**T.C.**

**FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİYOMEDİKAL MÜHENDİSLİĞİ**

**KENDİ ÖĞRENEN ÇARPIM TABLOSU ALGORİTMASI**



**HAZIRLAYAN:**

**Ömer Turgut Uysal**

**DANIŞMAN:**

**Yrd. Doç. Dr. Ebubekir KOÇ**

**Matlab ile Kendi Öğrenen Çarpım Tablosu**

# Ömer Turgut Uysal

Mühendislik Fakültesi, Fatih Sultan Mehmet Vakıf Üniversitesi, Beyoğlu, İstanbul, o.t.uysal@hotmail.com **1.GİRİŞ**

Bu çalışmada, program input değeri olarak bazı çarpım tablosu örnekleri verilmektedir. Daha sonrasında program başka test işlemrini kendi kendine örnek datalardan belirlediği algoritmalar sayesinde kendi kendine yapabilmektedir. Optimize etme sayısı ne kadar fazla ise işlem sonuçlarının o kadar gerçek değere yakın olduğu gözlemlenmiştir.

Matlab programının bazı hazır toolları kullanılarak gerçekleştirilen bu program değişik fikirlerle oluşturulabilir. Yani bu programdaki öğrenme olayı verilen örneklemler ile kendi kendine bir yol oluşturmaktadır.

**Peki Derin Öğrenme (Deep Learning) Nedir?**

Endüstri ve akademik çevrelerdeki veri bilimciler görüntü sınıflandırma, video analizi, konuşma tanıma ve doğal dil öğrenme süreci dahil olmak üzere çeşitli uygulamalarda çığır açan gelişmeler elde etmek üzere makineyle öğrenmede GPU’ları ([Grafik İşlemci Ünitesi](https://tr.wikipedia.org/wiki/Grafik_%C4%B0%C5%9Flemci_%C3%9Cnitesi)) kullanmaktadır. Özellikle, büyük miktarlarda etiketlenmiş eğitim verilerinden özellik saptama yapabilen sistemler oluşturmak için ileri teknoloji, çok seviyeli “derin” sinir ağların kullanılması olan Derin Öğrenme, önemli derecede yatırım ve araştırmanın yapıldığı bir alandır.

Makineyle öğrenme yıllardır kullanılan bir yöntem olmasına rağmen, iki yeni yeni trend makineyle öğrenmenin yaygın bir şekilde kullanılmasına yol açmıştır: çok büyük miktarlarda eğitim verisi ile GPU hesaplama ile elde edilen güçlü ve verimli paralel hesaplama. GPU’lar, çok daha büyük eğitim setleri kullanarak bu derin nöral ağları çok daha kısa sürelerde ve çok daha az veri merkezi altyapısı kullanarak eğitmek için kullanılmaktadır. GPU’lar aynı zamanda, çok daha fazla veri hacmi ve daha az güç ve altyapı destekleyerek, bulut içinde sınıflandırma ve tahmin yapmak için bu eğitilmiş makineyle öğrenme modellerini çalıştırmak için kullanılmaktadır.

Makineyle öğrenme için GPU’ları kullanmaya ilk başlayanlar arasında en büyük web ve sosyal medya şirketlerinin yanı sıra, veri bilimi ve makineyle öğrenme alanında çalışan üst düzey araştırma kuruluşları bulunmaktadır. Binlerce hesaplama çekirdeği ve tek başına çalıştırılan CPU’lar ([Merkezi İşlem Birimi](https://tr.wikipedia.org/wiki/Merkezi_i%C5%9Flem_birimi)) ile karşılaştırıldığında 10 ile 100 kat uygulama performansı sunan GPU’lar, veri bilimcilerin büyük verilerin işlenmesinde tercih ettikleri işlemci olmuştur.

GPU’lar ile önceden kaydedilen konuşmalar veya multimedya içerikleri çok daha hızlı bir şekilde yazıya geçirebilmektedir. Carnegie Mellon Üniversitesi’nden Profesör Ian Lane yürüttüğü çalışmalarında CPU uygulaması ile karşılaştırıldığında, GPU’ların 33 kata kadar daha hızlı tanıma yaptığını ortaya koymuştur.

**Yapay Zekanın Ortaya Çıkması ve Uygulamaları**

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;**

**1. Bulanık Mantık (BM):** Bulanık mantık, bulanık eseme ya da puslu mantık, 1961 yılında Lütfü Aliaskerzade'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):** İnsan beynindeki sinirleri baz alınarak geliştirilmiştir. Daha önce gerçekleştirmiş olduğum Fabrikada üretim sırasında gerçekleşebilecek hataları en aza indirme tasarımında Genetik Algoritma ve Yapay Zeka Algoritmalarını kullanmıştım. Gerçekleştirmek istediğim bu projemde ise Yapay Sinir Ağlarını Kullandım.

**Makine Öğrenmesi Nedir?**

Son dönemde adını sıkça duyduğumuz Makine Öğrenmesini mümkün olduğunca basit halde sizlere anlatmaya çalışacağım. [Makine Öğrenmesi](https://tr.wikipedia.org/wiki/Makine_%C3%B6%C4%9Frenimi) (Machine Learning), matematiksel ve istatistiksel yöntemler kullanarak mevcut verilerden çıkarımlar yapan, bu çıkarımlarla bilinmeyene dair tahminlerde bulunan yöntem paradigmasıdır. Makine öğrenmesine güncel hayatımızdan bazı örnekler: yüz tanıma, belge sınıflandırma, spam tespiti.

Bir benzetme yapacak olursak öğrenci konu çalışıp ders kitabındaki örnek çözümlü soruları çözer ve öğrenir. Ardından kitapta bulunmayan ama aynı bilgiye dayanan farklı bir test önüne konulur. Öğrenci cevapları bilmeden testi çözer. Sonra değerlendirmeye alınır, ne kadar başarılı olduğu görülür. Eğer ezberci bir öğrenciyse benzeri kitapta olmayan soruları muhtemelen yanlış yanıtlayacaktır. Eğer işin temelini anlayan bir öğrenciyse farklı tarzda soru gelse bile doğru çözebilecektir. Öğrenci test sonucunu ve nerelerde hata yaptığını inceler. "Hmm şu soruda şöyle bir genelleme yapmışım ama aslında iş bu kadar basit değilmiş", "Hmm şu etkeni hesaba katmayı düşünemedim" gibi dersler çıkarır kendisine. Ardından derslerini almış bir gözle kitabını tekrar çalışır ve tekrar testi çözer. Yeterince iyi sonuç alana kadar hatalarını keşfedip konunun püf noktalarını öğrenmeye çalışır.

Makine öğrenmesi de bunun çok benzeridir. Süreç aynıdır. Öncelikle terminolojiden bahsedelim. Ardından en temel 2 öğrenme türünü tanıyalım ve örnek bir makine öğrenme sürecini anlayalım.

**Terminoloji**

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

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

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

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

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

**İki Temel Öğrenme Türü**

**Gözetimli Öğrenme**

Etiketlenmiş gözlemlerden öğrenme sürecidir. Etiketler, algoritmaya gözlemleri nasıl etiketlemesi gerektiğini öğretir. Örneğin içinde "para kazan" ifadesi geçiyorsa spam demelisin gibi yol göstermelerde bulunur.

**Sınıflandırma (Classification):** Her bir gözleme bir kategori/sınıf atması yapar: *Örn: spam/spam değil*. Sınıflar ayrıktır (sayı değildir) ve birbirlerine yakın/uzak olmaları gibi bir durum söz konusu değildir.

**Regresyon (Regression):** Her gözlem için öğrendiklerine bakarak reel bir değer tahmini yapar. *Örn: "2011 model 40.000 km'de Mia Torento arabanın fiyatı 45.670 TL olmalıdır".*

**Gözetimsiz Öğrenme**

Etiketsiz gözlemlerden öğrenme sürecidir. Algoritmanın kendi kendine keşifler yapması, gözükmeyen örüntüleri keşfetmesi beklenir.

**Kümeleme (Clustering):** Gözlemleri homojen bölgelere ayırır. *Örn: bir okuldaki öğrenci gruplarını tespit etmek.*

**Boyut Azaltımı (Dimensionality Reduction):** Gözlemlerin mevcut özellik sayısını az ve öz hale indirir, bize en iyi öğrenme imkanı sunar.

**Peki nasıl öğreniyor bu algoritmalar?**

Öncelikle ham veri (raw data) buluyoruz. *Örn: Metin, görüntü, genetik verisi, sayısal ölçümler, sosyal ağlar, kullanıcı puanlamaları...*

Regresyon tipi gözetimli öğrenme üzerinden örnek verelim. Elimizde hastaların sağlık verileri olsun. Her hasta için: yaş, cinsiyet, sigara içiyor mu, günde kaç sigara içiyor, şeker hastalığı var mı, daha önce kalp krizi geçirdi mi, kolestrolü kaç, nabzı kaç, şeker seviyesi kaç?



**Özellik Çıkarımı (Feature Extraction)**

Gözlemlerimizi en iyi temsil edecek özellikleri kullanmak etmek için ham veriden Özellik Çıkarımı (Feature Extraction) ve çıkarılan özellikler üzerinde Özellik Seçimi (Feature Selection) yapıyoruz. Bu süreçte alan bilgisi (domain knowledge) uygulamak gerekiyor. Yani bu işi tecrübesiz bir insana anlatacak olsaydık hangi özelliklere bak derdik? Bu özellikleri tanımlayıp hesaplamak gerekli. Örneğin kalp hastalığı tespitinde "AnneAdı" özelliğinin pek de işe yaramayacağını öngörebiliriz. En çok şüphelendiğimiz sigara sayısı, daha önce kalp krizi geçirdi mi? gibi özellikleri hesaba katmakta fayda var. Gereksiz kısımları temizlemeliyiz.

**Model Oluşturma**

Özellikler çıktıktan sonra bir öğrenme algoritması bu matris üzerinde çalışıyor. Sonucunda bir takım kurallar çıkarıyor. Örneğin basit bir algoritma az sayıda veriye bakarak bazı şöyle bir model ortaya koyabilirdi:

Kriz olasılığı = günlük sigara sayısı x 0.05 + kolestrol seviyesi x 0.004

Ancak gözlem sayısı arttıkça daha karmaşık yorumlar yapabiliyor.

**Değerlendirme**

Elimizde artık bir modelimiz var. Şimdi test verisiyle test etmeliyiz:

Elimizde etiketi (10 yıl içinde kalp krizi geçirdi / geçirmedi) gizlenmiş hasta gözlemlerini (test verisini) modele uygulayarak her birisi için bir olasılık çıkarıyoruz. Ardından olasılık 0.5'den büyükse kriz geçireceğini iddia ediyorum diyoruz. Modelin ürettiği bu tahminleri elimizdeki gerçek kriz bilgileriyle karşılaştırıp ne kadar başarılı bir tahmin yaptığını değerlendiriyoruz.



Eşik olarak 0.5 kullandığımızda bu test verisindeki herkesin kalp krizi geçirmesini bekliyoruz ama Veysel 10 yıl içinde geçirmemiş. Eğer eşiği 0.9 deseydik Veysel'in kriz geçirmeyeceğini bilecektik ama gerçekte kriz geçiren Mert'in kriz geçireceğini iddia edemeyecektik.

**Değerlendirmede kullandığımız kıstaslar:**

**Accuracy (Doğruluk)**: *doğru bildiğimiz / toplam gözlem*. Bunu hem model tahminimiz için yapıyoruz hem de *baseline (dayanak)* denilen basit tahmin için. Dayanak, eğitim verisindeki gözlemlerin yüzde kaçı kriz geçirmiş sorusuna verdiğimiz cevap oluyor. Yani hiçbir fikrimiz olmadığında öne sürebileceğimiz en iyi cevap. Model doğruluğu, dayanak doğruluğundan ne kadar iyiyse o kadar iyi.

**Sensitivity (hassaslık)**, gerçekten kriz geçiren kimselerden yüzde kaçını "kriz geçirecek" diyerek bildik?

**Specificity (belirginlik)**, kriz geçirmeyen kimselerden yüzde kaçını "kriz geçirmeyecek" diyerek bildik?

Model oluşturma yöntemine göre bu değerleri kullanarak daha karmaşık metrikler hesaplanıp yorumlanabiliyor. 0,5 olarak kullandığımız karar katsayısını 0.95 yaparak "iddiamdan ancak çok eminsem kriz geçirecek derim" diyebiliriz, bu da hassaslığı düşürür. Çünkü kimse kriz geçirmeyecek demiş oluruz. Belirginliği artırır çünkü kriz geçirmeyecek insanları daha çok insanı bilmiş oluruz. Katsayıyı 0.3 yaparsak ise birçok kişiye potansiyel hasta muamelesi yapmış oluruz, durum tersine döner. Kalp krizi hayati bir olay olduğu için bütçe elverdiği ölçüde bu katsayıyı düşük tutmakta fayda var.

Bununla birlikte modelin karmaşıklığına göre tahmin sonuçları değişecektir. Eğer model çok basitse (az sayıda özelliğe bakarak karar veriyorsa) çok daha hızlı çalışacaktır. Eğer iyi genelleme yapabiliyorsa farklı veri kümelerinde iyi tahminler yapabilir. Ama az miktarda veriye bakarak genelleme yaptıysa bu genelleme farklı veri kümelerinde yanlış tahminlere yöneltecektir.

Bunu engellemek için modelin karmaşıklığını artırırsak bu sefer eğitim kümesini ezberleme durumu ortaya çıkar. Yani eğer hasta 53 yaşındaysa ve günde 9 sigara içip 45.9 kolestrole sahipse ve ... .. kalp krizi geçirir gibi ezber kokan yanıtlar vermeye başlar. Bu modeli eğitim kümesinde test edecek olursak %100'e yaklaşan başarı gösterir. Ama 53 yaşında 9 sigara içen ve 50 kolestrole sahip birisi hakkında doğru yorum yapamaz. Çünkü ezberlediği bilgiler arasında bu gözlem yoktur.

Modelin karmaşıklığı iyi ayarlanmalıdır. Ne eğitim verisini ezberlemeli ne de çok basit genellemeler yapmalıdır. Mümkün olduğunca (fazla değil) basit olmalıdır.

**İyileştirme**

Eğer modelin yeterince başarılı olmadığını düşünüyorsak nerelerde hata yaptığını inceleyip modelimize hangi özellikleri vermemz gerektiğini düşünmeliyiz. Bazı özellikleri çıkarmalı, bazı özellikleri eklemeli, yeniden bir model oluşturup tekrar değerlendirmeliyiz. Bu süreç tatmin olana kadar devam edebilir ama sonu yoktur.

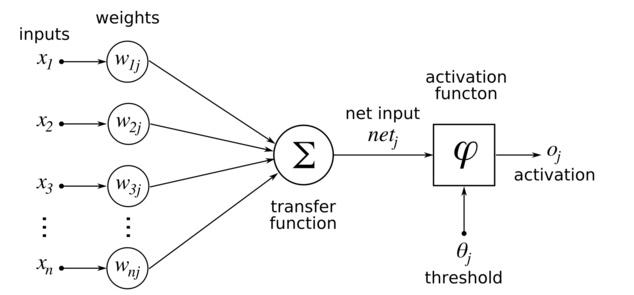
Örneğin ilk modelimizin yeterli olmadığını gördüğümüzde "bir de daha önce kalp krizi geçirmiş mi" özelliğine bakalım deyip ayrı bir model oluşturup onu test edip daha başarılı olduğunu görebiliriz. Modelin performansı, sağlamlığı konusunda çeşitli metrikler mevcuttur.

**Tahmin**

Eğer modelimizin test kümesinde başarılı şekilde tahmin yürüttüğünü düşünüyorsak başka veriler üzerinde modeli çalıştırıp tahminler yapmaya, modelden faydalanmaya başlayabiliriz.

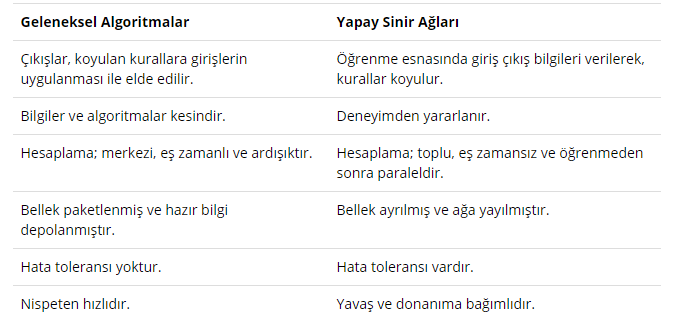
# 2.YAPAY SİNİR AĞLARI

Bilgisayar teknolojilerinin gelişmesiyle birlikte insanoğlu neredeyse tüm işlemlerini bu yenilikçi teknolojiler üzerinde yapmakta ve yeni yöntemlerin bulunmasına olanak sağlamaktadır. İşte bu yüzdendir ki, 1980’li yıllarda ortaya atılan makinanın insan gibi düşünebilmesi fikri ortaya atılmış, 1990’lı yıllara gelindiğinde ise Yapay Sinir Ağları teknolojisi iyiden iyiye hızlanmış ve büyük bir gelişme görülmüştür.



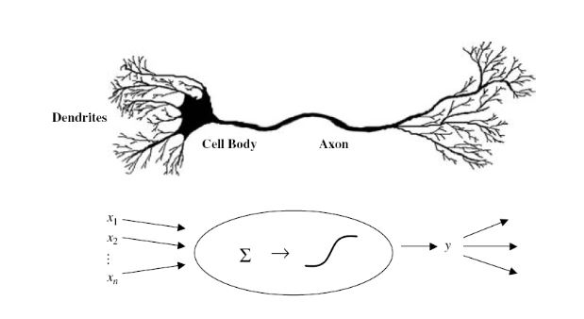
**Şekil 1**

Yapay Sinir Ağları, Yapay Zekâ kavramı altında oluşan bir alt başlık olup, bu konuya ilgi duyan araştırmacıların odak noktası haline gelmiştir. Benim projemde bu yöntemden esinlenme sebebim ise yapay sinir ağlarının karmaşık ve sonuç alınabilmesi için birçok işlem gerçekleştirilen uygulamaları kısa sürede halletmektir.



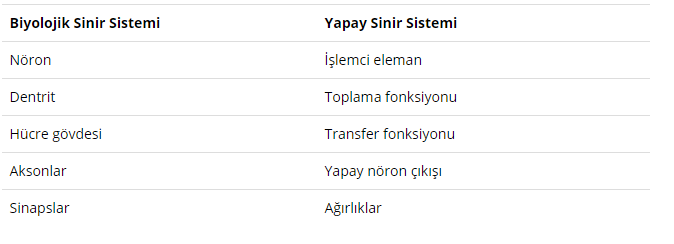
# BİYOLOJİK SİNİR HÜCRESİ

Biyolojik sinir ağlarının temel elemanları, biyolojik sinir hücreleridir. İnsan beyninin korteks kısmında yer alan sinir hücresi sayısı yaklaşık olarak 1011 olup her hücre sayısı, 1000-10000 arasında değişen başka hücrelerle karşılıklı ilişki içerisindedir. Şekil 1.’de görüldüğü gibi, bir sinir hücresinin temel elemanları hücre gövdesi, dendrit ve aksondur.



**Şekil 2**

Sinir hücresine diğer sinir hücrelerinden gelen uyarımlar, dendritler aracılığıyla hücre gövdesine taşınır ve hücre içi aktivasyonun/kararlılık halinin bozulmasıyla oluşan bir kimyasal süreç içerisinde diğer hücrelere aksonlarla iletilir; uyarımların diğer sinir hücrelerine taşınabilmesinde akson uçları ile dendritler arasındaki sinaptik boşluklar (sinaps) rol oynar. Sinaptik boşluk içinde yer alan “sinaptik kesecikler”, gelen uyarımların diğer hücrelere dendritler aracılığıyla geçmesini koşullayan elemanlardır. Sinaptik boşluğa, “sinaptik kesecikler” tarafından sağlanan nöro-iletken maddenin dolması uyarımların diğer hücrelere geçişini koşullar. Hücrelere gelen uyarımlarla uyumlu olarak hücreler arasındaki mevcut sinaptik ilişkilerin değişimi veya hücreler arasında yeni sinaptik ilişkilerin kurulması “öğrenme” sürecine karşılık gelir



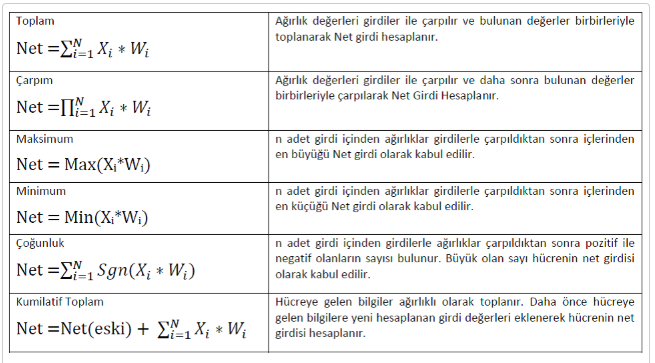
# YAPAY SİNİR YAPISI

Biyolojik sinir ağlarının sinir hücresi olduğu gibi yapay sinir ağlarının da yapay sinir hücreleri vardır. Yapay sinir hücreleri mühendislik biliminde proses elemanları olarak da adlandırılmaktadır.

Girdiler: Bir yapay hücresine dış dünyadan gelen bilgilerdir. Bunlar ağın öğrenmesini istenen örnekler tarafından belirlenir.

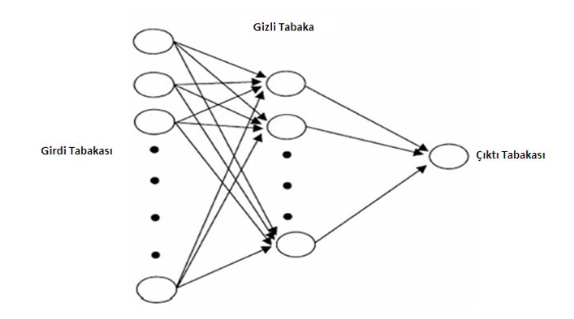
Ağırlıklar: Bir yapay hücreye gelen bilginin önemini ve hücre üzerindeki etkisini gösterir. Şekildeki ağırlık wi1, xi1 girdisinin hücre üzerindeki etkisini göstermektedir. Ağırlıkların büyük ya da küçük olması önemli veya önemsiz olduğu anlamına gelmez. Bir ağırlığın sıfır olması o ağ için en önemli olay olabilir.

Toplama Fonksiyonu (Birleştirme Fonksiyonu): Bu fonksiyon, bir hücreye gelen net girdiyi hesaplar. Bunun için değişik fonksiyonlar kullanılmaktadır. En yaygın olanı ise ağırlıklı toplamıdır. Burada her gelen bilgi kendi ağırlığı ile çarpılarak toplanır. Böylece ağa gelen net girdi bulunmuş olur. Aşağıdaki tabloda bazı toplama fonksiyonlarına yer verilmiştir.



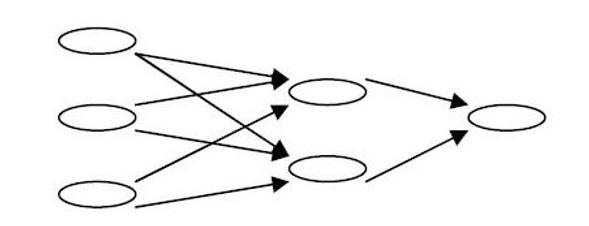
# YAPAY SİNİR AĞLARININ TEMEL BİLEŞENLERİ

# 1.MİMARİ YAPI

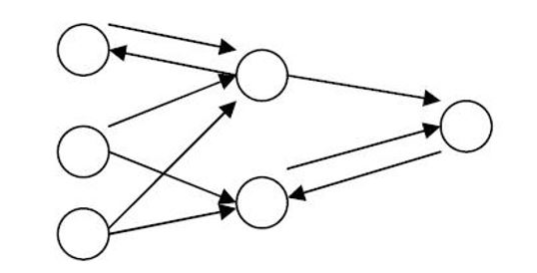


Girdi, Gizli ve Çıktı tabakalarından oluşan 3 tabakalı (ya da katmanlı) ileri beslemeli bir sinir ağı modeli görülmektedir.

* **İleri Beslemeli Yapay Sinir Ağı Modeli**



* Tek yönlü bilgi akışı söz konusudur.
* Bu ağ modelinde Girdi tabakasından alınan bilgiler Gizli katmana iletilir.
* Gizli ve Çıktı tabakalarından bilginin işlenmesi ile çıkış değeri belirlenir.
* **Geri Beslemeli Yapay Sinir Ağı Modeli**



* Bir geri beslemeli sinir ağı, çıkış ve ara katlardaki çıkışların, giriş birimlerine veya önceki ara katmanlara geri beslendiği bir ağ yapısıdır. Böylece, girişler hem ileri yönde hem de geri yönde aktarılmış olur.
* Bu çeşit Yapay Sinir Ağlarının dinamik hafızaları vardır ve bir andaki çıkış hem o andaki hem de önceki girişleri yansıtır. Bundan dolayı, özellikle önceden tahmin uygulamaları için uygundurlar. Geri beslemeli ağlar çeşitli tipteki zaman-serilerinin tahmininde oldukça başarı sağlamışlardır. Bu ağlara örnek olarak Hopfield, SOM (Self Organizing Map), Elman ve Jordan ağları verilebilir

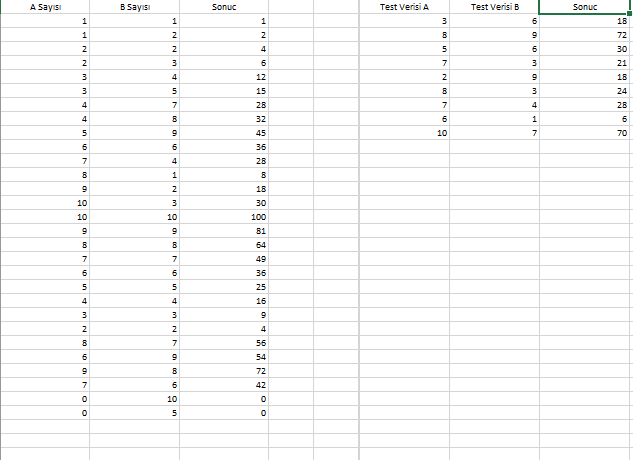
# YAPAY SİNİR AĞLARININ ÖZELLİKLERİ

* Doğrusal Olmama
* Paralel Çalışma
* Öğrenme
* Genelleme
* Hata Toleransı ve Esneklik
* Eksik Verilerle Çalışma
* Çok Sayıda Değişken ve Parametre Kullanma
* Uyarlanabilirlik

# 3.PROJENİN AŞAMALARI

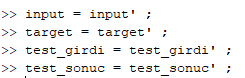
**1.Adım**

Örneklem için Excel Yardımıyla Data Listesi oluşturuldu.



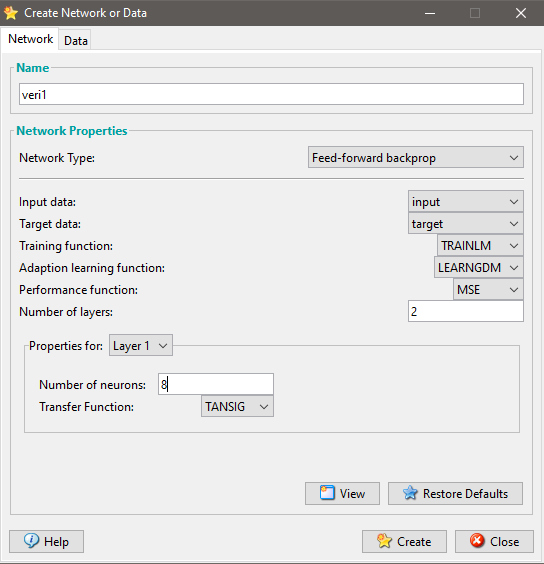
**2.Adım**

Excel yardımıyla oluşturduğumuz verileri İmport ediyoruz. Yukarıdaki tabloda görünen A Sayısı ve B Sayısı sütunlarını ‘input’, Sonuç sütununu target, Test Verisi A ve Test Verisi B sütunlarını test\_girdi, ikinci sonuç kısmını da test\_sonuc olarak tanımlıyoruz (Numeric Array Olarak). Daha sonrasında ‘nntoll’ işlemi satır mantığı ile çalıştığı için transpoze işlemini aşağıda verilen kodlar yardımı ile uyguluyoruz. Sonrasında ise nntoll fonksiyonunu kullanmaya başlıyoruz.

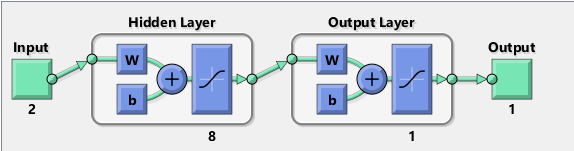


**3.Adım**

‘nntoll’ fonksiyonunu çalıştırmamız ile birlikte karşımıza Data Manager geliyor. Data Manager yardımı ile girdi ve çıktılarımızı belirliyoruz. Training Data olarak TRAINLM ve Öğrenme Fonksiyonu olarak LEARNGDM seçeneklerini seçiyoruz. Ardından nöron sayısını 8 yaparak yeni bir ağ oluşturuyoruz.

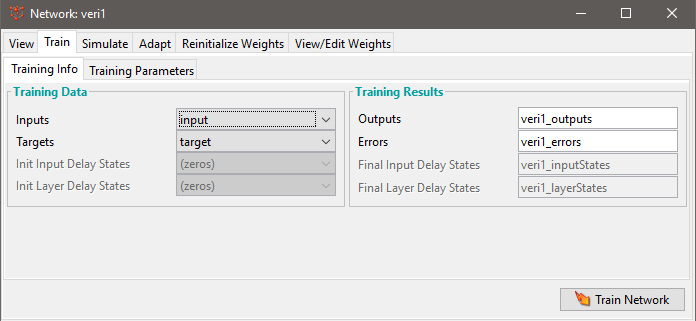


Bu işlemleri takiben ağımız aşağıda gösterilen şekildeki gibi oluşmuş oluyor.



**3.Adım**

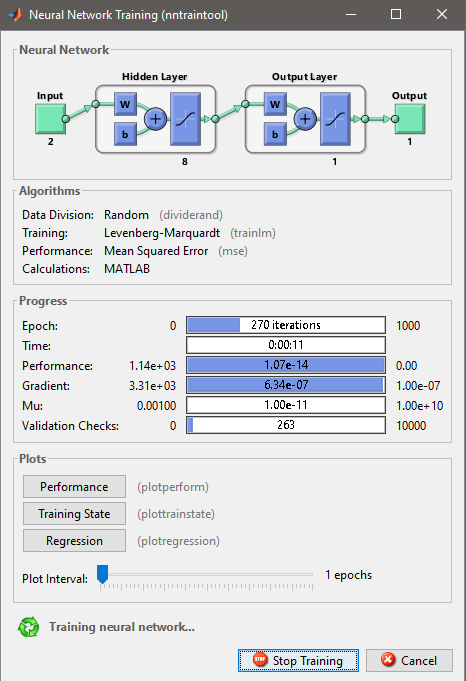
Bu aşamada oluşturulan ağı eğitme sürecine başlıyoruz. Yeniden training data kısmında input ve target verilerimizi işaretliyoruz. Buradan bize eğitim çıktıları olarak veri1\_output ve veri1\_errors olarak iki adet çıktı geliyor.



Bu işlemler gerçekleştirildikten sonra eğitim parametrelerinden iterasyon sayısını 10.000 olarak ayarlıyoruz.

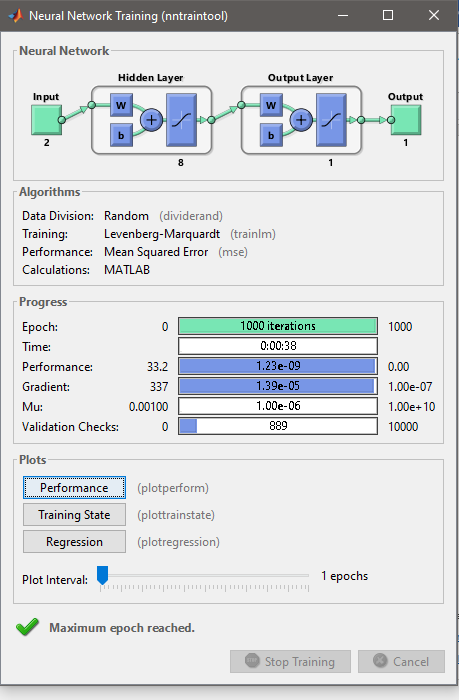
**4.Adım**

Bu aşamada ise ayarları yapılmış olan ağı eğitmeye başlıyoruz.

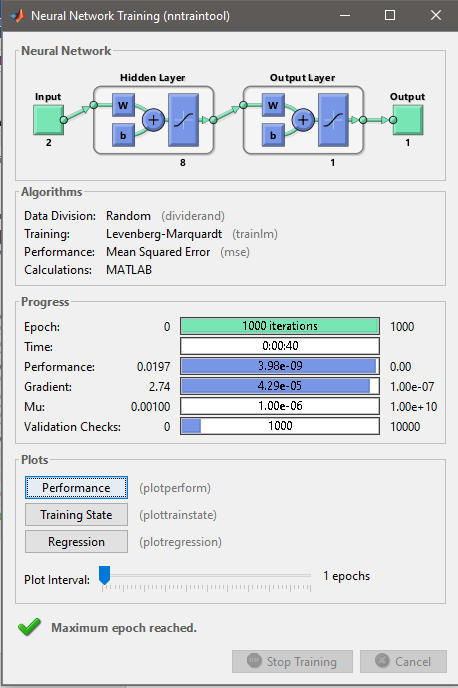


**(Şekil 1)**

**Ağın bir kere eğitilmiş hali**



**(Şekil 2)**



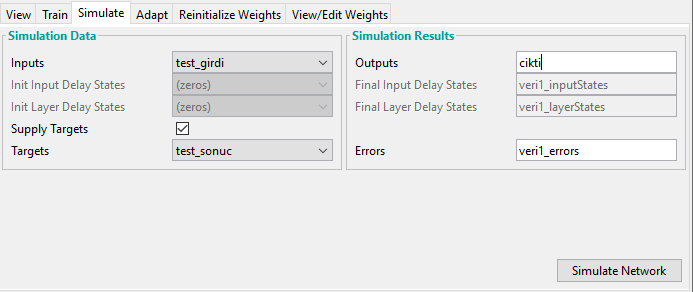
**(Şekil 3)**

**Ağın üçüncü eğitilmiş hali**

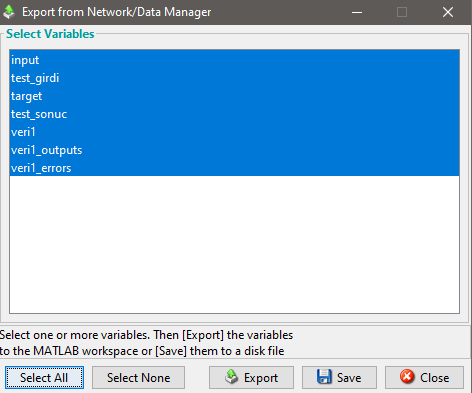
Biz bu eğitim aşamasında performans bölümündeki sayının en az seviyeye indirilmesi beklenmektedir. Yani ağ ne kadar eğitilirse o kadar performansın artması beklenmektedir.

**5.Adım**

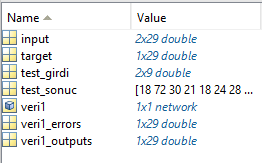
Bu aşamada ise Simulate ekranından girdi seçeneğini test\_girdi çıkışı ise sonuçlarla destekle sekmesini kullanarak test\_sonuc olarak seçiyoruz.



Daha sonrasında ise son aşama olarak bütün değişkenleri dışarı aktarıyoruz.

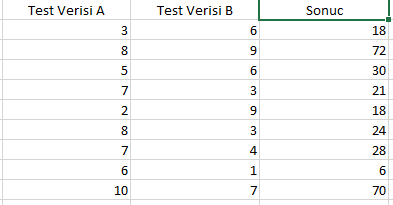


Böylelikle yapay sinir ağımız dahil tüm değişkenler workspace ekranımıza gelmiş bulunmaktadır.



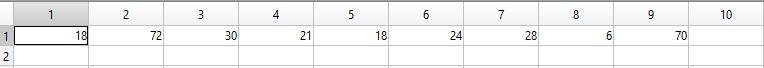
**6.Adım**

Son adım olarak ise test için belirlenen çarpım değerlerini karşılaştırıyoruz.



**(Şekil 1)**

**Oluşturulan Test Verileri**



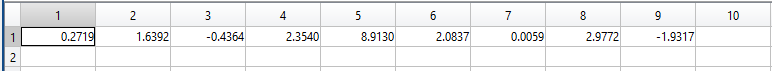
**(Şekil 2)**

**Gerçek Sonuçlar**



**(Şekil 3)**

**Programın Elde Ettiği Sonuçlar**



**(Şekil 4)**

**Sonuçlar Arasındaki Sapma**

# REFERANSLAR

Yapay Sinir Ağları İlkeleri; Zekâi Şen, Su Vakfı Yayınları

<http://www.derinogrenme.com/2017/03/04/yapay-sinir-aglari/>

<http://kod5.org/yapay-sinir-aglari-ysa-nedir/>

<https://medium.com/@yasinguzel/yapay-zeka-ders-notlar%C4%B1-03-biyolojik-sinir-sistemi-ve-yapay-sinir-a%C4%9F%C4%B1-h%C3%BCcresi-6555add68d80>

<http://www.derinogrenme.com/2015/07/21/derin-ogrenme-deep-learning-nedir/>

<https://www.emrealadag.com/makine-ogrenmesi-nedir.html>

# EKLER

<https://github.com/OmerTurgut/Nesne_Takibi>