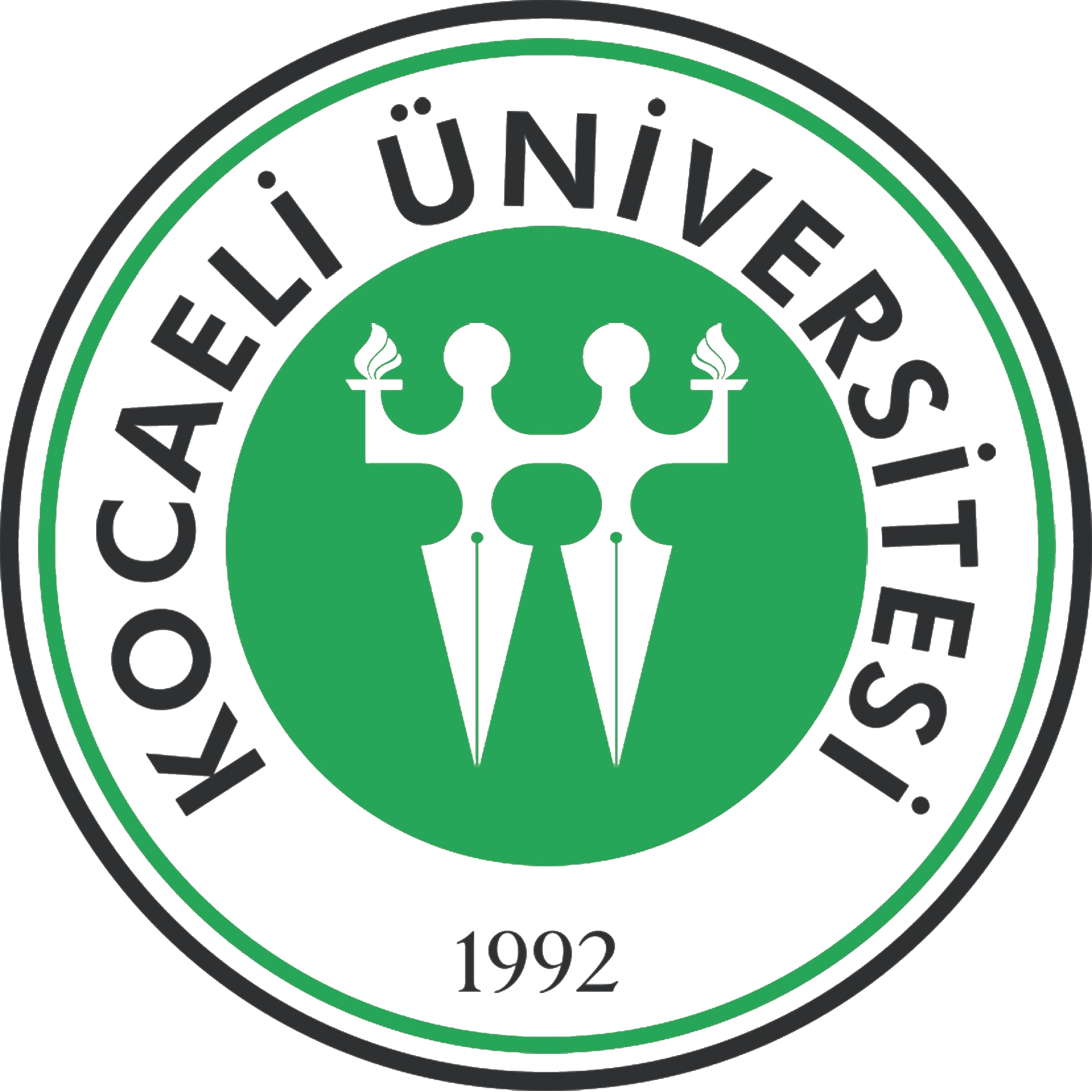
**NÖRON SİNİR AĞLARI İLE CHATBOT UYGULAMASI**

**Bu çalışma, Kocaeli Üniversitesi Bilgisayar Mühendisliği öğrencilerinden Pınar Sarı (170202049), Veli Kaan Çetinel (170202081), Alper Aktepe (160202056), Şamil Asaroğlu (170202091), Fetih Yenerdağ(170202012) tarafından yapılmıştır.**

****

Kocaeli Üniversitesi,Bilgisayar Mühendisliği

**Özet**

Bu projede nöron sinir ağları ile chatbot yapımı gerçekleştirilmiştir. Projemiz Python programlama dili kullanılarak geliştirilmiştir. Geliştirme ortamı olarak kullanılan Spyder 4.1.5 ve Visual Studio Code 1.55.2 ortamlarında başarılı olarak çalıştırılmıştır. Geliştirdiğimiz bu projede, bir arayüz bulunmaktadır. Bu arayüz ile kullanıcıdan bir cümle ya da kelime girişi aldıktan sonra chatbot bir cevap döndürmektedir. Bu aşamaların hepsi arayüzde gözlemlenmektedir.

# Giriş

Yapay sinir ağları (YSA), insan beyninin öğrenme yolunu taklit ederek beynin öğrenme, hatırlama, genelleme yapma yolu ile topladığı verilerden yeni veri üretebilme gibi temel işlevlerin gerçekleştirildiği bilgisayar yazılımlarıdır. Yapay sinir ağları; insan beyninden esinlenerek, öğrenme sürecinin matematiksel olarak modellenmesi uğraşı sonucu ortaya çıkmıştır. Yapay sinir ağları, paralel dağıtılmış ağlar, bağlantılı ağlar, nuromorfik ağlar gibi adlarla da tanımlanmaktadır.

**NEDEN YAPAY SİNİR AĞLARI?**

Uzun evrim süreci, insan beynine, Neumann veya modern paralel bilgisayarlarda bulunmayan pek çok arzu edilen özellik kazandırmıştır. Bu özellikler şunlardır :

* büyük paralellik,
* dağıtılmış temsil ve hesaplama, öğrenme yeteneği,
* genelleme yeteneği,
* uyarlanabilirlik,
* doğal bağlamsal bilgi işleme, hata toleransı ve

düşük enerji tüketimi.

Biyolojik sinir ağlarına dayalı cihazların bu istenen özelliklerden bazılarına sahip olacağı umulmaktadır.

Modern dijital bilgisayarlar, sayısal hesaplama ve ilgili sembol manipülasyonu alanında insanlardan daha iyi performans gösterir. Buna rağmen, insanlar da karmaşık algısal sorunları (bir kalabalığın içindeki bir adamı sadece yüzüne bakarak tanımak gibi), yüksek hızda ve ölçüde çözebilirler. Performansta neden bu kadar belirgin bir fark var? Biyolojik nöral sistem mimarisi von Neumann mimarisinden tamamen farklıdır.Von Neumann’ın merkezileştirilmiş mimarisine dayanan "akıllı" programlar geliştirmeye yönelik çokça çabaya rağmen program sonuçlanamadıı. Biyolojik sinir ağlarından esinlenen YSA'lar, birçok ara bağlantıya sahip son derece büyük bir basit işlemciden oluşan devasa paralel hesaplama sistemleridir.

Projemizde kullanılan kütüphaneler:

1. *nltk:*

Doğal Dil Araç Seti veya daha yaygın olarak NLTK, Python programlama dilinde yazılmış İngilizce için sembolik ve istatistiksel doğal dil işleme için bir kütüphaneler ve programlar paketidir. Pennsylvania Üniversitesi Bilgisayar ve Bilişim Bilimleri Bölümü'nde Steven Bird ve Edward Loper tarafından geliştirilmiştir.

1. *snowballstemmer:*

Stemming, aynı sözcüğün farklı biçimlerini ortak bir "kök" ile eşler. Örneğin, İngilizce sözcük ayırıcı bağlantı, bağlantılar, bağlayıcı, bağlı ve bağlanmak için bağlanma eşler. Dolayısıyla, bağlantılı arama, yalnızca diğer biçimlere sahip belgeleri de bulacaktır.

Bu kök biçimi genellikle bir kelimenin kendisidir, ancak bu her zaman geçerli değildir, çünkü bu, amaçlanan kullanım alanı olan metin arama sistemleri için bir gereklilik değildir. Ayrıca ortak bir dil kökü olan tüm sözcükler yerine aynı anlama sahip sözcükleri birleştirme hedeflenir.

1. *keras:*

Keras, Python'da yazılmış açık kaynaklı bir sinir ağı kütüphanesidir. Keras TensorFlow, Microsoft Cognitive Toolkit, R, Theano veya PlaidML ile beraber çalışabilir. Derin sinir ağları ile hızlı deney yapabilmek için tasarlanan bu cihaz kullanıcı dostu, modüler ve genişletilebilir olmaya odaklanıyor.

1. *numpy:*

NumPy, Python programlama dili için bir kütüphane olup, büyük, çok boyutlu diziler ve matrisler için destek eklerken, bu dizilerde çalışmak için yüksek düzeyli matematiksel fonksiyonların geniş bir koleksiyonudur.

1. *random:*

Rastgele veri üretimi için kullanılan kütüphanedir.

1. *tkinter:*

Python programlama dili ile birlikte gelen grafiksel kullanıcı arayüzü aracıdır. Python'la birlikte gelmesi ve basit bir yapıya sahip olması, Tkinter'in yaygın kullanımına neden olmuştur. Eleman eksikleri çeşitli paketlerle kapatılmaya çalışılmaktadır.

1. *functools :*

Çağrılabilir nesneler üzerinde daha yüksek dereceli fonksiyonlar ve işlemler için kullanılan kütüphanedir. Functools modülü üst düzey fonksiyonlar içindir: diğer fonksiyonlar yerine getiren veya döndüren fonksiyonlardır. Genel olarak, herhangi bir çağrılabilir nesne, bu modülün amaçları doğrultusunda bir fonksiyon olarak değerlendirilebilir.

1. *datetime:*

zaman, saat ve tarihlerle ilgili işlemler yapabilmemiz için bize çeşitli fonksiyon ve nitelikler sunan bazı sınıflardan oluşur. Bu kütüphane içinde temel olarak üç farklı sınıf bulunur.

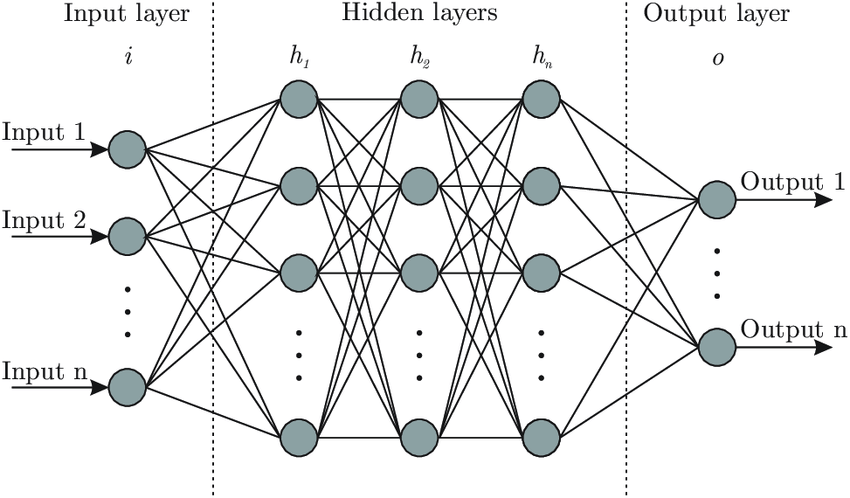
date sınıfı; tarihle ilgili işlemler yapabilmemizi sağlayan fonksiyon ve nitelikleri barındırır.

time sınıfı; zamanla/saatle ilgili işlemler yapabilmemizi sağlayan fonksiyon ve nitelikleri barındırır.

datetime sınıfı; date ve time sınıflarının birleşiminden ve ilave birkaç nitelik ve fonksiyondan oluşur.

# Mimari

Mimari: Yapay sinir ağları; giriş katmanları, gizli katmanlar ve çıktı katmanları olmak üzere üç tür katmandan oluşur. Gizli katmanlar isteğe bağlıdır. Giriş katmanları, girdileri alır ve bu girdileri herhangi bir işlem yapmadan diğer katmanlara iletir. Bir sonraki katman bu girdiyi alır ve üzerinde işlem yapar (toplama ve T ile karşılaştırma) ve sonuçlarını daha sonraki işlemler için sonraki katmanlara iletir.



Öncelikle ağ eğitilir, sonra test edilir ve son olarak uygulanır.Eğitim sırasında ağırlıklara rastgele değerler atanır.Bu değerler eğitim sırasında çıktıya göre değişir.Sinir ağını uygulamak için izlenen bazı adımlar vardır:

1) Veri toplama

2) Veri ayırma eğitimi ve testi

3) Ağ mimarisini seçme

4) Parametre ayarlama ve ağırlık başlatma

5) Veri dönüşümü

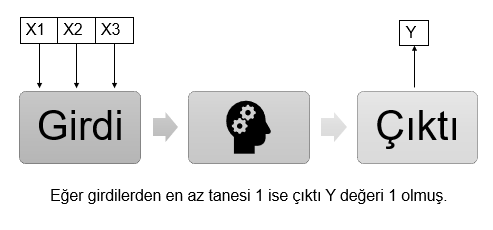
6) Eğitim

7) Test

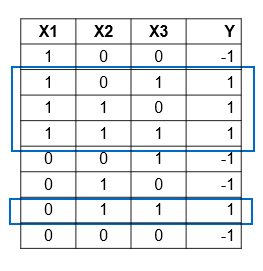
8) Uygulama

1. **Öğrenme Tekniği**

Sinir ağlarının arkasındaki fikir öğrenmek olduğu için deneyimlerinden öğrenen bu tür sistemler geliştirmekle ilgileniyoruz. Uygulamaya geçmeden önce ağı eğitiyor, test ediyor ve son olarak uyguluyoruz. Denetimli öğrenmede, ağa girdi ve çıktı sağlanır. Ağ, girişleri ve çıkışı karşılaştırarak gerekli modeli öğrenir ve ağırlıkları değiştirir. Denetimsiz öğrenmede, ağa yalnızca girdi verilir ve verilen verilerden örüntüyü anlar. Güçlendirilmiş öğrenmede, ağa girdi verilir ve yalnızca çıktısının doğru olup olmadığına dair göstergeler sağlanır.



Öncelikle tablodaki gibi bir veri setimiz olduğunu düşünelim. Bu veri setinde 3 adet girdi değişkeni bir adet çıktı değişkenine dönüşüyor. YSA’da X’ler arasında nasıl bir işlem gerçekleşirse sonuçta çıkan Y’ler gerçeğe daha yakın olur sorusu cevaplanır. Genel olarak girdiler ve girdilerin arasında yapılan bir takım matematiksel işlemler ardından çıktı ya da çıktılar hesaplanır. Bu veri setindeki her bir satırı ya da veriyi sisteme verdiğimizde alacağımız değer var olan Y değerlerine nasıl benzetilebilir şimdi beraber bakalım.

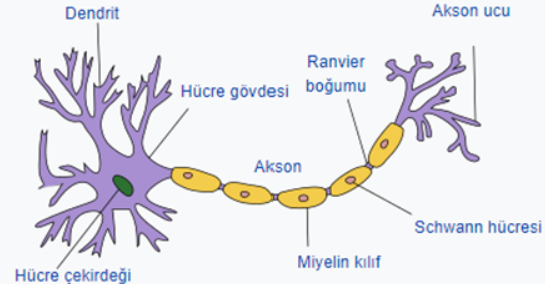


Bir defa veri setine dikkatli bakarsak Y=1 değerinin döndüğü durumlarda X değerlerinden en az 1 tanesinin 1 olduğunu görülür. Bunun gibi kuralları bizim kurabildiğimiz gibi acaba makine de bu kuralları kurabilir mi diye inceleyelim.

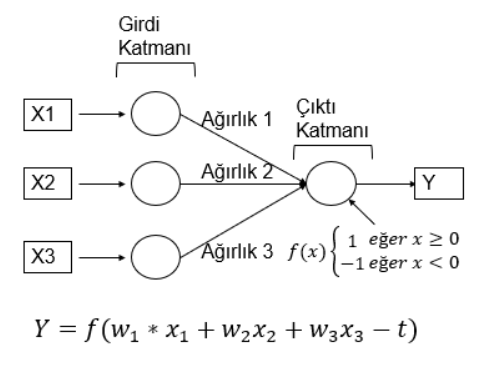
Şimdi ise Ysa modellerinden bahsedeceğiz.

**Ysa Modelleri**

YSA modelleri insan sinir hücrelerinin yapısı taklit edilerek ortaya çıkmıştır. Biraz biyoloji bilgilerimizi tazeleyecek olursak. Sinir hücreleri ya da başka ifade ile nöronların gövdesinde aksonlar bulunur. Aksonlar nöronlar arasındaki bilginin aktarılmasını sağlar. 2 nöron birbirlerine Dendritler ile bağlanır. Dentrit ile akson arasındaki bağlantı noktasında sinapslar vardır. Sinir bilimciler insandaki öğrenmenin bu sinapslardaki enerji değişimlerinden kaynaklandığını keşfetmişler. İnsan vücudunda bu nöronlardan milyarlarca var ve sürekli sinapslar arasında enerji değişimleri oluyor ve öğrenme olayı sinir hücrelerinde gerçekleşiyor.

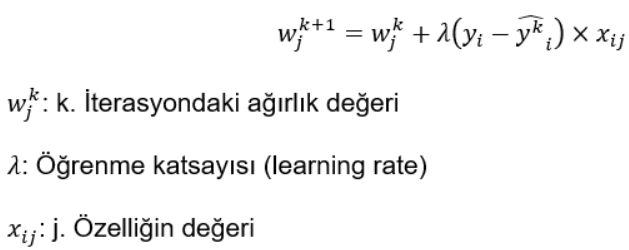


Yapay sinir hücresine bakacak olursak basit bir perceptron yani girdi ve çıktıdan oluşan bir sinir hücresi şekildeki gibi gösterilebilir. YSA’da az önce bahsettiğmiz sinapslar YSA’da ağırlıklar olarak isimlendirilir ve öğrenmeden kasıt bu ağırlıkların ve parametrelerin belirlenmesidir. Girdi katmanındaki her bir sinir hücresine nöron ya da birim(unit) denir. Girdi katmanında bulunan X değerleri bir ağırlık alarak Y değişkenine katkı sunar. Girdi katmanında özellik sayısı kadar birim ya da nöron bulunur. Çıktı katmanında ise hedef değerimizin alacağı değerler hesaplanır. Az önceki örneğimizde çıktı değişkenlerimiz sadece 1 ve -1 değeri aldığı için aşağıdaki gibi bir fonksiyon belirlememiz gerekir. Bu fonksiyon çıkan değer eğer negatif ise -1, pozitif ise 1 değerini dönderir. Çıktı denklemini yazacak olursak, her bir girdi değerimizi bir ağırlıkla çarpıp çıktı katmanındaki bias(ek bir girdi) değerinden çıkararak sonuca ulaşmış oluruz.



Örneğin x\_1 = 1, x2=1, x3=0 olursa ve tüm ağırlık değerleri 0,3 ek girdi bias değerimiz 0,4 olursa y değerimiz 1\*0,3 + 1\*0,3 +0\*0,3–0,4 = 0,2 yani pozitif çıkar. Fonksiyona bu değeri verdiğimizde çıktımız +1 olarak verilecek. Burada kullanılan fonksiyona işaret fonksiyonu diyebiliriz. Çünkü çıktının işaretini belirtiyor. Aslında başka bir ifade ile aktivasyon fonksiyonu da denilebilir. Yani sistemin aktive edilmesini sağlayan bir fonksiyon görevini ifa etmekte.

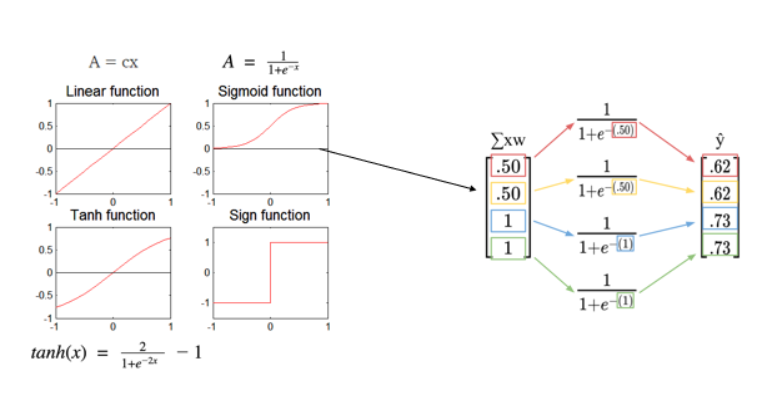
Öğrenme işlemi bahsettiğimiz bu ağırlıkların ve bias değerlerinin değiştirilmesi ile gerçekleştirilir ve öğrenmenin iyi bir hale gelmesine kadar devam eder. Peki ağırlıklar nasıl değişiyor? Bunun için şekildeki formülü inceleyelim.



Yeni ağırlık değeri bir önceki ağırlık değeri ile öğrenme oranının öğrenme kalitesi ile çarpılması ile elde edilen değerle toplanarak bulunuyor. Y-y\_şapka değeri öğrenmenin hatasını gösteriyor. Bu değer ne kadar düşük olursa ağırlık değeri bir önceki iterasyondaki ağırlık değerine o kadar yakın olur. Ne kadar büyük olursa o kadar uzak olur. Peki ya 0 olursa? İşte o zaman da bir önceki ağırlık değeri değişmeden kalır. Öğrenme katsayısı da sabit bir parametre olarak belirlenir. 0 ile 1 arasında bir değer alır. Her bir iterasyondaki ayarlamaları bu parametre ile kontrol ederiz. Eğer öğrenme oranı 1'e yakın olursa yeni ağırlık değeri mevcut iterasyondaki ayarlamaya daha yakın olmaya çalışır. Eğer 0a yakın olursa yine ağırlık değeri bir önceki iterasyondaki değere daha yakın değer alma hassasiyetindedir deriz.

**Aktivasyon Fonksiyonları**

Aktivasyon fonksiyonları öğrenme aşamasında çok önemli bir yer tutar. Eğer bu fonksiyonlar olmasaydı ağırlıklar ile çarptığımız ve bias ile çıkardığımız değeri çıktı olarak almamız gerekirdi. Bu değer de - sonsuz ile + sonsuz arasında değişir. Ancak bizim çıktı değerimiz örneğimizde ya -1 ya da +1. Bu nedenle bir aktivasyon fonksiyonuna ihtiyaç duyuyoruz. Bu örnekte kullandığımız aktivasyon fonksiyonu işaret ya da sign olarak geçen fonksiyon, basitçe negatif değerlere -1 pozitif değerlere +1 değeri dönderir. Eşik değeri de 0 noktasıdır. Tabi bu işaret fonksiyonu 2 değerin olduğu durumlarda işe yarayabilir ancak 2'den fazla sınıfın olduğu durumlarda bu fonksiyon işe yaramaz. Varsayalım üçüncü sınıfımız da +2 olsun. Bu durumda aktivasyon fonksiyonu pozitif değerde +1'i mi yoksa +2'yi mi aktive edeceğini bilemez. Bu duruma aktivasyon fonksiyonu kullanarak çözüm getirilebilir.



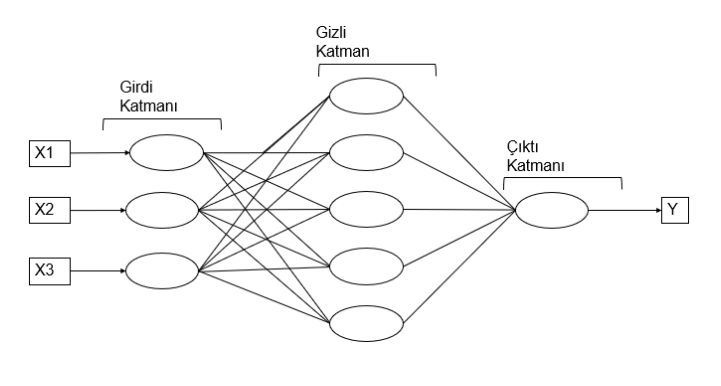
Görselde 4 adet aktivasyon fonksiyonu var. Birincisi lineer fonksiyondur. Bu fonksiyon ağırlıklarla toplanmış değeri alarak belirli bir katsayıyla çarpar ve gönderir. Yani f(x) = ax + b gibi bir fonksiyondur. Bu fonksiyonu kullanarak yine bir eşik değeri belirleriz ve eşik değerinin üstüne ve altına farklı çıktı sınıflarını atayabiliriz.

2. grafikte de sigmoid fonksiyonu var. Bu fonksiyon da sürekli bir fonksiyondur. - sonsuz ile + sonsuz arasındaki değerleri 0–1 aralığına getirir. Modelimize bağlı olarak kullanabiliriz ki yaygın bir şekilde bu fonksiyon aktivasyon fonksiyonu olarak kullanılır.

Diğer grafikte tanh fonksiyonu var bu da sigmoid fonksiyonuna benzer ancak farklı olarak verileri -1 +1 aralığında döndürür.

Bunlar gibi başka aktivasyon fonksiyonları da vardır. Aktivasyon fonksiyonları geri beslemede türev işlemleri ile hesaplamalar yapılacağı için türevi kolay alınabilen fonksiyonlar arasından seçilir. Daha fazla aktivasyon fonksiyonu örneği verebilir örneğin Relu aktivasyon fonksiyonu vardır ve özellikle derin öğrenmede çokça kullanılır.

**Çok Katmanlı YSA**

****

Bir önceki sinir ağı yapısında perceptronu gördük. Sadece bir çıktı ve bir girdi katmanlarından oluşuyordu. Ancak YSA modelleri bu basit yapıdan daha karmaşık şekilde çalışır. Bu karmaşıklık araya gizli katmanlar eklenerek artırılır. Şekilde çok katmanlı bir YSA yapısını görüyorsunuz. İleri beslemeli bir modelde bilgiler önce girdi katmanına oradan gizli katmana en son da çıktı katmanına geçer. İleri ve geri beslemeli bir modelde çıktı katmanından sonra tekrar girdi katmanına doğru geri besleme işlemi gerçekleştirilir. Gizli katmanda da aktivasyon fonksiyonu kullanılır. Az önce bahsettiğimiz aktivasyon fonksiyonlardan birisini kullanarak gizli katmandaki birimlerin aktive edilmesini veya edilmemesini sağlayabiliriz. İşte bu çok katmanlı yapı YSA’nın lineer olmayan ilişkileri de ortaya çıkarmasını sağlamaktadır. Yani diyebiliriz ki YSA aslında gücünü bu katmanların fazlalığından almaktadır. Derin öğrenme konusunu incelerseniz buradaki yapının benzerinin orada da geçerli olduğunu görürsünüz. İşte gizli katman sayısı derin öğrenmede birden fazla olur. Ayrıca nöronların öğrenme şekillerinde de farklı davranışlar söz konusu olabilir. Derin öğrenme de YSA’nın bir alt konusu ancak biraz ileri seviyesi diyebiliriz diyelim şimdilik derin öğrenme ile ilgili.

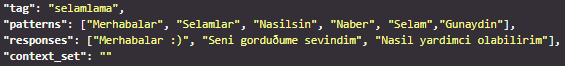
1. **Yöntem**

Bot, kendi başına ve bir insanın yardımı olmadan bir dizi görevi gerçekleştirmek için tasarlanmış yapay bir zeka yazılımıdır. Bir botun yapabileceği görevler, bir restoranda rezervasyon yapmak, takvimde bir tarih işaretlemek veya kullanıcılarına bilgi toplamak ve görüntülemek ve kullanıcıyı hava durumu hakkında bilgilendirmek gibi şeylerden farklı olabilir.

Chatbotları 2 kısıma ayırabiliriz:

1. ***Kural Tabanlı Yaklaşıma sahip olanlar:*** Chatbotlar eğitildiği bazı kurallara göre soruları cevaplar. Tanımlanan kurallar çok basit veya çok karmaşık olabilir,farklı senaryolar için kural yazmak çok zaman alır ve olası her senaryo için kural yazmak imkansızdır. Botlar basit sorguları işleyebilir ancak karmaşık sorguları yönetemez.
2. ***Kendi kendine öğrenilebilir botlar:*** Kural tabanlı botlardan daha verimli olmalarını sağlayan bazı Makine Öğrenmesi tabanlı yaklaşımları kullanan botlardır. Çok fazla veriye ihtiyaçları vardır. Kodlayacağımız bot melez bir versiyon olacaktır.

Proje için bir veri setine ihtiyaç duyulmaktadır. Veri seti json formatında hazırlanmıştır. Veri seti içeriği şu şekildedir:



Yukarıda json dosyasından örnek bir bölüm gösterilmektedir. Burada;

* Tag,
* Patterns,
* Responses,

şeklinde değerlerimiz tanımlanmıştır. Tag, “patterns” kısmındaki soru ya da cümlelerle alakalı bir anahtar kelime tutmaktadır. Patterns, kullanıcının girdiği cümleler ya da girdiği cümlelere yakın cümleler içermektedir. Responses, girilen cümlelere karşı olası cevaplardır. Bu cevaplar içerisinden random şekilde cevap çekilmektedir.

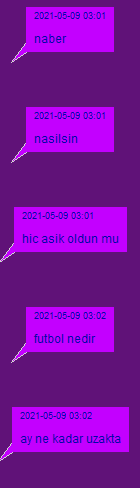
Projemiz python yazılım dili ile yazılmıştır. AI projeleri, geleneksel yazılım projelerinden farklıdır. Farklılıklar teknoloji yığınında, yapay zeka tabanlı bir proje için gerekli becerilerde ve derin araştırma gerekliliğinde yatmaktadır. Yapay zeka hedeflerinizi uygulamak için kararlı, esnek ve mevcut araçları olan bir programlama dili kullanmalısınız. Python tüm bunları sunuyor, bu yüzden bugün çok sayıda Python AI projesi görüyoruz.

Python, geliştirmeden devreye alma ve bakıma kadar geliştiricilerin geliştirdikleri yazılım konusunda üretken ve kendinden emin olmalarına yardımcı olur. Python'u makine öğrenimi ve yapay zeka tabanlı projeler için en uygun yapan avantajlar arasında basitlik ve tutarlılık, yapay zeka ve makine öğrenimi (ML) için harika kitaplıklara ve çerçevelere erişim, esneklik, platform bağımsızlığı ve geniş bir topluluk yer alır. Bunlar, dilin genel popülaritesine katkıda bulunur. Bizim de bu projede python seçme sebebimiz sıralanan sebeplere dayanmaktadır.

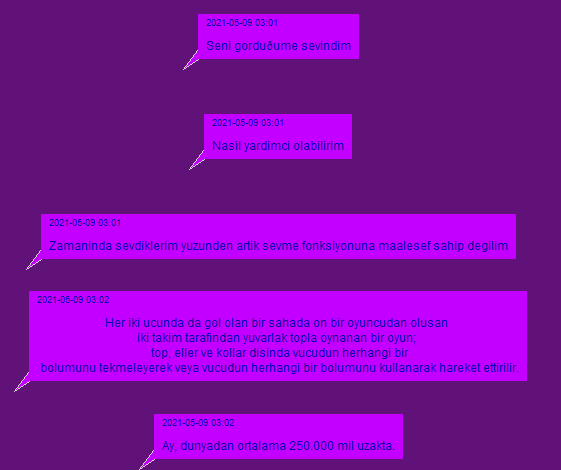
# Örnek Sonuçlar

Projemizde program çalıştığında kullanıcının göreceği arayüz: 

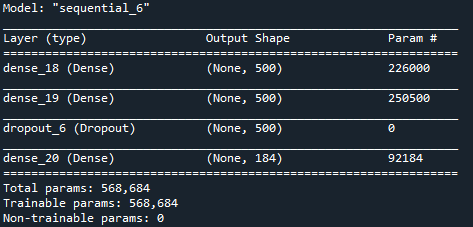
Chatbotumuzun ana ekranı bu şekildedir.



Chatbot içerisinde kullanıcının girdiği cümleler/sorular sohbet ekranında sırasıyla ortaya çıkıyor.



Sorulara verilen sıra ile cevaplar bu şekilde ekranın sol tarafında sırasıyla üretiliyor.



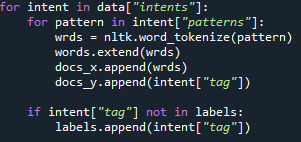
Terminalde eğiteceğimiz verinin katmanları ve boyutları bu şekilde yazdırılıyor. Toplam ve eğitilebilir parametreler karşılık değerleriyle bastırılıyor.



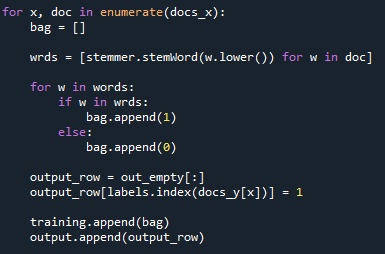
Eğitim Tur Sayısı yani Epoch adetimiz toplam 300 adet. Batch yani girdi sayımız ile doğru orantıda arttırılmıştır. Accuracy yani başarı tutarlılığımız %99 değerlerine kadar eğittiğimiz veri ile yükselebiliyor.



Veri dosyamızı bu şekilde okuyup içeriğini bir değişkene atıyoruz.



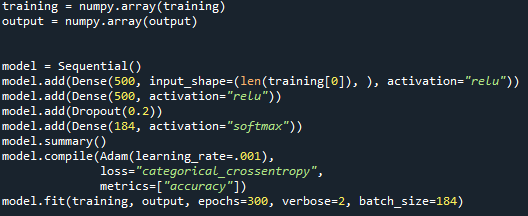
Elde ettiğimiz verinin içerisinde etikete(tag) sorulacak sorulara(pattern) göre cevap verebilmek için verileri işlemek üzere liste değişkenlerimize atıyoruz. Tabi hemen öncesinde kelimeleri ayırmak için nltk ile word tokenize yapıyoruz.



Türkçe stemmerimiz ile (Stemmer kelime köklerini elde etmek için kullandığımız fonksiyon)

Elimizdeki türkçe kelimelerin köklerini eğitmek üzere ayırıyoruz ve listemize ekliyoruz.

Daha sonra bu listeyi eğitilmek üzere training listimize ekliyoruz.



Bu verileri numpy kütüphanesi yardımıyla dizilere dönüştürdükten sonra keras kütüphanesinin Sequential fonksiyonu ile modelimizi oluşturuyoruz.

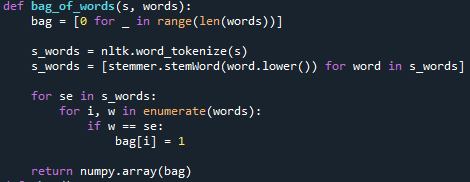
Eğitilmek üzere train edilecek verimizin uzunluğu boyutunda 500 unit(birim)lik 2 adet katman oluşturuyoruz. Ardından bir Dropout(Düğüm Seyreltme) katmanı ile eğitmede işimize yaramayacak node(düğüm)leri işlemden düşüyoruz. (0.2 Oranında)

Modelimizi Summary ile özetleyecek fonksiyonu çağırıyoruz ve terminalde görüntülüyoruz.

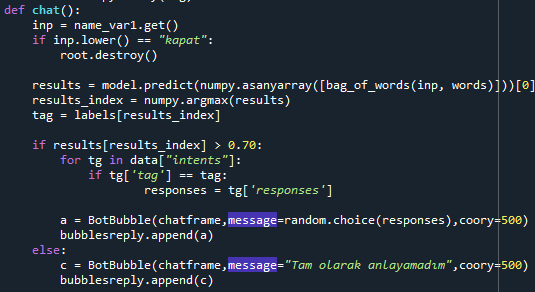
Modelimizi Adam optimizerı ile optimize ediyor ve learning\_rate(Öğrenme Oranını) .001e atıyoruz.

Loss(Kayıp) ve Accuracy(Başarı Oranı) metricleri ile optimize ediyoruz.

model.fit Modeli sabit sayıda eğitim tur sayısı(epoch) için eğitir (Veri kümesindeki her yineleme(iteration) için).



Kullanıcının girdiği cümle/soruyu ve elimizdeki kelime verilerini gönderip. Burada kelimelere word tokenize ile ayırıp stemmer ile köklerini alıyoruz. Ardından kullanıcı girdisi olan kelimeler ile elimizdeki kelime verileri için bir binary dizisi oluşturuyoruz. Kelimelerin karşılıkları için diziye tamamen 0 atıp eşleşen harfler için 1 verisi atıyoruz. Ardından kelimeler 0 1 şeklinde atamasının yapıldığı veriyi bag listesine kaydedip numpy ile dizi şeklinde return ediyoruz.



Chatbotun chat kısmı için fit edilmiş modelimizin sonuçlarını predict(Tahmin) fonksiyonu ile tahmin ediyoruz. Argümanlarımızı bag of words fonksiyonuna gönderip return edilmiş olan kelime binary yapısını alıyoruz. bu sonucu model.predict methodu ile çalıştırdıktan sonra results değişkenine atıyoruz. Sonra numpy kütüphanesi ile resultsindex değişkenine gönderip eşleşen indexlere bakıyor ve tag değişkeni label indexine göre atamasını yapıyoruz. Eğer elimizdeki sonucun indexi %70 oranından fazla ise kullanıcının girdiği kelimeler ile taglerdeki kelimeleri eşliyoruz. Bu koşullar sağlanırsa veri içerisinde gezilerek o tag’e ait olan sorulardan biri olduğunu varsayarak o soruların karşılığı olan cevaplardan bir tanesini rastgele bir şekilde seçerek ana ekran üzerinde gösteriyoruz.

# Karşılaşılan Problemler ve Çözümler

* Kullanılan kütüphanelerde versiyon uyumsuzluğu yaşandı. Kütüphaneler dökümantasyonlarından uyumlu versiyonlar belirlenerek kütüphane versiyonları değiştirilince düzeldi.
* Projenin verisi için ingilizce chatbot sohbet verileri bulduk. Türkçeye sırasıyla çevirip fazladan cevaplar ekledik. Verilerin işlenmesi için gerekli olan dense unit değerleri dropout değeri ve learning\_ratein belirlenmesinde problem yaşadık. Deneme yanılma ile doğru değerleri bulduk.
* Arayüz ekranında kullanıcı girdisi ile botun verdiği cevapların sohbet ekranında sırasıyla hareket etmesi bize problem çıkardı. Sohbet ekranını Canvas yapısına çevirdikten sonra her gelen mesajdan sonra Canvasımızı belli bir birim yukarı kaydırarak sorunu çözdük.

# Kaynakça

1. R. Lippmann, “An introduction to computing with neural nets”, IEEE ASSP Magazine, c. 4, sy 2, ss. 4-22, Nis. 1987, doi: 10.1109/MASSP.1987.1165576.
2. A. K. Jain, J. Mao, ve K. M. Mohiuddin, “Artificial neural networks: a tutorial”, Computer, c. 29, sy 3, ss. 31-44, Mar. 1996, doi: 10.1109/2.485891.
3. K. Öztürk ve M. E. Şahin, “Yapay Sinir Ağları ve Yapay Zekâ’ya Genel Bir Bakış”, s. 16.
4. J. Zupan, “Introduction to Artificial Neural Network (ANN) Methods: What They Are and How to Use Them”, Acta Chimica Slovenica, c. 41, Oca. 1994.
5. T. Kohonen, “An introduction to neural computing”, Neural Networks, c. 1, sy 1, ss. 3-16, Oca. 1988, doi: 10.1016/0893-6080(88)90020-2.
6. J. Bishop ve R. J. Mitchell, “Neural networks-an introduction”, Şub. 1991, s. 1/1-1/3.
7. [J. H. Lee, H. Yang, D. Shin, ve H. Kim, “Chatbots”, ELT Journal, c. 74, sy 3, ss. 338-344, Ağu. 2020, doi: 10.1093/elt/ccaa035.
8. C. Erden, “Giriş Seviyesinde Yapay Sinir Ağları”, Medium, Nis. 15, 2020. https://cerden.medium.com/giri%C5%9F-seviyesinde-yapay-sinir-a%C4%9Flar%C4%B1-2a76ba31b345 (erişim May. 09, 2021).
9. nilgunsengoz, “Yapay Sinir Ağları”, Derin Öğrenme | Deep Learning, Mar. 03, 2017. https://www.derinogrenme.com/2017/03/04/yapay-sinir-aglari/ (erişim May. 09, 2021).
10. Akküçük, Ulaş. “Veri madenciliği: kümeleme ve sınıflama algoritmaları”. İstanbul: Yalın Yayıncılık 18 (2011).
11. Han, Jiawei, Jian Pei, ve Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011
12. Kantardzic, Mehmed. Data mining: concepts, models, methods, and algorithms. John Wiley & Sons, 2011..
13. Sumathi, Sai, ve S. N. Sivanandam. Introduction to data mining and its applications. C. 29. Springer, 2006.
14. Tan, Pang-Ning, Michael Steinbach, ve Vipin Kumar. Introduction to data mining. Pearson Education India, 2016.
15. VanderPlas, Jake. Python Data Science Handbook. OReilly Media. Inc, 2017. •Wilson, Aidan. “Inroduction to Neural Networks in Python”. Medium, 24 Ekim 2019.
16. Towards Data Science. “Towards Data Science”. Erişim 29 Mart 2020.
17. “Understanding Activation Functions in Neural Networks”. Erişim 15 Nisan 2020.
18. J.Feldman, M.A. Fanty, and N.H. Goddard, “Computingwith Structured Neural Networks,” Computer, Vol. 21, No. 3, Mar. 1988, pp. 91-103
19. J. Hertz, A. Krogh, and R.G. Palmer, Introduction to the “he- oryofNeural Computation,Addison-Wesley,Reading, Mass., 1991.
20. D.E. Rumelhart and J.L. McClelland, ParallelDistributedRocessing: Exploration in the Microstructure of Cognition, MIT Press, Cambridge, Mass., 1986.