problemden yeni öğrenimler çıkararak bir sonraki karşılaşabilecekleri olaylara aktarmasıdır.

Yaptığımız tanımdan yola çıkarsak Yapay Zekânın temel amacı aşağıdakiler olduğunu düşünebiliriz. [1]

1. Zekânın ne olduğunu anlamak ve nasıl uygulanıcını tespit etmek
2. Makineleri daha zeki yapmak(temel amaç)
3. Makineleri daha kullanışlı hale getirmek

Şimdi ise birazda Yapay Zekânın Kavramın tarihçesini inceleyelim

MS 1.yy: İskenderiyeli Heron adına otomotlar dediği, su ve buharla çalışan mekanik düzenekler yaptı.

1206: Artuklu sarayında yaşayan Ebu'l İz El Cezri, suyla çalışan otomatlar yaptı.
1623: Alman Matematikçi Wihelm Shickard ilk mekanik hesap makinesini yaptı.
1672: Gottfired Leibniz ikili sayı sistemini geliştirdi. Günümüz bilgisayarlarında bu bilgisayarlar kullanılıyor.
1822-1859: Charles Babbage ve Ada Lovelace, programlanabilir mekanik hesap makineleri yaptılar. Ada Lovelace delikli kartlar kullanarak Babbace'ın makinelerini yeniden programladığı için ilk programcı sayılabilir.
1923: Karel Capek'in R.U.R. adlı tiyatro oyunu robot sözcüğünü ilk kez telafuz etti.
1931: Kurt Göldel ünlü eksiklik teoremini ortaya attı.
1936: Konrad Zuse Z1 adını verdiği programlanabilir 64K hafızaya sahipbir bilgisayar yaptı.
1946: ENIAC adlı bir oda büyüklüğündeki bilgisayar çalışmaya başladı.
1948: John von Neumann kendini kopyalayabilen bilgisayar programı fikrini ortaya attı.
1950: Alan Turing "Turing Testi" kavramını ortaya attı.
1951: İlk yapay zekâ parogramları Ferranti Mark 1 adlı aygıt için Manshester Üniversitesi'nde yazıldı.
1956- Dartmaouth Görüşmesi: “Yapay Zeka” ismi ortaya atıldı.
1958: MIT’den John Mc Carty LISP dilini yarattı.
1960: J.C.R. Licklider yaptığı bir makalede insan- makine simbiyozunu anlattı.
1962: Endüstüriyel robot üreten ilk firma Unimation kuruldu.
1965: ELİZA adlı yapay zekâ programı yazıldı.
1966: Stanford Üniversitesi'nde ilk hareketli robot "Shakey" üretildi.
1973: DARPA’da TCP/IP olarak adlandırılan protokollerde geliştirme çalışmaları başladı.
1974: İnternet sözcüğü, ilk olarak Vint Cerf ve Bob Kahn tarafından bir yazıda kullanıldı.
1978: Herbet Simon yapay zeka alanındaki önemli adımlardan biri olan Sınırlı Rasyonalite Teorisiyle Ekonomi Dalından Nobel Ödülü Kazandı.
1979: Standford Yapay Zekâ Labaratuvarı'nda Hans Maravec, Standford Arabası'nı başarıyla denedi.
1981: IBM ilk kişisel bilgisayarını piyasaya sürdü.
1993 : MIT'de Cog adlı insan biçimli bir robotun yapımına başlandı.
1997 : Deep Blue adlı süper bilgisayar satrançta dünya şampiyonu Gary Kasparov'u yendi.
1998 : Tiger Electronics firması evlere girmeyi başaran ilk yapay zekâ oyuncağı Furby'yi piyasaya sürdü.
2000 : Cynthia Braezeal, "Kısmet" adını verdiği robotu tanıttı. Bu robot karşısındaki kişiyle konuşurken mimik ve yüz hareketlerini kullanabiliyor.
2005: Honda Firması Asimo adını verdiği yapay zekâ sahibi insansı robotu tanıttı. ASIMO o güne dek yapılmış en becerikli insansı robottu.
2005 ve sonrası: IBM Watson, Siri Google Search gibi sayabileceğimiz bir sürü yapay zeka uygulaması
Yukarıdaki yazdığım gibi son yıllarda yapay zekâ araştırmacılar tarafından en çok

Yapay zekâ teknolojisi tarihçesinde gördüğümüzdeki gibi her geçen gün daha fazla gelişmektedir. Yeni ürünler ortaya çıkmakta ve daha çok günlük hayatta kendisini göstermektedir. Otomasyon sistemleri de yapay zekâ teknolojisi ile donatılarak bilgisayarın karar verme gücünden faydalanılmaktadır. Her geçen gün daha yeni ticari sistemler ortaya çıkmakta ve sistemlerin fonksiyonel özellikleri artmaktadır. Bununla birlikte yapay zekânın geliştirilmesi için farklı yöntemler ve teknolojiler kullanılmamıştır Yapay zekâ teknolojilerinden özellikle; [2]

1. Bulanık Mantık (BM) :Bulanık mantık, bulanık eseme ya da puslu mantık, 1961 yılında Lütfü Aliasker Zade'nin yayınladığı bir makalenin sonucu oluşmuş bir mantık yapısıdır. Bulanık mantığın temeli bulanık küme ve alt kümelere dayanır.
2. Genetik Algoritma (GA) Biyolojinin temel aldığı Genlerin değişmesi üzerine kurulu bir sistemdir
3. Uzman Sistemleri (US): Belirli bir alanda derlenen bilgileri temel alarak kendisini bu alanda geliştiren yazılım sistemidir
4. YZ Optimizasyon Algoritmaları Karınca algoritması arı algoritması gibi en uygun şekle sokma algoritmalarındır

5. Yapay Sinir Ağları (YSA): insan beynindeki sinirleri baz alınarak geliştirilmiştir.
6. …..

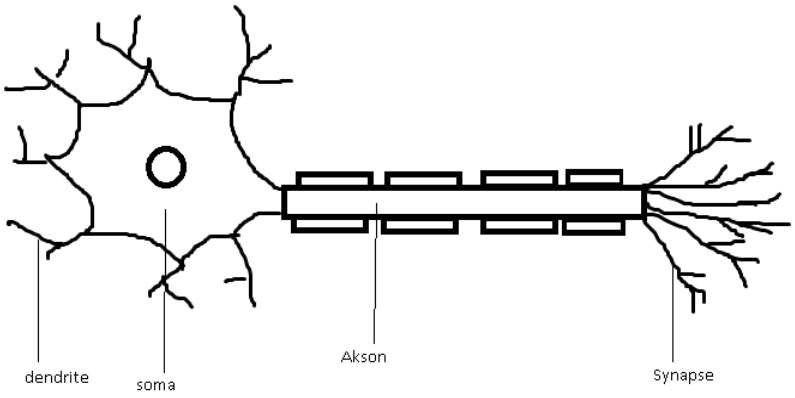
2. YAPAY SİNİR AĞLARI

Yapay Zekâ ortaya çıkması ile birlikte makinelerin eğitilmesi için farklı yöntemler çıkmıştır. Zeka işin içene girdiği için insan beynin örnek alan bir akım oluşmuştur Yapay Sinir Ağları (YSA) insan beyninden esinlenerek geliştirilmiş, ağırlıklı bağlantılar aracılığı ile birbirine bağlanan işlem elemanlarından oluşan paralel ve dağıtılmış bilgi işleme yapılarıdır. En önemli özelliği, deneyimlerden (tecrübe) yararlanarak öğrenebilmesidir. Yapay sinir ağları, insan beyninin özelliklerinden olan öğrenme yolu ile yeni bilgiler türetebilme, yeni bilgiler oluşturabilme ve keşfedebilme gibi yetenekleri herhangi bir yardım almadan otomatik olarak gerçekleştirmek amacı ile geliştirilmişlerdir. Yapay sinir ağları, öğrenmenin yanı sıra bilgiler arasında ilişkiler oluşturma yeteneğine de sahiptir.[3]
Yapay sinir ağlarının dayandığı ilk hesaplama modelinin temelleri 1940'ların başında araştırmalarına başlayan W.S. McCulloch ve W.A. Pitts'in, 1943 yılında yayınladıkları bir makaleyle atılmış olmuştur. Daha sonra 1954 yılında B.G. Farley ve W.A. Clark tarafından bir ağ içerisinde uyarılara tepki veren, uyarılara adapte olabilen model oluşturulmuştur. 1960 yılı ise ilk neural bilgisayarın ortaya çıkış yılıdır. 1963 yılında basit modellerin ilk eksiklikleri fark edilmiş, ancak başarılı sonuçların alınması 1970 ve 1980'lerde termodinamikteki teorik yapıların doğrusal olmayan ağların geliştirilmesinde kullanılmasına kadar gecikmiştir. 1985 yapay sinir ağlarının oldukça tanındığı, yoğun araştırmaların başladığı yıl olmuştur (Mehra Pankaj Wah W Benjamin, 1992).

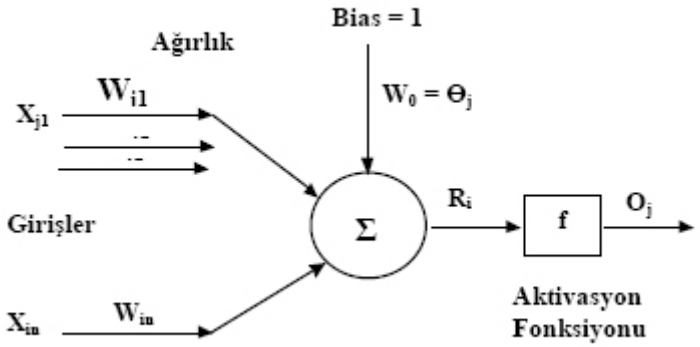
İnsan beyninin çalışma prensibini taklit ederek çalışan bu sistemler, her ne kadar bilgisayar teknolojisi hızlı bir gelişim göstermiş, işlem hızları nano saniyeler mertebesine inmiş olsa da, bırakalım insan beynini, ilkel bir canlı beyninin fonksiyonları dahi baz alındığında, böyle bir organizmanın yanında çok ilkel kalmaktadır. Nano saniyeler bazındaki işlem hızları ile YSA'lar, mili saniyeler mertebesindeki işlen hızları ile işlem yapan insan beyninin işlevselliğinin henüz çok uzağındadır[4]

Yapay sinir ağlarında öğrenme işlemi ağırlıklara verilen değerler ile gerçekleştirilir. Öğrenilen bilgi bütün ağa yayılmış olan ağırlıklarda saklıdır. [1]Yapay sinir ağları çok fazla girdinin, özelliğinin bulunduğu sistemlerde anlaşılması zor olan bağlantıları, ilişkileri ortaya çıkartabilir.[3] Yapay sinir ağları sınıflandırma, tahmin, örüntü tanıma vb. gibi birçok uygulamada başarılı olarak kullanılmaktadır. Problem tipleri farklılık gösterdiğinden dolayı ortak bir yapay sinir ağı modeli bulunmamaktadır. Problemin girdileri, girdilerin tipleri (tam sayı, metin, reel sayı vb.), beklenen çıktılar, çıktıların şekli (kategorik, ikili) gibi özellikler problemin zorluğunu ve aynı zamanda yapay sinir ağı modelinin yapısını değiştirmektedir. Oluşturulan yapı belirsizliklere, hatalara ve daha önce karşılaşılmamış olan örneklere, durumlara karşı toleranslı olmalıdır. [1]

Biyolojik Sinir Sistemi	**Yapay Sinir Sistemi**
Nöron	İşlemci eleman
Dentrit	Toplama fonksiyonu
Hücre gövdesi(soma)	Transfer fonksiyonu
Aksonlar	Yapay nöron çıkışı
Sinapslar	Ağırlıklar



Biyoloji beyin sinirinin temsili resmi



Bir yapay sinir ağ modeli[3?]

Yapay sinir ağları, ağırlıklı bağlantılar aracılığıyla birbirine bağlanan ve her biri kendi belleğine sahip işlem elemanlarından oluşan paralel ve dağıtılmış bilgi işleme yapılarıdır[6] ypay sinir ağını eğitmek için veri seti verilir verilen öğrenme kuralına göre ağırlık değerleri eşit şekilde dağıtlmalıdır bu eğitimden sonra verilen data setlerine göre tahminler başlar
Yapay Sinir ağları bakıldığında genel olarak 3 katmandan söz edebiliriz

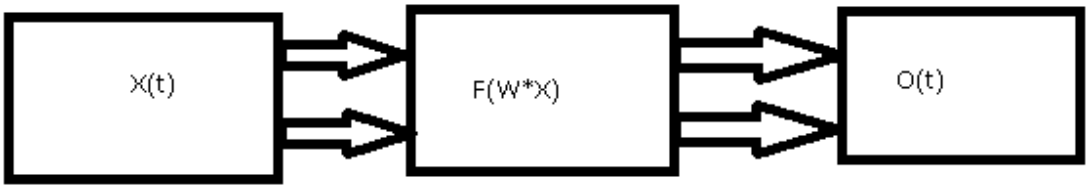
2.1 Giriş Katmanı
Bu katmanda herhangi bir işlem olmaz sadece dışardan girilen bilgileri toplar ve ara katmana iletir

2.2 Ara Katman (Gizli Katman)
Bu katmanda giriş katmanından gelen verilerin işlenmesi sağlanır yani bu katman işlemci katmanıdır. Ve birden fazla işlemci bu katmanda olabilir karmaşık problemlerin çözümünü kolaylaştırmak için

2.3 Çıkış Katmanı
Çıkış katmanı işlenmiş bilgiyi çıkaran yâda tekrar işlenerek ağın geri beslemesini sağlayan katmandır Burada ağırlık düzenlemesi yapılarak tekrardan ağa gönderilirken yapılan düzenleme farklı yöntemler kullanılabilir ayrıca bu kullanılan yöntemler yapay sinir ağlarının sınıflandırılmasını sağlar.

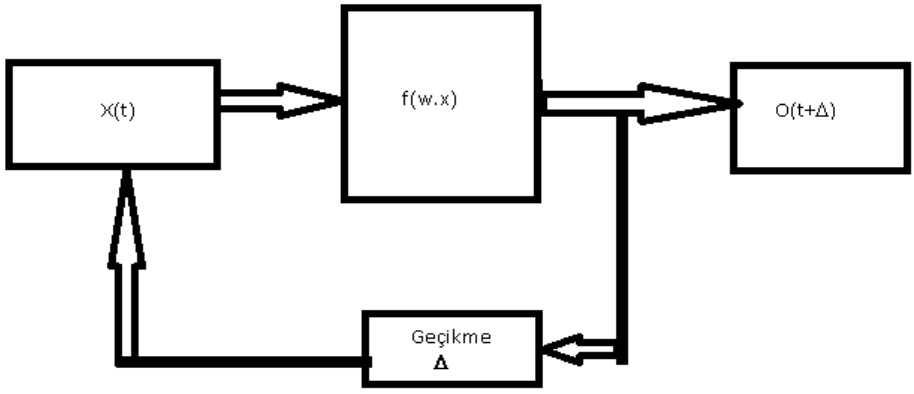
İleri Beslemeli Ağlar: İleri beslemeli ağlarda nöronlar girişten çıkışa doğru düzenli katmanlar şeklindedir. Bir katmandan sadece kendinden sonraki katmanlara bağ bulunmaktadır. Yapay sinir

ağına gelen bilgiler giriş katmanına daha sonra sırasıyla ara katmanlardan ve çıkış katmanından işlenerek geçer ve daha sonra dış dünyaya çıkar. [4]



İleri beslemeli yapay sinir ağ modeli

Geri Beslemeli Yapay Sinir Ağları:
Geri beslemeli yapay sinir ağlarında ileri beslemeli olanların aksine bir hücrenin çıktısı sadece kendinden sonra gelen hücrenin katmanına girdi olarak verilmez. Kendinden önceki katmanda veya kendi katmanında bulunan herhangi bir hücreye de girdi olarak bağlanabilir.[4]

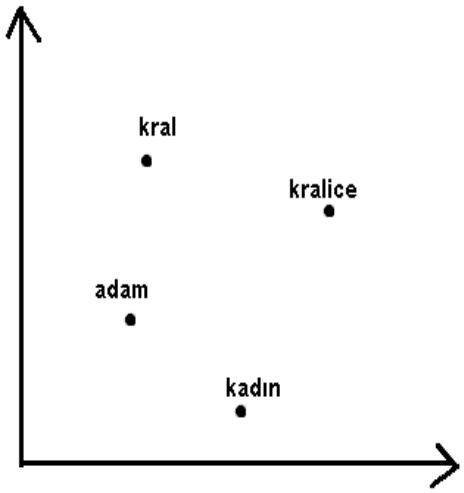


Geri beslemeli Yapay sinir ağ modeli

Yapay sinir ağlarının başarısını doğrudan etkileyen birçok parametre bulunmaktadır. Bu parametrelerin optimize edilmesi yapay sinir ağının başarısını olumlu bir şekilde arttıracaktır. Bazı parametreler: [2]

- Örnek Sayısı
- Verilerin normalize edilmesi
- Katman sayısı
- Girdi ve çıktı verilerinin uygun bir şekilde temsil edilmesi
- Aktivasyon fonksiyonlarının doğru seçilmesi
- Öğrenme oranı
- Hata hesaplama ve hata optimizasyonu fonksiyonlarının doğru seçilmesi

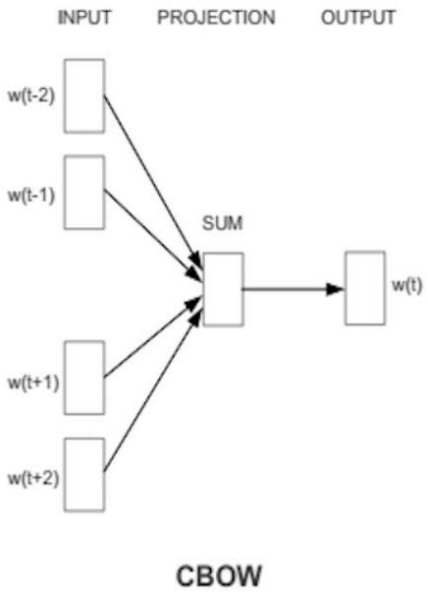
2.2 Kelime Vektörleri (Word2Vec)
WordVec kabaca kelimeleri vectör haline getiren bir çeşit algoritma sistemidir. Kelimelerin bir birbirine olan uzaklığının tespiti ile kelimeler arasındaki ilişkiyi bulur ve bu şekilde kelimeleri birleştirerek uygun önerilerde bulunabilirsiniz



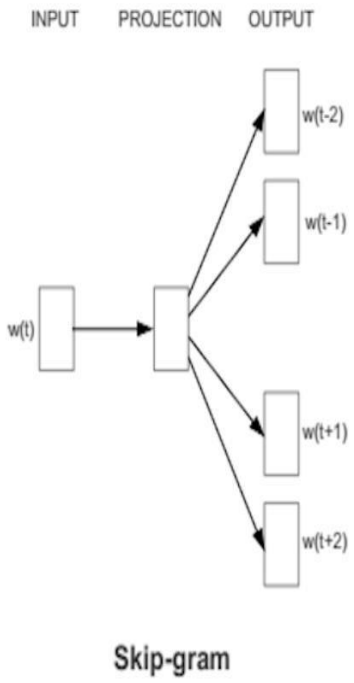
Kral ve kraliçe ilişkisni göstersen birwor2vek örneği

Google araştırmacı Tomas Mikolov ve ekibi tarafından 2013 yılında icat edilmiştir. 2 çeşit alt yöntemi vardır: CBOW(Continous Bag of Words) ve Skip-Gram. 2 yöntem de genel olarak birbirine benzemektedir.

CBOW ve Skip-Gram modelleri birbirlerinden giriş ve çıkış paramtresi alma açısından farklılaşıyor.
CBOW modelinden "windows size" merkezinde olmayan kelimeleri alıp merkezde olan kelimeleri tahmin etmeye çalışırken Skip Gram modelinde ise merkezde olan kelimeleri alıp merkezde olmayan kelimeleri bulunma prensibine göre çalışır
Örnek Bugün hava güzeldir
CBOW göre güzeldir+hava >>Bugün
Bugün hava+ güzeldir sonuçlarını üreti
Skip Gram
güzeldir>>> hava+Bugün
Bugün+hava güzeldir



Şekil 4. CBOW Yapısı[2]



Şekil 4. Skip-gram Yapısı[2]

3. Uygulamalar

3.1 Yapay Sinir Ağları Kullanılarak Pima İndians Diabetes Verisinin Sınıflandırılması Uygulaması

Bu uygulamada yapay sinir ağları kullanılarak verilen özelliklere bulunacaktır. Yapılan çalışmanın yazılan uygulama kodu ekler bölümünde bulunmaktadır.(Ek- A)

Veri Seti Özellikleri:

Veri setinde diabet hastası olabilecek ait 8 özellik ve diabet hastası olup olmadığı bilgisi verilmiştir. Veri seti 800 örnekten oluşmaktadır.

1. Hamilelik sayısı
2. Plazma glukoz konsantrasyonu
3. Diastolic kan basıncı
4. Deri kalınlığı
5. Kullanılan insulin miktarı
6. Vucut kitle indexi
7. Diyabet soyağacı
8. Yaş

5. Sınıf (Çıktı):

a. 0 Diyabet değildir

b. 1 Diyabettir

1. Girdi Katmanı (Input Layer) :
    a. Düğüm Sayısı (Input Units): 20
2. Gizli Katman 1 (Hidden Layer 1) :
    a. Düğüm Sayısı (Hidden Units): 12
    b. Aktivasyon Fonksiyonu: relu
    c. Dropout: 0.1
3. Gizli Katman 2 (Hidden Layer 2):
    a. Düğüm Sayısı (Hidden Units): 12

b. Aktivasyon Fonksiyonu: relu
c. Dropout: 0.1

4. Çıktı Katmanı (Output Layer):
    a. Düğüm Sayısı (Output Units): 1
    b. Aktivasyon Fonksiyonu: Sigmoid
    c. Hata Optimizasyonu (Optimizasyon Fonksiyonu): Adam
    d. Hata Hesaplama (Loss Fonksiyonu): binary_crossentropy
    e. Değerlendirme Kriteri (Metric): Accuracy
5. Diğer:
    a. İterasyon Sayısı(Number epoch): 100
    b. Eğitim verileri yüzdesi: %80
    c. Test verileri yüzdesi: %20

Algoritma Akışı:

1-Datasetimizi yüklüyoruz.
2-Datesetimizi ayrıştırıyoruz. X, 8 adet girdimiz. Yani 600 satır ve 8 sutun
3-Şimdi modelimizi oluşturup katmanlarımızı yerleştirelim
4-düğüm sayısı ve özellikleri belirlenir 1 giriş 2 gizli 1 çıkış katmanı
5-modelimizi derliyoruz
6-modelimizin itarasyona sokarız
7-Modelimizi itarasyon sonucunda başarı yüzdesini hesaplıyoruz.
8-Kaç kişinin diyabet hastası olduğunu

TEST SONUÇLARI

Düğüm sayısı	Deney sayısı(100)	Deney sayısı(150)	Deney sayısı(200)
10	0.77	0.77	0,81
15	0.69	0,82	0.84
20	0,72	0,83	0.66

3.2 TenserFlow kullanarak Word2vec uygulaması yapılması

Bu Çalışmada kelime Tensor flow kullanılarak kullanılarak verilen metinde kullanılan kelimelerin bir birine olan uzaklıkları skip diyagram kullanılarak analiz edilmiştir

Word2Vec Parametreleri:

1. **num_skips**: Eğitim algoritmasını tanımlar. 2 olarak seçtik burada skip algorimasını kullanmak için
2. **valid_size:** Benzerliği değerlendirmek için rastgele sözcük grubu.
3. **top_k** komşuluk sayısı

4 average_loss: her döngüde kabul edelebicek kayıp sayısı

Algoritma Akışı:

1.Data okunur
2.Verileri dizelerin bir listesine okunur
3.Listedeki gereksiz kelimeleri atın
4.Değişken doldurmaların yap
    Doldurma 4 genel değişken:
    data - kodların listesi (0'dan vocabulary_size-1'e kadar olan tamsayılar).
    # Bu orijinal metindir ancak sözcüklerin yerini kendi kodlarıyla değiştirir.

count - olayların sayısına göre kelimeler (dizeler) eşleme
dic - sözcüklerin kodlarına eşleştirilmesi
reverse_dictionary - kodları kelimelere eşler

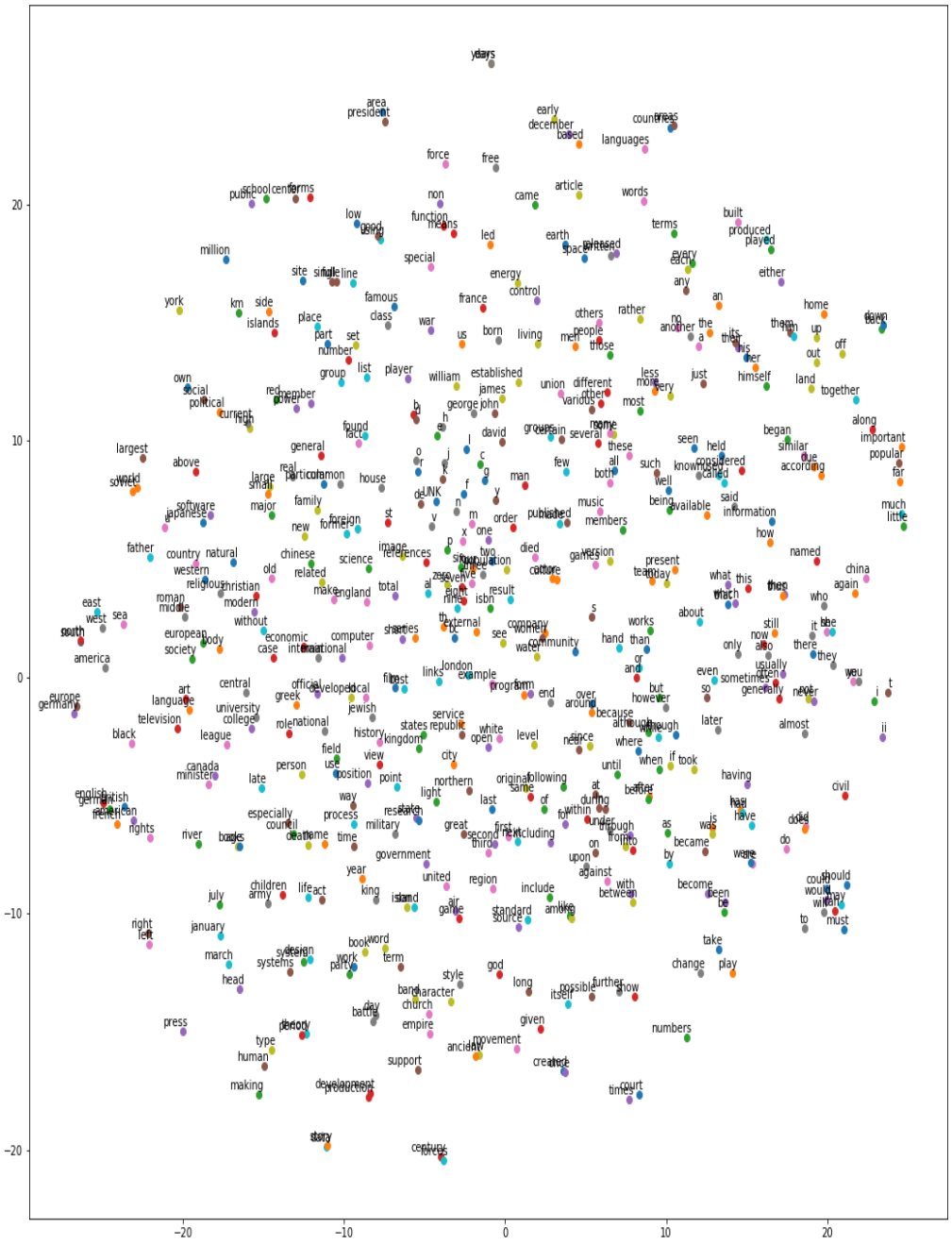
5.Skip gramı modeli içi toplu eğitim işlemi oluşturulur(generate_batch)

6.Eğtim işlemi başlatılır

Tenserfolow Session.run () için döndürülen değerler listesinde en iyileştirici değeri değerlendirerek bir güncelleştirme adımı gerçekleştirilir en yakın 8 komşu bulunur

7 Grafik çizilir

Çıktı Sonucu



3.3 Gensim kullanılarak Makale analizi

Bu uygulama gensim kütüphanesi kullanılarak makale analizi ve anahatar kelimeler bulunması hedeflenmiştir

Veri Seti Özellikleri:

Veri seti internetten makale inceleme için Prof Dr Fevzi Yılmaz hocanın makaleleri baz alınmıştı

Kullanılan parameterler

Ratio(summurize) Özetin uzunluğunu metnin bir kısmı olarak tanımlayın

Words(summurize) özette kullanacağı kelime sayısı

Language özette kullandığı dil Türkçeyi desteklemiyor

Algoritma Akışı

1. Makalenin çekileceği web sayfasındaki linkleri çekilir
2. Çekilen linkler içinden makaleleri gidilir ve makaleler çekilir
3. Çekilen makalenin içindeki cümleleri ağırlıkları hesaplanır
4. Makalenin anahtar kelimeleri ve özeti yazılır

Örnek Bir Makalenin çekilen Datası ve sonuçları

Input text:

….. Örneğin, buhar makinesinin icadı termodinamik bilimi ile olmamıştır. Aksine termodinamik bilimi buhar makinesinin icadından sonra gelişmiş ve yeni teknoloji dalgalarını doğurmuştur. Portekizlilerin 15. YY'da sahip oldukları denizcilik, haritalama ve navigasyon bilgisi denizcilerin deneme sınama faaliyetleri ile elde edilmiştir. 18.YY'da icad edilmiş birçok makine ve cihaz derin bilimsel çalışmalarla değil alan çalışanlarının hünerleri ve fikirleriyle icad edilmiştir. İcat ve ürün geliştirme fikirlerinde kullanıcıların payı da çoktur. Örneğin, geçişli elektron mikroskobu (TEM) böyle bulunmuştur. …

Summary: ratio=0.5

Portekizlilerin 15.Y'da sahip oldukları denizcilik, haritalama ve navigasyon bilgisi denizcilerin deneme sınama faaliyetleri ile elde edilmiştir.

Summary: ratio=0.25

18.YY'da icad edilmiş birçok makine ve cihaz derin bilimsel çalışmalarla değil alan çalışanlarının hünerleri ve fikirleriyle icad edilmiştir.

Summary: word_count=50

Portekizlilerin 15.YY'da sahip oldukları denizcilik, haritalama ve navigasyon bilgisi denizcilerin deneme sınama faaliyetleri ile elde edilmiştir. 18.YY'da icad edilmiş birçok makine ve cihaz derin bilimsel çalışmalarla değil alan çalışanlarının hünerleri ve fikirleriyle icad edilmiştir.
İcat ve ürün geliştirme fikirlerinde kullanıcıların payı da çoktur.

Keywords:

icad
edilmistir
denizcilik
oldukları
kullanıcıların
gelistirm

KAYNAKLAR

[1] Sonmez Ç. http://web.itu.edu.tr/~sonmez/lisans/ai/yapay_zeka_icerik1_1.6.pdf
[2] Öztemel E. YAPAY SİNİR AĞLARI papayta yayınev 2013
[3] Aybars U. Kınacı C. Yapay Zeka Teknikleri ve Yapay Sinir Ağları Kullanılarak Web Sayfalarının Sınıflandırılması
[4] Cayırcıoğlu İ. İLERİ ALGORİTMA ANALİZİ Ders notları
[5]. Hamzaçebi, C., & Kutay, F. (2004). Yapay sinir ağlari ile Türkiye elektrik enerjisi tüketiminin 2010 yilina kadar tahmini. *Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi*, *19*(3).
[6]Elmas, Ç. (2003). Yapay Sinir Ağları (Kuram, Mimari, Eğitim, Uygulama). Ankara: Seçkin Yayıncılık.

Ekler:

EK-A : https://github.com/ulasdemirci/word2vec.git

EK-B :https://github.com/ulasdemirci/diabet.git

EK-C : https://github.com/ulasdemirci/makaleozeti.git

YAPAY SİNİR AĞLARI, KELİME VEKTÖRLERİ VE DERİN ÖĞRENME METOTLARI İLE UYGULAMA ÖRNEKLERİ

Sedrettin ÇALIŞKAN[1]

ÖZET

Bu çalışmada, yapay sinir ağları, kelime vektörleri ve derin öğrenme metotları ile yapılan uygulamalardan bahsedilmiştir. Bu çalışmalarda kullanılan metodolojiler ve veri setleri açıklanmıştır. Yapay sinir ağları kullanılarak yapılan geri yayımlı algoritma iyileştirmesi, derin öğrenme metotları kullanarak twitter veri analizi ve kelime vektörleri kullanılarak verilen metin içerisinde geçen kelimelerin birbirleri ile olan ilişkilerinin görselleştirilmesi ve türkçe metinlerin anlam analizinin gerçekleştirilmesi anlatılmıştır. Yapılan farklı çalışmaların sonuçları, parametre analizi ve kaynak kodları makale içerisinde ve ekler kısmına konumlandırılmış bir şekilde paylaşılmıştır.

Anahtar Kelimeler: Yapay sinir ağları, geri yayımlı, kelime vektörleri, twitter, derin öğrenme.

1. GİRİŞ

Yapay sinir ağları, insan beyninin sinir hücrelerinden oluşmuş katmanlı ve paralel olan yapısının, tüm fonksiyonlarıyla beraber sayısal dünyada gerçeklenmeye çalışılan modellenmesidir. Sayısal dünya ile belirtilmek istenen donanım ve yazılımdır. Bir başka ifadeyle yapay sinir ağı hem donanımsal olarak hemde yazılım ile modellenebilir. Bu bağlamda, yapay sinir ağları ilk elektronik devreler yardımıyla kurulmaya çalışılmış ancak bu girişimi kendini yavaş yavaş yazılım sahasına bırakmıştır. Böylesi bir kısıtlanmanın sebebi; elektronik devrelerin esnek ve dinamik olarak değiştirilememesi ve birbirinden farklı olan ünitelerin bir araya getirilememesi olarak ortaya konmaktadır.[1]

Derin Öğrenme , Makine Öğrenmesi (Machine Learning) tekniklerinden sadece biri. Genel olarak kastedilen şey ise çok katmanlı Yapay Sinir Ağları'ndan başka bir şey değil. Yapay Sinir Ağları'nı temel alan sistemler önce Ses ve Konuşma Tanıma alanında mevcut sistemlerden daha iyi performans göstermeye ve hayatımıza girmeye başladılar. Aynı şekilde bugün cep telefonlarımız

[1] Mühendislik ve Fen Bilimleri Enstitüsü, Fatih Sultan Mehmet Vakıf Üniversitesi, Beyoğlu,İstanbul. sdrttnclskn@gmail.com

sesli komutları anlarken bu teknolojiden yararlanıyor. Öz olarak derin öğrenme, büyük verinin hızlı işlemcilerde yapay zeka teknikleri ile işlenip bilgiye dönüştürülmesidir.

Word2Vec , kelimeleri vektör uzayında ifade etmeye çalışan, tahmin temelli bir modeldir. Kelimelerin bilgisayarın anlayacağı şekilde vektörler halinde temsil edilmesini ve kelimeler arasındaki uzaklığı vektörel olarak hesaplamanızı sağlayan bir algoritma araç kitidir. Bu vektörel yapının üzerine yazılmış araçlar ile bir kelimeye en yakın kelimleri listeletebilirsiniz. Kelimeler arası ilişki kurabilirsiniz.[2]
Yapılan bu çalışmada Yapay sinir ağları kullanılarak Geri Yayılımlı Yapay Sinir Ağları Hata fonksiyonunu ve toplam hatayı minimuma yaklaştırma çalışmaları yapılmaktadır. Derin Öğrenme ile Twitter Duyarlılık Analizi yapılmakta ve Kelime Vektörleri ile de metin analizi yapılmaktadır.

Bölüm 2'de Kullanılan yöntemlerden ,Yapay Sinir Ağları, Kelime Vektörleri tanımılar ve kullanımlarından bahsedilmiştir.. **Bölüm 3**'de Yapılan uygulamlar ve uygulama sonuçlarına dair anlatımlar yapılmıştır.

2. KULLANILAN YÖNTEMLER

2.1 Yapay Sinir Ağları(YSA)

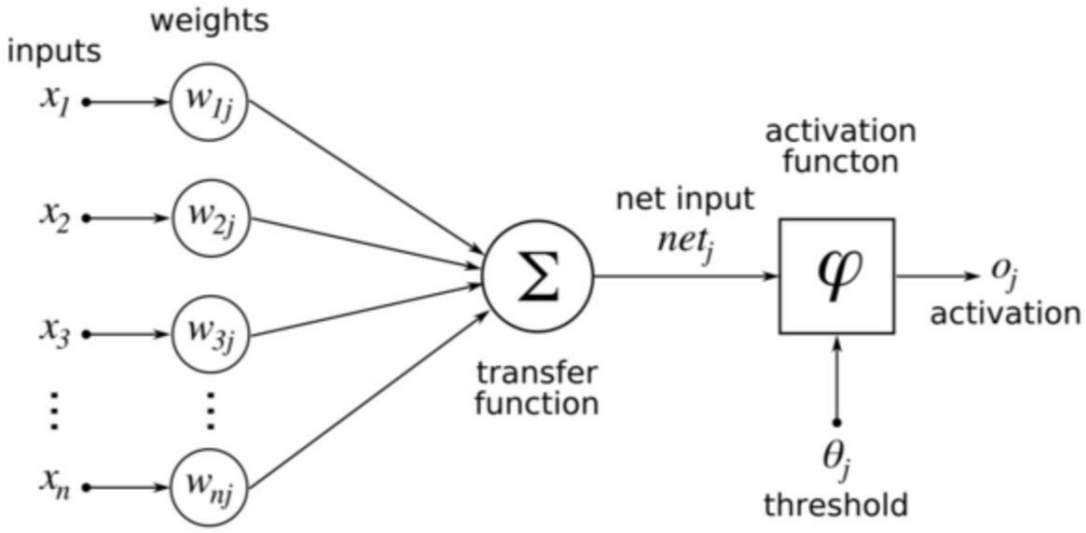
İlk yapay nöron, 1943 yılında nöropsikiyatrist Warren McCulloch ve bilim adamı Walter Pits tarafından üretilmiştir. Ancak dönemin kısıtlı olanakları nedeniyle, bu alanda çok gelişme sağlanamamıştır. Bundan sonra 1969'da Minsky ve Papert bir kitap yayınlayarak, yapay sinir ağları alanında duyulan etik kaygıları da ortadan kaldırmış ve bu yeni teknolojiye giden yolu açmışlardır. İlk gözle görülür gelişmeler ise 1990'lı yıllara dayanmaktadır.[4]

Genel anlamda YSA, beynin bir işlevi yerine getirme yöntemini modellemek için tasarlanan bir sistem olarak tanımlanabilir. YSA, yapay sinir hücrelerinin birbirleri ile çeşitli şekillerde bağlanmasından oluşur ve genellikle katmanlar halinde düzenlenir. Donanım olarak elektronik devrelerle veya bilgisayarlarda yazılım olarak gerçekleşebilir. Beynin bilgi işleme yöntemine uygun olarak YSA, bir öğrenme sürecinden sonra bilgiyi saklama ve genelleme yeteneğine sahip paralel dağılmış bir işlemcidir [3]. Turing makineleriyle temeli atılan yapay zeka üzerinde en fazla

araştırma yapılan konu “Yapay Sinir Ağları”dır. Yapay sinir ağları, temelde tamamen insan beyni örneklenerek geliştirilmiş bir teknolojidir .Bir sinir ağı, bilgiyi depolamak ve onu kullanışlı hale getirmek için doğal eğilimi olan basit birimlerden oluşan paralel dağıtılmış bir işlemcidir. İnsan beyni ile iki şekilde benzerlik göstermektedir: 1.Bilgi, öğrenme süreci yoluyla ağ tarafından elde edilir. 2. Sinaptik ağırlıklar olarak bilinen nöronlar arası bağlantı kuvvetlerini, bilgiyi depolamak için kullanır .[5]

Yapay sinir ağları başlıca; Sınıflandırma, Modelleme ve Tahmin uygulamaları olmak üzere, pek çok alanda kullanılmaktadır. Başarılı uygulamalar incelendiğinde, YSA'ların çok boyutlu, gürültülü, karmaşık, kesin olmayan, eksik, kusurlu, hata olasılığı yüksek sensör verilerinin olması ve problemi çözmek için matematiksel modelin ve algoritmaların bulunmadığı, sadece örneklerin var olduğu durumlarda yaygın olarak kullanıldıkları görülmektedir. Bu amaçla geliştirilmiş ağlar genellikle şu fonksiyonları gerçekleştirmektedirler; Muhtemel fonksiyon kestirimleri, sınıflandırma, ilişkilendirme veya örüntü eşleştirme, zaman serileri analizleri, sinyal filtreleme, veri sıkıştırma, örüntü tanıma, doğrusal olmayan sinyal işleme, doğrusal olmayan sistem modelleme, optimizasyon, Kontrol YSA'lar pek çok sektörde değişik uygulama alanları bulmuştur.[6]

2.1.1 Yapay Sinir Ağı Yapısı

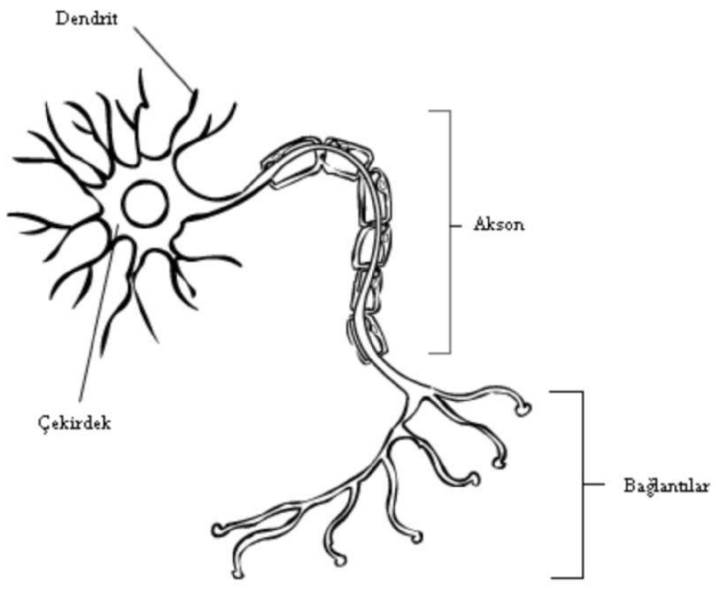


Şekil 1: Yapay sinir hücre yapısı

YSA, insan beyninin çalışma mekanizmasını taklit ederek beynin öğrenme, hatırlama genelleme yapma yolu ile yeni bilgiler türetebilme gibi temel işlevlerini gerçekleştirmek üzere geliştirilen mantıksal yazılımlardır. YSA biyolojik sinir ağlarını taklit eden sentetik yapılardır.YSA, biyolojik sinir ağları taklit eden sentetik ağlardır. Yapay Sinir Ağı modelindeki terimler (Tablo 1) [7]

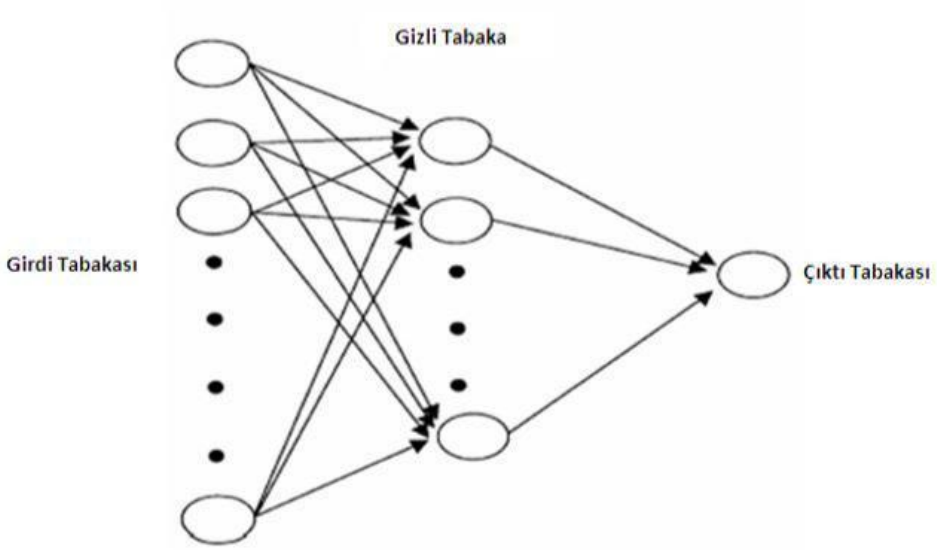
Tablo 1: Yapay Sinir Ağı modelindeki terminolojisi

Sinir Sistemi	Yapay Sinir Ağı
Nöron	İşlem Elemanı
Dentrit	Toplama Fonksiyonu
Hücre Gövdesi	Aktivasyon Fonksiyonu
Akson	Eleman Çıkışı
Sinaps	Ağırlıklar



Şekil 2: Biyolojik sinir hücre yapısı

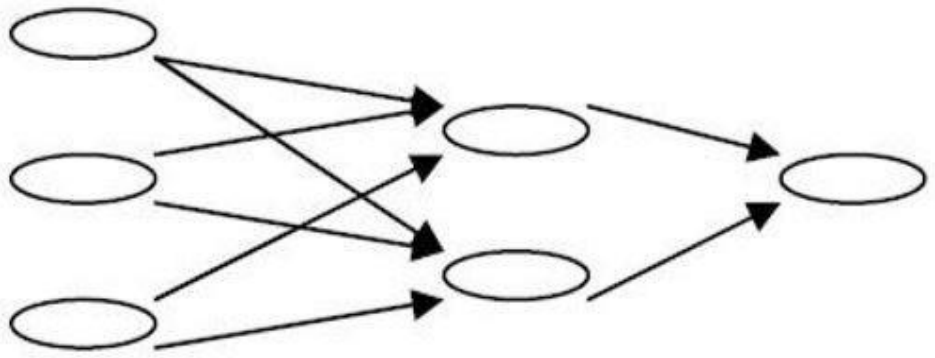
2.1.2 Yapay Sinir Ağlarının Mimari Yapısı



Şekil 3: Mimari Yapısı

Girdi, Gizli ve Çıktı tabakalarından oluşan 3 tabakalı (ya da katmanlı) ileri beslemeli bir sinir ağı modeli görülmektedir.[8]

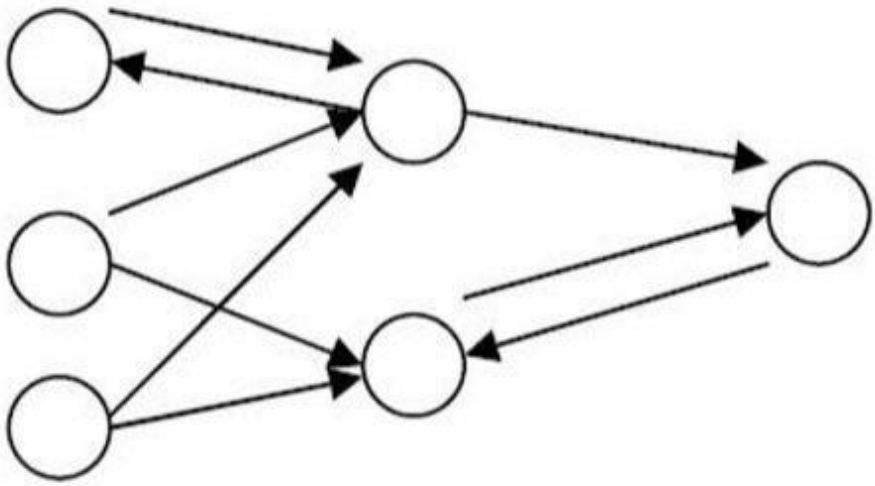
İleri Beslemeli YSA Modeli



- Tek yönlü sinyal akışı için izin verir.
- İleri beslemeli yapay sinir ağında, hücreler katmanlar şeklinde düzenlenir ve bir katmandaki hücrelerin çıkışları bir sonraki katmana ağırlıklar üzerinden giriş olarak verilir.

- Giriş katmanı, dış ortamlardan aldığı bilgileri hiçbir değişikliğe uğratmadan ara (gizli) katmandaki hücrelere iletir.
- Gizli ve çıktı tabakalarından bilginin işlenmesi ile çıkış değeri belirlenir.

Geri Beslemeli YSA Modeli

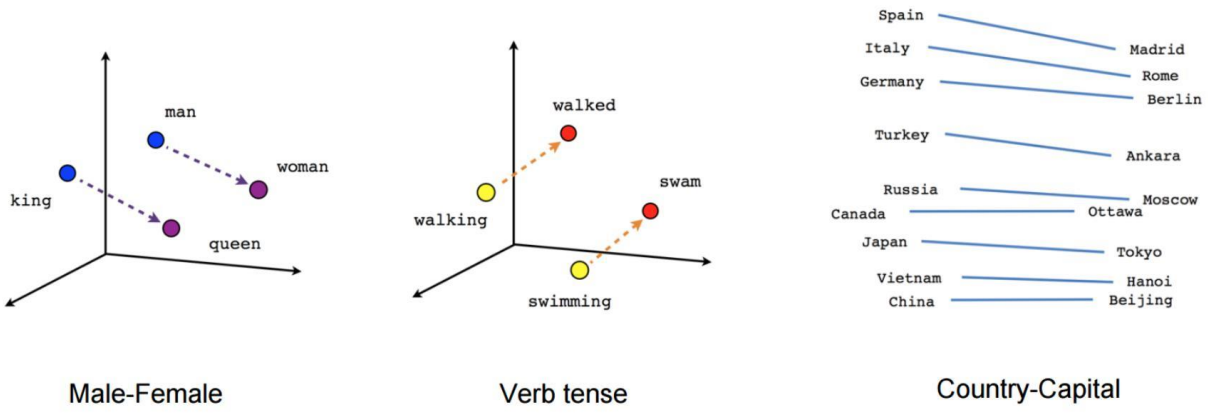


- Geri beslemeli Yapay Sinir Ağları (YSA)' da, en az bir hücrenin çıkışı kendisine ya da diğer hücrelere giriş olarak verilir ve genellikle geri besleme bir geciktirme elemanı üzerinden yapılır.
- Geri besleme, bir katmandaki hücreler arasında olduğu gibi katmanlar arasındaki hücreler arasında da olabilir.
- Bu çeşit sinir ağlarının dinamik hafızaları vardır. Bu yapıdaki nöronların çıkışı sadece o anki giriş değerlerine bağlı değildir ayrıca önceki giriş değerlerine de bağlıdır. Bundan dolayı, bu ağ yapısı özelikle tahmin uygulamaları için uygundur. Bu ağlar özellikle çeşitli tipteki zaman serilerinin tahmininde oldukça başarı sağlamıştır [9]

2.2 Kelime Vektörleri (Word2Vec)

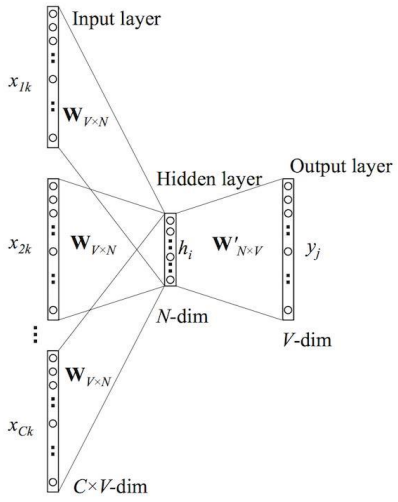
Word2Vec , kelimeleri vektör uzayında ifade etmeye çalışan ve tahmin temelli bir modeldir. Google araştırmacı Tomas Mikolov ve ekibi tarafından 2013 yılında icat edilmiştir. İki çeşit alt yöntemi

vardır: CBOW(Continous Bag of Words) ve Skip-Gram. iki yöntem de genel olarak birbirine benzemektedir. Word2Vec kelimeler arasındaki uzaklığı vektörel olarak hesaplamanızı sağlayan bir algoritma araç kitidir. Bu vektörel yapının üzerine yazılmış araçlar ile bir kelimeye en yakın kelimleri listeletebilirsiniz. Kelimeler arası ilişki kurabilirsiniz.



Şekil 4: Üç boyutlu uzayda Word2Vec örneği [10]

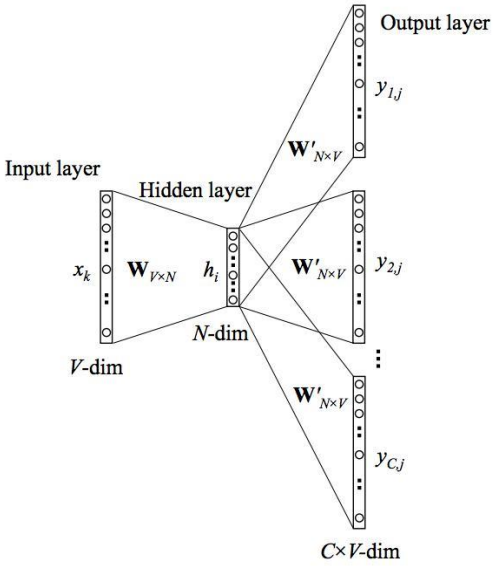
Word2Vec kelimelerin vektörel temsili için iki farklı model mimariden birini kullanabilir: CBOW ve skip-gram. CBOW ve Skip-Gram modelleri birbirlerinden output'u ve input'u alma açısından farklılaşmaktadır.



Şekil 5: CBOW Yapısı[11]

Örnek cümle: ''*Gözümün önünden yorgun insanlar geçiyordu birer birer. "* Bu cümleyi input olarak alan ve ***window_size***=1 olan CBOW modeli çalışma süreci: Önce "Gözümün" kelimesini window'un merkezine oturtuyor, sonra sağındaki ve solundaki 1'er kelimeyi ayrı ayrı input olarak alıp (çünkü ***window_size***=1) merkeze oturttuğu "Gözümün" kelimesini Neural Network modeli ile

tahmin etmeye çalışmaktadır. Sonra ***window***'u 1 sağa kaydırmakta, bu sefer window'un merkezine "önünden" kelimesi gelmektedir.



Şekil 6: Skip-Gram Yapısı[11]

Aynı cümlenin ***window_size*** =1 olan Skip-gram modeli çalışma süreci: Merkezdeki kelime input olarak alınıp merkezdeki kelimeye ***window_size*** büyüklüğünden daha az yakın olan kelimeler tahmin edilmektedir. Skip-gram modeli ilk önce cümledeki "Gözümün" kelimesini merkeze oturtuyor ve "Gözümün" kelimesini kullanarak "önünden" kelimesini tahmin etmeye çalışıyor.

Sonuç olarak, CBOW modelleri genel yapısı gereği küçük datasetlerde daha iyi çalışırken, büyük datasetlerde Skip-gram daha iyi çalışmaktadır.

3. Uygulamalar

3.1 Yapay sinir ağları kullanılarak Geri Yayılımlı Yapay Sinir Ağları Uygulaması

Geri yayılımlı yapay sinir ağları iki temel aşamadan oluşur: İleri besleme ve geri yayılım. İleri besleme, ağa giriş verilerinin verildiği aşamadır. Bu aşamanın sonunda elde edilen çıkışlar hata fonksiyonuna girilir ve hatalar geriye yayılarak ağırlıklar güncellenir. Hata fonksiyonunu ve

dolayısıyla toplam hatayı minimuma yaklaştırmak için gradyan iniş metodu kullanılır. Uygulama kodlarının ekler bölümünde linkleri verilmiştir.(Ek - 1)

Veri Seti Özellikleri:

X : Her satırın bir eğitim örneği olduğu girdi veri kümesi matrisi.

y : Her satır bir eğitim örneği olduğu çıktı veri kümesi matrisi.

l0 : Giriş verisi tarafından belirtilen Ağın Birinci Katmanı

l1 : Ağın İkinci Katmanı, aksi halde gizli katman olarak bilinir.

syn0 : Ağırlıkların ilk katmanı, Synapse 0, l0'dan l1'e bağlanma.

* : Elementwise çarpma, böylece eşit boyuttaki iki vektör, özdeş boyutta bir nihai vektör oluşturmak için karşılık gelen 1-to-1 değerlerini çarpar.

- : Eşit boyuttaki son vektör oluşturmak için, eşit boyuttaki iki vektör, karşılık gelen değerleri 1'den 1'e çıkartarak, öğe temelinde çıkarma.

x.dot(y) : Eğer x ve y vektörler ise, bu bir noktalı bir üründür. Her ikisi de matris ise, matris-matris çarpımıdır. Tek bir matris ise, o zaman vektör matris çarpımı olur.

Algoritma Akışı:

Geri yayılım ağındaki öğrenme aşağıdaki adımlardan oluşur:

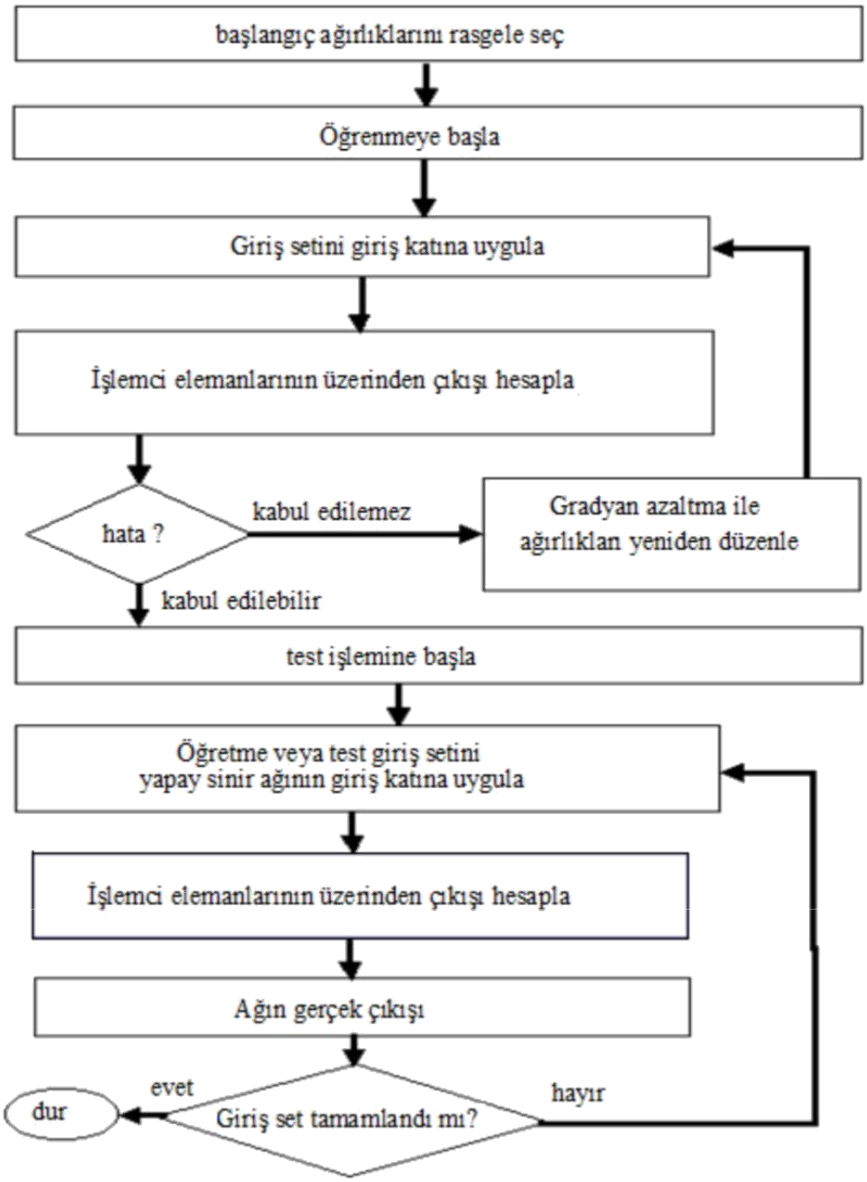
Adım 1. Eğitim kümesinden bir sonraki örneği seçme ve ağ girişine girdi vektörü uygulama.

Adım 2. Ağın çıktısını hesaplama.

Adım 3. Ağın çıktısı ile istenen hedef vektör arasındaki hatayı hesaplama.

Adım 4. Hatayı küçültecek şekilde ağın ağırlıklarını ayarlama.

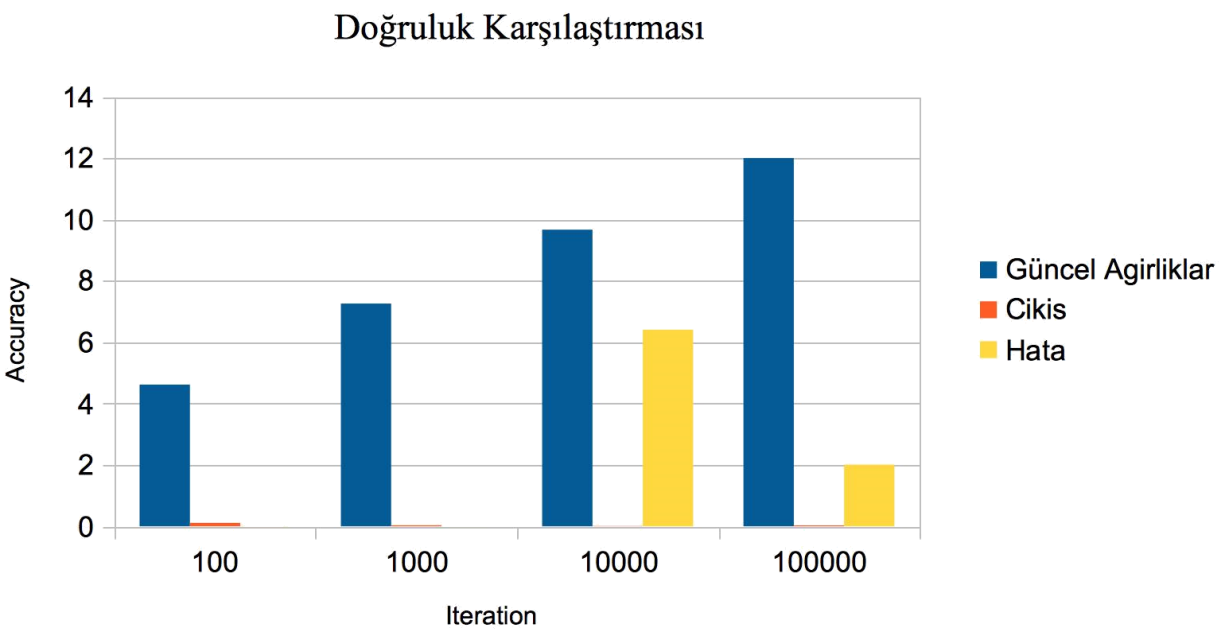
Çok katmanlı ileri beslemeli Y.S.A. modelindeki geri yayılım **Şekil 7'** de verilmiştir.



Şekil 7 : Geri Yayılımlı Yapay Sinir Ağları akış şeması [12]

Sonuçlar:

Tablo 2.'de bulunan değerler modelin eğitimi ve test edilmesi sırasında elde edilen doğruluk değerlerini göstermektedir.



Şekil 8: Eğitim doğruluk oranlarının karşılaştırılması

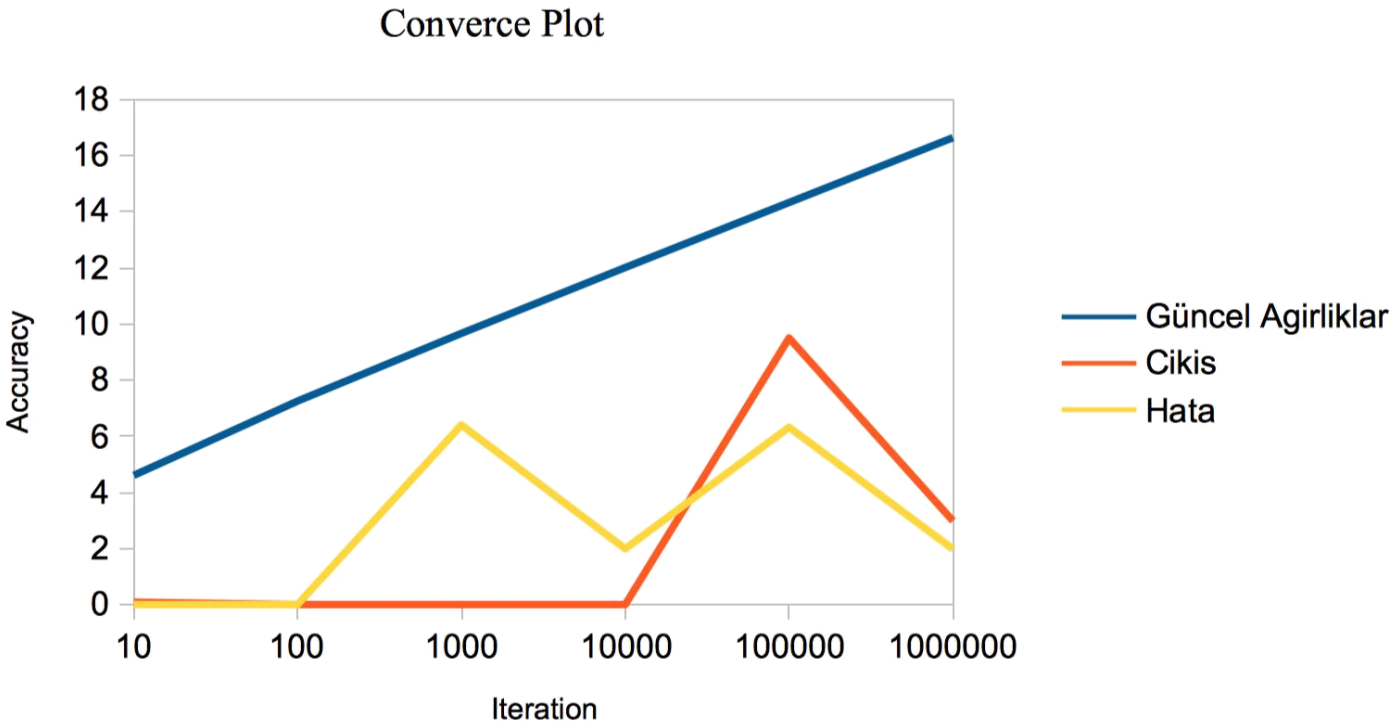
```
Güncel Agirliklar
[[ 9.67299303]
 [-0.2078435 ]
 [-4.62963669]]
Cikis
[[ 0.00966449]
 [ 0.00786506]
 [ 0.99358898]
 [ 0.99211957]]
Hata
[[ -9.66449169e-05]
 [ -7.86505967e-05]
 [  6.41101950e-05]
 [  7.88042678e-05]]
```

Güncel ağırlıklarımız: Öğrendiğimiz değerler

Çıkış değerleri : Sistemin kendi öğrenme değerleri

Hata : Kendi öğrenme sürecindeki hata payı

Şekil 9: 1000 iteration ‘da doğruluk değerleri



Şekil 10: Eğitim doğruluk oranlarının yakınsama grafiği

3.2 Derin Öğrenme Metotları kullanılarak Twitter Duyarlılık Analizi Uygulaması

Bu uygulamada belirtilen anlam kümesi ile ilgili atılan tweet'lerin anlamsal analizi yapılacaktır. Verilen cümlenin olumlu yada olumsuz bir cümle olduğu sınıflandırılacaktır. Yapılan çalışmanın uygulama kodu ekler bölümünde bulunmaktadır.(Ek- 2)

Veri Seti Özelikleri

wiki = TextBlobg("") : Twitter da aranacak olacak kelime dizisi

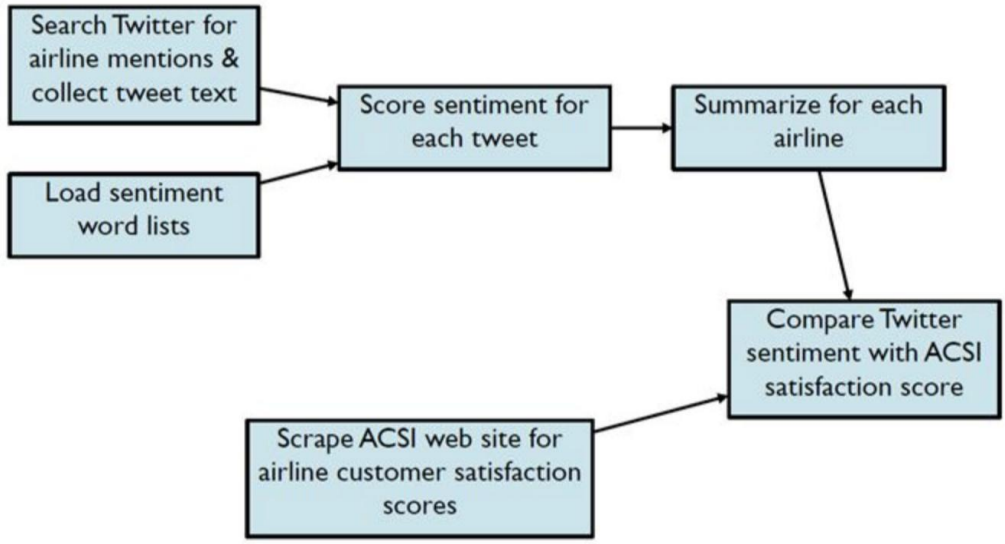
wiki.tags : Twitter da arancak olan kelime dizisin string'lere ayırıp tag'ler haline getirilmesi.

wiki.words : Dizi string 'leri.

wiki.sentiment.polarty = -1 < sentiment < 1 : Duyarlılık aralığımız.

public_tweets = api.serach(''): Tweet oluşturma, tweet silme ve twitter kullancısı bulma parametreleri kullanılabilir.

Akış Diyagramı



Şekil 11: Twitter duyarlılık analiz akış şeması[13]

Sonuçlar:

Örnek alınan kelime dizisi , wiki = TextBlobg (“İlem demek ilmi geleneğin yuvası demektir.”). Kelime dizi ile ilgili bir kelime(search (‘ilem’)) aradığımızda bululan tweet kümesi aşağıdaki gibidir.

```
Yer: İlmi Etüdler Derneği
Zaman: YARIN (13 Ocak Cumartesi) - 17:30

Engin Koca, Yeni doğa felsefesini…
Sentiment(polarity=0.0, subjectivity=0.0)

İLEM SUNUMLARI 131

Yer: İlmi Etüdler Derneği
Zaman: YARIN (13 Ocak Cumartesi) - 17:30

Engin Koca, Yeni doğa fels… https://t.co/PXD0Ul1Mxr
Sentiment(polarity=0.0, subjectivity=0.0)

RT @ilemihtisas: İLEM SUNUMLARINDA YARIN (13 Ocak Cumartesi), 17:30

"Modern Fiziğin Doğuşu"

https://t.co/e8kUTIxecP

#ilemsunumları #ile…
Sentiment(polarity=0.2, subjectivity=0.3)

RT @ilemihtisas: İLEM SUNUMLARINDA YARIN (13 Ocak Cumartesi), 17:30

"Modern Fiziğin Doğuşu"

https://t.co/e8kUTIxecP

#ilemsunumları #ile…
Sentiment(polarity=0.2, subjectivity=0.3)
```

#ilem ile ilgili atılan tweet’ler de olumlu, olumsuz mesajlarda **polarity** ve **subjectivity** -1 ile 1 arasında değiştiğini görüyoruz.

3.3 Kelime Vektörleri (Word2Vec) ile Metin Analizi Uygulaması

Bu çalışmada kelime vektörleri kullanılarak verilen metinlerin analizi gerçekleştirilmiştir. Metin içerisinde geçen kelimelerin birbirleri ile olan ilişkileri hesaplanarak tahminlerde bulunulmuş ve kelimeler arası uzaklık ve yakınlıklar grafik üzerinde görselleştirilmiştir. Yapılan çalışmanın uygulama kodu ekler bölümünde bulunmaktadır.(Ek- 3)

Veri Seti Özellikleri:

Veri seti olarak *"Şakir KOCABAŞ'ın, Yapay Zeka ve Bilim Felsefesi"* makalesi kullanılmıştır.
Word2Vec Parametreleri:

1. **Sg**: Eğitim algoritmasını tanımlar. Ön Tanımlı olarak "0" verilir. "0" ise "CBOW" yöntemi, "1" ise "skip-gram" yöntemi kullanılır. Çalışmamızda "skip-gram" kullanılmıştır.

2. **Seed**: Rastgele sayı üretir.

3. **Workers**: Modeli eğitmek için kullanılacak iş parçacıkları sayısı.

4. **Size**: Özellik vektörünün boyutudur. Kelimelerin temsil edileceği vektör boyutu.

5. **Min_count**: Toplam frekanstan düşük olan bütün kelimeler yok sayılır.

6. **Window**: Cümle içerisinde geçerli ve tahmin edilen kelime arasındaki maksimum uzaklıktır. Yani seçilen kelime ile ilişkili olan kelimeler aranırken, seçilen kelimenin sağındaki ve solundaki kelimelerden kaçar tanesinin inceleneceğini belirtir.

Algoritma Akışı:

1. "Data" klasöründen "Ş*akir KOCABAŞ*" makalesinin olduğu veri seti okunmaktadır.

2. Veri seti istenmeyen karakterlerden temizlenir. Sadece harf ve rakamlar tutulur. Özel karakterler veri setinden temizlenmektedir.

3. Veri seti içerisindeki cümleler analiz edilerek, cümlelerde geçen bütün kelimeler ayrıştırılır ve elde edilen kelimelerin frekans değerleri belirlenmektedir.

4. Word2Vec eğitim modeli “Word2Vec Parametreleri” bölümünde belirtilen değerlere göre oluşturulmaktadır.

5. Oluşturulan model örnek veri seti ile eğitilmektedir.

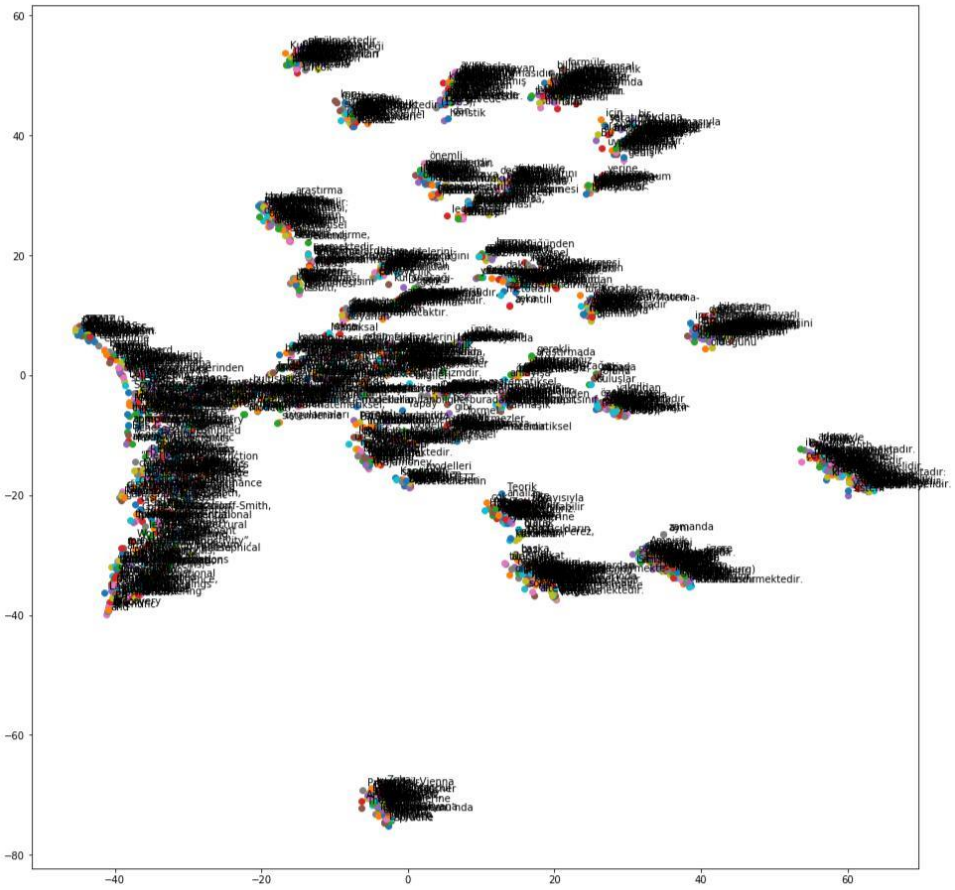
6. Model eğitildikten sonra veri setinde geçen bütün kelimelerin grafik üzerinde gösterilmek üzere “x” ve “y” noktaları belirlenir. Yani bütün kelimelerin birbirleri ile olan yakınlık ve uzaklık ilişkileri belirlenmiş olmaktadır.

7. Kelimeler için belirlenen “x” ve “y” noktaları koordinat düzlemine yerleştirilmektedir.

8. Kelimeler arasındaki ilişkiler grafik üzerinde görselleştirilmektedir.

9. İstenirse elle verilen kelime ile benzerlik gösteren kelimeleri belirlemek üzere benzerlik (similarity) fonksiyonu kullanılmaktadır.

Sonuçlar:



Şekil 12: Veri setinde kelimelerin birbirleri ile olan ilişkilerinin genel görünümü.

Bu çalışmada kullanılan kelimelerin birbirleri ile olan benzerlikleri, farklılıkları, verilen iki kelimenin birbirine ne kadar benzer oldukları belirlenmektedir.

model.similarity ('zeka','bilim') ——->> *0.88671410881046953*
model.similarity('yaratıcılık','yapay') ——->> *0.90104469558231726*
model.similarity('bilgi','metafizik') ———>> *0.89918061140991656*

Şekil 13*:* Benzer kelimelerin yakınlığını göstermektedir.

Şekil 13'te kelimelere atanan vektörler arasındaki kosinus mesafelerine (yakınlıklar/benzerlikler) bakacak olursak, model benzer kelimelerin yakınlığını yüksek, benzer olmayan kelimelerin de yakınlığını düşük olarak tanımlamaktadır.

```
model.most_similar('metafizik')
```

```
[('ontolojik', 0.9992651343345642),
 ('listede', 0.9992560148239136),
 ('motivasyonu', 0.999171257019043),
 ('bkz.', 0.9990996718406677),
 ('eğilimler', 0.9989954233169556),
 ('etkileyebileceği', 0.9989340901374817),
 ('Âlem', 0.9988895058631897),
 ('şimdiki', 0.9988783001899719),
 ('son', 0.9988157749176025),
 ('yer', 0.9987466931343079)]
```

Şekil 14 : "Metafizik" kelimesi ile benzer olan kelimeler ve benzerlik oranları.

Şekil 14'de bulunan sonuçlara bakıldığında verilen kelime ile "ontolojik, motivasyonu, son, yer ,eğilimler" kelimelerinin yüksek oranda benzer oldukları görülmektedir. Metin bağlamına bakıldığında "ontolojik", "Alem", ''yer" kavramların arasında güçlü bir anlam ilişkisi olduğu tespit edilmektedir.

KAYNAKÇA

1. Cinsdiki, M. (1997). *Neural Network Solutions for ATM Routing & Multicasting Problems.* Yayınlanmış Yüksek Lisans Tezi, Ege Üniversitesi, Fen Bilimleri Enstitisü, İzmir.

2 . https://medium.com/@bakiiii/deep-learning-ile-t%C3%BCrk%C3%A7e-film-yorumlar%C4%B1ndan-duygu-ve-puan-tahmini-78a606d86dde, *erişim tarihi: 08. 01.2018*

3. Der: Becerikli, Y. *Yapay Sinir Ağları ile Duygu Analizi.* Kocaeli Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği.

4. Ergezer, H.& Dikmen, M.& Özdemir, E. (2003). *Yapay Sinir Ağları Ve Tanıma Sistemleri Pivolka.* Uygulamalı Yerbilimleri Dergisi, 2(6), s.71-79, Kocaeli.

5. Ataseven, B. (2013) *Yapay Sinir Ağları ile Öngörü Modellemesi.* Dergipark, s.101-115, İstanbul.

6. Çayıroğlu, İ. *İleri Algoritma Analizi-5 Yapay Sinir Ağları.* Karabük Üniversitesi Mühendislik Fakültesi.

7.-http://kod5.org/yapay-sinir-aglari-ysa-nedir/, *erişim tarihi: 09.01.2018*

8. http://www.derinogrenme.com/2017/03/04/yapay-sinir-aglari, *erişim tarihi : 09.01.2018*

9. Aşkın, D. & İskender, İ. & Mamızadeh, A. (2011) *Farklı Yapay Sinir Ağları Yöntemlerini Kullanarak Kuru Tip Transformatör Sargısının Termal Analizi.* Gazi Üniv. Müh. Mim. Fak. Der.Cilt 26, No 4, 905-913, Ankara.

10. https://www.tensorflow.org/tutorials/word2vec , *erişim tarihi: 10.01.2018*

11. https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/, *erişim tarihi: 10.01.2018*

12. Doğan, G. (2010). *Yapay Sinir Ağları Kullanılarak Türkiye'deki Özel Bir Sigorta Şirketinde Portföy Değerlendirmesi.* Yayınlanmış Yüksek Lisans Tezi, Hacettepe Üniversitesi, Ankara.

13. Altun, İ.& Dündar, S. (2005). *Yapay Sinir Ağları ile Trafik Akım Kontrolü.* Deprem Sempozyumu, Kocaeli.

EKLER

EK -1: https://github.com/sdrttnclskn/YapaySinirAglari-GeriYayilimUygulamasi

EK -2 : https://github.com/sdrttnclskn/DataScience-TwitterSentimentAnalysis

EK -3 : https://github.com/sdrttnclskn/Word2Vec-TextAnalysis