

Makine Öğrenmesi

Temelleri

Ders Notu

Dr. Cahit Karakuş

“Elektronik sistemlerin hiçbir kişi⁺ki kişisel yeteneklerin üzerinde değildir ve kıyaslanamaz, Ckk”

İçindekiler

Özet	3
Giriş.....	4
1. Genel Kavramlar	6
2. Tarih.....	9
3. Neden Makine Öğrenmesi?	13
4. Yapay Zeka	20
5. Makine Öğrenmesinde Temel Kavramlar	29
5.1. Özelliğin (Öznitelik) Vektörleri	33
5.2. Eğitim Modelleri	35
5.3. Entropi (Bilgi Kazancı)	38
5.4. Kayıp Veri	40
5.5. Veri Madenciliği	41
5.6. Veri Tabanı Yönetimi.....	42
6. Makine Öğreniminin Türleri.....	47
6.1. Denetimli Öğrenme	52
6.2. Denetimsiz Öğrenme	55
6.3. Yarı denetimli öğrenme	57
6.4. Takviyeli Öğrenme	57
6.5. Özelliğin Öğrenme	58
6.6. Diğer Öğrenme Yöntemleri.....	59
7. Makine Öğrenmesi Algoritmaları.....	62
7.1. Kümeleme (Clustering)	68
7.2. Sınıflandırma	77
7.2.1. Karar Ağacıları.....	78
7.2.2. K-En Yakın Komşu	98
7.2.3. Destek Vektör Makineleri	123
7.2.4. Naïve Bayes.....	132
7.3. Sıra Desen Madenciliği.....	136
7.3.1. Rassal Orman Modeli Algoritması.....	138
8. Derin Öğrenme Algoritmaları	143
8.1. Anomali Tespit:	145
8.2. Yapay Sinir Ağları	146
8.3. Pekiştirmeli öğrenme	157
9. Genetik Algoritmalar	167

Özet

Bilgisayar sistemleri öğrenmeye başladılar ise, robotlar dünyayı ele geçirecekler mi? Yazılan komutlar ile robot sürüsü neler yapmaz ki? Robotlardan birşeyler yapılması isteniyorsa, kodların yazılması ve uygulamasının görülmESİ gerekir. Devasa veri yiğinları ve makine öğremesi algoritmaları ile gelecekte hangi sektörlerde organize, otonom ve gezgin makineler yoğun olarak kullanılacaktır? İzlerimizden kimliğimizin belirlenmesine izin veriyoruz. Sosyal medya ortamında paylaştığınız fotoğraflar, düşünceler; beğendiğiniz yorumların derinliğinde sizi birilerinin esiri yapacak mahremiyetler bulunmaktadır. Paylaştığınız bir fotoğrafı analiz edelim ve senaryonun sorularını hazırlayalım. Paylaştığınız bilgiler sınıflandırılarak kümelendirilmektedir. En anlamlı nokta ise, toplanan veriler ile sizin hangi sınıfı ait olduğunuzun bilinmesidir.

Makine Öğrenmesi: Deneyim yoluyla veri yiğininden otomatik olarak öğrenen ve otonom davranış sergileyen bilgisayar algoritmalarının oluşturulmasıdır.

Makine öğrenimi, bilginin deneyim yoluyla elde edildiği mekanizmaları araştırır. Makine Öğrenimi, tümevarım algoritmalarına ve `` öğrendiği '' söylenebilecek diğer algoritmala odaklanan bir alandır. Herhangi bir makine öğrenimi uygulamasında bir öğrenme modeli esastır:

- kim öğreniyor (bir bilgisayar programı)
- ne öğrenildi (bir alan)
- öğrencinin öğrendiklerinden (bilgi kaynağı)

Makine Öğreniminin yardımıyla, otonom olarak karar alabilen akıllı sistemler geliştirebilmektedir. Geliştirilen algoritmalar, istatistiksel analiz ve model eşleştirme yoluyla geçmiş veri örneklerinden öğrenir. Daha sonra öğrenilen verilere dayanarak tahmin edilen sonuçları sağlar. Veri, makine öğrenimi algoritmalarının temel omurgasıdır. Geçmiş veriler yardımıyla makine öğrenmesi algoritmaları eğitilerek daha fazla veri oluşturabilir.

Makine Öğrenimi, veri bilimi uygulamaları için büyük bir potansiyel açmıştır. Makine Öğrenimi, bilgisayar bilimi, matematik ve istatistikleri birleştirir. İstatistikler, verilerden çıkarımlar yapmak için gereklidir. Matematik, makine öğrenimi modelleri geliştirmek için yararlıdır ve son olarak, algoritmaları uygulamak için bilgisayar bilimi kullanılır.

Ancak, sadece model oluşturmak yeterli değildir. Ayrıca, size doğru sonuçlar vermesi için modeli uygun şekilde optimize edilmeli ve katsayılar ayarlanmalıdır. Optimizasyon teknikleri, optimum sonuca ulaşmak için hiperparametrelerin ayarlanması içerir.

Makine Öğrenimi her alanda kullanılmaktadır. Statik sistemlere zeka vermek için kullanılıyor. Verilerden elde edilen bilgi ile akıllı ürünler oluşturmak için kullanılır.

Giriş

Öğrenme, deneyim yoluyla bilgi, anlayış ya da beceri kazanma ve deneyime göre davranışsal eğilimin iyileştirilmesi, düzeltilmesi, güncellenmesidir.

Bilgiyi sayısallaştırip veriye dönüştüren, toplayan, sınıflandıran, birleştiren ve kıyaslama yapan algılayıcıların ortak değerlendirme ile doğru karar vermesinin nasıl yapılacağının iyi anlaşılması için, insan ve hayvanların bu işlevleri nasıl yaptığıın çok iyi araştırılması gerekmektedir; Baykuşlar bir farenin ayak sesini 1km öteden algılar. Filler çok uzaklardan gelen sismik sinyalleri hisseder. Yaban arıları yuvalarını tahrif edebilecek tehditleri önceden fark ederek, topluca saldırıyla geçerler ve düşmanlarını kilometrelerce kovalarlar. Hayvanların birbirleri ile bilgi paylaştıkları ve risk analizi yaptıkları da gözlenmiştir. Bunlara benzer yüzlerce örnek verebiliriz, burada önemli olan, hayvanların bu yetenekleri nasıl kazandıkları ve nasıl algıladıklarıdır.

Makine öğrenimi (ML - Machine Learning), anormallik algılama, tahmin, sınıflandırma, kümeleme, ilişkilendirme, boyut azaltma, desen bulma, eğilim ya da yön bulma ve öngörücü bakım gibi kullanım durumları için IoT uygulamalarında giderek daha yaygın hale geliyor.

Bilgisayarlara otonom olarak görevlerini nasıl gerçekleştireceklerini öğreten makine öğrenmesi algoritmaları ve matematiksel modelleri geliştirmede kullanılan yörünge denklemleri geniş bir uygulama alanı bulmaktadır.

Örneğin, geçerli e-posta mesajları ile istenmeyen postaları ayırt etmek için bir program yazmak istediğimizi varsayıyalım. Sahte başlık gibi belirli özellikleri içeren iletileri işaretlemek gibi bir dizi basit kural yazmaya çalışabiliriz, Bununla birlikte, hangi metnin geçerli olduğunu doğru bir şekilde ayırt etmek için kurallar yazmak aslında oldukça zor olabilir, bu da ya birçok kaçırılmış spam mesajına ya da daha kötüsü birçok kayıp e-postaya neden olur. Daha da kötüsü, spam gönderenler bu stratejileri kandırmak için spam gönderme yöntemlerini aktif bir şekilde değiştireceklerdir. Etkili kurallar yazmak - ve bunları güncel tutmak - hızla aşılmaz sıkıcı bir görev haline gelir.

Neyse ki, makine öğrenimi bir çözüm sağladı. Modern spam filtreleri örneklerden öğrenir. Öğrenme algoritmasına manuel olarak "ham" (geçerli e-posta) veya istenmeyen e-posta olarak etiketlenen örnek e-postalar sağlanır ve algoritmalar bunları otomatik olarak ayırt etmeyi öğrenir.

Makine öğrenimi çok çeşitli ve heyecan verici bir alandır ve onu tanımlamanın birçok yolu vardır:

- 1) **Yapay Zeka Görünümü:** Öğrenme, insan bilgisi ve zekasının merkezinde yer alır ve aynı şekilde akıllı makineler inşa etmek için de gereklidir. Yapay zeka konusunda yıllarca süren çaba, tüm kuralları programlayarak akıllı bilgisayarlar oluşturmaya çalışmanın yapılamayacağını göstermiştir; otomatik öğrenme çok önemlidir. Örneğin, biz insanlar dili anlama becerisiyle doğmadık - onu öğreniyoruz - ve hepsini programlamaya çalışmak yerine bilgisayarların dili öğrenmesini sağlamaya çalışmak daha mantıklı.
- 2) **Yazılım Mühendisliği Görünümü:** Makine öğrenimi, bilgileri örnekleyerek programlamamıza olanak tanır; bu, geleneksel şekilde kod yazmaktan daha kolay olabilir.
- 3) **İstatistik Görünümü:** Makine öğrenimi, bilgisayar bilimi ve istatistiğin birleşimidir: hesaplama teknikleri istatistiksel problemlere uygulanır. Makine öğrenimi, tipik istatistik problemlerinin ötesinde, birçok bağlamda çok sayıda soruna uygulanmıştır. Makine öğrenimi genellikle istatistiklerden farklı değerlendirmelerle tasarılanır (*Örneğin, hız genellikle doğruluktan daha önemlidir*).

1. Genel Kavramlar

Sistem: Amaç doğrultusunda çıkış sinyalleri üretmek için giriş sinyallerini işleyen, giriş sinyalini başka bir sinyale dönüştüren birimlerdir.

Zeka: İnsanın düşünme, akıl yürütme, algılama, kavrama, yargılama ve sonuç çıkarma yeteneklerinin tümüdür.

Akıł: Düşünme, kavrama, anlama yetisidir. Doğru ve yanlış, yalan ve gerçeği ayırt edebilme yetisidir.

Sinyal: Genellikle zaman içinde üretilen değerler dizisidir, bilgi taşırlar ve matematiksel olarak değişkenlerin fonksiyonu biçiminde gösterilir. İletilecek veriler elektromanyetik veya elektriksel sinyallere dönüştürülür. Yakın gelecekte veriler elektronlara dönüşecektir. Sinyaller bilgi taşıyan değişkenlerin fonksiyonel gösterimidir. Bir sinyal, nicelik gibi bir nitelikte gözlemlenebilir bir değişiklik olarak da tanımlanabilir.

Bilgi veriden doğmaktadır ve veri bilgiye dönüşmektedir. Verilerin günümüzde hız, çeşitlilik, kapasite (hacim) açısından büyük artış göstermesi ve bu artıya teknolojinin de destek vererek, yeni çözümler üretmesi ile birlikte "Büyük Veri" kavramı ortaya çıkmıştır. Veri, içerik işlemleriyle değer kazandırılarak bilgiye dönüştürülmektedir.

Veri (Data): Bilgisayarın belleğine aktarılan sinyaller, resimler, görüntüler, şekiller, rakamlar, metinler ve ses gibi sembollerdir. Veri, bilgi taşıyan fiziksel büyülüktür ve yorumlardır. Anlam kazanmamış, ilişkilendirilmemis, özümlenmemiş, işlenmemiş gerçekler ya da bilgi parçacıklarıdır. Herhangi bir içerikten yoksun formlardadırlar. Yorum taşımazlar ancak işlenmek için hazırlırlar. Karar vermede etkili değildir.

Bilgi (Information): İşlenmiş, düzenlenmiş, anlaşılmış verilerdir. Bilgi, organize, anlamlı ve yararlı verilerdir. Çıktı aşaması sırasında, oluşturulan bilgiler basılı rapor, garfik ve görseller ile sunum formuna sokulur. Bilgiler ileride kullanılmak üzere bilgisayar saklanır.

Yetenek - Tecrübe (Knowledge): Karar vermede, kestirim yapmadı, doğrulu aramada performansı yükseltmektir.

Understand (Bilinç): Anlayarak, kavrayarak, hissederek anlamaktır.

Wisdom (Bilgelik): Değerlendirilmiş anlayıştır. Sorulayarak, kestirim yaparak karar vermede ve yorumlamada etkindir.

Veri gürültüsü: Makine öğrenimi algoritmasının amacıyla alakalı olmayan herhangi bir veridir. Veri gürültüsü makine öğrenimi algoritmasının verimliliğini azaltabileceğinden elde edilecek sonuçlar olabildiğince kesin olmayacağındır.

Örüntü - Pattern: Bir nesnenin ya da olayın davranışının iki veya üç boyutlu, uzaysal ve geometriksel gösterildiği desenlerdir. Diğer bir ifadeyle örüntü, ilgilenilen varlığın davranışıyla ilgili uzayda gözlenebilir veya ölçülebilir geometrik bilgilerdir.

Olgu: Doğruluğu ispatlanmış önerme veya beklenen eylem.

Olay: Vakadır. Yağmur yağacak olması olgu, bunun yağması olaydır.

Hipotez: Bir problemin çözümünün ya da doğruluğunun araştırılmasına yön veren temel düşünceler, varsayımlar ve önermelerdir.

Öznitelik: Bir olgunun anlaşılır, ayırt edici ve bağımsız ölçülebilir özelliklerine öznitelik denir. Özellikler belirleme etkili örüntü tanıma, sınıflandırma ve regresyon algoritmaları için kritik bir adımdır. Özellikler genellikle sayısaldır ancak sentaktik örüntü analizinde kelimeler ve çizgiler de kullanılır. İşlenmemiş öznitelikler kümesi gereksiz öğeler içerebilir ve büyüklüğünden ötürü yönetilmesi zor olabilir. Bu yüzden, makine öğrenmesi ve örüntü tanıma uygulamalarından çoğu özniteliklerin bir alt kümesinin seçilmesini ya da yeni ve indirgenmiş bir öznitelikler kümesinin oluşturulmasını içerir. Kullanılacak özniteliklerin öğrenmeyi kolaylaştırması, genelliği ve yorumlanabilirliği artırması amaçlanır. Özniteliklerin çıkarılması ya da seçilmesi öznitelik mühendisliği olarak adlandırılır. Birçok farklı ihtimalin deneylenmesi ve hazır yöntemler ile bir alan uzmanının önsezilerinin bir araya getirilmesini gerektirir.

Öznitelikler vektörü:

Sayısal öznitelikler kümesinin matematiksel tanımlanmasında öznitelik vektörü kullanılır. Bir öznitelik vektörü kullanılarak iki ihtimalli sınıflandırma yapabilmek için öznitelik vektörü ve bir ağırlıklar vektörünün skaler çarpımı alınır ve çarpım sonucu bir eşik değeri ile karşılaştırılır. Bir öznitelikler vektörü kullanılarak yapılan sınıflandırma algoritmalarından bazıları en yakın komşu sınıflandırması, yapay sinir ağları ve Bayes yaklaşımlarıdır.

Hesaplamalı düşünme: bilgisayar yoluyla problemlere yaklaşmanın yeni bir yolu Soyutlama, ayrıştırma, modülerlik, ...

Veri bilimi: veri açısından zengin sorunları çözmek için disiplinler arası bir yaklaşım Makine öğrenimi, büyük ölçekli bilgi işlem, semantik meta veriler, iş akışları, ...

Kesikli (Süreksiz) Değişken: Tanımlı olduğu aralıklarda ayrık değerler alan değişkenlerdir.

Sürekli Değişken: Tanımlı olduğu aralıkta tüm değerleri (sonsuz sayıda) alabilen değişkenlerdir.

Nicel (Kantitatif) Değişkenler: Ölçüm sonucu değerleri saptanan sayısal özelliklerini belirten değişkenlerdir. Sayılabilir veya ölçülebilir büyüklüklerdir.

Nitel (Kalitatif) Değişkenler: Karakteristik özelliklerini, durumlarını ve pozisyonlarını belirten değişkenlerdir. Sayılamayan, birimlendirilemeyen ve ölçülebilir olmayan büyüklüklerdir. Bir şeyin nasıl olduğunu belirten, onu başka şeyleinden ayıran özellik (sifat).

Sıklık (Frekans) Dağılımı: Verilerin gösterdiği dağılıma sıklık (frekans) dağılımı denir. Aynı peryoda sahip sinyalin bütün içerisinde kaç kez tekrar etmesidir.

Sınıf: Eşit ya da birbirine yakın değerli deneklerin oluşturduğu her bir gruba sınıf denir.

Eşik seviye değeri (Bias): Bir fikir veya şey lehine veya aleyhine orantısız bir ağırlıktır. Uydu haberleşmesinde işaretin gürültüye oranı analiz edilirken, belirli eşik seviyenin altındaki değer (13dB) gürültü olarak kabul edilir.

2. Tarih

Hiç düşündünüz mü, ilk teknolojik icat ne zaman ortaya çıkmıştır? En eski zamandan bu zamana kadar gelmiş olan semboller bize tılsımsal ve büyüşel güçleri, bilinçaltı ve evren bağlantısı ve bazı ezoterik yani gizli sırları anlatırlar. Sümer tabletleri üzerindeki şekiller, semboller sesli anlatımın ifadesine dönüştürülürken bilginin gücü de ortaya çıkmaya başlamıştır. Fikirler kil tabletler üzerinde var olabiliyordu.

Joseph Marie Jacquard (1752 – 1834) 1804 yılında ipek dokumacılığında çok karmaşık bir mekanizmaya sahip olan desenleri ve sembollerini oluşturan bir yaratıcılık mucizesi bir alet tasarladı. Semboller, desenler 0 ve 1 lere dönüştürülüyor ve desenli ipek kumaşlar çok hızlı dokunuyordu. Bu makineler ilk talimatları işleyen ilk bilgisayar kontrollü makinelerdir. Elektrik yok... Desen ve Sembollerin Kodları üretilmiş, hafızaya yüklenmiş...

1831 yılının Ekim ayında Faraday bir diske iki tel bağladı ve diski bir at nali mıknatısının karşılıklı kutupları arasında döndürerek tel üzerinde akan bir elektrik akımı oluşturdu. 1 amperlik akımın oluşabilmesi için İletkenin herhangi bir noktasından 1 saniyede $6,25 \times 10^{18}$ elektron geçmesi gerektir.

19 uncu yüzyılda bilginin taransfer edilme hızında inanılmaz bir gelişme yaşandı. Bu gelişmeyi tetikleyen şletken üzerinden akan elektrik akımı idi. Karmaşık semboller basit bir sinyal ile elektrik teller üzerinde nasıl gönderilebillirdi? 1840 yılında Samuel Morse (1791 – 1872) ve arkadaşı Alfred Vail tarafından geliştirilen cihaz kısa ve uzun vuruşlardan oluşan elektrik akımları kullanarak alfabedecki harfler nokta ve uzun çizgi ile gösteriliyordu. Telgraf bilginin bir araçtan diğerine dönüştürülebileceğini gösteriyordu. İnsan beyninde yer alan bilgi basit sembollerle gösterilmişti. Telgraf diye adlandırılan sistemde bilgi Mors alfabesiyle sembollere dönüştürülmüştü. Bilgi elektrikle birleşmişti. Telgraf ağı bütün dünyaya yayıldı ve modern bilgi çağının temelleri atılmış oldu. Bilgi kablolar aracılıyla dünyanın her tarafına çok hızlı iletilebiliyordu.

Nikola Tesla (1856, 1943, New York). Sırp asıllı mucit, elektrik ve makine mühendisidir. Alternatif akım ile çalışan sistemlerin ilk mucididir. Tesla Kulesi olarak da bilinen Wardenclyffe Kulesi (1901–1917), Nikola Tesla tarafından tasarlanan ve ticari Atlantik ötesi kablosuz telefonculuk, yayın ve kabloları birbirine bağlamadan güç aktarımını göstermek için tasarlanan erken bir kablosuz telekomünikasyon kulesiydi. Ana tesis mali sorunlar nedeniyle tamamlanmadı ve hiçbir zaman tam olarak faaliyete geçmedi. Marconi (1874 – 1937), kendinden önce gelen fizikçi ve araştırmacıların, özellikle Tesla'nın çalışmalarını kullanarak ve değişiklikler yaparak radyonun ticari bir başarı kazanmasını sağlamıştır.

Alan Turing (1912 – 1954) bilgisayarın temelini oluşturan ilk insandır. Turing aslında matematiksel bir problemin çözümünü düşünüyordu. Matematikteki problemler basit kurallar dizisi takip edilerek çözülürse ne olur? Bu da bilgisayarlar hakkında düşünmesini sağladı. Beklenmedik bir şey oldu ve bilgisayar ortaya çıktı. Turing'in muhteşem fikri ilk kez 24 yaşındayken 1936 yılında yazdığı günümüzde efsane olan "Hesaplanabilir sayılarla karar veren problemlerin uygulanması" isimli 36 sayfalık kitapta yayınlandı.

Turing, hesaplama işlemesinde belirli kuralların tekrar edildiğini ve tüm hesaplamaların ikili boyutta olduğunu gördü. Bunun üzerine, makinelerin aritmetik işlemleri yapan talimatları insanların ki gibi anlamalarını sağlayan bir metod buldu. Aritmetik işlemleri makinelerin anlayabileceği bir dile çevirmek istiyordu. Turing bunu başardı; bir şeritte 1 ve 0 lardan oluşan talimatlar bilgisayara komut olarak verildiğinde makinenin insan beyni gibi işlevleri yerine getireceğini gösterdi. Şeritler bilginin ve komutların saklandığı ve işlendiği ortamlara dönüştürüldü. Yeterince büyük hafızası olan bir bilgisayar nerdeyse sınırsız sayıda iş yapabiliyordu. Turing'in çok sayıdaki farklı görevin heplama yapan makineye uzun bir dizi talimat verilerek yapılabileceğini savunan fikri en büyük mirasıdır. Günümüzde telefon ederken, hareketli görüntülerini kayt ederken, mektup yazarken, müzik dinlerken ayrı bir makine gerekmeyi. Resim, müzik, yazılar, ses, görüntü hepsi tek bir makine tarafından işlenebiliyor. Programlar, yazılım ya da uygulamalar dediğimiz bilgisayara ne yapacağını söyleyen 1 ve 0 dan oluşan çok uzun şeritlerdeki verilerden başka bir şey değildir. İnanılmaz boyuttaki şerit üzerindeki 1 ve 0 lar gözünüzün önündeki ekranada koca bir evrenin nasıl yaratıldığını size gösterebiliyor. Talimatları sembollere dönüştüren makine sadece basit bir resmi ya da sesi değil değişen bir sistemi bile yaratabiliyor. İnsan beyninin nasıl işlediğini düşünerek onu komut ve talimatlar ile makineye uygulama metodolojisini bulan Turing yirinci yüzyılın en önemli fikirlerinden birini üretti. Bilgisayar bilginin gücünü gösteriyordu.

Turing bir soru sordu:

- Hesaplama yapan, bir insanın zihninde neler oluyor?
- Hesaplama yapan kişi için hayatı öneme sahip olan şey nedir?
- Hesaplama işlemesinde insan beyninde anahtar işlev nedir?

Bilgiyi işleyen ve değiştiren bir makine düşüncesi Turing'e aittir? İnsan zihninde hesaplama işlemlerinde belirli kuralların tekrar edildiğini fark etti. Tüm hesaplamaların ikili boyutta (0/1) olduğunu gördü.

Makine öğrenmesi terimi 1959'da bilgisayar oyunu ve yapay zeka alanında Amerikalı Arthur Samuel tarafından üretildi. 1960'lı yıllarda makine öğrenmesi araştırmasının temsili bir kitabı, en çok desen sınıflandırması için makine öğrenmesi ile ilgilenen Nilsson'un Öğrenme Makineleri kitabıdır. Desen tanıma ile ilgili ilgi, Duda ve Hart tarafından 1973'te tarif edildiği

gibi 1970'lere kadar devam etti. 1981'de bir sinir ağının 40 karakteri, 26 harf, 10 basamak ve 4 özel simbol bir bilgisayar terminalinden üretildi.

Tom M. Mitchell, makine öğrenmesi alanında incelenen algoritmaların genel olarak alıntılmış, daha resmi bir tanımını yapmıştır: Bir bilgisayar programının, bazı görevler sınıfına ve performans ölçümüne göre görevlerindeki performanslarını artırarak deneyimlerinden ders alacağını söylemektedir. Makine öğrenmesinin söz konusu olduğu görevlerin tanımı yerine temelde operasyonel bir tanım sunar. Bu, Alan Turing'in "Bilişim Makineleri ve İstihbarat" adlı makalesinde, "Makineler düşünebilir mi?" "Makineler bizim gibi düşünen varlıklar olarak yapabileceğimizi yapabilir mi?" sorusuyla değiştirilebilir.

O halde Turing'in sorularını değiştirerek yeniden sorsam:

- Karar veren, bir insanın zihninde neler oluyor?
- Karar veren kişi için hayatı öneme sahip olan şey nedir?
- Karar verme aşamasında insan beyنinde anahtar işlev nedir?

Akıllı algoritmalar ve matematiksel kestirim modellerin temelini, gözetleme, ölçme, sorgulama ve kıyaslamaya dayalı kestirim yapmak ve karar vermek oluşturmaktadır. İnsanoğlunun kendisi gibi zeki, düşünen bir ürünü meydana getirme düşü, 1920'li yıllarda yazılan ve sonraları Isaac Asimov'u etkileyen bilim kurgu edebiyatının öncü yazarlarından Karel Čapek'in eserlerinde dışa vurmuştur. 1920 yılında yapay zekâının insan aklından bağımsız gelişebileceği öngörülümüştü. 1970'li yıllarda itibaren büyük bilgisayar üreticileri olan Microsoft, Apple, Xerox, IBM gibi şirketler kişisel bilgisayar modelleri ile bilgisayıları popüler hale getirdiler ve yaygınlaştırdılar. Günümüz bilgisayarlarında geliştirilen yazılımlar ile gündelik hayatımızın sorunlarını çözmeye yönelik kullanım alanları daha çok yaygınlaşmıştır.

Eb-Ül-İz El Cezeri'nin Otomatik Makinaları:

Su saatleri, su robotları, otomatik termos gibi birçok teknik ve mekanik buluşlar gerçekleştiren Eb-ul-iz El-Cezire 1136 yılında Cizre'de doğmuştur. Dünya bilim tarihi açısından bugünkü sibernetik ve robot teknolojileri üzerinde çalışan Ebu'l İz El Cezeri (1136-1206), bu çalışmalarını Artukoğulları Sultanı için yazdığı Kitab-ül Cami Beyn'el İlmi ve el Ameli'en Nafi fi Sı-naati'l Hiyel (Mekanik Hareketlerden Mühendislikte Faydalananmayı İçeren Kitap) adlı eserinde ortaya koymuştur. Cezeri'nin kitabının orijinali bugüne ulaşamamasına rağmen, on kopyası Avrupa'nı farklı müzelerinde, beş kopyası ise Topkapı Sarayı ve Süleymaniye kütüphanelerinde saklanıyor. Kitab-ül Hiyel adıyla bilinen eser, 6 bölümden oluşuyor. Başta Eb-Ül-İz olmak üzere çok sayıda alim, o dönemlerde Cizre yetişmesi rastlantı değildir. O dönemlerde Cizre; farklı kültürleri içerisinde barındıran, dini ilimler ile birlikte bilimsel araştırmaların da yapıldığı bir kent olarak karşımıza çıkmaktadır.

Otomatik Abdest Alma Makinesi:

Cezeri'nin bu makineyi yapımı hakkında, kitabının 332. sayfasında şu bilgilere yer veriliyor: Hükümdar Mahmut, hizmetçilerin ve cariyelerin abdest suyu dökmelerinden iğrenmektedir. Bunun için de Ebu'l İz'in yaptığı makinenin tavus kuşlarından faydalanan ve bunların döktüğü sularla abdest alır. Bu sistemde gerekli olan otomatik hareketler, hidrolik güçle sağlanmıştır. Su basıncı ve akış hızı en üst düzeyde olacak şekilde en üstteki depoya doldurulmuştur. Su deposuyla hizmetçinin elinde bulunan testi, sütun ve hizmetçinin elbiselerinin altından geçen U biçimindeki ince boruyla birleştirilmiştir. Testi iki bölümden meydana gelmiştir. Testinin önce alt bölümünü dolar; testinin su akıtma ağzı sifon şeklinde yapılmış olduğundan; su akmadan testiyi doldurmaya devam eder. Aynı zamanda su, hizmetçinin eli içinde bulunan ve elbisesi içine gizlenmiş ince borudan gereker sağ kolunun dirsek bölümüne basınç yapar.



Bu basınç düdük sesinin çıkışmasını sağlar. Dündük sesi, testi dolana kadar devam eder. Testi dolunca, suyun ağırlığıyla hizmetçinin kolu aşağıya doğru uzanır; dolayısıyla su testinin ağızından aşağıdaki kaba akmeye başlar. Gökhan Tok, Asya'da Rönesans, Bilim ve Teknik, Ekim 2001, s. 92-96

3. Neden Makine Öğrenmesi?

Makine öğrenimi: Örnek verileri veya geçmiş deneyimleri kullanarak bir performans kriterini optimize etmek için bilgisayarları programlamaktır. İyi olasılıklı modeller oluşturarak bir veri kütlesinden faydalı bilgilerin otomatik olarak çıkarılması. Genel bir teorinin olmadığı durumlarda çok fazla veriye sahip alanlar için idealdir.

Makine Öğrenimi çalışması: Deneyimle bazı görevlerde performanslarını artıran algoritmaların geliştirilmesidir. Bir çözüm süreci, model uydurma veya örneklerden öğrenme yoluyla, verilerden teoriyi otomatik olarak öğrenmek.

- Örnekler dışında bazı görevler iyi tanımlanamaz (örneğin yüzlerin veya insanların tanınması).
- Büyük miktarda verinin gizli ilişkileri ve korelasyonları olabilir. Bunları yalnızca otomatik yaklaşımlar tespit edebilir.
- Belirli bir sorun / görev hakkındaki bilgi miktarı, insanlar tarafından açık kodlama için çok büyük olabilir (örneğin, tıbbi teşhislerde)
- Ortamlar zamanla değişir ve sürekli olarak yeni bilgiler keşfedilir. Sistemlerin "elle" sürekli olarak yeniden tasarlanması zor olabilir.

Makine öğrenmesinde:

- Öğrenme Görevi: Ne öğrenmek ya da tahmin etmek istiyoruz?
- Veriler ve varsayımlar: Elimizde hangi veriler var? Kaliteleri nedir? Verilen problem hakkında ne varsayılabılır?
- Temsil: Sınıflandırılacak örneklerin uygun bir temsili nedir?
- Yöntem ve Tahmin: Olası hipotezler var mı? Tahminlerimizi verilen sonuçlara göre ayarlayabilir miyiz?
- Değerlendirme: Yöntem ne kadar iyi performans gösteriyor? Başka bir yaklaşım / model daha iyi performans gösterebilir mi?
- Sınıflandırma: Bir öğe sınıfının tahmini.
- Öngörü: Bir parametre değerinin tahmini.
- Karakterizasyon: Öğe gruplarını tanımlayan hipotezler bulun.
- Kümeleme: (Atanmamış) veri kümесinin ortak özelliklere sahip kümelere bölünmesi. (Denetimsiz öğrenme)

Örneğin, bir dizi kedi ve köpek resmimiz var. Yapmak istediğimiz şey onları bir grup kedi ve köpek olarak sınıflandırmak. Bunu yapmak için farklı hayvan özelliklerini bulmamız gerekiyor, örneğin:

- Her hayvanın kaç gözü vardır?
- Her hayvanın göz rengi nedir?
- Her bir hayvanın boyu kaçtır?
- Her bir hayvanın uzunluğu nedir?
- Her bir hayvanın bacak uzunlukları nedir?
- Her bir hayvanın ağırlığı nedir?
- Her hayvan genellikle ne yer?

Bu soruların her birinin cevabı için bir vektör oluşturulur. Ardından, bir dizi kural uygulanır: Boy > 30cm ve ağırlık > 2Kg ise, o zaman bir köpek olabilir. Her veri noktası için böyle bir dizi kurallar dizisi oluşturulur. Ayrıca, if, else if, else ifadelerinden oluşan bir karar aacı oluşturulur ve kategorilerden birine girip girmedigini kontrol edilir. Makine öğreniminin yaptığı şey, verileri farklı algoritmalarla işlemektir ve bize bunun bir kedi mi yoksa köpek mi olduğunu belirlemeye **hangi özelliğin daha önemli ve etkin olduğunu söyle**. *O halde sonuca etkisi fazla olan parametrelerin belirlenmesi çok daha önemsenmelidir.* Bu nedenle, birçok kural seti uygulamak yerine, bunu iki veya üç özelliğe göre basitleştirebiliriz ve sonuç olarak bu bize daha yüksek bir doğruluk sağlar.

Genellikle, makine öğrenimi yöntemleri iki aşamaya ayrılır:

- 1) Eğitim modeli: Verilerin bir koleksiyonundan öğrenilir.
- 2) Uygulama modeli: Yeni test verileri hakkında kararlar almak için kullanılır.

Makine öğrenimi türlerinden bazıları şunlardır:

- Eğitim verilerinin doğru yanıtlarla etiketlendiği denetimli öğrenim. En yaygın iki denetimli öğrenme türü, sınıflandırma ve regresyondur.
- Analiz etmek ve keşfetmek istediğimiz kalıpları, etiketlenmemiş verilerden oluşan bir koleksiyondan öğrenen denetimsiz öğrenme. En önemli iki örnek, boyut küçültme ve kümelenmedir.
- Robot veya kontrolör gibi bir temsilcinin geçmişteki eylemlerin sonuçlarına dayalı olarak uygun eylemleri öğrenmeye çalıştığı pekiştirmeli öğrenme.
- Eğitim verilerinin yalnızca bir alt kümesinin etiketlendiği yarı denetimli öğrenme.
- Mali piyasalarda olduğu gibi zaman serisi tahmini
- Fabrikalarda ve gözetimde arıza tespiti için kullanılanlar gibi anormallik tespiti.
- Verilerin elde edilmesinin pahalı olduğu aktif öğrenme

Bu nedenle bir algoritmanın hangi eğitim verilerinden elde edileceğini ve diğerlerini belirlenmesi gereklidir.

Makine öğrenimi modeli nedir?

Bir makine öğrenimi modeli, makine öğrenimiyle ilgili görevleri işleme koyan bir soru ya da yanıtlama sistemidir. *İş sonuçlarını iyileştirmek için değerli içgörüler toplamak, etkin olan parametreleri belirlemek için verileri kullanmayı amaçlar.*

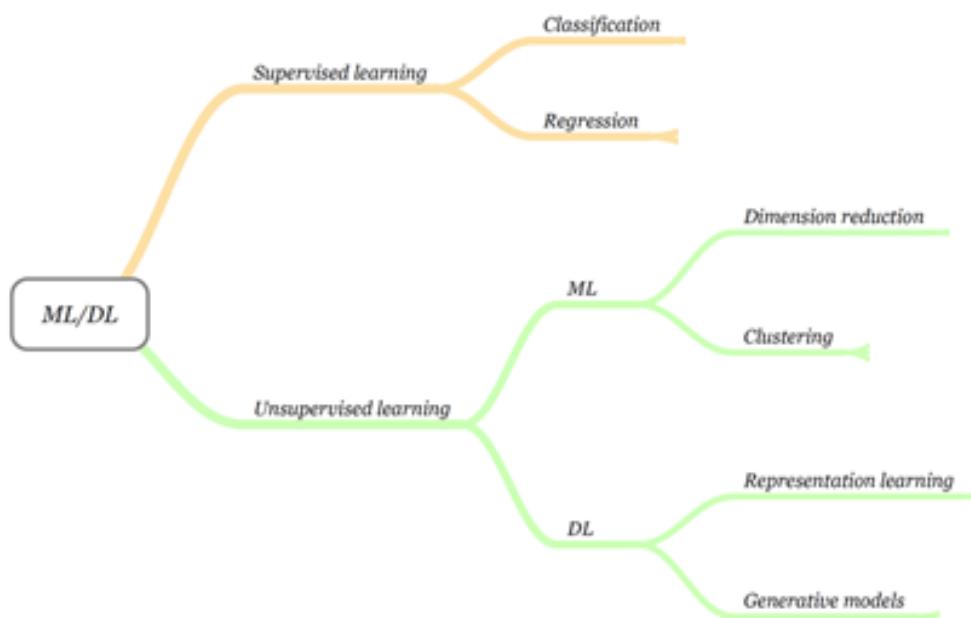
Disiplinler:

Mühendislikler: Bilgisayar, Elektronik, Makine

Matematik: Olasılık, istatistik, Nümerik analiz, İşaret işleme, Uygulamalı matematik, Stokastik, Algoritma

Antropoloji, Zooloji, Biyoloji, Bakteriyoloji, Genetik, Nörobiyoloji, Paleontoloji

Uzay bilimleri



Deneyim Performans ölçüsü:

Makine öğrenimi, bir çocuğun büyümeye benzer şekilde davranıştır. Bir çocuk büyündükçe, görevi yerine getirme deneyimi artar ve bu da performans ölçüsü ile sonuçlanır. Örneğin bir çocuğa “şekil ayırma bloğu” oyuncağı verilirse, (Artık hepimiz bu oyuncakta farklı şekil ve şekillerde deliklerimiz olduğunu biliyoruz). Bu durumda, görev, bir şekil için uygun bir şekil deliği bulmaktadır. Daha sonra çocuk şeklini gözlemler ve şekilli bir deliğe yerleştirmeye çalışır. Diyelim ki bu oyuncağın üç şekli var: bir daire, bir üçgen ve bir kare. Şekilli bir delik bulmaya yönelik ilk denemesinde performans ölçüsü $1/3$ 'tür, bu da çocuğun 3 doğru şekil deligidenden 1'ini bulduğu anlamına gelir.

İkincisi, çocuk başka bir zaman dener ve bu görevde biraz tecrübeli olduğunu fark eder. Kazanılan deneyime bakıldığından çocuk bu görevi başka bir zaman dener ve performansı ölçerken 2/3 olduğu ortaya çıkar. Bu görevi 100 kez tekrarladıktan sonra bebek şimdi hangi şeklärin hangi şekil deliğine gireceğini anladı. Böylece deneyimi arttı, performansı da arttı ve sonra bu oyuncaga yapılan deneme sayısı arttıkça fark edilimi, performansı da artar ve bu da daha yüksek doğruluk sağlar. Bu tür bir uygulama, makine öğrenimine benzer. Bir makinenin yaptığı şey, bir görevi alır, onu yürütür ve performansı ölçer. Artık bir makinenin çok sayıda verisi vardır, bu nedenle bu verileri işledikçe deneyimi zamanla artar ve daha yüksek bir performans ölçüsü ile sonuçlanır. Dolayısıyla, tüm verileri gözden geçirdikten sonra, makine öğrenimi modelimizin doğruluğu artar, bu da modelimiz tarafından yapılan tahminlerin çok doğru olacağı anlamına gelir.

Bu çalışmada, hesaplama yapıları da dikkate alınacaktır:

- Fonksiyonlar
- Mantıksal tasarım kuralları, sonlu durum makineleri
- Dilbilgisi
- Problem çözme mantıkları
- İstatistiksel analiz metodları
- Stokastik süreçler teorisi
- Uygulamalı matematik
- Kaos ve Sapma analizi
- Akıl Oyunları
- Belirsizlik
- Hata Kaynakları

Büyük Veri :

Yapay zekâ teknolojisinin altında büyük veri (big data) yatıyor. Verinin gerçek zamanlı işlenebilmesi, bu datanın farklı kanallardan gelmesi, kendi kendine öğrenen teknolojilere sahip uygun produktlere ulaşabilmesini sağlamak önemlidir. Artık kural bazlı statik teknolojiler yetmiyor. Yapay zekâyı bu yüzden teknoloji açısından değer yaratacak şekilde kullanıyoruz.

Gelecekte teknoloji departmanlarının yarısından fazlası veri bilimcilerden (Data Scientist) oluşacaktır. Beklenen önemli değişimlerden biri fiziksel dünya ile dijital dünyanın artık iç içe geçmişliği örneğin süt bittiğine otomatik sipariş geçen buz dolaplarından insanların çalışmadığı mağazalara uzanan bir dönüşümden bahsediyoruz. Bu dönüşümün ortasında da yapay zekâ teknolojileri oturuyor.

Şurası çok önemli; çağımızın insanı kendisine özel bir dokunuş istiyor. Yani bir grubun içerisinde addedilerek, o grubun bir parçası olan ortak bir kimlikmiş gibi addedilmek istemiyor. Mutlaka ona özel bir dokunuş" dedim. Tek tek herkese ayrı hitap etmeniz gerekiyor. Mesela ben şahsına özelleşmemiş hiçbir maili cevaplamam. İsmen hitap edilen, hukukumun gereği olan, karşılığı olan, üslubu olmayan standart hiçbir maili cevaplamam. Ben yokum ki orada. Bir topluluğa atılmış. Kim cevabını veriyorsa versin.

İnsanoğlunun anne karnından itibaren hareket ettiğini ve bunun hayatın sonuna kadar değişmedi. Yapılan bir araştırmaya göre sayfa yüklenme süresindeki her 1 saniyelik hızlanma internet satışlarını yüzde 5 arttırıyor.

Bilgisayar kontrolündeki bir sistemin ya da cihazın faaliyetlerini hafızasındaki verileri kullanarak insan zekasına benzer şekilde yerine getirme yeteneği kazandırılması, veri yoğunlarından öğrenen zekanın geliştirilmesi ile mümkün olabilir. Diğer bir anlatımla, ***kendi kendine öğrenen matematiksel modeller ve algoritmalar ile insandan bağımsız davranışı geliştirmesi gerekmektedir.***

Gezgin algılayıcıların, sistemlerin ve makinelerin artması, sosyal ağlara gezgin erişimin yaygınlaşması, çeşitli takip (sensörler, barkodlar, karekodlar, RFID sistemleri... vs.), nesnelerin interneti (IOT) ve otomasyon teknolojilerinin gelişmesi, iletişim teknolojilerinin ulaşılabilirliğinin artması, başta ticari işlemler olmak üzere pek çok iş kolunun elektronik ortama taşınması ile birlikte hem üretilen verinin çeşitliliği hem de toplanma hızı ve miktarı da ciddi oranlarda artmıştır. Bu artış üstel olarak devam etmektedir.

Öte yandan cihazlara takılan sim kartlar, algılayıcılar, elektronik ölçerler, bilgisayar sistemleri ve yazılımları sayesinde, cihazların uzaktan izlenmesini, yönetilmesini ve internet ağı ile birbiriyle iletişim kurabilmesini sağlayan bir teknoloji olan Makineler Arası İletişim (M2M) ve

Nesnelerin İnterneti, hem bireylerin hem de şirketlerin hayatında geniş bir kullanım alanı bulmaktadır.

M2M sistemleri, günümüzde neredeyse herhangi bir donanımın çeşitli uygulamalar veya cihazlarla birbirine kolayca bağlanıldığı Nesnelerin İnterneti (IoT), Her şeyin İnterneti (IoE), Nesnelerin Ağrı (WoT) ve Her şeyin Ağrı (WoE) gibi ortamlara evrilmüştür. Akıllı ortamların meydana geldiği bu sistemlerde, muazzam bir veri hacmi üretilir ve üstelik bu verilerin çoğu yapılandırılmamıştır. Resim, ses, metin, video gibi pek çok türde olabilen ve ağlar üzerinden aktarılan bu veriler bulut ortamlarda da saklanmaya başlamıştır. Bu verilerle ilgili bir başka husus ise sosyal medya verileri gibi insan kaynaklı veriler başta olmak üzere, değişken ve dinamik bir diğer deyişle akan bir yapıya sahip olmalarıdır. Bir yandan sisteme cihazlardan yeni veriler dahil olmakta veya bazı veriler kesintiye uğramakta, öbür yandan mevcut verilerde değişiklik meydana gelebilmektedir. Toplanan verilerin analizi bu sebeple daha karmaşık bir hal almaktadır. "Big data" yani "büyük veri" kavramı bu sebeple de özellikle son yıllarda çokça tartışırlı hale gelmiştir.

İnternet ve kablosuz teknolojilerin hayatın her alanında daha çok yer almasıyla birlikte, sayılar, metinler, ifadeler, görüntüler, şekiller, grafikler, sunumlar gibi anlamlı bilgilere dönüşen verilerin devasa miktarlarda depolanması söz konusu olmaktadır. *Hareket halindeki mobil uygulamalar ile birlikte, toplanan verinin hem çeşitliliği hem de hacmi inanılmaz boyutlarda artmaya devam etmektedir.* Verilerin işlenmesi, sınıflandırılması, analizi, ve anlamlı bilgilere dönüştürülmesi ile kendi kendine karar veren sonuçların elde edilebilmesine de imkân vermektedir. Bu ders çalışmasında, veri analizinde yapay zekâ ve makine öğrenmesi tekniklerinin kullanımı anlatılacaktır. Başlıca yapay zekâ ve makine öğrenmesi teknikleri öğretilecektir. Kümeleme, sınıflandırma, yapay sinir ağları, metin ve web madenciliği, fikir madenciliği, duygusal analizi, olasılık ve öğrenerek karar veren algoritmalar konusunda büyük verilerle yapılan çalışmalar anlatılacaktır.

Büyük Veri Analizi

Büyük veri, kısaca 11V diyebileceğimiz beş kavram ile betimlenmektedir:

- 1) Volume (Hacim),
- 2) Velocity (Hız),
- 3) Variety (Çeşitlilik),
- 4) Verification (Doğrulama)
- 5) Value (Değer).
- 6) Veracity (Gerçeklik),
- 7) Volatility (Oynaklık)
- 8) Validity (Geçerlik)
- 9) Vulnerability (Hassaslık),
- 10) Variability (Değişkenlik),
- 11) Visualization (Görselleştirme).

“Data never sleeps” projesi kapsamında Haziran 2016 verilerine göre, yalnızca bir dakika içerisinde;

- Youtube video paylaşım sitesine kullanıcılar tarafından 400 saatlik video yüklemesinin gerçekleştirildiği,
- Twitter üzerinden 9.678 adet emoji içerikli tweet atıldığı,
- Google’da 69.500.000 kelime tercüme edildiği, sadece Amerikalı kullanıcıların mobil cihazlarla yaklaşık 18.000 GB veri kullandıkları,
- Facebook Messenger kullanıcılarının 216.302 adet fotoğraf paylaştıkları,
- Instagram kullanıcılarının paylaşılan görüntüler için 2.430.555 adet beğeni yaptıkları
- Amazon web sitesinden 222.283 \$ satış yapıldığı tespit edilmiştir.

Büyük veriden kastedilen yalnızca hacimsel büyülüklük değildir. Farklı kaynaklardan ve farklı biçimlerde toplanan verilerin anlamlı ve işlenebilir hale getirilmesi gerekmektedir. Veriler hacim ve tür yanında sürekli artan bir hızda oluşmakta ve depolanmaktadır.

Verilerin depolanma ve değişme hızının yanı sıra büyük verilerin çoğu zaman karmaşık, düzensiz olduğu ve yanlışlar içerebileceği gerçeği, bu verilerin düzenlenmesi ve ayıklanması sorununu doğurmaktadır.

Üstelik anlık alınan verilerden hemen bilginin elde edilmesi yani verinin toplandığı anda analiz edilmesi gerekmektedir. Özellikle internete bağlı cihazların kaynaklık ettiği verilerin analizinde veri madenciliği yöntemleri yanında pazarlama, algı yönetimi, izleme, web, metin ve multimedya madenciliği teknikleri kullanılmaktadır. Burada unutulmaması gereken, verilerin yalnızca dijital değil fotoğraf, resim, video, ses, metin, konum (GPS) bilgisi vs. gibi pek çok çeşitte ve her biri için çeşitli boyutlarda olduğudur. Böyle olunca da asıl önemli olan, bu kadar büyük, hızlı ve çeşitli olan veri topluluğundan anlamlı ve değerli bilgiyi elde etmek olmaktadır.

4. Yapay Zeka

Doğadaki varlıkların akıllı davranışlarını yapay olarak üretmeyi amaçlayan (Charniak ve McDermot, 1985; Akt. Nabihev, 2012: 25), ve işini mükemmel yapan canlı sistemlerini ve insan beynini model alan yapay zeka çalışmaları; günlük hayatın farklı alanlarında ürünler vermesinin yanında, tahmin, sınıflandırma, kümeleme gibi amaçlar için de kullanılmaktadır.

Başlıca olarak uzman sistemler, genetik algoritmalar, bulanık mantık, yapay sinir ağları, makine öğrenmesi gibi teknikler, genel olarak yapay zeka teknolojileri olarak adlandırılmaktadır. Bu tekniklerin yanı sıra doğanın taklidi amacıyla da canlılar incelenmekte ve benzeri akıllı yöntemler önerilmektedir. Karınca kolonisi, parçacık sürü ve yapay arı gibi algoritmalar, yapay zeka optimizasyon teknikleri olarak kullanılmaktadır. Genel anlamda yapay zekadan kastedilen; insan zekasının, sinir sistemi, gen yapısı gibi fizyolojik ve nörolojik yapısının ve doğal olayların modellenerek makinelere (bilgisayar ve yazılımlara) aktarılmasıdır. Özette yapay zeka; "insan gibi düşünen, insan gibi davranışan, akılçıl (rasyonel) düşünen ve akılçıl davranışan" canlıların zekice olarak kabul edilen davranışlarına sahip bilgisayar sistemleridir ve makine öğrenmesi bu anlamda yapay zekanın son evresi olarak kabul edilmektedir.

Yapay zeka, bilgisayar kontrolündeki bir makineye kendisinden beklenen işlevleri canlılara benzer şekilde yerine getirme yeteneği kazandırılmasıdır. Yapay zeka çalışmaları genellikle canlıların bir işlevi yerine getirken geliştirdikleri davranış yöntemleri analiz edilerek, matematiksel modellerinin geliştirilmesine yönelikir. Yapay zeka araştırmacıları, insan gibi düşünebilen sistemleri araştırmaya devam ederken, rasyonel karar alan sistemler (Uzman sistemler) üzerine yoğunlaşan çalışmalar da hız kazanmıştır.

Yapay sinir ağları, insan beyninin bilgiyi nasıl öğrendiği mantığının araştırılmasıdır. Yapay sinir ağları, biyolojik sinir sisteme benzer nöronlar içerir ve bu nöronlar çeşitli şekillerde birbirlerine bağlanarak iletişim ağını oluştururlar. Bilgisayarın temel matematiğini geliştiren Macar asıllı matematikçi John Von Neumann'ın kurduğu mimariye göre bilgisayarın işlemcisi ve belleği ayrıdır. İşlemci ve bellek arasında bilgi iletişim yolları vardır. İşlemcinin işlem yapma hızı ile veri iletim yollarının veriyi okuma ve yazma hızları bilgisayarın kapasitesini belirler. Fakat insan beyni bilgisayardan farklıdır ve Von Neumann mimarisini ile çalışmaz. Beynimizdeki nöronlar (10 milyar) ve bağlantı noktaları (Synapses – 10 trilyon) paralel çalışır. Bu nedenle beyin sürekli öğrenen bir yapıya sahiptir.

Öğrendiklerinden karar veren makine: Bilgisayar donanım ve yazılımların kontrolündeki bir makineye kendisinden beklenen işlevleri canlılara benzer şekilde yerine getirme yeteneği kazandırılmasıdır. Öğrenerek Karar veren makine çalışmaları genellikle canlıların bir işlevi yerine getirken geliştirdikleri davranış yöntemleri analiz edilerek, matematiksel modellerinin geliştirilmesine yönelikir. Öğrenen makine araştırmacıları, insan gibi düşünebilen

sistemleri araştırmaya devam ederken, rasyonel karar alan sistemler (Uzman sistemler) üzerine yoğunlaşan çalışmalar da hız kazanmıştır. "Öğrenerek karar veren makine" kavramının fikir babası, "Makineler düşünebilir mi?" tartışmasını başlatan Alan Mathison Turing'dir. 1943'te II. Dünya Savaşı sırasında Kripto analizi gereksinimleri ile üretilen elektromekanik cihazlar sayesinde bilgisayar bilimi ve yapay zekâ kavramları doğmuştur.

Programlanmış bir bilgisayarın düşünmesi anlam olarak programın algoritmalarında kendi kendine öğrenen yazılımlar ile insandan bağımsız davranış geliştirmesi ile mümkün olmaktadır. Akıllı algoritmalar ve matematiksel kestirim modellerin temelini, insanın evreni ve doğayı gözetlemesi, ölçmesi, sorgulaması ve kıyaslaması oluşturur. İnsanoğlunun zeki, düşünen bir ürünü geliştirme düşü, 1920'li yıllarda yazılan ve sonraları Isaac Asimov'u etkileyen bilim kurgu edebiyatının öncü yazarlarından Karel Čapek'in eserlerinde dışa vurulmuştur. 1920 yılında öğrenen zekânın insan aklından bağımsız gelişebileceği öngörülümüştü.

Öte yandan çevremizde oluşan ve periyodik olarak tekrarlanan olaylardan geleceğe yönelik kestirim yapma binlerce yıldır insanlar tarafından yapılmaktadır. Bilgiyi sayısallaştırıp veriye dönüştüren, toplayan, sınıflandıran, birleştirilen ve kıyaslama yapan algılayıcıların ortak değerlendirme ile doğru karar vermesinin nasıl yapılacağına iyi anlaşılması için, insan ve hayvanların bu işlevleri nasıl yaptığına çok iyi araştırılması gerekmektedir. "Algılamak öğrenmektir." Karar vermek; risk satın alarak seçim yapmaktır. Günümüzde kritik alt yapıları uzaktan izleyen, yöneten ve tehditleri erken belirleyen algılayıcıların sayısında, çeşidinde ve konumsal dağılığında yoğun artış görülmektedir.

Gelecekte öğrenerek karar veren makineler konsuda araştırmalardaki tüm alanların birleşeceğini öngörmek zor değildir. Sibernetik bir yaklaşımla modellenmiş insan aklına benzetilmiş bilişsel süreçler ve insan aklı kadar esnek ve duyguları olan bir İrade (Karar alma yetisi), Uzman sistemler kadar yetkin bir bilgi birikimi ve rasyonel yaklaşımın dengeli bir karışımı sayesinde Öğrenen Zekâ, gelecekte insan zekâsına bir alternatif oluşturabilir.

"Yapay zekâ" kavramının fikir babası, "Makineler düşünebilir mi?" tartışmasını başlatan Alan Mathison Turing'dir. 1943'te II. Dünya Savaşı sırasında Kripto analizi gereksinimleri ile üretilen elektromekanik cihazlar sayesinde bilgisayar bilimi ve yapay zekâ kavramları doğmuştur.

Yapay zekâ konusundaki ilk çalışmalar McCulloch ve Pitts tarafından yapılmıştır. Araştırmaları, yapay sinir hücrelerini kullanan hesaplama modeli, önermeler mantığı, fizyoloji ve Turing'in hesaplama kuramına dayanıyordu. Herhangi bir hesaplanabilir fonksiyonun sinir hücrelerinden oluşan ağlarla hesaplanabileceğini ve "mantıksal dallanma" işlemlerinin gerçekleştirilebileceğini gösterdiler. Bu ağ yapılarının uygun şekilde tanımlanmaları halinde öğrenme becerisi kazanabileceğini de ileri sürdürüler. Hebb, sinir

hücreleri arasındaki bağlantıların şiddetlerini değiştirmek için basit bir kural önerince, öğrenebilen yapay sinir ağlarını gerçekleştirmek de olası hale gelmiştir.

1950'lerde Shannon ve Turing bilgisayarlar için satranç programları yazıyorlardı. İlk yapay sinir ağı temelli bilgisayar SNARC, MIT'de Minsky ve Edmonds tarafından 1951'de yapıldı. Çalışmalarını Princeton Üniversitesi'nde sürdürən Mc Carthy, Minsky, Shannon ve Rochester'le birlikte 1956 yılında Dartmouth'da düzenlenen toplantıda Mc Carthy tarafından önerilen yapay zeka konusunun tartışılmıştır. Bu toplantıda Logic Theorist (Mantık kuramcısı) Newell ve Simon tarafından tanıtılmıştır.

Newell ve Simon, insan gibi düşünme yaklaşımına göre üretilmiş ilk program olan Genel Sorun Çözücü (General Problem Solver)'ı geliştirmiştirlerdir. Ardından Simon, fiziksel simge varsayımini ortaya atmış ve bu kuram, insandan bağımsız zeki sistemler yapma çalışmalarıyla uğraşanların hareket noktasını oluşturmuştur. Simon'ın bu tanımlaması bilim adamlarının yapay zekaya yaklaşımlarında iki farklı akımın ortaya çıktığını belirginleştirmesi açısından önemlidir: Sembolik Yapay Zeka ve Sibernetik Yapay Zeka.

Gelecekte Yapay Zeka:

Gelecekte yapay zekâ araştırmalarındaki tüm alanların birleşeceğini öngörmek zor değildir. Sibernetik bir yaklaşımla modellenmiş bir Yapay Beyin, Sembolik bir yaklaşımla insan aklına benzetilmiş bilişsel süreçler ve Yapay Bilinç sistemi, insan aklı kadar esnek ve duyguları olan bir İrade (Karar alma yetisi), Uzman sistemler kadar yetkin bir bilgi birikimi ve rasyonel yaklaşımın dengeli bir karışımı sayesinde Yapay Zekâ, gelecekte insan zekâsına bir alternatif oluşturabilir.

Bilginin hesaplanması matematiksel gelişme ile mümkün olabilir. Çok yüksek döngü gerektiren NP problemlerin çözümü, satranç oyununda en iyi hamleyi hesaplamak veya görüntü çözümleme işlemlerinde bilgiyi saymak yerine hesaplamak sureti ile sonuca ulaşılabilir.

Yeni matematik kuantum parçacık davranışlarını açıklayacağı gibi kuantum bilgisayarın yapılmasına olanak verir .

Alt dallar:

- Makine Zekâsı (Sembolik Yapay Zekâ)
- Yapay Sinir Ağları (Sibernetik Yapay Zekâ)
- Doğal Dil işleme (Dil ile düşünme)
- Konuşma Sentezi (Yapay Konuşma)
- Konuşma Anlama (Konuşma Analizi)
- Uzman sistemler
- Örütü Tanıma
- Genetik Algoritmalar
- Genetik Programlama
- Bulanık Mantık
- Çoklu Örnekle Öğrenme(Multiple Instance Learning)

Sembolik Yapay Zeka:

Geliştirilen programlaryalnızca sentaktik süreçleri benzeşimlendirerek anlam çıkarma, bağlantı kurma ve fikir yürütme gibi süreçleri modelleme temelli olduğundan başarısız olmuştur. Weizenbaum tarafından geliştirilen Eliza algoritmaları, yalnızca karşısındaki insanın cümleleri üzerinde bazı işlemler yapıyordu. İlk makine çevirisini çalışmaları sırasında gülünç çevirilerle karşılaşılınca bu çalışmaların desteklenmesi durdurulmuştu. Bu yetersizlikler aslında insan beynindeki semantik süreçlerin yeterince incelenmemesinden kaynaklanmaktadır.

Sibernetik yapay zeka:

Sibernetik, karar çıktısının geri bildirim girişi olarak sistemi teiklemesi esasına dayanmaktadır. İnsani müdahaleye gerek duymadan, makinenin değişen durumlara göre kendi kendini düzenleyebilen sistemleri inceleyen bilim dalıdır.

Yapay sinir ağları konusundaki çalışmaların Minsky ve Papert'in 1969'da yayınlanan Perceptrons adlı kitaplarında tek katmanlı algaçların bazı basit problemleri çözemeyeceğini gösterip aynı kısırlığın çok katmanlı algaçlarda da beklenilmesi gerektiğini ileri sürdüler. Sibernetik akımın görevle ilgili varlıkların veya sonuçların bir yargıya dönüşerek diğer kavramlar ile bir ilişki kurulamamasından kaynaklanan başarısızlıklar görülmektedir. Bu durum aynı zamanda semantik süreçlerin de benzeşimlendirilememesi gerektiğini doğurdu.

Uzman sistemler:

Sembolik ve sibernetik zeka konularında yapılan çalışmalarda görülen başarısızlıklar, her sorunu çözecek genel amaçlı sistemler yerine belirli bir uzmanlık alanındaki bilgiyle donatılmış programları kullanma fikrinin gelişmesine sebep oldu ve bu durum yapay zeka alanında yeniden bir canlanmaya yol açtı. Kısa sürede "Uzman sistemler" adı verilen bir metodoloji gelişti. Fakat burada çok sık rastlanan tipik bir durum, bir otomobilin tamiri için

önerilerde bulunan uzman sistem programının otomobilin ne işe yaradığından haberi olması gerekliliğidir.

Uzman sistemler, çözümü bir uzmanın bilgi ve yeteneğini gerektiren problemleri, bilgi ve mantıksal çıkarım kullanarak o uzman gibi çözebilen sistemlerdir. Yani problemi çözmede uzman kişi veya kişilerin bilgi ve mantıksal çıkarım mekanizmasının modellemesi amaçlanmaktadır. Uzman sistemlerde, bilgiler depolanıp daha sonra bir problemle karşılaşıldığında bu bilgi üzerinden yapılan çıkarımlarla sonuçlara ulaşılmaya çalışılmaktadır. Böylelikle insan zekasının muhakeme etme sürecine, bilgisayarın kesinlik ve hızının katılması amaçlanmaktadır.

Uzman sistemler şu temel ögelerden teşekkür edilir: Bilgi tabanı (kural tabanı), veri tabanı, çalışan bellek (yardımcı yorumlama modülü), çıkarım motoru (karar verme mekanizması, mantıksal çıkarım modülü) ve kullanıcı arayüzü. Bilgi tabanı, bilgilerin tutulduğu ve tutulan bilgilerden yeni bilgiler üretilmesine imkan sağlayan birim olup uzman sistemin beyni ve yapı taşıdır.

Veri tabanı ise bilgi tabanı ile sürekli ilişki halinde olmalıdır. Kullanıcı arayüzü; bilgi kazanma, bilgi tabanı ile hata ayıklama ve deneme, test durumlarını çalışma, özet sonuçlar üretme, sonuca götüren nedenleri açıklama ve sistem performansını değerlendirme gibi görevleri yerine getirmektedir. Çıkarım motoru, arama ve çıkarımın yer aldığı ögedir. Uygun bilgi için bilgi tabanını taramakta ve mevcut problem verisine dayanarak çıkarımda bulunmaktadır. Çalışan bellekte, problem ile ilgili soruların cevapları ve tanışsal testlerin sonuçları gibi mevcut problem verisi saklanmaktadır. Çıkarım motorunda mantıksal sonuçlar elde edilmesine yardımcı olur.

Doğal dil işleme:

Antropoloji bilimi, gelişmiş insan zekası ile dil arasındaki bağlantıyı gözler önüne serdiğinde, dil üzerinden yürütülen yapay zeka çalışmaları tekrar önem kazandı. İnsan zekasının doğrudan doğruya kavramlarla düşünmediği, dil ile düşündüğü, dil kodları olan kelimeler ile kavramlar arasında bağlantı kurduğu anlaşıldı. Bu sayede insan aklı kavramlar ile düşünen hayvan beyninden daha hızlı işlem yapabilmekteydi ve dil dizgeleri olan cümleler yani şablonlar ile etkili bir öğrenmeye ve bilgisini soyut olarak genişletebilme yeteneğine sahip olmuştu. İnsanların iletişimde kullandıkları doğal dilleri anlayan bilgisayarlar konusundaki çalışmalar başladı. Sembolik Yapay Zeka araştırmacıları özel Yapay Zeka dillerini kullanarak verileri birbiri ile ilişkilendirebilmekte, geliştirilen özel prosedürler sayesinde anlam çıkarma ve kestirim yapma gibi ileri seviye öğrenen matematiksel fonksiyonları geliştirmeye çalışmaktadır.

Android:

Google, Open Handset Alliance ve özgür yazılım topluluğu tarafından geliştirilmiş olan, Linux tabanlı, mobil cihaz ve cep telefonları için geliştirilmekte olan, açık kaynak kodlu bir mobil işletim sistemidir. Android, makinelerin fonksiyonellliğini genişleten uygulama yazılımlarıdır. Android uygulama yazılımları ise, Apache harmony üzerine kurulu Java-uyumlu kütüphaneleri içine alan uygulama iskeleti üzerinden çalışmaktadır. Android, derlenmiş Java kodunu çalıştırmak için dinamik çevirmeli (JIT) Dalvik sanal makinasını kullanır ve cihazların fonksiyonellliğini artıran uygulamaların geliştirilmesi için çalışan geniş bir programcı-geliştirici çevresine sahiptir.

Sözelimi oturduğunuz yerden evinizdeki cihazların çalıştırılması, kontrol edilmesi, acil durumlarda uyarılmanız gibi pek çok işi yönetebilirsiniz. Ayrıca kullanıcıların internet üzerindeki izlerinden alışkanlıklarının belirlenmesi konusunda yapılan çalışmalar ile pazarlama teknikleri geliştirmektedir.

Mühendisler uzak bir gelecekte el değimeden her işi halleden mutfaklar hayal etmekten de geri durmuyorlar.

Android işletim sisteme sahip bir mikrodalga fırın ve bir de pirinç pişirme makinesi geliştirdiğini biliyor muydunuz? Android buzdolabını duydunuz mu? İnternet üzerinden diğer cihazlara bağlanıp tarifler alan ya da stok tutan bu gibi mutfak cihazları hayli ilgi çekiyor.

Android cep telefonunu ile akıllı otomobile dönüsen araçlar geliştirmek üzere. Japonya kökenli Seraku firması tarafından geliştirilen Smart Basin adlı prototip akıllı banyo aynası, aynı zamanda dokunmatik bir ekran görevi görüyor ve arkasındaki gizli Android tablet yardımıyla su sıcaklığını ayarlamaktan, hava durumunu kontrol etmeye kadar pek çok iş yapabiliyor. Bir gün evlerin standart donanımı olabilecek olan bu prototip, şimdilik sadece fuarlarda boy gösteriyor.

Samsung tarafından piyasaya sürülen Galaxy fotoğraf makinesi gerçekte Android işletim sistemli bir akıllı telefon ya da tabletten çok farklı sayılmazlar. İçerideki gelişmiş işlemciler ve Android sayesinde fotoğraf ve video işleme konusunda hayli büyük esneklik sunan bu cihazlara rahatlıkla yeni foto uygulamaları eklenebilir, dosya paylaşımı gerçekleştirilebilir.

NASA ve Google tarafından ortak yürütülen Project Tango, Android işletim sistemli telefonların uzaya adım atmasının ardından en büyük sebep. NASA tarafından geliştirilen ve SmartSphere adı verilen robotik kärelerin kalbinde, ciddi şekilde modifiye edilmiş Androidlı cep telefonları bulunuyor. Bu cihazlar ilk adımda uzay istasyonlarının içinde süzülerek astronotların günlük işlerine yardımcı olacaklar. Bunlardan elde edilecek tecrübe ve verilerle gelecekte uzayda otonom olarak dolaşıp, istasyon inşaatı ya da uzay gemisi onarımı gibi işler yapacak yeni nesil robotlar geliştirilecek!

Cyborg:

Biyolojik ve yapay (örneğin elektronik, mekanik veya robot) kısımları olan varlıklara verilen isimdir. Sibernetik organizma teriminin kısaltılmasıdır. ABD'li bilim insanları Manfred Clynes ve Nathan S. Kline tarafından 1960 yılında icat edilen terim, ikilinin uzayda kendi kendini düzenleyen insan-makine sistemlerinin avantajlarını anlattıkları bir makalede kullanıldı.

Beyin dalgalarını alarak yorumlayan cihazlar, insan beyni ve kendi bilgisayarlarını kullanarak iş yapacak süreçleri kontrol edecektir.

İnsanlar vücutlarını destekleyen iskeletleri ile ayakta kalmaktadır. Deri ve kasların altındaki kemikler ile oluşturulmuş iskelet, vücutun içindeki mekanik tasarım harikasıdır. Bu mekanik tasarım harikasının zarar görmesi halinde insanlar fiziksel yeteneklerinin çoğunu kaybetmektedir. Japonya merkezli Cyberdyne adlı şirket bu kusursuz tasarıma bir alternatif geliştirmek için çalışıyor. Hybrid Assistive Limb (HAL) isimli bir dış iskelet robottu üzerinde çalışan şirketin geliştirdiği robot, görünüşü ile Cyborg'u anımsatıyor.

Sürü zekâsı:

Kendi kendine organize olan algıların sergiledikleri toplu davranış biçimidir. Bir araya kümelenmiş hayvanların kitle halinde hareket etmeleri ya da göç etmeleri ile sergilenen toplu bir davranıştır. Ana kural: komşuların davranışını izle, aynı yöne ilerle, yakın dur, çarpmaktan kaçın.

Yapay Zeka, Makine Öğrenmesi ve Derin Öğrenme

Arasındaki Temel Farklar

Profesör Andrew Moore tarafından tanımlanan yapay zeka (AI), yakın zamana kadar insan zekası gibi bilgisayarların davranış gelişirmesini sağlayan bilimi ve mühendisliğidir.

Bunlar şunları içerir:

- Bilgisayar görüşü
- Dil İşleme
- Yaratıcılık
- Öztleme

Profesör Tom Mitchell tarafından tanımlandığı gibi, makine öğrenimi, bilgisayar programlarının deneyim yoluyla otomatik olarak öğrenmesine izin veren bilgisayar algoritmaları çalışmasına odaklanan yapay zekanın bilimsel bir dalını ifade eder.

Bunlar şunları içerir:

- Sınıflandırma
- Sinir ağı
- Kümeleme

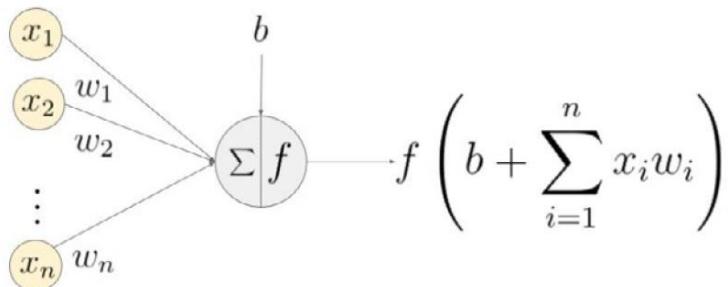
Yapay zekâ, ister makine öğrenmesi kullanınsın ister kullanmasın herhangi bir tahmin veya karar işlemini gerçekleştiren teknolojilerin genel adıdır. Genel kanaatin aksine yapay zekâ makine öğrenmesi veya derin öğrenme algoritmaları olmaksızın da çalışan bir algoritma olabilir. Makine öğrenmesi algoritmaları ortaya çıkana kadar yapay zekâ çalışmaları “hard-coded” olarak nitelendirilen yani tüm mantıksal ve matematiksel işlemlerin yazılımcı tarafından bizzat kodlandığı bir yapıya dayanmaktadır. Örneğin ilk satranç oyuncusu yapay zekâ algoritmaları tamamen böyledi. Yapay zekânın bu türü sembolik yapay zekâ olarak adlandırılmaktadır.

Makine öğrenmesinde algoritmalar tamamen veriden öğrenir. Makine öğrenmesini hard-coded olarak kodlanmış sembolik yapay zekâ algoritmalarından ayıran özellik algoritmanın tamamen veriden öğrenmesidir. Doğal olarak akla makine öğrenmesi algoritmalarının verilerden öğrenmesinin riskli tarafları gelebilir. Sorulara yanıt veren yazılımsal makine öğrenmelerine ahlaksız ve ırkçı söylemler sıkça sorulursa, günün sonunda yazılımın kendisi sapık ve ırkçı bir karaktere bürünebilmektedir.

Derin öğrenme modeli, verinin yapısına göre hangi parametrelere ne ağırlık verileceğini kendisi keşfetmektedir. Derin öğrenme algoritması da veriye dayalı öğrenme gerçekleştirmekle birlikte, öğrenme süreci standart makine öğrenmesi algoritmalarında olduğu gibi tek bir matematiksel modele değil sinirsel ağ (neural network) olarak ifade edilen ağ diyagramlarına benzeyen yapıda geliştirilen hesaplamalarla çalışmaktadır. Her

derin öğrenme algoritması bir makine öğrenmesi algoritmasıdır çünkü verilerden öğrenme gerçekleştirmektedir. Ancak her makine öğrenmesi algoritması derin öğrenme algoritması değildir; nitekim derin öğrenme, makine öğrenmesinin spesifik bir türüdür. Derin öğrenme, Yüksek bilgi işleme gücü ve büyük veri kümeleriyle birlikte katmanlı yapay sinir ağlarının güçlü matematiksel modelleri oluşturabildiği bir makine öğrenimi alt kümesidir.

Örnek: n giriş değerlerinin ağırlık değerleri (w) ile çarpılarak elde edilen fonksiyonun birde bağımsız değişken girişi olsun.



Bu algoritma öncelikle ağırlık değerleri (w)'yi tesadüfi olarak atar, bu değerlerin sonuçları ile gerçek sonuçlar arasındaki sayısal farka göre tipki Gradient Descent algoritmasından olduğu gibi kendini yeniler. Eldeki verilerle etiketler arasındaki ilişkiyi en iyi ifade eden konfigürasyon elde edildiğinde ise durur. Elde edilen konfigürasyonda artık w değerleri bilindiğinden algoritma belirli bir doğruluk gücünde tahmin yapar. Davranış analiz edilmiş oldu. Geleceğe yönelik öngörüde bulunurken w değerleri değişecektir. Makinenin nasıl karar verceğinin de öngörülmesi gerekmektedir. Göktaşında çalışan bir makine ile dünyadaki kullanıcıları arasındaki iletişim koptu. Makinen kullanıcısının ne yapacağını önceden kestirecek. Dünyadaki uzman da aynı şekilde makinenin ne yapacağını ön görecektir.

Derin Öğrenme algoritmalarının standart makine öğrenmesi algoritmalarına göre en büyük üstünlüğü feature engineering adı verilen süreci gerektirmemektedir. Bu durum bir örnekle şöyle ifade edilebilir. Diyelim ki bir makineye ait 100 parametre ile karar verme tahmini yapan bir algoritma olsun. Standart makine öğrenmesi algoritmasında hangi parametrelerin işe yarar olduğu ve modele konması gereken katsayının zarfı verilmek zorundadır. Bunun için teoriden ya da parametre seçim algoritmalarından faydalana bilir. Bu süreç feature engineering olarak adlandırılmaktadır. Derin öğrenmede ise bu süreç yoktur. **Derin öğrenme modeli, verinin yapısına göre hangi parametrelere ne ağırlık verileceğini kendisi keşfetmektedir.** Derin öğrenmenin bu özelliği bir makine öğrenmesi algotimasını daha esnek hale getirmekte ve ürüne dönüşümünde daha fazla kolaylık sağlamaktadır. Bugünlerde ürüne dönüşmüş olan insansız otomobiller gibi teknolojilerin arasında ise derin öğrenme vardır. Derin öğrenme algoritmalarının makine öğrenmesi algoritmalarına göre dezavantajı ise ihtiyaç duyulan donanım ve veri kapasitesidir.

5. Makine Öğrenmesinde Temel Kavamlar

Bilgi toplama gibi öğrenme, kesin olarak tanımlanması zor olan çok çeşitli süreçleri kapsar. Öğrenmenin sözlük anlamı, bilgi edinmek, çalışmak, deneyimlemek suretiyle anlama, beceri kazanma gibi ifadelerin ve deneyimle **davranışsal eğilimlerin değiştirilmesini** içerir. Makine öğrenmesi alanında araştırmacılar tarafından keşfedilen kavram ve tekniklerin biyolojik öğrenmenin bazı yönlerini aydınlatabileceğini de muhtemel görünüyor. Öte yandan biyolojik öğrenme metotları da makinelerin öğrenmesine inanılmaz katkı vereceği öngörmektedir.

Yapay zekanın en aktif olarak kullanıldığı alan kuşkusuz robot teknolojileridir. Yapay zekanın gelişmesi robot teknolojilerinin gelişimini de doğrudan etkiledi. Robotlarda gerçekleşen performans problemlerini kolay bir şekilde algılayabilen yapay zeka, ihtiyaç halinde sorunları giderebiliyor. Böylece robotlar kendini yenileyebiliyor.

Son yıllarda taşımacılık alanında yaşanan en büyük gelişme sürücüsüz araçların geliştirilmesi oldu. Google, Tesla, Uber gibi büyük firmalar bu alana önemli yatırımlar yaptı ve sürücüsüz araç teknolojileri ivme kazandı. Sürücüsüz araç teknolojilerinin en büyük destekçisi ise yapay zeka teknolojisidir. Sürücüsüz araçların yanı sıra drone teknolojileri de yapay zekadan faydalaniyor. Hem sürücüsüz araçlar hem de drone teknolojileri yapay zeka olmadan mümkün olamazdı.

Makine Öğrenmesi Tenimlar:

- Makine öğrenimi, deneyimle otomatik olarak öğrenmek ve gelişmek için sistem programlamaya ilgilenen bir bilgisayar bilimi dalıdır. Örneğin: Robotlar, sensörlerden topladıkları verilere göre görevi yerine getirebilecek şekilde programlanmıştır. Verilerden programları otomatik olarak öğrenir.
- Makine Öğrenmesi: veri yığınından öğrenen ve otonom davranış sergileyen algoritmaların ve matematiksel modellerin oluşturulmasıdır.
- Veri yığınından tahmin etmeye ya da karar vermeye yönelik otonom davranış paternleri geliştiren algoritmalar ve matematiksel modellerin oluşturulmasıdır.

- Veri yiğininden kendi kendine öğrenen matematiksel modeller ve algoritmalar ile insandan bağımsız otonom davranış geliştirilmesidir.
- Yapısal işlev olarak veri yiğininden öğrenebilen ve veriler üzerinden karar vermeye yönelik tahmin yapabilen algoritmaların çalışma ve işlerini araştıran bir sistemdir.

Makine öğrenmesi (ML) Yapay zekanın bir alt kümesi olarak görülür. Makine öğrenmesi algoritmalarından oluşur. Açık bir şekilde tahminler veya kararlar vermek için "öğrenme verileri" olarak bilinen verilere dayalı kendi kendine öğrenen bir matematiksel model oluşturulmasıdır. **Makine öğrenmesi, bilgisayarları açıkça programlanmaksızın görevleri nasıl gerçekleştirebileceklerini keşfetmeyi içerir. Belirli görevleri yerine getirmeleri için verilerden öğrenen algoritmaları içerir.**

Atanan basit görevler için, makineyi eldeki sorunu çözmek için gereken tüm adımların nasıl yürütüleceğini söyleyen algoritmaları programlamak mümkündür; bilgisayar tarafından öğrenmeye gerek yoktur. Daha gelişmiş görevler için, bir insanın gerekli algoritmaları manuel olarak oluşturması zor olabilir. **Makine Öğrenmesi uygulamasında, programcılar gereken her adımı belirtmesinden ziyade makinenin kendisinin algoritmaları geliştirmesine yardımcı olmaktadır.**

Makine öğrenmesinin omurgasını oluşturan disiplinler:

- Makine öğrenmesi, tahmin yapmaya odaklandığından istatistiksel hesaplama ile yakından ilgilidir.
- Matematiksel optimizasyon çalışması, makine öğrenmesi alanına yöntemler, teori ve uygulama alanları sağlar.
- Veri madenciliği, denetimsiz öğrenim yoluyla keşifsel veri analizine odaklanan ilgili bir çalışma alanıdır.
- Uygulamalı Matematik
- Bilgisayar sistemleri ve yazılımlar

Makine öğrenmesi (ML), yapay zekanın bir alt kümesi olarak görülür. Makine öğrenmesi algoritmalarından oluşur. Açık bir şekilde programlanmadan tahminler veya kararlar vermek için "öğrenme verileri" olarak bilinen örnek verilere dayalı bir matematiksel model oluşturulmasıdır. Makine öğrenmesi algoritmaları, gerekli görevleri yerine getirmek için geleneksel algoritmalar geliştirmenin zor veya mümkün olmadığı filtreleme ve katsayıları değişken fonksiyonlarının çok çeşitli uygulamalarında kullanılır.

Veri madenciliği, denetimsiz öğrenim yoluyla keşifsel veri analizine odaklanan ilgili bir çalışma alanıdır. Makine öğrenmesi, bilgisayarları açıkça programlanmaksızın görevleri nasıl gerçekleştirebileceklerini keşfetmeyi içerir. Belirli görevleri yerine getirmeleri için sağlanan verilerden öğrenen algoritmaları içerir. Makine öğrenmesi tam olarak tatmin edici bir algoritmanın bulunmadığı görevleri yerine getirmelerini öğrenmek için çeşitli yaklaşımalar

kullanır. Çok sayıda potansiyel cevap bulunduğu durumlarda, doğru cevapların bazlarını geçerli olarak etiketlemektir.

Makine öğreniminin (ML) yaygın olduğu sektörler; Görüntü tanıma dayalı tıbbi tanı, kendi kendine giden arabalar için navigasyon, gibi bir çok alanda yer bulur. Bilgi toplama ve veri anlamlandırma makine öğreniminin yardımıyla gelişen 5G standartları ile bağıdaştırılıyor. Büyük e-ticaret web siteleri tarafından uygulanan öneri motorları Makine Öğrenimini kullanır.

Makine öğrenimi karmaşık olarak algılanan problemlerin analizi için mükemmel bir yöntemdir. Genel anlamda makine öğrenimin görevi sorunları tespit etmek, gelecekte olabilecek senaryoları önceden tahmin etmek ve büyük veri yığınının genel olarak kalıplarını keşfedebilme yeteneğine sahiptir ancak her türlü teknolojide olduğu gibi makine öğrenimi de tamamen mükemmel değildir. Hata yapma olasılığını arttıran nedenler; verileri elde etmenin zorluğu gerekli hesaplama gücü getirilen karmaşıklık, bazı algoritmaların uzun süren eğitim süreleri vb. Bu nedenler makine öğrenimine geçişin engeli değil erteleyicisidir.

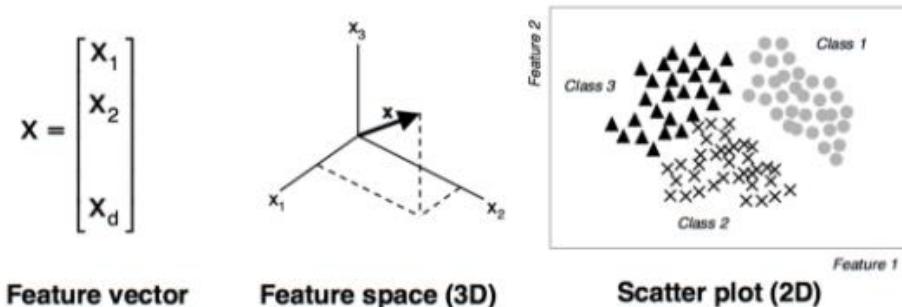
5.1. Özelliğ (Öznitelik) Vektörleri

Makine öğrenmesinde, özellik vektörleri, bir nesnenin özellikler adı verilen sayısal veya sembolik özelliklerini matematiksel olarak, kolayca analiz edilebilir bir şekilde temsil etmek için kullanılır. Makine öğreniminin ve kalıp işlemenin birçok farklı alanı için önemlidirler. Makine öğrenimi algoritmaları, algoritmaların işleme ve istatistiksel analiz yapabilmesi için tipik olarak nesnelerin sayısal bir temsilini gerektirir. Özellik vektörleri, doğrusal regresyon gibi istatistiksel prosedürlerde kullanılan açıklayıcı değişkenlerin vektörlerinin eşdeğeridir.

Aşina olabileceğiniz bir özellik vektörüne örnek RGB (kırmızı-yeşil-mavi) renk açıklamalarıdır. Bir renk, içinde ne kadar kırmızı, mavi ve yeşil olduğu ile tanımlanabilir. Bunun için bir özellik vektörü, renk = [R, G, B] olacaktır.

Bir vektör, tek sütunlu ancak çok satırlı bir matris gibi, genellikle uzamsal olarak temsil edilebilen bir sayı dizisidir. Bir özellik, bir nesnenin bir yönünün sayısal veya sembolik bir özelliğidir. Özellik vektörü, bir nesne hakkında birden çok öğe içeren bir vektördür. Nesneler için özellik vektörlerini bir araya getirmek bir özellik alanı oluşturabilir.

Özellikler, bir bütün olarak, yalnızca bir piksel veya bütün bir görüntüyü temsil edebilir. Ayrıntı düzeyi, bir kişinin nesne hakkında ne öğrenmeye veya temsil etmeye çalışığına bağlıdır. 3 boyutlu bir şekli, yüksekliğini, genişliğini, derinliğini vb. Gösteren bir özellik vektörüyle tanımlayabilirsiniz.



Özellik vektörleri, birçok analiz türüne yardımcı olmak için nesneleri sayısal bir şekilde temsil etmenin etkinliği ve pratikliği nedeniyle makine öğreniminde yaygın olarak kullanılmaktadır. Analiz için iyidirler çünkü öznitelik vektörlerini karşılaştırmak için birçok teknik vardır. İki nesnenin özellik vektörlerini karşılaştırmanın basit bir yolu Öklid mesafesini almaktır.

Görüntü işlemede, özellikler gradyan büyüğünü, renk, gri tonlama yoğunluğu, kenarlar, alanlar ve daha fazlası olabilir. Özellik vektörleri, bir görüntü hakkında özniteliklerin, listelenen örnekler gibi, öznitelik vektörlerine yerleştirildikten sonra sayısal olarak

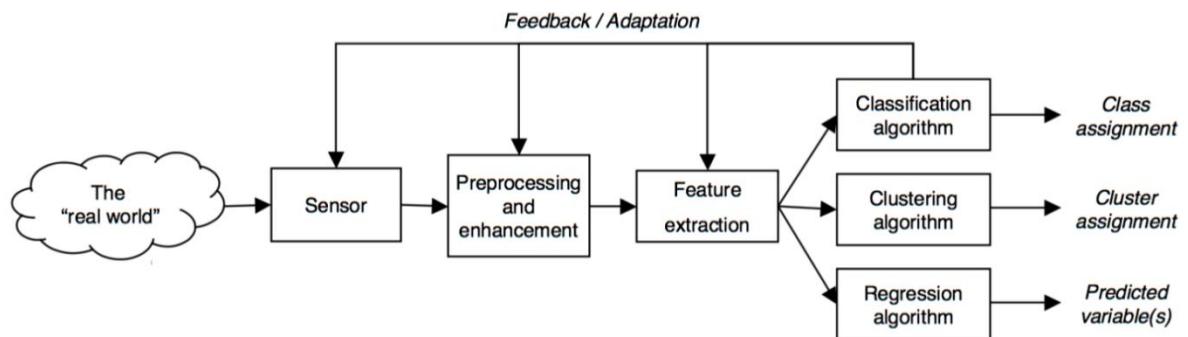
karşılaştırılabilmesinin uygun yolu nedeniyle özellikle görüntü işlemedeki analizler için popülerdir.

Konuşma tanımda özellikler ses uzunlukları, gürültü seviyesi, gürültü oranları ve daha fazlası olabilir.

İstenmeyen e-postayla mücadele girişimlerinde özellikler bol miktarda bulunur. IP konumu, metin yapısı, belirli kelimelerin sıklığı veya belirli e-posta başlıklarını olabilir.

Özellik vektörleri, makine öğrenmesinde sınıflandırma problemlerinde, yapay sinir ağlarında ve en yakın komşu algoritmalarında kullanılır.

Örütü tanıma süreçlerinde, özellik vektörleri, veri toplama ve verileri anlamlandırma arasında kullanılan araçlardır:



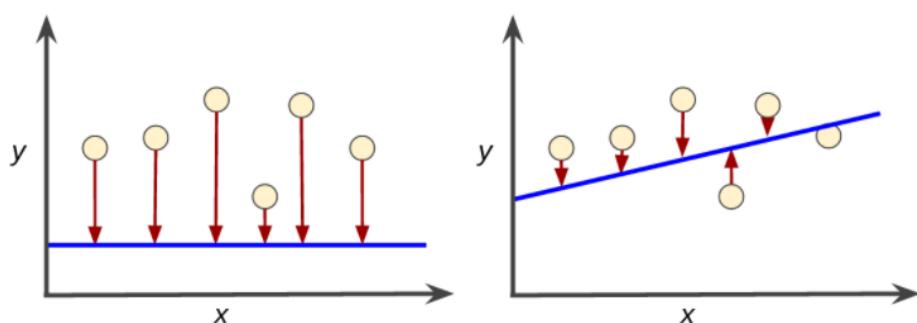
5.2. Eğitim Modelleri

Veri setlerini tanımlamak için kullanılan farklı matematiksel modeller arasından seçme işlemi “Eğitim Model Seçimi” olarak bilinir. Model seçimi istatistik, makine öğrenimi ve veri madenciliği alanlarına uygulanmaktadır.

Makine öğrenmesi modelleri iyi performans gösterebilmeleri için çok fazla veri gerektirir. Bir makine öğrenmesi modeli eğitilirken, temsili veri örneklerinin toplanması gereklidir. Eğitim setindeki veriler, bir metin topluluğuna, bir resim koleksiyonuna ve bir hizmetin tek tek kullanıcılarından toplanan verilerine kadar değişebilir.

Bir modeli eğitmek, tüm ağırlıklar ve etiketli örneklerden eşik seviye değeri bulabilmek için iyi değerleri öğrenmek (belirlemek) anlamına gelir. Denetimli öğrenmede, bir makine öğrenimi algoritması birçok örneği inceleyerek ve kaybı en aza indiren bir model bulmaya çalışarak bir model oluşturur; bu süreç empirik risk minimizasyonu denir.

Makine öğrenmesinde kayıp, kötü bir tahminin cezasıdır. Kayıp, modelin tahmininin ne kadar kötü olduğunu gösteren bir sayıdır. Modelin tahmini mükemmelse, kayıp sıfırdır; aksi takdirde kayıp daha büyütür. Makine öğrenmesinde bir modeli eğitmenin amacı, tüm örneklerde ortalama olarak düşük kayıplı bir eşik değer bulmaktır. Örneğin, aşağıdaki şekilde, solda yüksek kayıplı bir modeli ve sağda düşük kayıplı bir modeli gösterilmektedir. Şekilde okların uzunluğu kaybı temsil eder. Mavi çizgiler tahminleri temsil eder.



Şekil. Sol modelde yüksek kayıp; doğru modelde düşük kayıp.

Soldaki grafikteki okların sağdaki grafikteki benzerlerinden çok daha uzun olduğuna dikkat edin. Açıkçası, sağdaki çizimdeki çizgi, soldaki grafikteki çizgiden çok daha iyi bir tahmin modelidir. Kayıpları anlamlı bir şekilde bir araya getirecek bir matematiksel fonksiyon - bir kayıp fonksiyonu – kullanılır.

Doğrusal regresyon modelleri için karesel kayıp adı verilen bir kayıp fonksiyonu kullanılır.

$$\begin{aligned}
 \text{Tek bir örnek için kayıpların karesi} &= \text{etiket ve tahmin arasındaki farkın karesi} \\
 &= (gözlem - tahmin(x))^2 \\
 &= (y - y')^2
 \end{aligned}$$

Ortalama kareler hatası (MSE), tüm veri kümesi boyunca örnek başına düşen ortalama kare kayıptır. MSE'yi hesaplamak için, tek tek örnekler için tüm kayıpların karesi toplanır ve ardından örneklerin sayısına bölünür:

$$MSE = \frac{1}{N} \sum_{(x,y) \in D} (y - yt)^2 = \frac{1}{N} \sum_{i=1}^N (y_i - yt_i)^2$$

burada:

(x,y) bir örnektir

x, modelin tahminlerde bulunmak için kullandığı özellikler kümesidir.

y, örneğin etiketidir.

Tahmin(x), yt: özellikler kümesi ile birlikte ağırlıkların ve önyargının bir fonksiyonudur.

D, (x,y) çiftler olan birçok etiketli örneği içeren bir veri kümesidir.

N, D içindeki örneklerin sayısıdır.

MSE, makine öğreniminde yaygın olarak kullanılmasına rağmen, ne tek pratik kayıp işlevi ne de tüm koşullar için en iyi kayıp işlevi değildir.

Örnek:

Makine öğrenmesinde bir modeli eğitmenin amacı, tüm örneklerde ortalama olarak düşük kayıplı bir eşik değer bulmaktır. Bu eşik değer ortalama kareler hatası ile bulunur.

i=1...5

örnek değerler, yi= 1, 2, 3, 4, 5 ;

modelin örnekleme tahmin değerleri yti=1, 2, 7, 2, 3

$$\text{MSE} = ((1-1)^2 + (2-2)^2 + (3-7)^2 + (4-2)^2 + (5-3)^2) / 5 = (0+0+16+4+4) / 5 = 24 / 5 = 4.8$$

örnek değerler, yi= 1, 2, 3, 4, 5 ;

modelin örnekleme tahmin değerleri yti=1, 2, 4, 3, 5

$$\text{MSE} = ((1-1)^2 + (2-2)^2 + (3-4)^2 + (4-3)^2 + (5-5)^2) / 5 = (0+0+1+1+0) / 5 = 2 / 5 = 0.4$$

İkinci örnek ortalama düşük kayıplı olduğu görülmektedir.

Toplu öğrenme:

Belirli bir hesaplama programını çözmek için sınıflandırıcılar veya uzmanlar gibi birden çok model stratejik olarak oluşturulur ve birleştirilir. Bu süreç toplu öğrenme olarak bilinir. Toplu öğrenme, bir modelin sınıflandırmasını, tahminini, işlev yaklaşımını geliştirmek için kullanılır. Topluluk öğrenme, daha doğru ve birbirinden bağımsız bileşen sınıflandrıcılar oluşturduğunuzda kullanılır.

Toplu yöntemler:

Topluluk yöntemlerinin iki paradigması şunlardır:

- Sıralı topluluk yöntemleri
- Paralel topluluk yöntemleri

Bir topluluk yönteminin genel ilkesi, tek bir modele göre sağlamlığı artırmak için belirli bir öğrenme algoritması ile oluşturulmuş birkaç modelin tahminlerini birleştirmektir. Torbalama, istikrarsız tahmin veya sınıflandırma şemalarını iyileştirmek için topluluk içinde bir yöntemdir. Artırma yöntemi, kombine modelin yanılığını azaltmak için sırayla kullanılır. Artırma ve Torbalama, varyans terimini azaltarak hataları azaltabilir.

Bir öğrenme algoritmasının beklenen hatası, önyargı ve varyansa ayırtılabilir. Bir önyargı terimi, öğrenme algoritması tarafından üretilen ortalama sınıflandırıcının hedef işlevle ne kadar yakından eşleştiğini ölçer. Varyans terimi, öğrenme algoritmasının tahmininin farklı eğitim setleri için ne kadar dalgalandığını ölçer.

Toplulukta Artımlı Öğrenme algoritması, bir algoritmanın, sınıflandırıcı halihazırda mevcut olan veri kümesinden oluşturulduktan sonra mevcut olabilecek yeni verilerden öğrenme yeteneğidir.

5.3. Entropi (Bilgi Kazancı)

Rassal bir değişkenin belirsizlik ölçütü olarak bilinen Entropi, bir süreç için tüm örnekler tarafından içeren enformasyonun beklenen değeridir. **Enformasyon ise rassal bir olayın gerçekleşmesine ilişkin bir bilgi ölçütüdür.** Eşit olasılıklı durumlar yüksek belirsizliği temsil eder. Shannon'a göre bir sistemdeki durum değiştiğinde entropideki değişim kazanılan enformasyonu tanımlar. Buna göre maksimum belirsizlik durumundaki değişim muhtemelen maksimum enformasyonu sağlayacaktır. Shannon bilgisi bitlerle temsil ettiği için logartimayı iki tabanında kullanmıştır.

$$I(x) = \log_2 \frac{1}{P(x)} = -\log_2 P(x)$$

Shannon'a göre entropi, iletilen bir mesajın taşıdığı enformasyonun beklenen değeridir.

Shannon Entropisi (H) adıyla anılan terim, tüm x_i durumlarına ait $P(x_i)$ olasılıklarına bağlı bir değerdir.

$$H(X) = E(I(X)) = \sum_{1 \leq i \leq n} P(x_i) I(x_i) = \sum_{i=1}^n P(x_i) \log_2 \frac{1}{P(x_i)} = -\sum_{i=1}^n P_i \log_2 P_i$$

$$\text{Log}_2(P) = \frac{10}{3} \text{Log}_{10}(P)$$

$$H(X) = -\frac{10}{3} \sum_{i=1}^n P_i \text{Log}_{10} P_i$$

10 tabanında logaritmik ifadeler:

- $\text{Log}1=0$, $\text{Log } 2 \approx 0.3$, $\text{Log } 3 \approx 0.477$, $\text{Log } 5 \approx 0.7$, $\text{Log } 7 \approx 0.845$, $\text{Log}10=1$
- $\text{log}(a*b)=\text{log}a + \text{log}b$; $\text{log}a^n=n*\text{log}a$

Örnek:

Bir paranın havaya atılması olayı, rassal X sürecini temsil etsin. Yazı (1/2) ve tura (1/2) gelme olasılıkları eşit olduğu için X sürecinin entropisi 1 olan para atma olayı (X) gerçekleştiginde 1 bitlik bilgi kazanılacaktır.

$$H(X) = -\sum_{i=1}^2 p_i \log_2 p_i = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$$

Örnek:

Hava, nem, rüzgar, su sıcaklığı gibi değerlere göre pikniğe gidip gitmemeye karar verilmiş 4 olay sonucunda pikniğe gidildi mi? sorusunun iki yanıtı var. Evet yanıtının olasılığı: $\frac{3}{4}$, hayır yanıtının olasılığı: $\frac{1}{4}$.

Olay No	Hava	Nem	Rüzgar	Su sıcaklığı	Pikniğe gidildi mi?
1	güneşli	normal	güçlü	ılık	Evet
2	güneşli	yüksek	güçlü	ılık	Evet
3	yağmurlu	yüksek	güçlü	ılık	Hayır
4	güneşli	yüksek	güçlü	sıcak	Evet

$$H(X) = -\sum_{i=1}^2 p_i \log_2 p_i$$

$$\text{Entropy}(S)=H(x)=-(3/4)\log_2(3/4)-(1/4)\log_2(1/4)=0.811$$

Örnek:

Makine öğrenmesi algoritmasının olasılık hesaplanması sonucunda karar vermesi gerekmektedir. İki durum söz konusudur. Birinci durumun olma olasılığı, P1=0.6, olmama olasılığı, P2=0.4 ise entropisini hesaplayınız.

$$\log_2(0.6) = -0.743$$

$$\log_2(0.4) = -4/3$$

$$\text{Entropi, } H(x)=0.6*0.743+0.4*4/3=0.979$$

5.4. Kayıp Veri

Eğer veride bazı örneklerin bazı özellikleri kayıpsa izlenecek iki yol vardır:

- Kayıp özelliklere sahip örnekler veriden tamamen çıkartılır.
- Kayıp verilerle çalışabilecek şekilde algoritma düzenlenir.

Eğer kayıplı örneklerin sayısı birinci seçenek uygulanamayacak kadar çoksa ikinci seçenek uygulanmalıdır. Kayıp bilgiye sahip özellik vektörü için kazanç hesaplanırken kayıplı örnekler hariç tutularak bilgi kazancı normal şekilde hesaplanır ve daha sonra F katsayısıyla çarpılır. F, kayıpsız verinin tamamına oranıdır.

$$IG(X) = F \cdot (H(X) - H(V, X))$$

Kayıp bilgiye sahip özellik vektörü içinde en sık tekrarlanan değerin kayıp bilgi yerine yazılması da önerilen yöntemlerdendir.

Eksik değerler ortaya çıktığında veri noktalarını kolayca atamayız. Sınıflandırılacak bir test noktası da eksik değişkenlere sahip olabilir. Sınıflandırma ağaçlarının, eksik değerlerini tamamlamanın güzel bir yolu vardır. Sınıflandırma ağaçları, bir yedek ayrımla sorunu çözer. Başka bir değişkene dayalı başka bir ayrımla bulmak için, sınıflandırma ağaçları diğer tüm değişkenleri kullanarak tüm bölünmelere bakar ve optimum bölünmeye en çok benzeyen eğitim veri noktaları bölümünü veren birini arar. En iyi bölünmenin sonucunu tahmin etmeye çalışırlar.

5.5. Veri Madenciliği

Veri madenciliği, denetimsiz öğrenim yoluyla keşifsel veri analizine odaklanan ilgili bir çalışma alanıdır. Verilerdeki (geçmiş) bilinmeyen özelliklerin keşfedilmesine odaklanır. Bu veri tabanlarında bilgi keşfi analizinin bir adımıdır.

Veri Madenciliği ve Makine öğrenimi arasındaki fark nedir?

Makine öğrenimi, bilgisayarlara açıkça programlanmadan öğrenme yeteneği veren algoritmaların incelenmesi, tasarımları ve geliştirilmesi ile ilgilidir. Veri madenciliği, yapılandırılmamış verilerin bilgiyi veya bilinmeyen ilginç kalıpları çıkarmaya çalıştığı süreç olarak tanımlanabilir. Veri madenciliği işlemi sırasında, öğrenme algoritmaları kullanılır.

Makine öğrenmesi ve veri madenciliği genellikle aynı yöntemleri kullanır ve önemli ölçüde örtüşür, ancak makine öğrenmesi öğrenme verilerinden öğrenilen öngörüye odaklanırken, veri madenciliği verilerde (önceden) bilinmeyen özelliklerin keşfine odaklanır (bu veritabanlarında bilgi keşfinin analiz basamağı). Veri madenciliği birçok makine öğrenme yöntemi kullanır, ancak farklı hedefleri vardır.

Makine öğrenmesi ile genellikle aynı yöntemleri kullanır ve önemli ölçüde örtüşür, ancak makine öğrenmesi öğrenme verilerinden öğrenilen bilinen özelliklere dayanarak öngörüye odaklanırken, veri madenciliği verilerde (önceden) bilinmeyen özelliklerin keşfine odaklanır. Veritabanlarında bilgi keşfinin ilk basamağıdır.

Veri Madenciliği, büyük miktardaki veri yiğini içerisindeki desenlerin, ilişkilerin, önemli bilgilerin keşfedilmesi teknigidir. Kötüye kullanım tespiti ve anormallik tespiti yöntemleri kullanılırken yapay sinir ağları, bayes ağları ve KNN gibi sınıflandırma yöntemleri; bölünmeli, çizge tabanlı ve hiyerarşik demetleme yöntemleri, karar ağaçları ve genetik algoritmalar da mevcuttur. Ayrıca farklı yöntemlerin birleştirilmesi ile oluşturulan hibrit yöntemler de kullanılmaktadır.

Karar Destek Sistemleri

Karar Destek Sistemleri, değişik kaynaklardan toplanan bilgilerin düzenlenerek, karar modellenerek, bilgiler analiz edilerek ve değerlendirme sonuçlarını karar vericiye sunan bilgisayar tabanlı sistemlerdir. Bir karar verici için verilen kararın doğruluğu, onun yeteneklerine, deneyimine ve bilgi birikimine olduğu kadar sahip olduğu veri kümelerinin yeterliliğine de bağlıdır. Kararın başarısında, verilerin doğru depolanması, doğru sınıflanması, doğru ayıklanıp işlenmesi ve doğru yorumlanması çok önemli bir rol oynar. Bu nedenle, veri madenciliği, Karar Destek Sistemleri için etkili araçlardan biridir.

5.6. Veri Tabanı Yönetimi

Veri tabanları: Elde edilen verilerin tutulduğu alanlardır. Bir veri tabanı sistemi, birbiri ile ilişkili verilerin birikimini içeren, veriye erişimi sağlayarak veriyi yönetmeye yardımcı olan yazılım programları kümesidir.

Makine öğrenimi projelerindeki en kritik bileşenlerden biri veritabanı yönetim sistemidir. Bu sistemin yardımıyla çok sayıda veri sıralanabilir ve bunlardan anlamlı içgörüler elde edilebilir. 2019 Stack Overflow Survey raporuna göre Redis en çok sevilen veritabanı, MongoDB ise en çok aranan veritabanı.

Veri tabanları kullanım amaçlarına göre farklı isimler alır:

İlişkisel veritabanları, her biri farklı isimler alan tablolardan oluşur. Her tabloda her bir kaydın özelliklerinin değerlerini tutan alanlar ve her kayda ait bir tekil anahtar bulunur. Bir üniversitenin veritabanını ilişkisel veri tabanına örnek olarak verebiliriz. Zira her bir kişi için ayırt edici bir öğrenci numarası, hangi yılda kayıt yaptırdığı, hangi bölümde okuduğu gibi alanlar ile öğrenciye ait bilgiler saklanır. Buradan çeşitli sorular ile hangi bölümde kaç öğrencinin okuduğu, geçtiğimiz yıl kaç kişinin belli bir bölüme kayıt yaptırdığı gibi soruların cevapları bulunabilir.

İşlemsel veritabanında her bir kaydın bir işlem olduğu varsayılar. Bir marketin veri tabanını düşünecek olursak, her an bir satış yapıldığını ve her bir satışın işlemsel veri tabanında bir kayıt olarak göründüğü varsayılabılır. Bu veritabanından, bugün, ilgilenilen ürününden kaç tane satıldığı sorusunun cevabına ulaşılabilir.

Zaman serisi veritabanı düzenli zaman aralıkları ile elde edilmiş (yıllık, haftalık, günlük) verilerin tutulduğu alanlardır. Örnek olarak borsa verilerinin, stok kontrolleri sonucu alınan verilerin, sıcaklık ölçümlerinden elde edilen verilerin depolanması gösterilebilir.

Veri Ambarları:

Veri Ambarları: “Veri ambarları, tüm operasyonel işlemlerin en alt düzeydeki verilerine kadar inebilen, etkili analiz yapılabilmesi için özel olarak modellenen ve tarihsel derinliği olan veri depolama sistemi olarak tanımlanabilir.” Günlük işlemler sonucu, farklı kaynaklardan toplanan veriler, temizleme dönüştürme, birleştirme gibi işlemlerden geçirilerek, daha önce inşa edilmiş veri ambarının yapısına uygun hale getirilerek veri ambarına aktarılır. Veri ambarları, üzerinde, verilerin yüklenmesi ve erişimi dışında herhangi bir işlem yapılmasına izin vermez. Veri ambarları belirli aralıklar ile güncellenirler. Mimari açıdan veri ambarları üç farklı şekilde olabilir. İlk, işletmelerin farklı kaynaklardan (işletmenin kendi işlemsel veritabanı sistemleri ve dış kaynaklar dâhil olmak üzere) aldığı tüm verilerin tutulduğu “işletme ambarları”, ikincisi veri üzerinde çalışma yaparak karar alan kişiler için belirli

kurallara göre oluşturulmuş “veri pazarları” , sonuncusu ise işlemsel veri tabanlarının görsel hali olan “ görsel ambarlar” “dır.

Veri madenciliğinde kullanılan modeller:

Tahmin Edici Modeller : Tahmin edici modellerde, sonuçları bilinen verilerden hareket edilerek bir model geliştirilmesi ve kurulan bu modelden yararlanılarak sonuçları bilinmeyen veri kümeleri için sonuç tahmin edilmesi amaçlanmaktadır.

Tanımlayıcı Modeller : Tanımlayıcı modellerde, veri kümesinde bulunan gizli örüntülerin (olayların ve nesnelerin ortaya çıkardığı davranış değişikliklerinin desenleri) tanımlanması amaçlanmaktadır.

Veri madenciliği süreci dört aşama ile tanımlanabilir.

- İlk aşamada problem tanımlanarak veri kaynakları değerlendirilir.
- İkinci aşamada veriler kullanıma uygun hale getirilmek için hazırlanır.
- Arkasından model kurulur ve
- Nihai aşamada model değerlendirilerek kullanıma hazır hale getirilir.

Problemin Tanımlanması:

Amaç, işletme problemine verileri kullanarak çözüm getirmek olduğundan, ilk olarak ihtiyaç duyulan şey tam olarak tanımlanmalıdır. Bu problem, işletmenin ayrılmakta olan müşterisinin belirli özelliklerini tanımlayarak ona uygun davranış olabileceği gibi, kendi kaynaklarını optimum kullanabilmek için yapacağı bir planlamada gelecek dönemdeki harcamalarını tahmin etmek şeklinde de olabilir. “Bu adımda ihtiyaç duyulan şeyin tanımlanması için cevaplanması gereken sorular neyin otomatize edilmeye değer olduğu ve neyin insan içeren süreçlere bırakılması gereği, amacın ne olduğu ve hangi performans kriterlerinin daha önemli olduğu, sürecin sonucunda elde edilecek çıktıının keşif, sınıflandırma, özetleme gibi şeyler için kullanılıp kullanılmayacağı olabilir.” Problemin tanımlanması durumunda ihtiyaç duyulan iş modelinin kalibi da belirlenmiş olur.

Verilerin Hazırlanması:

Modelin kurulması için gerekli bilgilerin hazırlandığı aşamadır. Öncelikle toplam, maksimum, minimum değer gibi dağılım ölçüleri; aritmetik ortalama, ağırlıklı ortalama gibi cebirsel ölçüler veya serpilme, dağıılma diyagramı gibi grafiksel öğeler kullanılarak verilerin durumu hakkında bilgi edinilir. Verilerde eksik, hatalı, gürültülü bilgi olup olmadığı bu şekilde kontrol edilmiş olur. Eksik değerlerde kaydı dikkate almama, global sabit ile eksik değerleri doldurma, eksik değere o değişkenin ortalama değerini verme, gürültülü değerlerde regresyon ile belirli fonksiyonel kalıba sokma gibi yöntemler ile verilerdeki sıkıntı giderilebilir.

Farklı kaynaklardan gelen, aynı değişkene ait verilerin tiplerinde, alan isimlerinde uyuşmazlık olması halinde gerekli değişikliklere gidilerek tüm verileri bir arada tutabilecek yapı oluşturulmalıdır.

Bazı modellerin gereksinimlerini göz önünde bulundurmak açısından farklı dönüşümlere gitmek de veri hazırlanırken dikkate alınması gereken hususlardan olabilir. Örneğin bazı değişkenlerdeki değerler çok yüksek ise, bu değerleri normalize ederek, uzaklıklar ile çalışan kümeleme algoritmalarının öğrenme fazını hızlandırarak modelin oluşturulma aşaması için kolaylık sağlanması gereklidir.

Değişken sayısının çok yüksek olduğu, hangi değişkenlerin öneminin daha yüksek olduğuna karar verilemediği durumlarda faktör analizi, temel bileşenler analizi gibi yöntemler kullanılarak boyut indirmeleri yapılmalıdır. Zira bu indirmeler modele girecek değişken sayısını azaltarak modeli gereksiz bilgilerden ayıklar ve daha sağlıklı bir sonucun çıkışmasına zemin hazırlarlar.

Gerektiğinde kategorik değişkenlerde kategori aralıklarını genişleterek kategori sayısını azaltma veya sürekli bir değişkeni kategorik hale getirmek de verinin hazırlanmasında dikkat edilmesi gereken unsurlardandır. Çok kategorili değişkenler duruma göre modelin çalışma süresini ve sürecin performansını olumsuz etkileyebilmektedir.

Modelin Kurulması:

Modelin kurulması aşamasında birçok model denenerek veriyi en iyi temsil eden model seçilir. Verileri temsil eden en iyi modeli bulabilmek için çok sayıda model kurulmalı, en iyi sonucu alana kadar denemeye devam edilmelidir.

Modelin kuruluşu, amacımızın ne olduğuna, problemimizi ne şekilde çözmek istediğimize ve sonucun ne kadar işimize yarar olacağına göre değişebilir. Örneğin görmek istediğimiz gelecek dönemdeki tahmini ciromuz ise, sürekli bir değişkeni tahmin edeceğimiz doğrusal regresyon modelini; müşterilerimizin pasifleşme eğiliminde olup olmadıkları ise kategorik bir değişkeni tahmin edeceğimiz sınıflandırma modelleri olan karar ağaçlarını, yapay sinir ağını veya kategorik değişkenin olasılığını tahmin edeceğimiz lojistik regresyon modelini, hangi ürünlerimizin diğerlerine oranla daha çok beraber alındığı ise birelilik analizi, beraber alınan bu ürünlerin hangi sırayla alındığı, nedensellikleri ise sıralı örüntü algoritmaları kullanılabilir. Ayrıca müşterilerimizin sahip oldukları alışveriş özelliklerine göre (gelme sıklıkları, uğradıkları mağazalar, satın aldıkları ürünler vb.) belirli grplara ayırmak için kümeleme algoritmaları kullanılabilir.

Model kurulurken denetimli veya denetimsiz öğrenmeye göre farklı aşamalar uygulanmaktadır. Örneğin sınıflandırma algoritmaları kullanılırken tüm veri kümesi öğrenme

ve test kümesi olarak ayrılmalı; modelin verilerden öğrenerek oluşturulması öğrenme kümesi, doğruluğunun kontrolü ise test kümesi ile gerçekleştirilmelidir.

Kurulan modellerde birbiri ile ilişkili olan veya anlamsız olan değişkenlerin elenmesine dikkat edilmelidir. Amaç bilgi çıkarımı olduğundan ve birbiri ile ilişkili olan değişkenler bize ekstra bilgi vermediğinden, diğerine göre daha anlamlı olan değişkeni modele katmak faydamıza olacaktır.

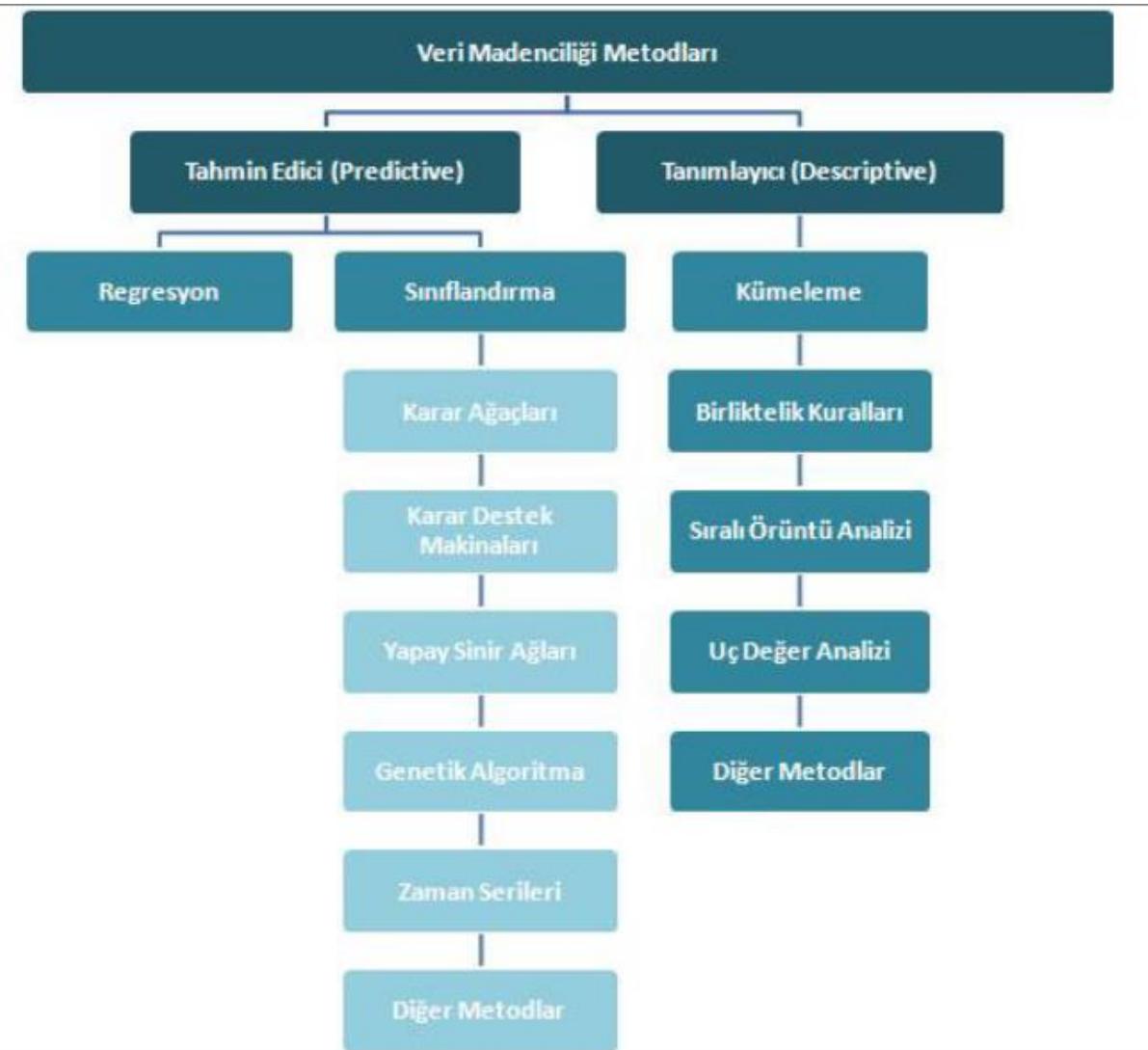
Modelin Değerlendirilmesi:

Kurulan modellerin karşılaştırılarak veri kümesini en iyi temsil eden modelin seçildiği aşamadır.

Karşılaştırma için, sınıflayıcının tahmin ettiği sınıfların oranını belirten doğruluk oranı kullanılır. Sınıflayıcının doğruluk oranının görece yüksek olması, diğer modellere göre veri kümesini daha iyi ifade ettiğini gösterebilir. Doğruluğun testi için kullanılan geçerlilik yöntemleri basit geçerlilik yöntemi, çapraz geçerlilik yöntemi, n-katlı geçerlilik yöntemi olarak sıralanabilir.

Basit geçerlilik yönteminde verilerin bir kısmı test verisi olarak ayrılır, kalan kısım üzerinde modelin öğrenimi gerçekleştirildikten sonra ayrılan kısım üzerinde test işlemi yapılır. "Bir sınıflama modelinde yanlış olarak sınıflanan olay sayısının, tüm olay sayısına bölünmesi ile hata oranı, doğru olarak sınıflanan olay sayısının tüm olay sayısına bölünmesi ile doğruluk oranı hesaplanır." Çapraz geçerlilik yöntemi daha az sayıda veri kümesine sahip olunduğu durumlarda kullanılabilir. Bu yöntemde veri kümesi rastgele seçilerek iki eşit gruba ayrılır, gruplar sırayla öğrenme ve test kümesi yapılarak elde edilen doğruluk oranlarının ortalaması kullanılır. N-katlı geçerlilik yöntemi de çapraz geçerlilik yöntemi gibi küçük veri kümeleri için kullanılmaktadır. Veri kümesi birden fazla gruba ayrılır, bir tanesi test diğerleri öğrenim için kullanılır. Test kümesi değiştirilerek doğruluk oranı hesaplanır ve elde edilen oranların ortalaması kullanılır.

Risk matrisi geçerlilik yöntemlerini görselleştirmek için kullanılabilen bir araç olabilir. Yeni çıkan bir ürünü piyasaya sürmeden önce belli sayıda kişi ile görüşülerek ürünün tutup tutmayacağı konusunda bir araştırma yapıldığını ve ürün hakkındaki fikirleri iyi ya da kötü olarak sınıflandırmak istedığımızı düşünelim. Sonuçta karşılaşacağımız sınıflandırma algoritmalarının doğruluğunu aşağıdaki şekilde görselleştirebiliriz.



6. Makine Öğreniminin Türleri

Makine öğrenmesi yaklaşımları, öğrenme sistemi için mevcut olan "sinyal" veya "geri bildirim" in doğasına bağlı olarak geleneksel olarak üç geniş kategoriye ayrırlar:

Makine öğrenmesi, bir öğrenme algoritmasının, veri yiğini içinde öğrenme veri setini deneyimledikten sonra görevler üzerinde doğru bir şekilde performans gösterebilmesidir. Öğrenme örnekleri, genel olarak bilinmeyen bazı olasılık dağılımından (olayların alanını temsil eden kabul edilir) gelir ve makine öğrenmesi, bu alan hakkında yeni durumlarda yeterince doğru tahminler üretmesini sağlayan genel bir model oluşturmmalıdır.

Makine öğrenmesi algoritmalarının hesaplamalı analizi ve performansları, hesaplamalı öğrenme teorisi olarak bilinen teorik bilgisayar biliminin bir dalıdır. Öğrenme setleri sonlu ve gelecek belirsiz olduğundan, öğrenme teorisi genellikle algoritmaların performansının garantisini vermez. Bunun yerine, performans üzerindeki olasılık sınırları oldukça yaygındır. Eğilim-varyans ayırtması, genelleme hatasını ölçmenin bir yoludur.

Genelleme bağlamında en iyi performans için, hipotezin karmaşıklığı, verilerin altında yatan fonksiyonun karmaşıklığıyla eşleşmelidir. Hipotez işlevden daha az karmaşıksa, model verileri altına yerleştirmiştir. Yanıt olarak modelin karmaşıklığı artarsa, öğrenme hatası azalır. Ancak hipotez çok karmaşıksa, model aşırı uyuma tabidir ve genelleme daha zayıf olacaktır.

Performans sınırlarına ek olarak, öğrenme kuramcılar öğrenmenin zaman karmaşıklığını ve fizibilitesini inceler. Hesaplamalı öğrenme teorisinde, bir hesaplama polinom zamanında yapılabiliyorsa uygulanabilir olarak kabul edilir. İki tür zaman karmaşıklığı sonucu vardır. Olumlu sonuçlar, polinom zamanında belirli bir fonksiyon sınıfının öğrenilebileceğini göstermektedir. Olumsuz sonuçlar, bazı sınıfların polinom zamanında öğrenilemeyeceğini göstermektedir.

Makine öğreniminde "Aşırı Uyum"

Makine öğreniminde, istatistiksel bir model, temelde yatan ilişki yerine rastgele hata veya gürültüyü tanımladığında "aşırı uyum" ortaya çıkar. Bir model aşırı derecede karmaşık olduğunda, eğitim veri türlerinin sayısına göre çok fazla parametrenin olması nedeniyle normal olarak aşırı uyum gözlemlenir. Model, aşırı uyumlu olan zayıf performans sergiliyor.

Modeli eğitmek için kullanılan kriterler, bir modelin etkinliğini değerlendirmek için kullanılan kriterlerle aynı olmadığından, aşırı uyum olasılığı oluşur.

Çok fazla veri kullanarak aşırı uydurma önlenebilir, küçük bir veri kümeniz olduğundan ve siz ondan öğrenmeye çalışığınız için aşırı uydurma nispeten gerçekleşir. Ancak küçük bir veritabanınız varsa ve buna dayalı bir modelle gelmek zorunda kalırsınız. Böyle bir durumda çapraz doğrulama olarak bilinen bir teknik kullanabilirsiniz. Bu yöntemde veri kümesi, test ve eğitim veri kümeleri olmak üzere iki bölüme ayrılır, test veri kümesi yalnızca modeli test ederken, eğitim veri kümesinde veri noktaları modelle birlikte gelir.

Bu teknikte, bir modele genellikle eğitimin (eğitim veri seti) çalıştırıldığı bilinen bir veri kümesi ve modelin test edildiği bilinmeyen verilerden oluşan bir veri kümesi verilir. Çapraz doğrulama fikri, eğitim aşamasında modeli "test etmek" için bir veri kümesi tanımlamaktır.

Makine öğrenmesinin yapay zeka ile ilişkisi:

Bilimsel bir çaba olarak, makine öğrenmesi yapay zeka arayışından doğdu. Akademik bir disiplin olarak yapay zekanın (Artificial intelligence – AI) ilk günlerinde, bazı araştırmacılar makinelerin verilerden öğrenmesini istiyorlardı. Soruna çeşitli sembolik yöntemlerle ve daha sonra "sinir ağları" olarak adlandırılan yöntemle yaklaşmaya çalışılar; bunlar çoğunlukla algılayıcılar ve daha sonra genelleştirilmiş doğrusal istatistik modellerinin yeniden icadı olarak bulunan diğer modellerdi. Olasılıksal akıl yürütme, özellikle otomatik tıbbi tanıda da kullanılmıştır.

Yapay öğrenme ile makine öğrenimi arasındaki fark nedir?

Deneysel verilere dayalı davranışlara göre algoritma tasarlama ve geliştirme, Makine Öğrenimi olarak bilinir. Yapay zeka, makine öğrenimine ek olarak bilgi sunumu, doğal dil işleme, planlama, robotik gibi diğer konuları da kapsar.

Mantıksal, bilgiye dayalı yaklaşım artan vurgu, yapay zeka ve makine öğrenmesi arasında bir kaymaya neden oldu. Olasılıksal sistemler, veri toplama ve temsil etme ile ilgili teorik ve pratik problemlerle boğulmuştu. 1980'e gelindiğinde, uzman sistemler yapay zekaya hâkim olmuştu ve istatistikler kaybolmuştu. Yapay zeka içinde sembolik bilgiye dayalı öğrenme üzerine çalışmalar devam etmiş, bu da tümevarımsal mantık programlamaya yol açmıştır.

İstatistiksel araştırma çizgisi artık yapay zekâ alanının dışında, örüntü tanıma ve bilgi edinmede uygulanmaya başlamıştı. Yapay sinir ağları araştırması aynı zamanda yapay zeka ve bilgisayar bilimleri tarafından terk edilmişti. Bu çizgi de, Hopfield, Rumelhart ve Hinton da dahil olmak üzere diğer disiplinlerden araştırmacılar tarafından “bağlantıcılık” olarak AI alanının dışında devam etti. Başlıca başarıları 1980'lerin ortalarında geri yayılımın yeniden keşfedilmesiyle geldi.

Aynı bir alan olarak yeniden düzenlenen makine öğrenmesi 1990'larda gelişmeye başladı. Alan, yapay zeka elde etmekten pratik nitelikteki çözülebilir problemlerle başa çıkmak için amacını değiştirdi. Odağı AI'dan miras aldığı sembolik yaklaşımlardan uzaklaşarak istatistik ve olasılık teorisinden ödünç alınan yöntem ve modellere kaydırıldı. 2019 itibarıyle, birçok kaynak makine öğrenmesinin Yapay Zeka'nın bir alt alanı olduğunu iddia etmeye devam ediyor. Yine de bazı uygulayıcılar, örneğin hem AI öğreten hem de sahada faaliyet gösteren bir şirket işleten Dr Daniel Hulme, makine öğrenmesi ve AI'nın ayrı olduğunu savunuyor.

Makine öğrenmesinin veri madenciliği ile ilişkisi:

Makine öğrenmesi ve veri madenciliği genellikle aynı yöntemleri kullanır ve önemli ölçüde örtüşür, ancak makine öğrenmesi öğrenme verilerinden öğrenilen öngörüye odaklanırken, veri madenciliği verilerde (önceden) bilinmeyen özelliklerin keşfine odaklanır (bu veritabanlarında bilgi keşfinin analiz basamağı). Veri madenciliği birçok makine öğrenme yöntemi kullanır, ancak farklı hedefleri vardır; Öte yandan, makine öğrenmesi aynı zamanda veri denetim yöntemlerini "denetimsiz öğrenme" veya öğrenme doğruluğunu geliştirmek için bir ön işleme adımı olarak kullanır. Makine öğrenmesinde, performans genellikle Bilinen bilgiyi yeniden üretirken, bilgi keşfi ve veri madenciliği için temel görev önceden bilinmeyen bilgilerin keşfidir. Bilinen bilgilere göre değerlendirildiğinde, bilgisiz (denetimsiz) bir yöntem diğer denetimli yöntemlerle kolayca daha iyi performans gösterirken, tipik bir veri madenciliği görevinde denetimli yöntemler öğrenme verilerinin bulunamaması nedeniyle kullanılamaz.

Makine öğrenmesinin optimizasyon ile ilişkisi:

Makine öğrenmesinin de optimizasyon ile yakın bağlantıları vardır: birçok öğrenme problemi, bir öğrenme seti örneğindeki bazı kayıp işlevlerinin en aza indirilmesi olarak formüle edilmiştir. Kayıp fonksiyonları, eğitilmekte olan modelin tahminleri ile gerçek problem örnekleri arasındaki tutarsızlığı ifade eder (örneğin, sınıflandırmada, örnekler bir etiket atamak ister ve modeller, bir dizi grubun önceden atanmış etiketlerini doğru şekilde tahmin etmek için eğitilir örnekler). İki alan arasındaki fark genelleme amacından kaynaklanmaktadır: optimizasyon algoritmaları bir öğrenme setindeki kaybı en aza indirirken, makine öğrenmesi görünmeyen numunelerdeki kaybı en aza indirmekle ilgilidir. Parametrelerin hassasiyetinin belirlenmeside optimizasyon algoritmaları, makine öğrenmesinde kritik rol oynakmaktadır.

Makine öğrenmesinin istatistiklerle ilişkisi:

Makine öğrenmesi ve istatistikler, yöntemler açısından yakından ilişkili alanlardır, ancak temel amaçlarında farklıdır: **İstatistik, bir örneklemden analiz etmeye ya da yorumlamaya yönelik çıkarımlar elde ederken, makine öğrenmesi genelleştirilebilir tahmin modelleri bulur.** Metodolojik prensiplerden teorik araçlara kadar makine öğrenmesi fikirleri istatistiklerde uzun bir geçmişe sahiptir. Ayrıca, veri bilimi terimi, genel alanı çağırırmak için bir yer tutucudur. İstatistiksel modelleme paradigmاسını ayırt edilirse: veri modeli ve makine öğrenme algoritmaları aşağı yukarı, makine öğrenmesi yöntemlerinin benimsediği istatistiksel öğrenme birleşik bir alana yol açmıştır.

İstatistik ve Makine Öğrenimi verileri tanımlamak, analiz etmek ve modellemek için işlevler ve uygulamalar sağlar. Açıklayıcı veri analizi için tanımlayıcı istatistikleri, görselleştirmeleri ve kümelemeyi kullanabilir, olasılık dağılımlarını verilere uydurabilir, Monte Carlo simülasyonları için rasgele sayılar üretebilir ve hipotez testleri gerçekleştirebilirsiniz. Regresyon ve sınıflandırma algoritmaları, Sınıflandırma ve Regresyon Öğrenici uygulamalarını kullanarak etkileşimli olarak verilerden çıkarımlar yapmanızı ve tahmine dayalı modeller oluşturmanıza olanak tanır.

Çok boyutlu veri analizi ve özellik çıkarma için araç kutusu, değişkenleri en iyi tahmin gücüyle tanımlamanıza izin veren temel bileşen analizi (PCA), düzenleme, boyutsallık azaltma ve özellik seçme yöntemleri sağlar.

Araç kutusu, destek vektör makineleri (SVM'ler), artırılmış karar ağaçları, k-ortalamalar ve diğer kümeleme yöntemleri dahil olmak üzere denetimli, yarı denetimli ve denetimsiz makine öğrenimi algoritmaları sağlar. Kısmi bağımlılık çizimleri ve LIME gibi yorumlanabilirlik tekniklerini uygulayabilir ve gömülü dağıtım için otomatik olarak C/C++ kodu oluşturabilirsiniz. Birçok araç kutusu algoritması, bellekte saklanamayacak kadar büyük veri kümelerinde kullanılabilir.

Makine öğrenmesi ve matematik:

Faydalı sonuçlar elde etmek için, belirli genel makine öğrenimi ilkeleri ve bireysel algoritmaların iç işleyişi hakkında iyi matematiksel sevgilere ihtiyaç var. İyi bir matematik alt yapısı ile,

- Problem için **doğru algoritmalar** seçilir.
- **Parametre ayarları, doğrulama stratejileri** hakkında iyi seçimler yapılır.
- **Fazla veya yetersiz uyum** tanınır.
- **Yetersiz ya da belirsiz sonuçlar** giderilir.
- Sonuçlara uygun **güven ya da belirsizlik sınırları** konulur.
- **Kodlama algoritmalarında** daha iyi bir iş yapılır veya onlar daha karmaşık hale getirilir.

Makine öğreniminde hipotezler veya model oluşturmanın üç aşaması:

- Model oluşturma
- Model testi
- Modeli uygulamak

Makine Öğrenimindeki farklı yaklaşımlar:

- Konsept ve Sınıflandırma Öğrenimi
- Sembolik vs İstatistiksel Öğrenme
- Endüktif Vs Analitik Öğrenme

Makine Öğreniminin iki tekniği şunlardır:

- Genetik Programlama
- Endüktif Öğrenme

Makine Öğrenimindeki farklı teknik türleri:

Makine öğrenmesi algoritmalarının türleri, yaklaşımlarına, girdikleri ve çıktıları veri türlerine ve çözmeleri amaçlanan görev veya sorun türlerine göre farklılık gösterirler.

- Denetimli Öğrenme (Supervised Learning Algorithms)
- Denetimsiz Öğrenme (Unsupervised Learning Algorithms)
- Yarı denetimli Öğrenme
- Takviye Öğrenme
- Transdüksiyon
- Öğrenmeyi öğrenmek

6.1. Denetimli Öğrenme

Denetimli öğrenme, etiketli eğitim verilerinden bir fonksiyon oluşturulması ile ilgili makine öğreniminin bir dalıdır. Belki de şimdilik makine ya da derin öğrenmenin ana akımıdır. **Denetimli öğrenmede, eğitim verileri bir dizi giriş ve hedef çiftinden oluşur;** burada giriş, özelliklerin bir vektörü (Öznitelik Vektörü) olabilir ve hedef, işlevin çıktı vermesi için ne istediğimizi belirtir. Hedef, sınıfın veya değer etiketinin tahmin edilmesidir.

Hedefin türüne bağlı olarak, denetimli öğrenimi kabaca iki kategoriye ayırır: **Sınıflandırma ve regresyon.** (Kategori: Aralarında herhangi yönden benzerlik, bağ ya da ilgi bulunması)

- Sınıflandırma, aralarında herhangi yönden benzerlik, bağ ya da ilgi bulunan hedefleri içerir; Görüntü sınıflandırması gibi bazı basit durumlardan makine çevirileri ve resim yazısı gibi bazı gelişmiş konulara kadar değişen örnekler.
- Regresyon, nicel (sayısal) değişkenler arasındaki ilişkilerin belirlendiği hedefleri içerir. Uygulamaların tümü bu kategoriye girer. Örneğin, stok tahmini, görüntü maskeleme ve diğerlerini içerir.

Denetimli Öğrenmenin işlevleri:

- Sınıflandırmalar
- Konuşma tanıma
- Regresyon
- Zaman serileri tahmin edilir
- Dizelere açıklama (etiket) eklenir.

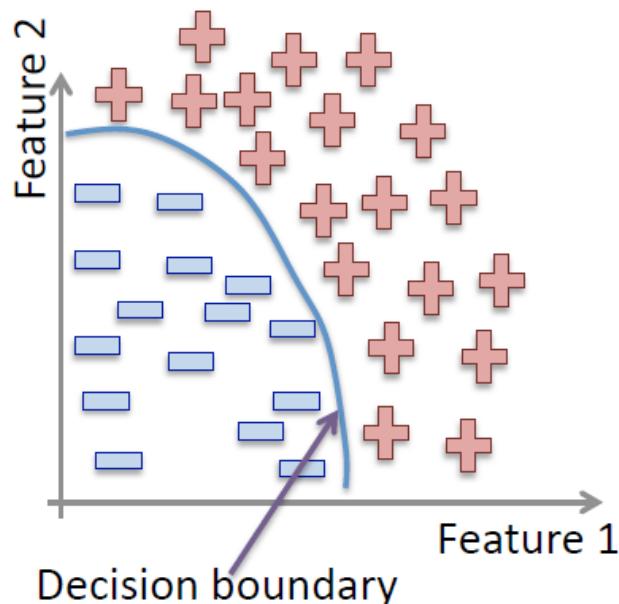
Denetimli öğrenmeye yönelik standart yaklaşım, örnek setini eğitim seti ve teste bölmektir. Makine öğrenimi gibi bilgi biliminin çeşitli alanlarında, "Eğitim Seti" olarak bilinen potansiyel olarak öngörücü ilişkiyi keşfetmek için bir dizi veri kullanılır. Eğitim seti öğrenen algoritmaya verilen bir örnektir, Test seti ise öğrenci tarafından oluşturulan hipotezlerin doğruluğunu test etmek için kullanılır ve öğrenen algoritmaların saklanan bir örnek setidir. Eğitim seti, test setinden farklıdır.

Meyvelerden oluşan torbayı taşıyan robotun taşıdığı torba parçalanır ve tüm meyveler birbirine karışır. Robot, topladığı meyveleri önceden etiketlediğinden hemen ayırtır. Bu örnekteki robotta hangi öğrenme algoritması bulunmaktadır.

Denetimli öğrenmenin ne olduğunu anlamak için bir örnek kullanacağımız. Örneğin, bir çocuğa içinde on aslan, on maymun, on fil ve diğerleri gibi her türden on hayvanın bulunduğu 100 oyuncak hayvan veriyoruz. Daha sonra, çocuğa bir hayvanın farklı özelliklerine (özelliklerine) dayalı olarak farklı hayvan türlerini tanımayı öğretiyoruz. Rengi turuncuya, o zaman bir aslan olabilir. Gövdesi büyük bir hayvansa, fil olabilir.

Çocuğa hayvanları nasıl ayırt edeceğini öğretiriz, bu denetimli öğrenmeye bir örnektir. Şimdi çocuğa farklı hayvanlar verdığımızde, onları uygun bir hayvan grubuna ayırmalıdır.

Aynısı bilgisayarlar için de geçerlidir. Onlara gerçek etiketli değerleri ile binlerce veri noktası sağlıyoruz (Etiketli veriler, özellik değerleriyle birlikte farklı grplara sınıflandırılır). Daha sonra eğitim döneminde farklı özelliklerinden ders çıkarır. Eğitim dönemi bittikten sonra eğitimli modelimizi tahmin yapmak için kullanabiliriz. Makineyi zaten etiketli verilerle beslediğimizi, bu nedenle tahmin algoritmasının denetimli öğrenmeye dayandığını unutmayın. Kısaca bu örnekteki tahminlerin etiketli verilere dayandığını söyleyebiliriz.



Denetimli öğrenme algoritmaları:

- K-En Yakın Komşular
- Doğrusal Regresyon
- Lojistik regresyon
- Rastgele Orman
- Gradyan Güçlendirilmiş Ağaçlar
- Destek Vektör Makineleri (SVM)
- Naive Bayes
- Nöral ağlar
- Karar ağaçları

Bilgisayar, bir "öğretmen" tarafından verilen örnek girişler ve istenen çıktıları ile sunulur ve amaç, girdileri çıktılarla eşleyen genel bir kural öğrenmektir.

Denetimli öğrenme algoritmaları, hem girdileri hem de istenen çıktıları içeren bir veri kümесinin matematiksel bir modelini oluşturur. Veriler, öğrenen verileri olarak bilinir ve bir dizi öğrenmeörneğinden oluşur. Her öğrenmeörneğinde, denetim sinyali olarak da bilinen bir veya daha fazla giriş ve istenen çıkış bulunur. Matematiksel modelde, her öğrenmeörneği baze özelliğin vektörü olarak adlandırılan bir dizi veya vektörle temsil edilir ve öğrenme verileri bir matrisle temsil edilir. Nesnel bir fonksiyonun yinelemeli optimizasyonu yoluyla, denetimli öğrenme algoritmaları yeni girdilerle ilişkili çıktıyı tahmin etmek için kullanılabilecek bir işlevi öğrenir. Optimal bir fonksiyon, algoritmanın egzersiz verilerinin bir parçası olmayan girişler için çıkışını doğru bir şekilde belirlemesine izin verecektir. Zaman içinde çıktılarının veya tahminlerinin doğruluğunu artıran bir algoritmanın bu görevi yerine getirmeyi öğrendiği söylenir.

Denetimli öğrenme algoritması türleri sınıflandırma ve regresyonu içerir. Sınıflandırma algoritmaları, çıktılar sınırlı bir değer kümesiyle sınırlandığında ve regresyon algoritmaları, çıktılar bir aralık içinde herhangi bir sayısal değere sahip olduğunda kullanılır. Örnek olarak, e-postaları filtreleyen bir sınıflandırma algoritması için girdi gelen bir e-posta olur ve çıktı da e-postanın dosyalanacağı klasörün adı olur.

Benzerlik öğrenme, regresyon ve sınıflandırma ile yakından ilişkili olan denetimli bir makine öğrenmesi alanıdır, ancak amaç iki nesnenin ne kadar benzer veya ilişkili olduğunu ölçen bir benzerlik işlevi kullanan örneklerden öğrenmektir. Sıralama, öneri sistemleri, görsel kimlik takibi, yüz doğrulaması ve konuşmacı doğrulaması gibi uygulamaları vardır.

Denetimli Öğrenmede iyi olasılıkları tahmin etmek için kullanılan iki yöntem şunlardır:

- Platt Kalibrasyonu
- İzotonik Regresyon

Bu yöntemler ikili sınıflandırma için tasarlanmıştır ve önemsiz değildir.

Sıralı Denetimli Öğrenme problemlerini çözmek için farklı yöntemler şunlardır:

- Sürgülü pencere yöntemleri
- Tekrarlayan sürgülü pencereler
- Gizli Markow modelleri
- Maksimum entropi Markow modelleri
- Koşullu rastgele alanlar
- Grafik trafo ağları

6.2. Denetimsiz Öğrenme

Öğrenme algoritmasına hiçbir etiket verilmez ve girdisinde yapı bulmak için tek başına bırakılır. Denetimsiz öğrenme kendi içinde bir hedef (verilerdeki gizli kalıpları keşfetme) veya bir sona doğru bir araç (özellik öğrenme) olabilir.

- k-means clustering
- t-SNE (t-Distributed Stochastic Neighbor Embedding)
- PCA (Principal Component Analysis)
- Association rule

Denetimsiz öğrenme algoritmaları, yalnızca girdileri içeren bir veri yiğinini alır ve verileri yapısal olarak gruplar veya kümeler. Bu nedenle algoritmalar, etiketlenmemiş, sınıflandırılmamış veya kategorize edilmemiş test verilerinden öğrenilir. Geri bildirime yanıt vermek yerine, denetimsiz öğrenme algoritmaları verilerdeki ortaklıkları tanımlar ve her yeni veri parçasında bu tür ortak özelliklerin varlığına veya yokluğuna bağlı olarak tepki verir. Denetimsiz öğrenmenin merkezi bir uygulaması, olasılık yoğunluk işlevini bulmak gibi istatistiklerde yoğunluk tahmini alanıdır. Denetimsiz öğrenme, veri özelliklerini özetleme ve açıklamayı içeren diğer alanları kapsar.

Denetimsiz Öğrenmenin işlevleri:

- Veri kümeleri bulunur,
- Verilerin düşük boyutlu temsilleri bulunur,
- Verilerde ilginç yönler bulunur,
- İlginç koordinatlar ve korelasyonlar elde edilir,
- Yeni gözlemler ya da veritabanı elde edilir.

Elmalardan oluşan torbayı taşıyan robotun taşıdığı torba parçalanır ve tüm elmalar (Çürüük, büyük, küçük, ort boy, ...) birbirine karışır. Robot, topladığı elmaları önceden etiketlemediğinden hemen ayırtıramaz. Tek başına farklılıklarını öğrenerek ayırtırma işleme yapar. Bu örnekteki robotta hangi öğrenme algoritması bulunmaktadır.

Küme analizi, bir grup gözlemin alt kümelere (kümeler olarak adlandırılır) atanmasıdır, böylece aynı kümedeki gözlemler önceden belirlenmiş bir veya daha fazla kriter'e göre benzerken, farklı kümelerden alınan gözlemler farklıdır. Farklı kümeleme teknikleri, genellikle bazı benzerlik ölçütleri ile tanımlanan ve örneğin iç kompaktlık veya aynı kümeyi üyeleri arasındaki benzerlik ve ayırmaya, kümeler arasındaki fark ile değerlendirilen verilerin yapısı üzerinde farklı varsayımlar yapar. Diğer yöntemler tahmini yoğunluk ve grafik bağlantısına dayanır.

Denetimli öğrenmenin aksine, denetimsiz öğrenme, verilerdeki gizli yapıları tanımlayan bir işlev olan etiketlenmemiş verilerden kaynaklanır.

Belki de denetimsiz öğrenmenin en temel türü, PCA, t-SNE gibi boyut azaltma yöntemleridir; PCA genellikle veri ön işlemede kullanılır ve t-SNE genellikle veri görselleştirmede kullanılır.

Daha gelişmiş bir dal, verilerdeki gizli kalıpları araştıran ve daha sonra bunlar hakkında tahminlerde bulunan kümelemedir; örnekler arasında K-ortalama kümeleme, Gauss karışım modelleri, gizli Markov modelleri ve diğerleri bulunur.

Derin öğrenmenin rönesansı ile birlikte, denetimsiz öğrenme, verileri manuel olarak etiketlemekten bizi kurtardığı için gittikçe daha fazla dikkat çekiyor. Derin öğrenmeninlığında, iki tür denetimsiz öğrenmeyi ele alıyoruz: temsili öğrenme ve üretken modeller.

Temsil öğrenme, bazı aşağı akış görevleri için yararlı olan üst düzey bir temsili özelliği damıtmayı amaçlarken, üretken modeller bazı gizli parametrelerden girdi verilerini yeniden üretmeyi amaçlamaktadır.

Denetimsiz öğrenme göründüğü gibi çalışır. Bu tür algoritmalarla, etiketlenmiş verilere sahip değiliz. Bu yüzden makinenin giriş verilerini işlemesi ve çıktı hakkında sonuçlara varmaya çalışması gereklidir. Örneğin, şekil oyuncağı verdığımız çocuğu hatırlıyor musunuz? Bu durumda, farklı şekiller için mükemmel şekil deliğini bulmayı kendi hatalarından öğrenecekti.

Ancak asıl mesele, çocuğu şekillere uyacak yöntemleri öğreterek beslemiyor olmamızdır (etiketli veriler olarak adlandırılan makine öğrenimi amaçları için). Ancak çocuk oyuncağın farklı özelliklerinden öğrenir ve onlar hakkında bir sonuca varmaya çalışır. Kısacası, tahminler etiketlenmemiş verilere dayanmaktadır.

Denetimsiz öğrenme algoritmalarına örnekler:

- Boyut Küçültme
- Yoğunluk Tahmini
- Pazar Sepeti Analizi
- Üretken düşmanlık ağları (GAN'lar)
- Kümeleme

6.3. Yarı denetimli öğrenme

Yarı denetimli öğrenme, denetimsiz öğrenme (herhangi bir etiketlenmiş öğrenme verisi olmadan) ve denetimli öğrenme (tamamen etiketlenmiş öğrenme verisi ile) arasındadır. Bazı öğrenme örnekleri öğrenme etiketlerinin eksik olmasına rağmen, birçok makine öğrenmesi araştırmacısı, etiketlenmemiş verilerin, az miktarda etiketlenmiş verilerle birlikte kullanıldığında, öğrenme doğruluğunda önemli bir gelişme sağlayabildiğini bulmuştur.

Zayıf denetimli öğrenmede, öğrenme etiketleri gürültülü, sınırlı veya kesin değildir; bununla birlikte, bu etiketlerin elde edilmesi genellikle daha ucuzdur, bu da daha büyük etkili öğrenme setleriyle sonuçlanır.

6.4. Takviyeli Öğrenme

Bir bilgisayar programı, belirli bir hedefi (araç kullanmak veya rakibe karşı oyun oynamak gibi) gerçeklestirmesi gereken **dinamik bir ortamla etkileşime girer**. Sorun alanı içinde ilerledikçe, program ödüllendirmeye benzeyen ve performansı en üst düzeye çıkarmaya çalışan geri bildirim sağlar.

Takviye öğrenmesi, yazılım temsilcilerinin kümülatif ödül kavramını en üst düzeye çıkarmak için bir ortamda nasıl işlem yapmaları gereğiyle ilgili bir makine öğrenmesi alanıdır. **Genelliği nedeniyle, oyun teorisi, kontrol teorisi, yüneylem araştırması, bilgi teorisi, simülasyon tabanlı optimizasyon, çok etmenli sistemler, sürü zekası, istatistikler ve genetik algoritmalar gibi birçok disiplinde çalışılmaktadır**. Makine öğrenmesinde, ortam tipik olarak bir Markov Karar Süreci (MDP) olarak temsil edilir. Pek çok takviye öğrenme algoritması dinamik programlama teknikleri kullanır. Takviye öğrenme algoritmaları, MDP'nin kesin bir matematiksel modeli hakkında bilgi sahibi değildir ve kesin modeller mümkün olmadığında kullanılır. Takviye öğrenme algoritmaları otonom araçlarda veya bir insan rakibe karşı bir oyun oynamayı öğrenmek için kullanılır.

- Q-Learning
- Temporal Difference (TD)
- Monte-Carlo Tree Search (MCTS)
- Asynchronous Actor-Critic Agents (A3C)

6.5. Özellik Öğrenme

Öğrenme algoritmaları, genellikle eğitim sırasında sağlanan girdilerin daha iyi temsilini keşfetmeyi amaçlamaktadır. Klasik örnekler temel bileşenler analizi ve küme analizini içerir. Temsili öğrenme algoritmaları olarak da adlandırılan özellik öğrenme algoritmaları, genellikle girdilerindeki bilgileri korumaya çalışır, ancak sınıflandırma veya tahminler gerçekleştirmeden önce genellikle bir ön işleme adımı olarak yararlı hale getirecek şekilde dönüştürür. Bu teknik, bilinmeyen veri üreten dağıtımdan gelen girdilerin yeniden yapılandırılmasına izin verirken, bu dağıtım altında mantıksız olan yapılandırmalara mutlaka sadık kalmaz. Bu, manuel özellik mühendisliğinin yerini alır ve bir makinenin hem özelliklerini öğrenmesini hem de belirli bir görevi gerçekleştirmek için kullanmasını sağlar.

Özellik öğrenmesi denetimli veya denetimsiz olabilir. Denetimli özellik öğrenmede, özellikler etiketli giriş verileri kullanılarak öğrenilir. Örnekler arasında yapay sinir ağları, çok katmanlı algılayıcılar ve denetimli sözlük öğrenmesi sayılabilir. Denetimsiz özellik öğrenmede, özellikler etiketlenmemiş girdi verileriyle öğrenilir. Örnekler arasında sözlük öğrenmesi, bağımsız bileşen analizi, otomatik kodlayıcılar, matris çarpanlarına ayırma ve çeşitli kümeleme biçimleri bulunmaktadır.

Manifold öğrenme algoritmaları, öğrenilen sunumun düşük boyutlu olması kısıtlaması altında bunu yapmaya çalışır. Seyrek kodlama algoritmaları, öğrenilen sunumun seyrek olduğu, yani matematiksel modelin çok sayıda sıfır olduğu kısıtlaması altında bunu yapmaya çalışır. Çok satırlı altuzay öğrenme algoritmaları, düşük boyutlu göstergeleri, çok boyutlu veriler için tensör göstergelerinden, daha yüksek boyutlu vektörlere dönüştürmeden doğrudan öğrenmeyi amaçlamaktadır. Derin öğrenme algoritmaları birden çok temsil düzeyini ya da bir özellik hiyerarşisini keşfeder; daha düşük düzey özellikler (ya da üretme) açısından daha yüksek düzey, daha soyut özellikler tanımlanır. Akıllı bir makinenin, gözlemlenen verileri açıklayan temel varyasyon faktörlerini çözen bir temsili öğrenen bir makine olduğu ileri sürülmüştür.

Özellik öğrenme, sınıflandırma gibi makine öğrenmesi görevlerinin genellikle matematiksel ve hesaplamaya uygun olarak işlenmesi için girdi gerektirmesi gerçeğe motiva edilir. Bununla birlikte, görüntüler, video ve duyusal veriler gibi gerçek dünya verileri, belirli özellikleri algoritmik olarak tanımlama girişimlerine yol açmamıştır. Bir alternatif, açık algoritmalara dayanmadan, bu özellikleri veya göstergeleri muayene yoluyla keşfetmektedir.

6.6. Diğer Öğrenme Yöntemleri

Kendi kendine öğrenme:

Makine öğrenmesi paradigmaları olarak kendi kendine öğrenme 1982'de Crossbar Adaptive Array (CAA) adı verilen kendi kendine öğrenebilen bir sinir ağı ile tanıtıldı. Dış ödüller ve dış öğretmen tavsiyeleri olmayan bir öğrenmedir. CAA kendi kendine öğrenme algoritması, çapraz çubuk şeklinde, sonuç durumlarıyla ilgili eylemler ve duygular (duygular)larındaki kararları hesaplar. Sistem, biliş ve duyu arasındaki etkileşim tarafından yönlendirilir. Kendi kendine öğrenme algoritması $W = || w(a, s) ||$ bellek matrisini günceller böylece her yinelemede aşağıdaki makine öğrenme rutini yürütür.

Seyrek sözlük öğrenme:

Seyrek sözlük öğrenmesi, bir eğitim örneğinin temel işlevlerin doğrusal bir kombinasyonu olarak temsil edildiği ve seyrek bir matris olduğu varsayılan bir özellik öğrenme yöntemidir. Yöntem güçlü bir şekilde NP-zordur ve yaklaşık olarak çözülmesi zordur. Seyrek sözlük öğrenmesi için popüler bir sezgisel yöntem K-SVD algoritmasıdır. Seyrek sözlük öğrenmesi çeşitli bağamlarda uygulanmıştır. Sınıflandırmada sorun, daha önce görülmemiş bir eğitim örneğinin ait olduğu sınıfı belirlemektir. Her sınıfın önceden oluşturulduğu bir sözlük için, sınıfla ilgili sözlük tarafından en iyi şekilde temsil edilen yeni bir eğitim örneği ilişkilendirilir. Görüntü parazitlenmesinde seyrek sözlük öğrenmesi de uygulanmıştır. Ana fikir, temiz bir görüntü yamasının bir görüntü sözlüğü ile seyrek olarak temsil edilebileceğidir, ancak gürültü olamaz.

Robot öğrenme:

Gelişimsel robot biliminde, robot öğrenme algoritmaları, öz rehberli keşif ve insanlarla sosyal etkileşim yoluyla kümülatif olarak yeni beceriler kazanmak için müfredat olarak da bilinen kendi öğrenme deneyimleri dizilerini oluşturur. Bu robotlar aktif öğrenme, olgunlaşma, motor sinerjileri ve taklit gibi rehberlik mekanizmalarını kullanır.

Birleşik öğrenme:

Birleşik öğrenme, eğitim sürecini ademi merkeziyetçi hale getiren ve verilerini merkezi bir sunucuya göndermeye gerek kalmadan kullanıcıların gizliliğinin korunmasına izin veren eğitim makinesi öğrenme modellerine uyarlanmış bir Dağıtılmış Yapay Zeka biçimidir. Bu, eğitim sürecini birçok cihaza dağıtarak verimliliği de arttırmır. Örneğin, Gboard, bireysel aramaları Google'a geri göndermek zorunda kalmadan kullanıcıların cep telefonlarında arama sorusu tahmin modellerini eğitmek için birleşik makine öğrenmesi kullanır.

Sıralı öğrenme:

Sıralı öğrenme, mantıklı bir şekilde öğretme ve öğrenme yöntemidir.

Sıralı öğrenme sürecini kategorize edebileceğiniz farklı kategoriler:

- Sıra tahmini
- Sıra oluşturma
- Sıra tanıma
- Sıralı karar

Toplu istatistiksel öğrenme:

Istatistiksel öğrenme teknikleri, görünmeyen veya gelecekteki veriler hakkında tahminlerde bulunabilen bir dizi gözlemlenen veriden bir işlevi veya öngörücüyü öğrenmeye izin verir. Bu teknikler, öğrenilen tahmincinin gelecekteki görünmeyen veriler üzerindeki performansı hakkında, veri oluşturma sürecine ilişkin istatistiksel bir varsayıma dayalı olarak garantiler sağlar.

PAC öğrenimi:

PAC (Probably Approximately Correct) öğrenme, öğrenme algoritmalarını ve bunların istatistiksel verimliliklerini analiz etmek için tanıtılan bir öğrenme çerçevesidir.

PCA (Temel Bileşenler Analizi), KPCA (Çekirdek Tabanlı Temel Bileşen Analizi) ve ICA (Bağımsız Bileşen Analizi), boyut azaltma için kullanılan önemli özellik çıkarma teknikleridir.

Endüktif (Tümevarım) makine öğrenimi:

Tümevarımlı makine öğrenimi, bir sistemin gözlemlenen bir dizi örnekten genel bir kural oluşturmaya çalıştığı örneklerle öğrenme sürecini içerir.

Endüktif Mantık Programlama (ILP), arka plan bilgisini ve örnekleri temsil eden mantıksal programlamayı kullanan bir makine öğrenimi alt alanıdır.

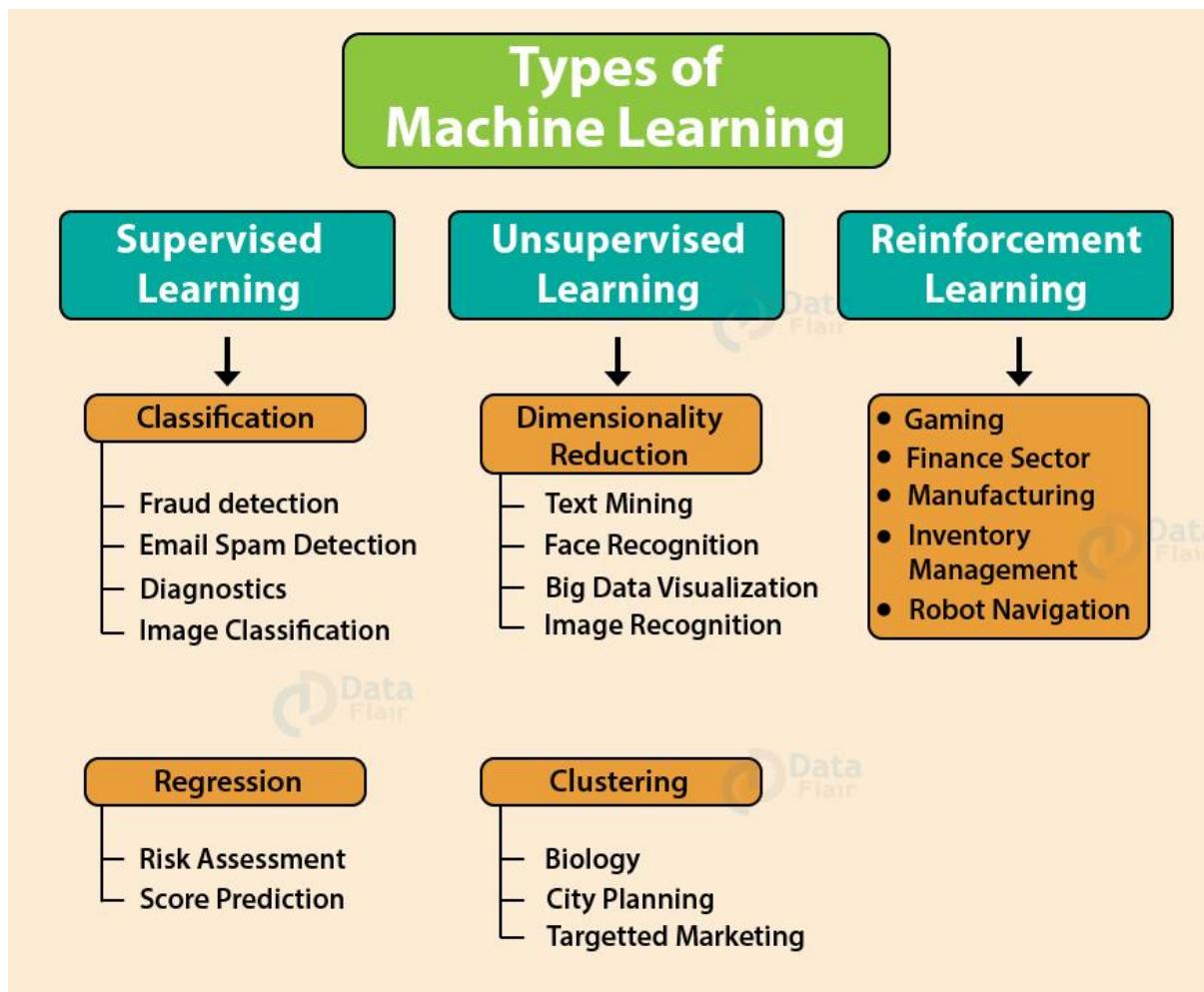
Endüktif mantık programlama (ILP), giriş örnekleri, arka plan bilgisi ve hipotezler için tekdüze bir sunum olarak mantık programlamayı kullanarak kural öğrenmeye bir yaklaşımdır. Bilinen arka plan bilgisinin bir kodlaması ve gerçeklerin mantıksal bir veritabanı olarak temsil edilen bir dizi örnek önüne alındığında, bir ILP sistemi, tüm olumlu ve olumsuz örnekleri içeren varsayılmış bir mantık programı türetecektir. Endüktif programlama, fonksiyonel programlar gibi hipotezleri (ve sadece mantık programlamayı değil) temsil etmek için her türlü programlama dilini göz önünde bulunduran ilgili bir alandır.

Endüktif mantık programlama özellikle biyoinformatik ve doğal dil işlemede yararlıdır. Gordon Plotkin ve Ehud Shapiro, endüktif makine öğrenmesi için ilk teorik temeli mantıksı bir

ortamda ortaya koydu. Shapiro ilk uygulamalarını (Model Çıkarım Sistemi) 1981'de kurdu: mantık programlarını pozitif ve negatif örneklerden induktif olarak çıkartan bir Prolog programı. Buradaki tümevarım terimi, iyi düzenlenmiş bir kümenin tüm üyeleri için bir özellik kanıtlayan, matematiksel tümevarım yerine gözlemlenen gerçekleri açıklayan bir teori öneren felsefi tümevarım anlamına gelir.

Tembel öğrenme algoritması:

Örnek tabanlı öğrenme algoritması, sınıflandırma gerçekleştirilinceye kadar induksiyon veya genelleme sürecini geciktirdiği için Tembel öğrenme algoritması olarak da adlandırılır.



7. Makine Öğrenmesi Algoritmaları

Makine Öğrenimi, kurala dayalı çıkarımlar yapar. İlişkilendirme kuralı öğrenme, büyük veritabanlarındaki değişkenler arasındaki ilişkileri keşfetmek için kural tabanlı bir makine öğrenme yöntemidir. Makine öğrenmede veritabanlarında keşfedilen güçlü kuralların bir miktar "ilginçlik" ölçüsü kullanılarak belirlenmesi amaçlanmıştır.

Kural tabanlı öğrenme, bilgiyi depolamak, manipüle etmek veya uygulamak için "kuralları" tanımlayan, öğrenen veya geliştiren herhangi bir makine öğrenme yöntemi için genel bir terimdir. Kural tabanlı öğrenme algoritmasının belirleyici özelliği, sistem tarafından yakalanan bilgileri topluca temsil eden bir dizi ilişkisel kuralın tanımlanması ve kullanılmasıdır. Bu, bir tahminde bulunmak için herhangi bir örneğe evrensel olarak uygulanabilen tekil bir modeli yaygın olarak tanımlayan diğer makine öğrenme algoritmalarının aksine, kural tabanlı makine öğrenme yaklaşımları, öğrenme sınıflandırıcı sistemleri, ilişkilendirme kuralı öğrenme ve yapay bağışıklık sistemlerini içerir.

Makine öğrenmesi algoritmalarının temel bileşenleri:

- Hesaplamalar için çeşitli kütüphaneler kullanıldığından gerekli kütüphaneler aktarılır.
- Veritabanı dosyaları okunur.
- Etiketli verilerin ağırlıklı ortalama, standart sapma gibi istatistiksel analiz özellikleri belirlenir ve yorumlanır.
- Verilerin grafiği çizilir.
- Değerler tahmin edilirken dikkate alınmak istenilen özellikler ve katsayılar seçilir.
- Bir modelin doğruluğunu kontrol etmek için, veriler eğitim ve test veri setlerine ayrıılır.
- Model eğitilir
- Test veri seti için bir tahmin fonksiyonu bulunur.
- Test verilerinin doğruluğu kontrol edilir: Gerçek değerleri veri setindeki tahmin edilen değerlerle karşılaştırılarak bir modelin doğruluğu kontrol edilebilir.

Makine Öğreniminin beş popüler algoritması:

- Karar ağaçları
- Sinir Ağları (geri yayılım)
- Olasılık ağları
- En yakın komşu
- Vektör makineleri desteklemek

Algoritmadan bağımsız makine öğrenimi nedir?

Matematiksel temellerin belirli bir sınıflandırıcıdan bağımsız olduğu veya öğrenme algoritmasının algoritmadan bağımsız makine öğrenimi olarak adlandırıldığı yerlerde makine öğrenimi?

Makine Öğreniminde Algılayıcı Nedir?

Makine Öğreniminde, Perceptron, girdinin birkaç olası ikili olmayan çıktıdan birine denetimli sınıflandırılması için bir algoritmadır.

İlişkisel değerlendirme tekniklerinin önemli bileşenleri şunlardır:

- Veri toplama
- Ground Truth Edinimi
- Çapraz Doğrulama Tekniği
- Sorgu Türü
- Puanlama Metriği
- Önem Testi

Robotik ve bilgi işlemede sıralı tahmin probleminin ortaya çıktığı alanlar:

- Taklit Öğrenme
- Yapılandırılmış tahmin
- Model tabanlı pekiştirmeli öğrenme

Makine Öğrenmesi Algoritmaları:

1- Regresyon (Tahmin)

Sürekli değerleri tahmin etmek için regresyon algoritmaları kullanılır.

Regresyon algoritmaları:

- Doğrusal Regresyon
- Polinom Regresyon
- Üstel Regresyon
- Lojistik regresyon
- Logaritmik Regresyon

2-Sınıflandırma

Bir dizi ögenin sınıfını veya kategorisini tahmin etmek için sınıflandırma algoritmaları kullanılır.

Sınıflandırma algoritmaları:

- K-En Yakın Komşular
- Karar ağaçları
- Rastgele Orman
- Destek Vektör Makinesi
- Naive Bayes

3- Kümeleme

Özetlemek veya verileri yapılandırmak için kümeleme algoritmaları kullanılır.

Kümeleme algoritmaları:

- K-means
- DBSCAN
- Mean Shift
- Hierarchical

4- İlişkilendirme

Birlikte meydana gelen öğeleri veya olayları ilişkilendirmek için ilişkilendirme algoritmaları kullanıyoruz.

İlişkilendirme algoritmaları:

- Apriori

5- Anomali (Sapma) Algılama:

Anormal etkinlikleri ve dolandırıcılık tespiti gibi olağanüstü durumları keşfetmek için anormallik algılama kullanılır.

Veri madenciliğinde, aykırı tespit olarak da bilinen anomali tespiti, verilerin çoğundan önemli ölçüde farklılık göstererek şüphe uyandıran nadir maddelerin, olayların veya gözlemlerin tanımlanmasıdır. Tipik olarak, anormal kalemler banka sahtekarlığı, yapısal bir kusur, tıbbi sorunlar veya bir metindeki hatalar gibi bir konuyu temsil eder. Anomalilere aykırı değerler, yenilikler, gürültü, sapmalar ve istisnalar denir.

Özellikle, kötüye kullanım ve ağ izinsiz giriş tespiti bağlamında, ilginç nesneler genellikle nadir nesneler değil, faaliyette beklenmedik patlamalardır. Bu model, bir aykırı değerin nadir bir nesne olarak ortak istatistiksel tanımına uymaz ve uygun şekilde toplanmadığı sürece birçok aykırı algılama yöntemi (özellikle denetimsiz algoritmalar) bu verilerde başarısız olur. Bunun yerine, bir küme analiz algoritması bu örüntüler tarafından oluşturulan mikro kümeleri tespit edebilir.

Üç geniş anomali tespit tekniği kategorisi bulunmaktadır. Gözetimsiz anomali tespit teknikleri, veri kümesindeki örneklerin çoğunun normal olduğu varsayımlıyla etiketlenmemiş bir test veri kümesindeki anormallikleri, veri kümesinin geri kalan kısmına en az uyan görünen örnekleri arayarak tespit eder. Denetimli anomali algılama teknikleri, "normal" ve "anormal" olarak etiketlenmiş ve bir sınıflandırıcıyı (diğer birçok istatistiksel sınıflandırma problemi için temel fark, aykırı algılamanın doğasında dengesiz doğasıdır) içeren bir veri seti gerektirir. Yarı denetimli anomali tespit teknikleri, belirli bir normal eğitim veri setinden normal davranışını temsil eden bir model oluşturur ve daha sonra model tarafından bir test örneğinin üretilme olasılığını test eder.

6- Sıra Desen Madenciliği

Örütü - Pattern: Bir nesnenin ya da olayın iki veya üç boyutlu, uzaysal ve geometriksel davranışının matematiksel ifadesidir. Diğer bir ifadeyle, nesnenin davranışları ile ilgili uzayda gözlemebilir veya ölçülebilir geometrik bilgilerdir. Tersinden desen madenciliği. Örütü hazırlanır. Veri yiğini içersinde gezinir. Hedef geldiğinde uyanır, kendisini kabul ettirir. Ayrıca veri yiğisi içerisinde dolaşan ve arayan desenler geliştirilmektedir.

Örütü Tanımanın kullanıldığı alanlar:

- Örütü Tanıma şu alanlarda kullanılabilir:
- Bilgisayar görüşü
- Konuşma tanıma
- Veri madenciliği
- İstatistik
- Gayri Resmi Erişim
- Biyo-Bilişim

Bir dizideki veri örnekleri arasındaki sonraki veri olaylarını tahmin etmek için sıralı model madenciliği kullanıyoruz.

Dünya ve ötesinde kainat sürekli veri oluşturan sinyal kaynakları (Elektromanyetik, elektrik, ısı, renk, ses, titreşim, çekimsel kuvvetler, ...) ile doludur. Bu kaynaklarından yayılan sinyal örüntüleri yayıldıkları oratmlar ile etkileşime girerler. O halde bu örüntülerin davranış değişikliklerinden kestirimsel tahminler yapılabilir.

7- Boyut Küçültme (Dimensionality reduction)

Makine Öğrenimi ve istatistikte boyut küçültme, dikkate alınan rastgele değişkenlerin sayısını azaltma işlemidir ve özellik seçimi ve özellik çıkarımı olarak ikiye ayrılabilir.

Bir veri kümesinden yalnızca yararlı özellikleri çıkarmak için verilerin boyutunu küçütmek için boyut azaltma kullanılır. Boyut azaltma, denetimsiz bir öğrenme tekniğidir.

Veri biliminde, boyut indirgeme, bir verinin yüksek boyutlu bir uzaydan, düşük boyutlu bir uzaya, anlamını kaybetmeyecek şekilde dönüştürülmesidir. Yüksek boyutlu bir veriyi işlemek daha fazla işlem yükü gerektirir. Bu yüzden, yüksek sayıda gözlemin ve değişkenin incelendiği sinyal işleme, konuşma tanıma, nöroinformatik, biyoinformatik gibi alanlarda boyut indiremesi sıkça kullanılır.

Boyut indirgeme yaklaşımları doğrusal ve doğrusal olmayan olarak ikiye ayrılır. Boyut indirgeme var olan özniteliklerin bir alt kümesini seçerek ya da yeni öznitelikler çıkararak yapılabilir. Boyut indirgemesi gürültü filtreleme, veri görselleştirme ya da kümeleme analizi

amacıyla kullanılabileceği gibi, diğer makine öğrenimi yöntemlerinin ön adımı olarak uygulanabilir.

8- Önerilerden eğilim ya da yön bulma

Öneri motorları oluşturmak için öneri algoritmalarını kullanılır.

Örnekler:

- Netflix öneri sistemi.
- Bir kitap tavsiye sistemi.
- Amazon'da bir ürün öneri sistemi.

Günümüzde yapay zeka, makine öğrenimi, derin öğrenme ve diğerleri gibi pek çok vizültli kelime duyuyoruz.

Makine öğrenimi algoritmalarını uygulamak için neden Python?

Python, popüler ve genel amaçlı bir programlama dilidir. Python kullanarak makine öğrenimi algoritmaları yazabiliyoruz ve bu iyi çalışır. Python'un veri bilimcileri arasında bu kadar popüler olmasının nedeni, **Python'un hayatımıza daha rahat hale getiren çok çeşitli modül ve kütüphanelerin halihazırda uygulanmış olmasıdır.**

Bazı heyecan verici Python kütüphalerine kısaca bir göz atalım:

1. Numpy: Python'da n boyutlu dizilerle çalışmak için bir matematik kitaplığıdır. Hesaplamaları etkili ve verimli bir şekilde yapmamızı sağlar.
2. Scipy: Sinyal işleme, optimizasyon, istatistik ve çok daha fazlasını içeren sayısal algoritmalar ve alana özgü araç kutusudur. Scipy, bilimsel ve yüksek performanslı hesaplamalar için işlevsel bir kitaplıktır.
3. Matplotlib: 2D çizimin yanı sıra 3D çizim de sağlayan modaya uygun bir çizim paketidir.
4. Scikit-learn: Python programlama dili için ücretsiz bir makine öğrenimi kitaplığıdır. Sınıflandırma, regresyon ve kümeleme algoritmalarının çoğuna sahiptir ve Numpy, Scipy gibi Python sayısal kütüphaneleri ile çalışır.

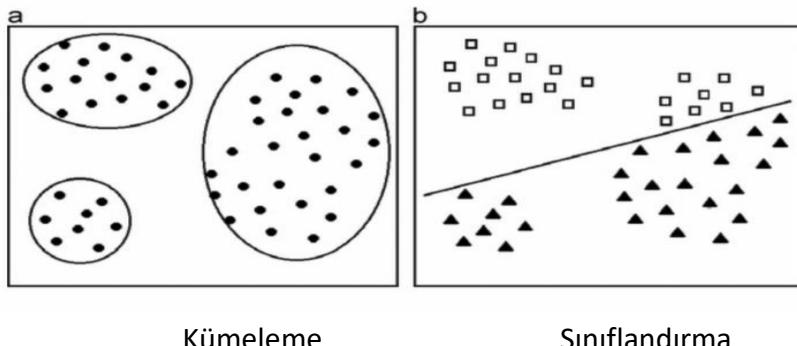
7.1. Kümeleme (Clustering)

Kümeleme, denetimsiz öğrenmenin bir yöntemidir ve birçok alanda kullanılan istatistiksel veri analizi için yaygın bir tekniktir. Denetimsiz öğrenme, veri kümesi ile çıktıların olmadığı bir öğrenme metodudur. Veri kümesindeki verileri yorumlayarak ortak noktaları bulmak ve bunları kümelenştirme işlemi yapılarak anlamlı bir veri elde edebilmektir. Sistem, öğreten olmadan öğrenmeye çalışır. Ham verileri organize verilere dönüştüren bir makine öğrenimi türüdür.

Denetimsiz öğrenmede sadece veriler vardır onlar hakkında bilgi verilmez. Bu verilerden sonuçlar çıkarılmaya çalışılır. En baştan veriler hakkında herhangi bir bilgi verilmemiş için çıkartılan sonuçların kesinlikle doğru olduğu söylenemez. Veriyi değişkenler arasındaki ilişkilere dayalı olarak kümelenerek çeşitli modeller, yapılar oluşturabiliriz.

Örneğin, bir alışveriş sitesinde alınan bir ürünün yanında kullanıcıların alabileceği diğer ürünlerin tavsiye olarak belirlenmesi. Ya da bir hizmet satınaldığında, o hizmetle etkileşimli diğer hizmetlerin müşterinin ilgi alanına girmesi.

Kümeleme işlevinden kimlik belirlenebilir mi? Özellikle alış veriş yaparken aldıklarımız, aynı anda kimliğimiz olabilir mi?



Denetimsiz öğrenmede kümeleme söz konusudur. Kümeleme algoritmaları basitçe, veri kümesindeki elemanları kendi aralarında gruplamaya çalışır. Burada kaç grup olacağı veya en uygun küme sayısını algoritmanın kendisi belirler. Verilerin yakınlık, uzaklık, benzerlik gibi ölçütlerde göre analiz edilerek sınıflara ayrılmasına kümeleme denir. Örnek; alışveriş yapılan bir markette kasierin bir robot olduğun düşünün ve tüm ürünler birbirine karışmış olsun. Elma bulup, onu tanıayıp diğer elemanları yiğin içerisinde topladığını düşünün. Seçme işlemi devam ederken yetenek kazanarak performansını atırabilir; hatalı seçtiği ürünler var ise ayıkalayabilir. Böylece ürünlerin birbirlerine benzeme yakınlığı uzaklaşarak, ayırm yapma yeteneği artırılmış olur. Böylece sınıflandırma da yapılmış olur.

Elmalar da kendi aralarında kümeleme yapılabilir mi? Aynı tipte verilerin değişik segmentlere bölünmüş halidir. Elinde örnekler var ama hangi veya kaç sınıfa ait olduğunu bilmiyor. Sınıfları(kümeleri) kendisi inşa ediyor. Örneğin elimizde sadece domatesler varsa, ve bunları kalitelerine göre ayırlırsa bu kümeleme işlemidir.

Kümelemenin uygulama alanları:

Tıp'da elde edilen görüntülemeler üzerindeki farklıları analiz edilerek değişik nitelikler çıkartılabilir.

Suç Yerlerinin Belirlenmesi: Bir şehirdeki belirli bölgelerde mevcut olan suçlarla ilgili veriler, suç kategorisi, suç alanı ve ikisi arasındaki ilişki, bir şehirdeki ya da bölgedeki suça eğilimli alanlara ilişkin kaliteli bilgiler verebilir.

Oyuncu istatistiklerini analiz etmek: Oyuncu istatistiklerini analiz etmek, spor dünyasının her zaman kritik bir unsuru olmuştur ve artan rekabetle birlikte, makine öğrenmenin burada oynayacağı kritik bir rol vardır.

Çağrı Kaydı Detay Analizi: Bir çağrı detay kaydı (CDR), telekom şirketleri tarafından bir müşterinin araması, SMS ve internet etkinliği sırasında elde edilen bilgilerdir. Bu bilgiler, müşteri demografisiyle birlikte kullanıldığında, müşterinin ihtiyaçları hakkında daha fazla bilgi sağlar.

Müşteri Segmentasyonu:

Collaborative Filtering: Davranışları birbirine benzeyen insanların yaptıklarına bakılarak kümedeki birinin ne yapacağını kestirmek.

Tehdit ve Sahtekarlık Yakalama: Sınıflandırmada tehditlerin özellikleri makineye öğretilir. Kümelemede ise kümelerin dışında kalan, herhangi bir kümeye girmeyen örnek bir tehdit unsuru olarak tanımlanabilir.

Eksik Verilerin Tanımlanması: Örneğin birinin maaş bilgisi eksikse segmentte benzer kişilerin maaş ortalamasına göre bir maaş hesaplanabilir.

Pazar Segmentasyonu:

Davranışsal Segmentasyon: Örneğin ücretsiz uygulamaları yükleyenler neyi seçiyorlar?

Demografik Segmentasyon: Müşterilerin yaşı, cinsiyeti, eğitim düzeyi ile davranışlarının entegre edilmesi.

Psikolojik Segmentasyon: Müşterilerin hayal-beklentilerine göre farklı ürünler sunulabilir.

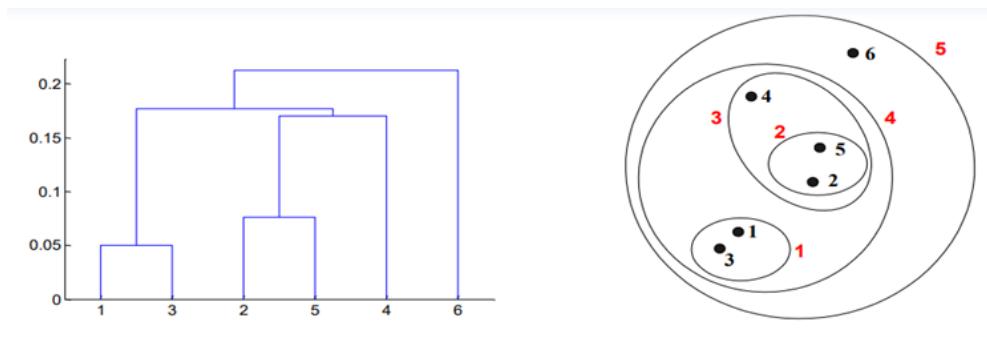
Coğrafi Segmentasyon: Ülkelere şehirlere göre vs.

Kümeleme Çeşitleri:

- Hiyerarşik Kümeleme
- Gürültülü Uygulamaların Yoğunluğa Dayalı Konumsal Kümelenmesi (DBSCAN) (DBSCAN)
- K-means Kümeleme
- Ağırlık Ortalama Kaydırma Kümelemesi
- Gauss Karışım Modelleri (GMM) kullanarak Beklenti-Maksimizasyon (EM) Kümeleme

Hiyerarşik Kümeleme:

Hiyerarşik kümeleme algoritmaları 2 kategoriye ayrılır: yukarıdan aşağıya veya aşağıdan yukarıya. Aşağıdan yukarıya algoritmalar, her veri noktasını başlangıçta tek bir küme olarak ele alır ve ardından tüm kümeler tüm veri noktalarını içeren tek bir kümede birleştirilene kadar küme çiftlerini art arda birleştirir (veya toplar). Bu nedenle aşağıdan yukarıya hiyerarşik kümeleme, hiyerarşik kümelemeli kümeleme (hierarchical agglomerative clustering) veya HAC olarak adlandırılır. Bu küme hiyerarşisi bir ağaç (veya dendrogram) olarak temsil edilir. Ağacın kökü, tüm örnekleri toplayan benzersiz kümedir, yapraklar yalnızca bir örnek içeren kümelerdir. Algoritma adımlarına geçmeden önce bir örnek için aşağıdaki grafiğe bakın



- Her veri noktasını tek bir küme olarak ele alıyoruz, yani veri kümemizde X veri noktası varsa, o zaman X kümemiz var. Daha sonra iki küme arasındaki mesafeyi ölçen bir mesafe ölçüsü seçiyoruz. Örnek olarak, iki küme arasındaki mesafeyi, birinci kümedeki veri noktaları ile ikinci kümedeki veri noktaları arasındaki ortalama mesafe olarak tanımlayan ortalama bağlantıyı kullanacağız.
- Her yinelemde, iki kümeyi tek bir kümede birleştiriyoruz. Birleştirilecek iki küme, en küçük ortalama bağlantıya sahip olanlar olarak seçilir. Yani, seçtiğimiz uzaklık ölçütümüze göre, bu iki küme birbirleri arasındaki en küçük mesafeye sahiptir ve bu nedenle en benzer olanlardır ve birleştirilmeleri gereklidir.
- Adım 2, ağacın köküne ulaşana kadar tekrar edilir, yani tüm veri noktalarını içeren tek bir kümeye sahibiz. Bu şekilde sonunda kaç tane kume istedigimizi seçebiliriz, sadece kümeleri birleştirmeyi ne zaman durduracağımızı seçerek, yani ağıacı oluşturmayı bıraktığımızda!

Hiyerarşik kümeleme, küme sayısını belirlememizi gerektirmez ve hatta bir ağaç oluşturduğumuz için hangi küme sayısının en iyi görüneceğini seçebiliriz. Ek olarak, algoritma mesafe ölçüsü seçimine duyarlı değildir; hepsi eşit derecede iyi çalışma eğilimindeyken, diğer kümeleme algoritmalarında uzaklık ölçüsü seçimi kritiktir. Hiyerarşik kümeleme yöntemlerinin özellikle iyi bir kullanım durumu, temeldeki verilerin hiyerarşik bir yapıya sahip olması ve hiyerarşiyi kurtarmak istemenizdir; diğer kümeleme algoritmaları bunu yapamaz. Hiyerarşik kümelemenin bu avantajları, K-Ortalamlarının ve GMM'nin doğrusal karmaşıklığından farklı olarak, $O(n^3)$ zaman karmaşıklığına sahip olduğundan, daha düşük verimlilik pahasına gelir.

K-Means Algoritması:

K-means algoritmasında kullanılan örneklem, k adet kümeye bölünür. Algoritmanın özü birbirlerine benzerlik gösteren verilerin aynı küme içerisinde alınmasına dayanır. Algoritmadaki benzerlik terimi, veriler arasındaki uzaklığa göre belirlenmektedir. Uzaklığın az olması benzerliğin yüksek, çok olması ise düşük olduğu anlamına gelmektedir.

K-means algoritmasının yapısı aşağıdaki gibidir;

- 1) K adet rastgele küme oluştur
- 2) Kare hata oranını hesapla
- 3) Verilerin kümelerin orta noktalarına olan uzaklıklarını bul
- 4) Her veri için en yakın kümeyi, o verinin kümesi olarak belirle
- 5) Yeni yerleşim düzeneğine göre hata oranını hesapla
- 6) Eğer önceki hata oranı ile şimdiki hata oranı eşit değilse 2,3,4,5 ve 6. adımları tekrarla
- 7) Eğer önceki hata oranı ile şimdiki hata oranı eşitse kümeleme işlemini sonlandır

Dirsek yöntemi, kümelere nasıl ihtiyaç duyacağımız konusunda kullanışlı olur. Dirsek noktasının belirlenmesi gereklidir.

Ağırlık Ortalama Kaydırma Kümelemesi:

Ortalama kaydırma kümeleme, veri noktalarının yoğun alanlarını bulmaya çalışan kayan pencere tabanlı bir algoritmadır. Centroid tabanlı bir algoritmadır, yani amacın her bir grubun / sınıfın merkez noktalarını bulmaktır, bu da kayan pencere içindeki noktaların ortalaması olacak merkez noktaları için adayları güncelliyor ve çalışır. Bu aday pencereler daha sonra, neredeyse kopyaları ortadan kaldırarak işlem sonrası aşamasında filtrelenir ve nihai merkez noktaları ve bunlara karşılık gelen grupları oluşturur.

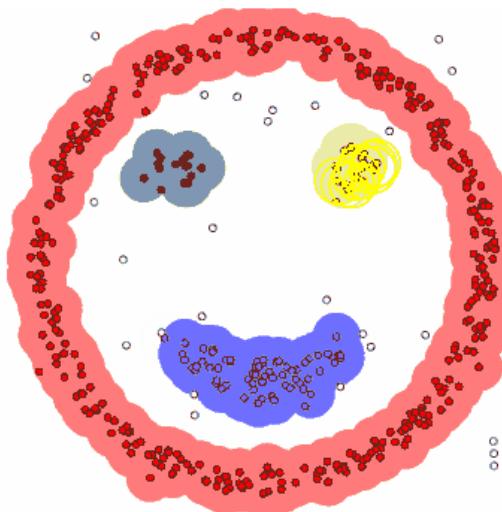
Sürgülü pencerelerin tümü ile uçtan uca tüm sürecin bir örneği aşağıda gösterilmiştir. Her siyah nokta, kayan bir pencerenin merkezini temsil eder ve her gri nokta bir veri noktasıdır.



- 1) Ortalama-kaymayı açıklamak için, yukarıdaki resimde olduğu gibi iki boyutlu uzayda bir dizi noktayı ele alacağız. Bir C noktasında ortalanmış (rastgele seçilmiş) ve çekirdek olarak r yarıçapına sahip dairesel bir kayan pencere ile başlıyoruz. Ortalama kayma, bu çekirdeği yakınsamaya kadar her adımda yinelemeli olarak daha yüksek yoğunluklu bir bölgeye kaydırmayı içeren bir tepe tırmanma algoritmasıdır.
- 2) Her yinelemede, kayan pencere, merkez noktası pencere içindeki noktaların ortalamasına kaydırılarak (dolayısıyla adı) daha yüksek yoğunluklu bölgelere kaydırılır. Sürgülü pencere içindeki yoğunluk, içindeki noktaların sayısı ile orantılıdır. Doğal olarak, penceredeki noktaların ortalamasına geçerek, yavaş yavaş daha yüksek nokta yoğunluğuna sahip alanlara doğru hareket edecektir.
- 3) Bir kaymanın çekirdek içinde daha fazla noktası barındırabileceği bir yön olmayana kadar ortalamaya göre kayan pencereyi kaydırmaya devam ediyoruz. Yukarıdaki grafiğe bakın; Artık yoğunluğu artırmayana kadar (yani penceredeki nokta sayısı) daireyi hareket ettirmeye devam ediyoruz.
- 4) 1'den 3'e kadar olan bu adım süreci, tüm noktalar bir pencerinin içinde kalana kadar birçok sürgülü pencerede yapılır. Birden çok sürgülü pencere örtüşüğünde, en çok noktayı içeren pencere korunur. Veri noktaları daha sonra bulundukları kayan pencereye göre kümelenir.

Gürültülü Uygulamaların Yoğunluğa Dayalı Konumsal Kümelenmesi (DBSCAN):

DBSCAN, ortalama kaymaya ekseninde, benzer ve yoğunluklu bölgeleri kümeleyen bir algoritmadır, ancak birkaç önemli avantajı vardır. Minimum bölge sayısında ve uzaklıkta maksimum yoğunluk bölgesi oluşturulması hedeflenir.



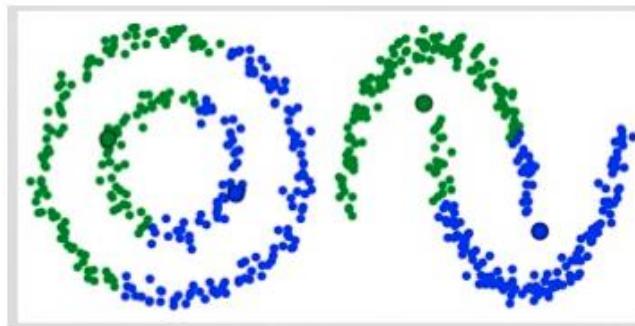
- DBSCAN, keyfi bir başlangıç veri noktasıyla başlar. Bu noktanın komşuluğu bir epsilon ϵ uzaklığı kullanılarak çıkarılır (ϵ mesafesi içindeki tüm noktalar komşuluk noktalarıdır).
- Bu mahalle içinde yeterli sayıda nokta (minPoints'e göre) varsa, kümelenme işlemi başlar ve mevcut veri noktası yeni kümedeki ilk nokta olur. Aksi takdirde, nokta gürültü olarak etiketlenecektir (daha sonra bu gürültülü nokta kümelenin parçası haline gelebilir). Her iki durumda da bu nokta "ziyaret edildi" olarak işaretlenir.
- Yeni kümedeki bu ilk nokta için, ϵ mesafesi komşuluğundaki noktalar da aynı kümelenin parçası olur. ϵ mahallesindeki tüm noktaları aynı kümeye ait yapma prosedürü, kümelerin yeni eklenen tüm yeni noktalar için tekrarlanır.
- 2. ve 3. adımlardan oluşan bu süreç, kümedeki tüm noktalar belirlenene kadar, yani kümelenin ϵ mahallesindeki tüm noktalar ziyaret edilmiş etiketlenene kadar tekrar edilir. Mevcut kümeye işlemiz bittiğinde, yeni bir ziyaret edilmemiş nokta alınır ve işlenir, bu da başka bir kümeye veya gürültü keşfine yol açar. Bu işlem, tüm noktalar ziyaret edildi olarak işaretlenene kadar tekrar eder.
- Bunun sonunda tüm noktalar ziyaret edildiğinden, her nokta bir kümeye ait veya gürültü olarak işaretlenecektir.

DBSCAN, diğer kümelenme algoritmalarına göre bazı büyük avantajlar sunar. İlk olarak, hiç bir kümeye gerekir. Ayrıca, veri noktası çok farklı olsa bile, aykırı değerleri basitçe bir kümeye

atan ortalama kaymanın aksine, gürültü olarak tanımlar. Ek olarak, keyfi olarak boyutlandırılmış ve keyfi olarak şekillendirilmiş kümeleri oldukça iyi bulabilir. DBSCAN'in ana dezavantajı, kümeler farklı yoğunlukta olduğunda diğerleri kadar iyi performans göstermemesidir. Bunun nedeni, komşu noktaların belirlenmesi için mesafe eşiği ϵ ve minPoints'in, yoğunluk değiştiğinde kümeden kümeye değişmesidir. Bu dezavantaj, çok yüksek boyutlu verilerde de ortaya çıkar, çünkü yine mesafe eşliğini ϵ tahmin etmek zorlaşır.

Gauss Karışım Modelleri (GMM) kullanarak Beklenti-Maksimizasyon (EM) Kümeleme:

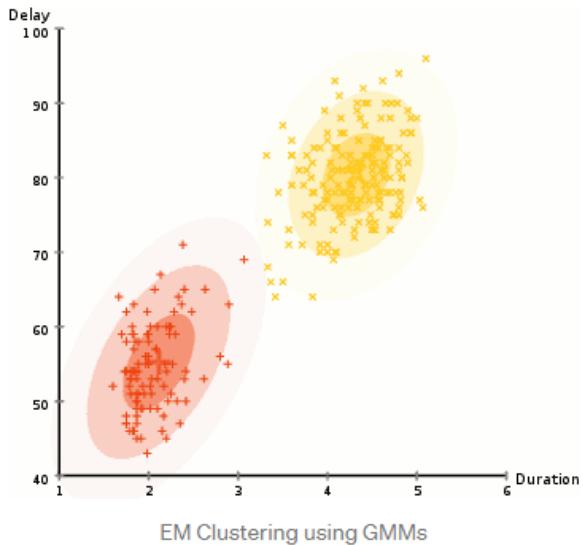
K-Ortalamalarının en büyük dezavantajlarından biri, küme merkezi için ortalama değerini naif kullanımızdır. Aşağıdaki resme bakarak bunun bir şeyleri yapmanın en iyi yolu olmadığını anlayabiliyoruz. Sol tarafta, aynı ortalamaya merkezlenmiş farklı yarıçaplara sahip iki dairesel küme olduğu insan gözüne oldukça açık görünüyor. K-Ortalamlar bunun üstesinden gelemez çünkü kümelerin ortalama değerleri birbirine çok yakındır. K-Ortalamlar, yine ortalamanın küme merkezi olarak kullanılması sonucunda kümelerin dairesel olmadığı durumlarda başarısız olur.



Two failure cases for K-Means

Gauss Karışım Modelleri (GMM'ler) bize K-Ortalamalarından daha fazla esneklik sağlar. GMM'ler ile veri noktalarının Gauss olarak dağıtıldığını varsayıyoruz; bu, ortalamayı kullanarak döngüsel olduklarını söylemekten daha az kısıtlayıcı bir varsayımdır. Bu şekilde, kümelerin şeklini açıklamak için iki parametremiz var: ortalama ve standart sapma! İki boyutlu bir örnek alırsak, bu, kümelerin her türlü eliptik şekli alabileceği anlamına gelir (çünkü hem x hem de y yönlerinde standart bir sapmaya sahibiz). Böylece, her Gauss dağılımı tek bir kümeye atanır.

Her küme için Gauss'un parametrelerini bulmak için (örneğin ortalama ve standart sapma), Beklenti-Maksimizasyon (EM) adı verilen bir optimizasyon algoritması kullanacağız. Kümelere uydurulan Gaussian'ların bir örneği olarak aşağıdaki grafiğe bir göz atın. Daha sonra GMM'leri kullanarak Beklenti-Maksimizasyon kümeleme sürecine geçebiliriz.



- 1) Küme sayısını seçerek (K-Means'in yaptığı gibi) ve her küme için Gauss dağılım parametrelerini rastgele başlatarak başlıyoruz. Verilere de hızla göz atarak ilk parametreler için iyi bir tahmin sağlamaya çalışılabilir. Yine de, yukarıdaki grafikte de görülebileceği gibi, bu% 100 gerekli değildir çünkü Gauss bize çok zayıf olarak başlarlar, ancak hızla optimize edilirler.
- 2) Her küme için bu Gauss dağılımları göz önüne alındığında, her veri noktasının belirli bir kümeye ait olma olasılığı hesaplanır. Bir nokta Gauss'un merkezine ne kadar yakınsa, o kümeye ait olasılığı o kadar artar. Gauss dağılımında verilerin çoğunun kümenin merkezine daha yakın olduğunu varsayıdığımız için, bu sezgisel bir anlam ifade etmelidir.
- 3) Bu olasılıklara dayanarak, kümeler içindeki veri noktalarının olasılıklarını maksimize edecek şekilde Gauss dağılımları için yeni bir dizi parametre hesaplıyoruz. Bu yeni parametreleri, veri noktası konumlarının ağırlıklı toplamını kullanarak hesaplıyoruz; burada ağırlıklar, o belirli kümeye ait veri noktasının olasılıklarıdır. Bunu görsel olarak açıklamak için yukarıdaki grafiğe, özellikle örnek olarak sarı kümeye bakabiliriz. Dağıtım ilk yinelemede rastgele başlar, ancak sarı noktaların çoğunun bu dağılımın sağında olduğunu görebiliriz. Olasılıklara göre ağırlıklandırılmış bir toplamı hesapladığımızda, merkeze yakın bazı noktalar olmasına rağmen çoğu sağdadır. Dolayısıyla, doğal olarak dağılımın ortalaması bu noktalar kümese daha da yaklaştırır. Ayrıca noktaların çoğunun “üstten-sağdan-aşağıya” olduğunu da görebiliriz. Bu nedenle standart sapma, olasılıkların ağırlıklandırdığı toplamı maksimize etmek için bu noktalara daha uygun bir elips oluşturmak üzere değişir.
- 4) Dağılımların yinelemeden yinelemeye pek değişmediği yakınsamaya kadar 2. ve 3. adımlar yinelemeli olarak tekrarlanır.

GMM kullanmanın 2 önemli avantajı vardır. Birincisi, GMM'ler küme kovaryansı açısından K-Ortalamlarına göre çok daha esnektir; standart sapma parametresi nedeniyle, kümeler

dairelerle sınırlı olmak yerine herhangi bir elips şeklini alabilir. K-Ortalamları aslında her kümenin tüm boyutlardaki kovaryansının 0'a yaklaştığı özel bir GMM durumudur. İkinci olarak, GMM'ler olasılıkları kullandığından, veri noktası başına birden çok kümeye sahip olabilirler. Dolayısıyla, bir veri noktası üst üste binen iki kümenin ortasındaysa, sınıf 1'e yüzde X ve yüzde 2'ye ait olduğunu söyleyerek sınıfını tanımlayabiliriz. Yani GMM'ler karma üyeliği destekler.

7.2. Sınıflandırma

Bir Makine Öğrenimindeki bir sınıflandırıcı, ayrık veya sürekli özellik değerlerinin bir vektörünü giren ve tek bir ayrık değer olan sınıfı çıkarır bir sistemdir. Bir dizi ögenin sınıfını veya kategorisini tahmin etmek için sınıflandırma algoritmaları kullanılır.

Sınıflandırma algoritmaları:

Diğer sınıflandırma tekniklerinden bazıları aşağıda verilmiştir:

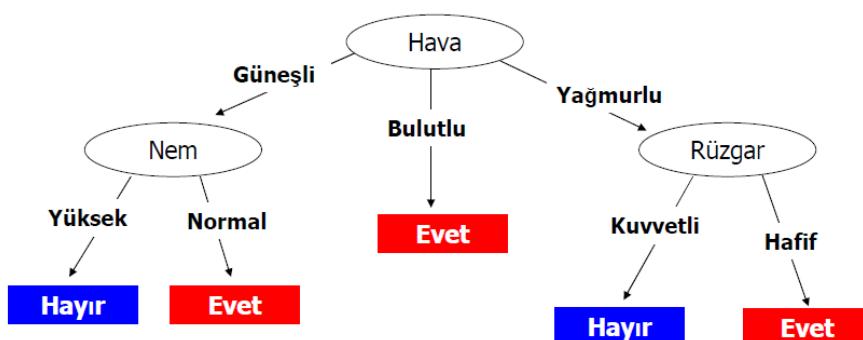
- K-En Yakın Komşu Algoritması (K-Nearest Neighbour Algorithm)
- Lojistik Regresyon (Logistic Regression)
- Destek Vektör Makineleri (Support Vector Machines)
- Karar Ağaçları (Decision Tree)
- Rasgele Orman Kümeleri (Random Forests)
- Yapay Sinir Ağları (Artifical Neural Networks)
- Naive Bayes

7.2.1. Karar Ağaçları

Karar ağaçları algoritması, denetimli öğrenme kategorisine girer. Hem regresyon hem de sınıflandırma problemlerini çözmek için kullanılırlar. Karar ağaçları, her yaprak düğümün bir sınıf etiketine karşılık geldiği ve özniteliklerin ağacın iç düğümünde temsil edildiği sorunu çözmek için ağaç temsilini kullanır. Karar ağaçını kullanarak herhangi bir boole fonksiyonunu ayrık öznitelikler üzerinde temsil edebiliriz.

Karar ağaçları öğrenmesi, bir karar ağaçını, bir öğeyle ilgili (dallarda temsil edilen) gözlemlerden ögenin hedef değeri (yapraklarda temsil edilen) ile ilgili sonuçlara gitmek için bir tahmin modeli olarak kullanır. İstatistik, veri madenciliği ve makine öğrenmesinde kullanılan öngörülü modelleme yaklaşımlarından biridir. Hedef değişkenin ayrı bir değer kümesi alabileceği ağaç modellerine sınıflandırma ağaçları denir; bu ağaç yapılarında yapraklar sınıf etiketlerini ve dallar bu sınıf etiketlerine yol açan özelliklerin birleşimlerini temsil eder. Hedef değişkenin sürekli değerler alabileceği karar ağaçlarına (tipik olarak gerçek sayılar) regresyon ağaçları denir. Karar analizinde, bir karar ağaçları, kararları ve karar almayı görsel ve açık bir şekilde temsil etmek için kullanılabilir. Veri madenciliğinde, bir karar ağaçları verileri tanımlar, ancak sonuçta ortaya çıkan sınıflandırma ağaçları karar verme için bir girdi olabilir.

Karar ağaçları metodu, giriş verisinin bir algoritma yardımıyla gruptara bölünderek tüm elemanlarının aynı sınıf etiketine sahip olması için yapılan sınıflama işlemidir. Giriş verisinin bir kümelenme algoritması yardımıyla tekrar tekrar gruptara bölünmesine dayanır. Grubun tüm elemanları aynı sınıf etiketine sahip olana kadar kümelenme işlemi derinlemesine devam eder.



Karar ağıacı kullanılırken yapılan bazı varsayımlar aşağıdadır:

- Başlangıçta, tüm eğitim seti kök olarak kabul edilir.
- Özelliğin değerlerinin kategorik olması tercih edilir. Değerler sürekli ise, model oluşturmadan önce ayriklaştırılırlar.
- Öznitelik değerleri temelinde, kayıtlar özyinelemeli olarak dağıtilır.
- Öznitelikleri kök veya dahili düğüm olarak sıralamak için istatistiksel yöntemler kullanılır.

Karar ağıacı tipleri ikiye ayrılır:

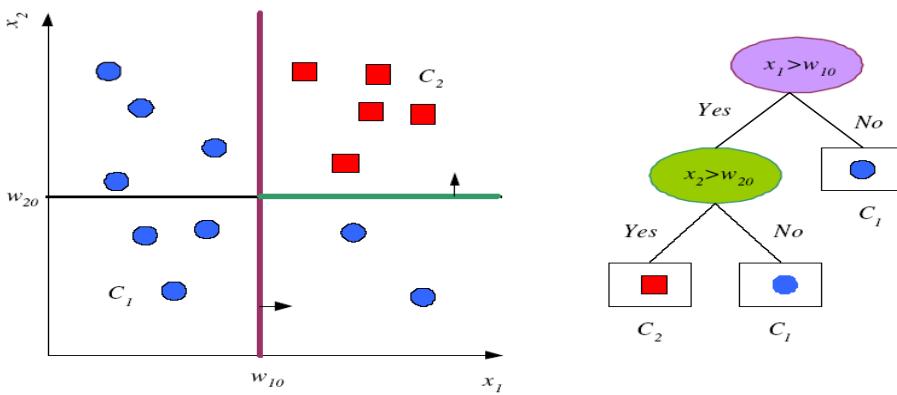
- Entropiye dayalı sınıflandırma ağaçları (ID3, C4.5)
- Regresyon ağaçları (CART).

Karar ağaçları çok boyutlu (özellikli) veriyi belirlenmiş şartlara bağlı olarak parçalara böler. Her adımda verinin hangi özelliği üzerinde işlem yapılacağına karar vermek çok büyük bir kombinasyonun çözümüyle mümkün değildir. Örneğin, 5 özellik ve 20 örnegé sahip bir veride 10^6 dan fazla sayıda farklı karar ağıacı oluşturulabilir. Bu sebeple her parçalanmanın metodolojik olması gereklidir. Quinlan'e göre veri bir özelliğine göre bölündüğünde elde edilen her bir veri kümelerinin belirsizliği minimum ve dolayısıyla bilgi kazancı maksimum ise en iyi seçim yapılmış demektir. Buna göre önerdiği ilk algoritma ID3'te tek tek özellik vektörleri incelenir ve en yüksek bilgi kazancına sahip özellik, ağaçta dallanma yapmak için tercih edilir.

Karar Ağacı Algoritması:

Karar ağaçları eğiticili öğrenme için çok yaygın bir yöntemdir. Algoritmanın adımları:

- 1) T öğrenme kümelerini oluşturular.
- 2) T kümelerindeki örnekleri en iyi ayıran nitelikler belirlenir.
- 3) Seçilen nitelik ile ağaçın düğümleri oluşturulur ve herbir düğümde alt düğümler veya ağaçın yapraklarını oluşturular. Alt düğümlere ait alt veri kümelerinin örneklerini belirlenir
- 4) 3. adımda oluşturulan her alt veri kümeleri için
 - Örneklerin hepsi aynı sınıfa aitse
 - Örnekleri bölecek nitelik kalmamışsa
 - Kalan niteliklerin değerini taşıyan örnek yoksa işlemi sonlandır. Diğer durumda alt veri kümelerini ayırmak için 2. adımdan devam edilir.



Ezber (Overfitting: Aşırı Uyum):

- Tüm makine öğrenmesi yöntemlerinde verinin ana hatlarının modellenmesi esas alındığı için öğrenme modelinde ezberden (overfitting) kaçınılmalıdır.
- Tüm karar ağaçları önlem alınmazsa ezber yapar. Bu yüzden ağaç olusturulurken veya olusturulduktan sonra budama yapılmalıdır.

Ağaç Budama:

Budama, sınıflandırmaya katkısı olmayan bölümlerin karar ağacından çıkarılması işlemidir. Bu sayede karar ağaçları hem sade hem de anlaşılabilir hale gelir. İki çeşit budama yöntemi vardır;

- Ön budama
- Sonradan budama

Ön budama işlemi ağaç oluşturulurken yapılır. Bölünen nitelikler, değerleri belli bir esik değerinin (hata toleransının) üstünde değilse o noktada ağaç böülüme işlemi durdurulur ve o an elde bulunan kümedeki baskın sınıf etiketi, yaprak olarak olusturulur.

Sonradan Budama: Sonradan budama işlemi ağaç olusturulduktan sonra devreye girer. Alt ağaçları silerek yaprak olusturma, alt ağaçları yükseltme, dal kesme şeklinde yapılabilir.

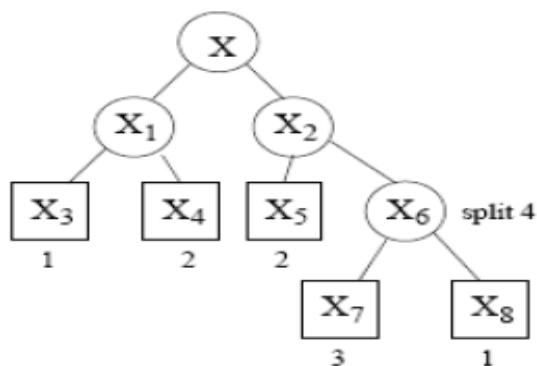
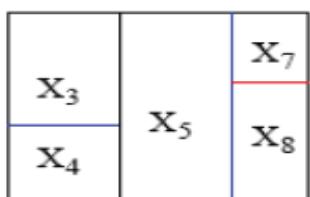
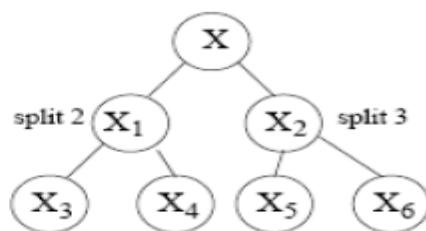
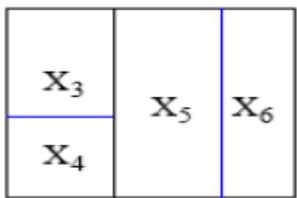
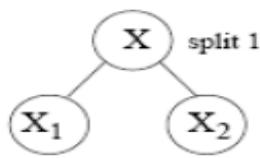
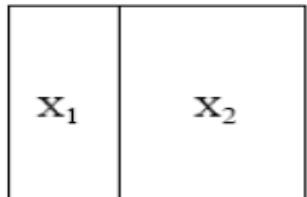
Aşırı uyumu önlemek için ağaç büyütmemeyi durdurabiliriz, ancak durdurma kriteri miyop olma eğilimindedir. Bu nedenle standart yaklaşım, "dolu" bir ağaç yetiştirmek ve ardından budama yapmaktadır. Düğümdeki noktalar için yanlış bir sınıflandırma yapma olasılığı olduğu da unutulmamalıdır. Tüm ağaç için yanlış sınıflandırma olasılığını elde etmek için, toplam olasılık formülüne göre yaprak düğüm içi hata oranının ağırlıklı toplamı hesaplanır.

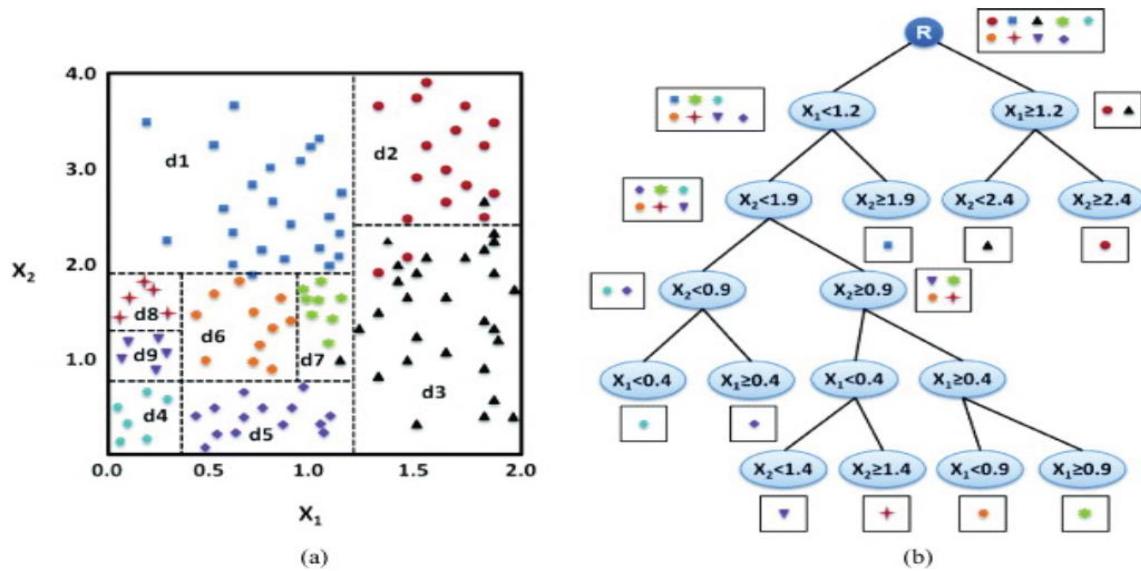
Spesifik olarak, ana düğüm için ağırlıklı yanlış sınıflandırma oranının, sol ve sağ alt düğümlerin ağırlıklı yanlış sınıflandırma oranlarının toplamından daha büyük veya buna eşit olacaktır. Yeniden ikame hata oranını en aza indirirsek, her zaman daha büyük bir ağaç tercih edeceğimiz anlamına gelir. Aşırı uyuma karşı hiçbir savunma yoktur. Aşırı büyümüş ağaç er ya da geç budanacaktır. Ne zaman duracağınızı karar vermenin birkaç yolu vardır:

- Tüm terminal düğümleri saf olana kadar devam edilir.
- Her bir terminal düğümündeki veri sayısı belirli bir eşikten, örneğin 5'ten, hatta 1'den büyük olmayana kadar devam edilir.
- Ağaç yeterince büyük olduğu sürece, ilk ağaçın boyutu kritik değildir.

Buradaki anahtar, ilk ağaçtı yeniden budamadan önce yeterince büyük yapmaktadır!

Sınıflandırma Ağaçları:





Karar Ağacı'nda en büyük zorluk, her seviyede kök düğüm için özniteliğin tanımlanmasıdır. Bu işlem öznitelik seçimi olarak bilinir. İki popüler öznitelik seçim ölçüsü bulunmaktadır:

- 1) Bilgi Kazancı
- 2) Gini İndeksi

1) Bilgi Kazancı

Eğitim örneklerini daha küçük alt kümelere bölmek için karar ağacında bir düğüm kullandığımızda entropi değişir. Bilgi kazancı, entropideki bu değişimin bir ölçüsüdür.

Tanım: Diyelim ki S bir örnekler kümesi, A bir nitelik, S_v , S 'nin $A = v$ ile alt kümesi ve Değerler (A), A 'nın tüm olası değerlerinin kümesidir, o zaman

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \cdot Entropy(S_v)$$

Örnek:

$X = \{a,a,a,b,b,b,b,b\}$ kümesi için

Toplam örnek: 8

b: 5 örnekleri

a: 3'ün örnekleri

$$\begin{aligned} EntropyH(X) &= - \left[\left(\frac{3}{8} \right) \log_2 \frac{3}{8} + \left(\frac{5}{8} \right) \log_2 \frac{5}{8} \right] \\ &= - [0.375 * (-1.415) + 0.625 * (-0.678)] \\ &= -(-0.53 - 0.424) \\ &= 0.954 \end{aligned}$$

Bilgi Kazanımını Kullanarak Karar Ağacı Oluşturma

Gereklilikler:

- Kök düğümle ilişkili tüm eğitim örnekleriyle başlanır
- Her bir düğümün hangi öznitelikle etiketleneceğini seçmek için bilgi kazancı kullanılır.
- Not: Hiçbir kökten yaprağa yol, aynı ayrık özniteligi iki kez içermemelidir
- Her alt ağacı, ağaçta o yolda sınıflandırılacak eğitim örneklerinin alt kümesinde yinelemeli olarak oluşturulur.

Sınır vakaları:

- Tüm pozitif veya tüm negatif eğitim örnekleri kalırsa, o düğümü buna göre “evet” veya “hayır” olarak etiketlenir.
- Hiçbir öznitelik kalmazsa, o düğümde kalan eğitim örneklerinin çoğunluk oyu ile etiketlenir.
- Örnek kalmadıysa, ebeveynin eğitim örnekleri çoğunluk oyu ile etiketlenir.

Örnek:

Şimdi aşağıdaki veriler için Bilgi kazanımını kullanarak bir Karar Ağacı çizelim.

Eğitim seti: 3 özellik ve 2 sınıf

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Burada 3 özellik ve 2 çıktı sınıf var. Bilgi kazanımını kullanarak bir karar ağacı oluşturmak. Her bir özellik alınır ve her bir özellik için bilgi kazancı hesaplanır.

ID3 Algoritması:

Sadece kategorik veri ile çalışan bir yöntemdir. Her iterasyonun ilk adımımda veri örneklerine ait sınıf bilgilerini taşıyan vektörün entropisi belirlenir. Daha sonra özellik vektörlerinin sınıfa bağımlı entropileri hesaplanarak ilk adımda hesaplanan entropiden çıkartılır. Bu şekilde elde edilen değer ilgili özellik vektörüne ait kazanç değeridir. En büyük kazanca sahip özellik vektörü ağaçın o iterasyonda belirlenen dallanmasını gerçekleştirir.

Örnek:

2 özellik vektörü (V_1 ve V_2) ile S sınıf vektörüne sahip 4 örnekli veri kümesi verilmistir. ID3 algoritması ile ilk dallanma hangi özellik üzerinde gerçekleşsir ?

- $H(S) - H(V_1, S)$
- $H(S) - H(V_2, S)$

V1	V2	S
A	C	E
B	C	F
B	D	E
B	D	F

Sınıf Entropisi

$$H(S) = -\left(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}\right) = 1$$

V_1 Entropisi

$$\begin{aligned} H(V_1) &= \frac{1}{4} H(A) + \frac{3}{4} H(B) \\ &= \frac{1}{4} 0 - \frac{3}{4} \left(\frac{1}{3} \log_2 \frac{1}{3} + \frac{2}{3} \log_2 \frac{2}{3} \right) = 0 + \frac{3}{4} 0,9183 = 0,6887 \end{aligned}$$

V_2 Entropisi

$$H(V_2) = \frac{1}{2} H(C) + \frac{1}{2} H(D) = \frac{1}{2} + \frac{1}{2} = 1$$

V1 seçilir...

C4.5 Algoritması:

ID3 algoritmasının nümerik özellik içeren veriye uygulanabilen seklidir. ID3'ten tek farkı nümerik özelliklerin kategorik hale getirilebilmesini sağlayan bir esikleme yöntemini içermesidir. Temel mantık nümerik özellik vektöründeki tüm değerler ikili olarak ele alınarak ortalamaları esik olarak denenir. Hangi esik değeriyle bilgi kazanımı en iyi ise o değer seçilir. Seçilen eseğe göre özellik vektörü kategorize edilir ve ID3 uygulanır.

Örnek:

Kredilendirmede "Mükemmel", "İyi" ve "Kötü" değerleri alabilen bir özelligimiz vardır. Toplam 14 gözlem var. Bunlardan 7'si Normal Sorumluluk sınıfına, 7'si ise Yüksek Sorumluluk Sınıfına aittir. Yani kendi başına eşit bir bölünmedir. En üst satırda özetlersek, kredi notu özelligi için Mükemmel değerine sahip 4 gözlem olduğunu görebiliriz. Ayrıca, "Mükemmel" Kredi Notu için hedef değişkeninin nasıl bölündüğünü de görülebiliyor. Kredi notu için "Mükemmel" değeri alan gözlemler için Normal Sorumluluk sınıfına ait 3 adet ve Yüksek Sorumluluk sınıfına ait sadece 1 adet gözlem bulunmaktadır. Benzer şekilde diğer Kredi Notu değerleri için de beklenmedik durum tablosundan bu değerler bulunabilir.

Credit Rating		Liability		
		Normal	High	Total
Excellent	3	1	4	1
Good	4	2	6	1
Poor	0	4	4	1
Total	7	7	14	1

Bu örnek için, beklenmedik durum tablosunu, hedef değişkeninin entropisini kendi başına hesaplamak için kullanılacak ve ardından özellik, kredi notu hakkında ek bilgiler verilen hedef değişkenin entropisi hesaplanacak. Bu, "Kredi Notu"nun hedef değişkenin "Yükümlülük" için ne kadar ek bilgi sağladığını hesaplanması olanağ sağlayacak.

$$\begin{aligned} E(\text{Liability}) &= -\frac{7}{14} \log_2 \left(\frac{7}{14} \right) - \frac{7}{14} \log_2 \left(\frac{7}{14} \right) \\ &= -\frac{1}{2} \log_2 \left(\frac{1}{2} \right) - \frac{1}{2} \log_2 \left(\frac{1}{2} \right) \\ &= 1 \end{aligned}$$

Hedef değişkenin entropisi, "Normal" ve "Yüksek" sınıf etiketi arasındaki eşit bölünme nedeniyle maksimum düzensizlikte 1'dir. Bir sonraki adım, kredi puanı hakkında ek bilgi verilen hedef değişken Yükümlülük'ün entropisini hesaplamaktır. Bunun için her Kredi Puanı değeri için Sorumluluk entropisi hesaplanacak ve her bir değerde sonuçlanan gözlemlerin

oranının ağırlıklı ortalaması kullanılarak bunlar eklenecektir. Neden ağırlıklı ortalama kullanıldığı, bunu karar ağaçları bağlamında tartışıldığında daha da netleşecektir.

$$E(\text{Liability} \mid CR = \text{Excellent}) = -\frac{3}{4} \log_2\left(\frac{3}{4}\right) - \frac{1}{4} \log_2\left(\frac{1}{4}\right) \approx 0.811$$

$$E(\text{Liability} \mid CR = \text{Good}) = -\frac{4}{6} \log_2\left(\frac{4}{6}\right) - \frac{2}{6} \log_2\left(\frac{2}{6}\right) \approx 0.918$$

$$E(\text{Liability} \mid CR = \text{Poor}) = -0 \log_2(0) - \frac{4}{4} \log_2\left(\frac{4}{4}\right) = 0$$

Weighted Average:

$$E(\text{Liability} \mid CR) = \frac{4}{14} \times 0.811 + \frac{6}{14} \times 0.918 + \frac{4}{14} \times 0 = 0.625$$

Credit Rating		Liability			
		Normal	High	Total	
Excellent	3	1	4		
Good	4	2	6		
Poor	0	4	4		
Total	7	7	14		

Kredi Derecelendirme özelliği verilen hedef değişken için entropi elde edildi. Artık bu özelliğin ne kadar bilgilendirici olduğunu görmek için Kredi Notundan Yükümlülük Bilgi Kazancı hesaplanmalıdır.

Information Gain:

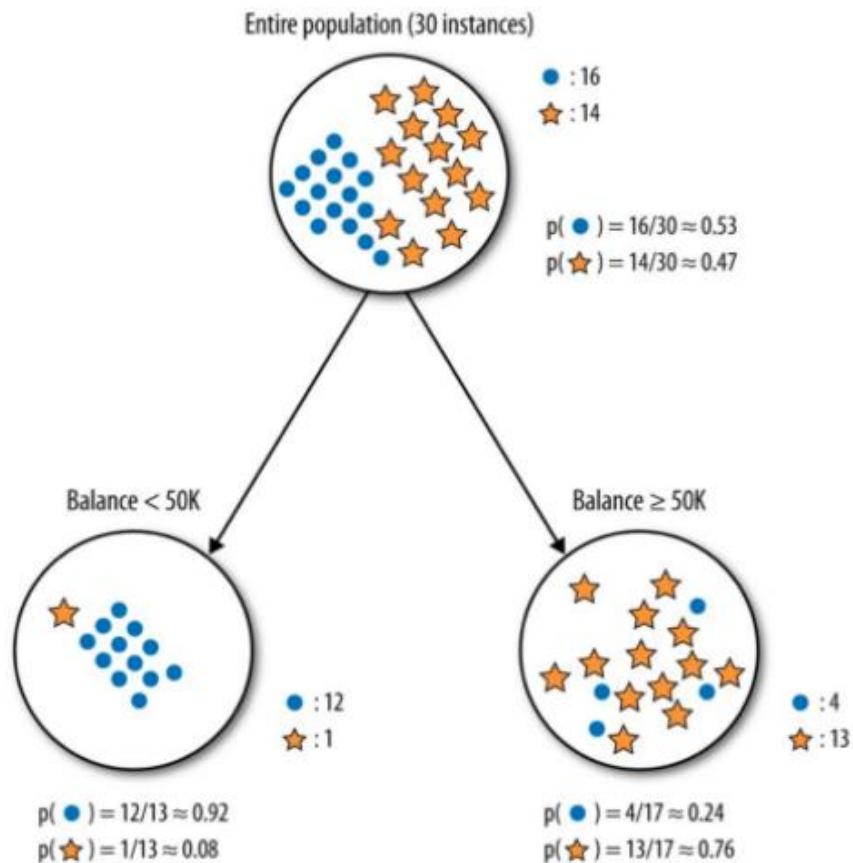
$$IG(\text{Liability}, CR) = E(\text{Liability}) - E(\text{Liability} \mid CR)$$

$$= 1 - 0.625 = 0.375$$

Kredi Notunu bilmek, hedef değişkenimiz olan Sorumluluk etrafındaki belirsizliği azaltmamıza yardımcı oldu! İyi bir özelliğin yapması gereken de bu değil mi? Hedef değişkenimiz hakkında bize bilgi verir misiniz? İşte tam olarak bu, karar ağaçlarının, her bir bölünmede hedef değişkeni tahmin etmeye yaklaşmak ve aynı zamanda ağaç bölmeyi ne zaman durduracağını belirlemek için düğümlerini hangi özelliğin üzerine böleceğini belirlemek için entropi ve bilgi kazancını nasıl ve neden kullandığıdır! (elbette maksimum derinlik gibi hiper parametrelere ek olarak).

Örnek: Karar Ağacı

Bir kişiye verilen bir kredinin bir zararla sonuçlanıp sonuçlanmayacağını tahmin etmek için bir karar ağıacı oluşturduğumuz bir örneği ele alalım. Tüm popülasyonumuz 30 örnekten oluşmaktadır. 16'sı silinen sınıfa, diğer 14'ü silinmeyen sınıfa aittir. İki değer alabilen "Bakiye" -> "< 50K" veya ">50K" ve üç değer alabilen "Konut" -> "KENDİ", "KİRALIK" veya "DİĞER" olmak üzere iki özelliğimiz var. Entropi ve Bilgi Kazanımı kavramlarını kullanarak bir karar ağıacı algoritmasının hangi özniteliğin ilk olarak bölüneceğine ve hangi özelliğin daha fazla bilgi sağladığına veya hedef değişkenimiz hakkındaki belirsizliği ikisinden daha fazla azaltlığına nasıl karar vereceğini göstereceğim.



Özellik 1: Denge

Noktalar, sınıf hakkı olan veri noktalarıdır ve yıldızlar, silinmeyenlerdir. Ana düğümü öznitelik dengesine bölmek bize 2 alt düğüm verir. Sol düğüm, silme sınıfından 12/13 (0.92 olasılık) gözlem ile toplam gözlemlerin 13'ünü ve sınıfın yazılmayan sınıfından sadece 1/13 (0.08 olasılık) gözlemi alır. Sağ düğüm, silinmeyen sınıfından 13/17(0.76 olasılık) ve silinen sınıfından 4/17 (0.24 olasılık) ile toplam gözlemin 17'sini alır.

Ana düğümün entropisini hesaplayalım ve Denge üzerinde bölerek ağacın ne kadar belirsizliği azaltabileceğini görelim.

$$E(\text{Parent}) = - \frac{16}{30} \log_2 \left(\frac{16}{30} \right) - \frac{14}{30} \log_2 \left(\frac{14}{30} \right) \approx 0.99$$

$$E(\text{Balance} < 50K) = - \frac{12}{13} \log_2 \left(\frac{12}{13} \right) - \frac{1}{13} \log_2 \left(\frac{1}{13} \right) \approx 0.39$$

$$E(\text{Balance} > 50K) = - \frac{4}{17} \log_2 \left(\frac{4}{17} \right) - \frac{13}{17} \log_2 \left(\frac{13}{17} \right) \approx 0.79$$

Weighted Average of entropy for each node:

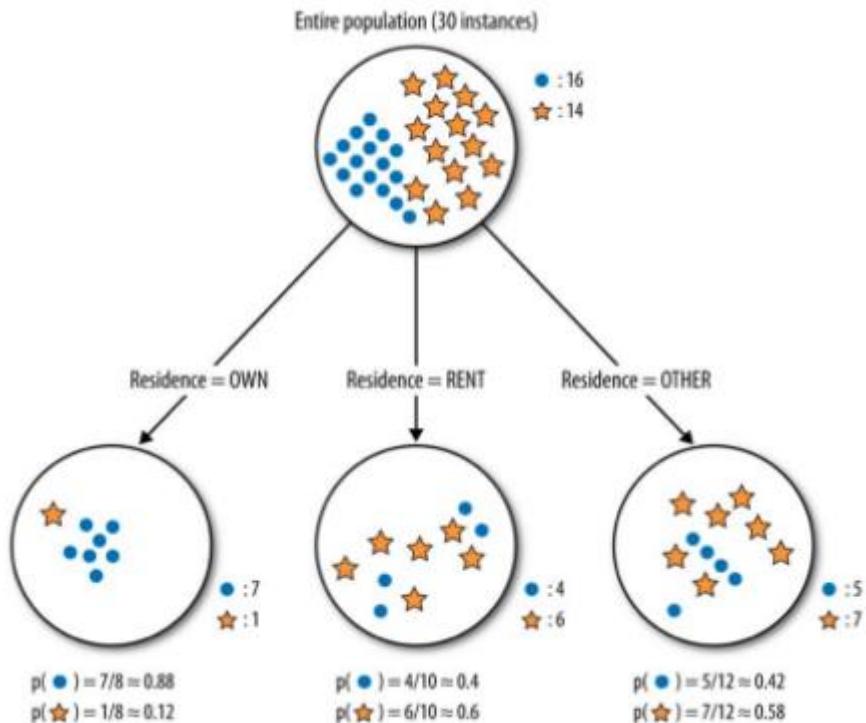
$$\begin{aligned} E(\text{Balance}) &= \frac{13}{30} \times 0.39 + \frac{17}{30} \times 0.79 \\ &= 0.62 \end{aligned}$$

Information Gain:

$$\begin{aligned} IG(\text{Parent}, \text{Balance}) &= E(\text{Parent}) - E(\text{Balance}) \\ &= 0.99 - 0.62 \\ &= 0.37 \end{aligned}$$

Özelliğe göre bölme, “Denge” hedef değişkenimiz üzerinde 0.37'lik bir bilgi kazanımına yol açar. Aynı şeyi nasıl karşılaştırdığını görmek için “Konut” özelliği için yapalım.

Özellik 2: Konut



Ağacı Residence'ta bölmek bize 3 alt düğüm verir. Sol alt düğüm, silinen sınıfından $7/8$ (0.88 olasılık) gözlem ile toplam gözlemlerin 8'ini ve silinmeyen sınıfından sadece $1/8$ (0.12 olasılık) gözlem alır. Orta alt düğümler, silinen sınıfından $4/10$ (0,4 olasılık) ve silinmeyen sınıfından $6/10$ (0,6 olasılık) gözlem ile toplam gözlemlerin 10'unu alır. Sağ alt düğüm, silme sınıfından $5/12$ (0.42 olasılık) gözlem ve silinmeyen sınıfından $7/12$ (0.58) gözlem ile toplam gözlemlerin 12'sini alır. Ana düğümün entropisini zaten biliyoruz. “Konut”tan elde edilen bilgi kazancını hesaplamak için bölmeden sonraki entropiyi hesaplamamız yeterlidir.

$$E(\text{Residence} = \text{OWN}) = -\frac{7}{8}\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right) \approx 0.54$$

$$E(\text{Residence} = \text{RENT}) = -\frac{4}{10}\log_2\left(\frac{4}{10}\right) - \frac{6}{10}\log_2\left(\frac{6}{10}\right) \approx 0.97$$

$$E(\text{Residence} = \text{OTHER}) = -\frac{5}{12}\log_2\left(\frac{5}{12}\right) - \frac{7}{12}\log_2\left(\frac{7}{12}\right) \approx 0.98$$

Weighted Average of entropies for each node:

$$E(\text{Residence}) = \frac{8}{30} \times 0.54 + \frac{10}{30} \times 0.97 + \frac{12}{30} \times 0.98 = 0.86$$

Weighted Average of entropies for each node:

$$E(\text{Residence}) = \frac{8}{30} \times 0.54 + \frac{10}{30} \times 0.97 + \frac{12}{30} \times 0.98 = 0.86$$

Information Gain:

$$\begin{aligned} IG(\text{Parent}, \text{Residence}) &= E(\text{Parent}) - E(\text{Residence}) \\ &= 0.99 - 0.86 \\ &= 0.13 \end{aligned}$$

Denge özelliğinden gelen bilgi kazanımı, Konut'tan elde edilen bilgiden neredeyse 3 kat daha fazla! Geri dönüp grafiklere bir göz atarsanız, Denge'de bölünen alt düğümlerin, Yerleşim düğümlerinden daha saf göründüğünü görebilirsiniz. Bununla birlikte, ikamet için en soldaki düğüm de çok saftır, ancak ağırlıklı ortalamaların devreye girdiği yer burasıdır. Bu düğüm çok saf olmasına rağmen, toplam gözlemlerin en az miktarına sahiptir ve bir sonuç, Yiğindaki bölmenden toplam entropiyi hesapladığımızda saflığının küçük bir kısmına katkıda bulunur. Bu önemlidir çünkü bir özelliğin genel bilgi gücünü arıyoruz ve sonuçlarımızın bir özellikteki nadir bir değer tarafından çarpıtılmasını istemiyoruz.

Kendi başına Balance özelliği, hedef değişkenimiz hakkında Konuttan daha fazla bilgi sağlar. Hedef değişkenimizde daha fazla düzensizliği azaltır. Bir karar ağacı algoritması, Balance kullanarak verilerimizde ilk bölmeyi yapmak için bu sonucu kullanır. Bundan sonra, karar ağacı algoritması, bir sonraki hangi özelliği böleceğine karar vermek için her bölmeye bu

süreci kullanacaktır. Gerçek dünya senaryosunda, ikiden fazla özellik ile ilk bölme en bilgilendirici özellik üzerinde yapılır ve daha sonra her bölmeye, her bir ek özellik için bilgi kazancının yeniden hesaplanması gereklidir, çünkü her birinden elde edilen bilgi kazancı ile aynı olmaz. özellik kendi başına. Entropi ve bilgi kazancı, sonuçları değiştirecek bir veya daha fazla bölme yapıldıktan sonra hesaplanmalıdır. Bir karar ağacı, önceden tanımlanmış bir derinliğe ulaşana ya da hiçbir ek bölümme, genellikle hiper parametre olarak da belirlenebilen belirli bir eşliğin ötesinde daha yüksek bir bilgi kazanımına neden olana kadar derinleşikçe ve derinleşikçe bu işlemi tekrarlar!

Example: Decision Tree - Classification

Karar ağacı, bir ağaç yapısı şeklinde sınıflandırma veya regresyon modelleri oluşturur. Bir veri kümesini giderek daha küçük alt kümelere ayırırken aynı zamanda ilgili bir karar ağacı aşamalı olarak geliştirilir. Nihai sonuç, karar düğümleri ve yaprak düğümleri olan bir ağaçtır. Bir karar düğümünün iki veya daha fazla Şubesi vardır (örn. Güneşli, Bulutlu ve Yağmurlu). Yaprak düğümü bir sınıflandırmayı veya kararı temsil eder. Kök düğüm adı verilen en iyi tahmin ediciye karşılık gelen bir ağactaki en üstteki karar düğümü. Karar ağaçları hem kategorik hem de sayısal verileri işleyebilir.



Algorithm

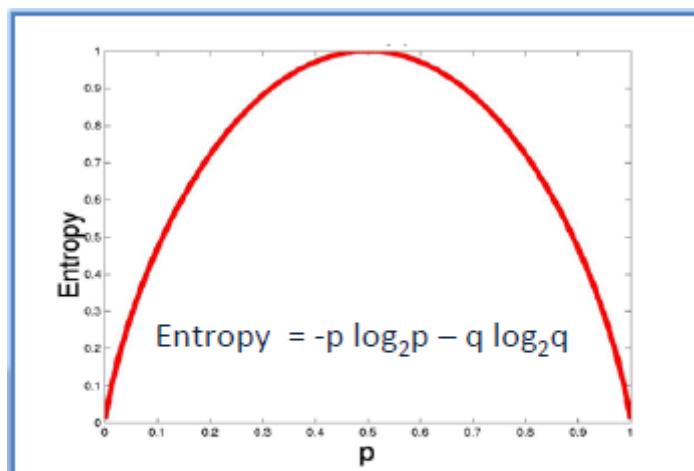
The core algorithm for building decision trees called ID3 by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses Entropy and Information Gain to construct a decision tree. In ZeroR model there is no predictor, in OneR model we try to find the single best predictor, naive Bayesian includes all predictors using Bayes' rule and the independence assumptions between predictors but decision tree includes all predictors with the dependence assumptions between predictors.

J. R. Quinlan tarafından ID3 olarak adlandırılan karar ağaclarının, temel algoritması, olası dallar alanında yukarıdan aşağıya, geri izleme olmadan bir arama yapar. ID3, bir karar ağacı oluşturmak için Entropi ve Bilgi Kazancı kullanır. ZeroR modelinde tahmin edici yoktur, OneR

modelinde tek en iyi tahmin ediciyi bulmaya çalışılır, saf Bayesian, Bayes kuralını ve tahmin ediciler arasındaki bağımsızlık varsayımlarını kullanan tüm tahmin edicileri içerir, ancak karar ağacı, tahmin ediciler arasındaki bağımlılık varsayımlarına sahip tüm tahmin edicileri içerir.

Entropy

Bir karar ağacı, bir kök düğümden yukarıdan aşağıya oluşturular ve verileri benzer değerlere sahip (homojen) örnekler içeren alt kümelere ayırmayı içerir. ID3 algoritması, bir örneğin homojenliğini hesaplamak için entropiyi kullanır. Örnek tamamen homojen (Olma olasılığı 1 ise olma olasılığı sıfırdır. Tüm olasıklar toplamı bire eşit olmak zorundadır) ise entropi sıfırdır ve örnek eşit olarak bölünmüşse entropisi birdir.



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

$$y = \text{Log2}(x) = \text{Log10}(x)/\log_2(10)$$

$$\text{Log10}(2)=0.3, \text{Log10}(3)=0.477, \text{Log10}(5)=0.7, \text{Log10}(7)=0.845$$

Bir karar ağacı oluşturmak için aşağıdaki gibi sıklık tablolarını kullanarak iki tür entropi hesaplamamız gereklidir:

- a) Bir özelliğin sıklık tablosunu kullanan entropi:

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5



$$\begin{aligned}\text{Entropy(PlayGolf)} &= \text{Entropy}(5,9) \\ &= \text{Entropy}(0.36, 0.64) \\ &= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\ &= 0.94\end{aligned}$$

$$5/14=0.36$$

$$9/14=0.64$$

b) İki özelliğin sıkılık tablosunu kullanan entropi:

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

Play Golf				
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14



$$\begin{aligned}E(\text{PlayGolf, Outlook}) &= P(\text{Sunny}) * E(3,2) + P(\text{Overcast}) * E(4,0) + P(\text{Rainy}) * E(2,3) \\ &= (5/14) * 0.971 + (4/14) * 0.0 + (5/14) * 0.971 \\ &= 0.693\end{aligned}$$

Information Gain

Bilgi kazancı, bir veri kümesi bir öznitelijke bölündükten sonra entropideki azalmaya dayanır. Bir karar ağacı oluşturmak, en yüksek bilgi kazancını (yani en homojen dalları) döndüren özniteligi bulmakla ilgilidir.

Adım 1: Hedefin entropisini hesaplanır.

$$\text{Entropy}(\text{PlayGolf}) = \text{Entropy}(5,9)$$

$$= \text{Entropy}(0.36, 0.64)$$

$$= - (0.36 \log_2 0.36) - (0.64 \log_2 0.64)$$

$$= 0.94$$

Adım 2: Veri kümesi daha sonra farklı niteliklere bölünür. Her dal için entropi hesaplanır. Daha sonra, bölme için toplam entropi elde etmek için orantılı olarak eklenir. Ortaya çıkan entropi, bölünmeden önceki entropiden çıkarılır. Sonuç, Bilgi Kazanımı veya entropideki azalmadır.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

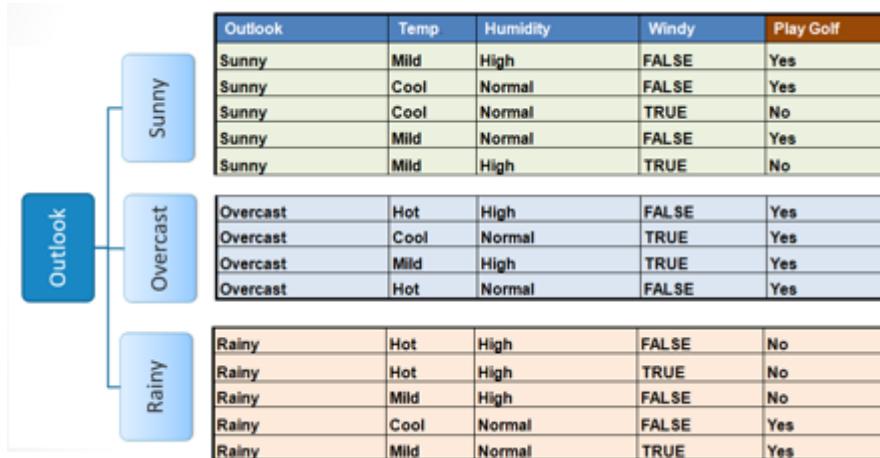
$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

$$G(\text{PlayGolf}, \text{Outlook}) = E(\text{PlayGolf}) - E(\text{PlayGolf}, \text{Outlook})$$

$$= 0.940 - 0.693 = 0.247$$

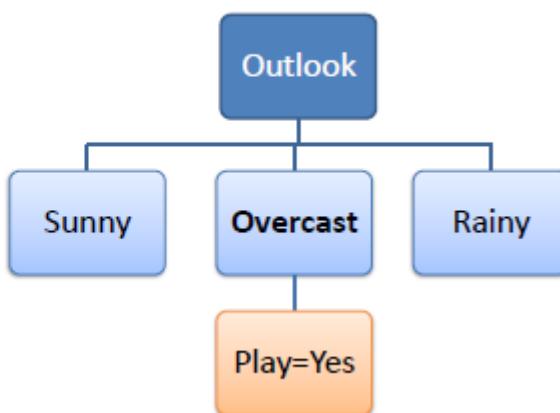
Adım 3: Karar düğümü olarak en büyük bilgi kazancına sahip öznitelik seçilir, veri seti dallarına bölünür ve aynı işlemi her dalda tekrarlanır.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



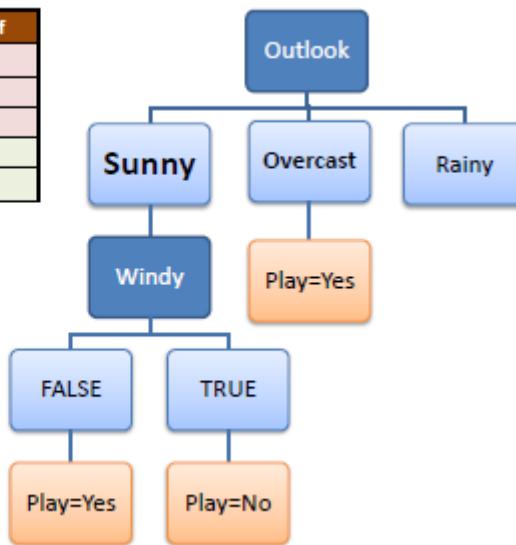
Step 4a: Entropisi 0 olan bir dal, bir yaprak düğümdür.

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



Adım 4b: Entropisi 0'dan büyük olan bir dalın daha fazla bölünmesi gereklidir.

Temp.	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Adım 5: ID3 algoritması, tüm veriler sınıflandırılana kadar yaprak olmayan dallarda özyinelemeli olarak çalıştırılır.

Karar Ağacından Karar Kurallarına

Bir karar ağacı, kök düğümden yaprak düğümlere tek tek eşlenerek kolayca bir dizi kurala dönüştürülebilir.

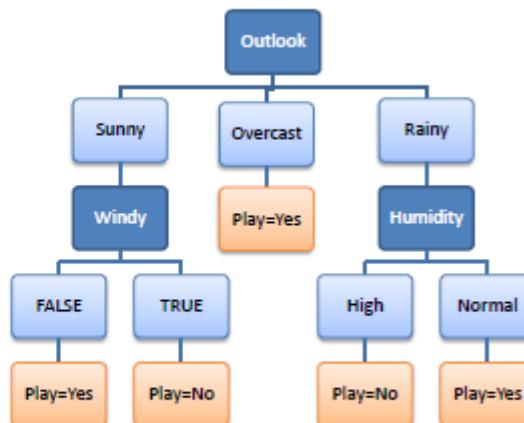
R₁: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R₂: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

R₃: IF (Outlook=Overcast) THEN Play=Yes

R₄: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R₅: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes



7.2.2. K-En Yakın Komşu

1967 yılında T. M. Cover ve P. E. Hart tarafından önerilen, örnek veri noktasının bulunduğu sınıfın ve en yakın komşunun, K değerine göre belirlendiği bir sınıflandırma yöntemidir. Denetimli öğrenmede sınıflandırma ve regresyon için kullanılan algoritmaların biridir. En basit makine öğrenmesi algoritması olarak kabul edilir.

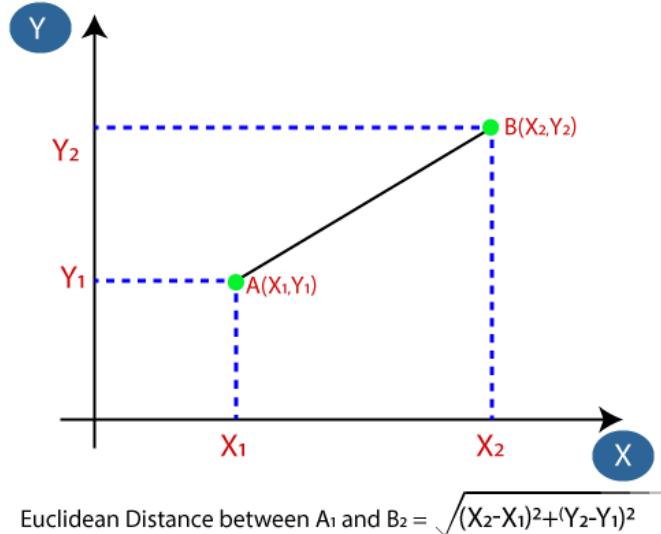
KNN amacı, yeni bir örnek geldiğinde var olan öğrenme verisi üzerinde sınıflandırma yaparak onun en yakın K komşusuna bakarak örneğin sınıfına karar verir.

K-NN algoritması sınıflandırma algoritmasıdır. Sınıflandırmak, belirli bir veri kümesini farklı sınıflara ayırmayı içermektedir. Sınıflandırma hem yapılandırılmış hem de yapılandırılmamış veri türleri üzerinde uygulanabilir.

KNN algoritması sınıflandırılmak istenen bir veriyi daha önceki verilerle olan yakınlık ilişkisine göre sınıflandıran bir algoritmadır. Algoritma adının içinde bulunduğu “K” algoritmaya dahil edilecek veri kümesindeki veri sayısını ifade etmektedir. Yani algoritmada “k” adet komşu aranır. Bir tahmin yapmak istediğimizde, tüm veri setinde en yakın komşuları arar. Algoritmanın çalışmasında bir K değeri belirlenir. Bu K değerinin anlamı bakılacak eleman sayısıdır. Bir değer geldiğinde en yakın K kadar eleman alınarak gelen değer arasındaki uzaklık hesaplanır. İlgili uzaklıklardan en yakın k komşu ele alınır. Öznitelik değerlerine göre k komşu veya komşuların sınıfına atanır. Seçilen sınıf, tahmin edilmesi beklenen gözlem değerinin sınıfı olarak kabul edilir. Yani yeni veri etiketlenmiş (label) olur.

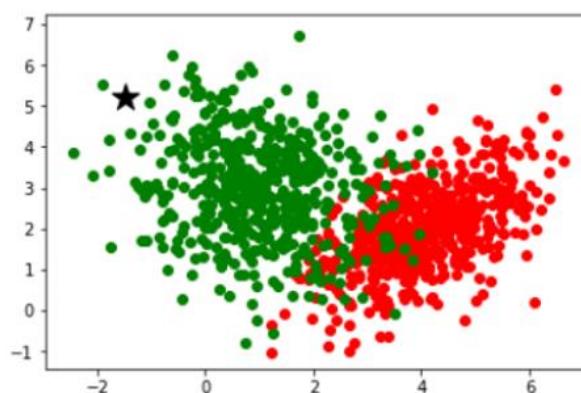
K-NN non-parametric (parametrik olmayan), lazy (tembel) bir öğrenme algoritmasıdır. Lazy kavramını anlamaya çalışırsak “eager learning” aksine “lazy learning” in bir eğitim aşaması yoktur. Eğitim verilerini öğrenmez, bunun yerine eğitim veri kümesini “ezberler”. Uzaklık hesaplama işleminde genelde Öklid fonksiyonu kullanılır.

Öklid fonksiyonuna alternatif olarak Manhattan, Minkowski ve Hamming fonksiyonlarında kullanılabilir. Uzaklık hesaplandıktan sonra sıralanır ve gelen değer uygun olan sınıfına atanır.



Bir tahmin yapmak için KNN algoritması, lojistik veya doğrusal regresyonda olduğu gibi bir eğitim veri kümesinden öngörücü bir model hesaplamaz. Aslında, KNN'nin tahmine dayalı bir model oluşturmamasına gerek yoktur. Bu nedenle, KNN için gerçek bir öğrenme aşaması yoktur. Bu yüzden genellikle tembel bir öğrenme yöntemi olarak kategorize edilir. Bir tahmin yapabilmek için, KNN herhangi bir eğitim aşaması olmadan bir sonuç üretmek için veri setini kullanır.

KNN, bir tahmin yapmak için tüm veri kümesini depolar. KNN herhangi bir tahmine dayalı modeli hesaplamaz ve tembel öğrenme algoritmaları ailesinin bir parçasıdır. KNN, bir girdi gözlemi ile veri kümesindeki farklı gözlemler arasındaki benzerliği hesaplayarak tam zamanında (anında) tahminler yapar.



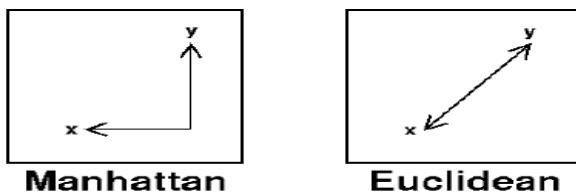
Yukarıdaki şekilde, kırmızı veya yeşil olarak sınıflandırılan veri noktalarında siyah bir veri noktası kırmızı ya da yeşil olabilecek iki sınıfı göstermektedir. Siyah noktanın ne olduğu KNN algoritması tarafından nasıl hesaplanır?

KNN algoritmaları, sınıflandırılacak veri noktasına en yakın komşu olan bir k sayısına karar verir. K değeri 5 ise, o veri noktasına en yakın 5 Komşuyu arayacaktır. Bu örnekte, k = 4. KNN en yakın 4 komşuyu bulur. Bu veri noktasının bu komşulara yakın olması nedeniyle sadece bu sınıfı ait olacağı görülmektedir.

K-en yakın komşu sınıflandırıcı algoritmalarının basit versiyonu, en yakın komşu sınıfı bularak hedef etiketini tahmin etmektir. Sınıflandırılacak noktaya en yakın sınıf, Öklid mesafesi kullanılarak hesaplanır.

Mesafe, benzerliği ölçmek için kullanılır. İki örnek arasındaki mesafeyi ölçmenin birçok yolu vardır.

- Manhattan Distance, $|X_1-X_2| + |Y_1-Y_2|$
- Euclidean Distance, $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$



Mesafe Ölçüm yöntemleri:

Minkowsky:

$$D(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^m |x_i - y_i|^r \right)^{1/r}$$

Euclidean:

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^m (x_i - y_i)^2}$$

Manhattan / city-block:

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^m |x_i - y_i|$$

Camberra:

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^m \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(\mathbf{x}, \mathbf{y}) = \max_{i=1}^m |x_i - y_i|$$

Quadratic:

$$D(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T Q (\mathbf{x} - \mathbf{y}) = \sum_{j=1}^m \left(\sum_{i=1}^m (x_i - y_i) q_{ji} \right) (x_j - y_j)$$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$$D(\mathbf{x}, \mathbf{y}) = [\det V]^{1/m} (\mathbf{x} - \mathbf{y})^T V^{-1} (\mathbf{x} - \mathbf{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occurring in the training set instances $1..n$.

Correlation:

$$D(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^m (x_i - \bar{x}_i)(y_i - \bar{y}_i)}{\sqrt{\sum_{i=1}^m (x_i - \bar{x}_i)^2 \sum_{i=1}^m (y_i - \bar{y}_i)^2}}$$

$\bar{x}_i = \bar{y}_i$ and is the average value for attribute i occurring in the training set.

Chi-square:

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^m \frac{1}{\text{sum}_i} \left(\frac{x_i}{\text{size}_x} - \frac{y_i}{\text{size}_y} \right)^2$$

sum_i is the sum of all values for attribute i occurring in the training set, and size_x is the sum of all values in the vector \mathbf{x} .

Kendall's Rank Correlation:

$$D(\mathbf{x}, \mathbf{y}) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^m \sum_{j=1}^{i-1} \text{sign}(x_i - x_j) \text{sign}(y_i - y_j)$$

$\text{sign}(x) = -1, 0 \text{ or } 1 \text{ if } x < 0,$
 $x = 0, \text{ or } x > 0, \text{ respectively.}$

Figure 1. Equations of selected distance functions.
(\mathbf{x} and \mathbf{y} are vectors of m attribute values).

Mesafe ölçüsünün de yalnızca sürekli değişkenler için geçerli olduğu unutulmamalıdır. Kategorik değişkenler durumunda Hamming mesafesi kullanılmalıdır. Veri setinde sayısal ve kategorik değişkenlerin bir karışımı olduğunda, 0 ile 1 arasındaki sayısal değişkenlerin standardizasyonu konusunu da gündeme getirir.

Hamming Distance

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

K'nin önemi nedir?

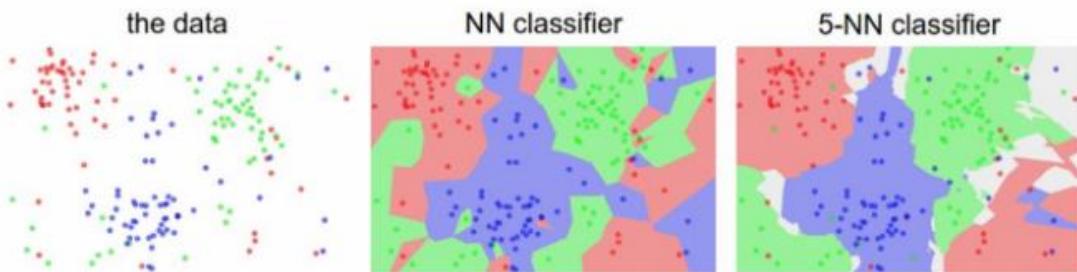
K değeri büyükçe tahmine duyulan güveni artırır. Öte yandan K çok büyük bir değere sahipse, kararlar çarpık olabilir.

K nasıl seçilir?

Algoritma, yeni bir veri noktasının diğer tüm eğitim veri noktalarına olan mesafesini hesaplar. Mesafe herhangi bir türde olabilir, örneğin Öklid, Manhattan, vb. Algoritma daha sonra k'ye en yakın veri noktalarını seçer, burada k herhangi bir tam sayı olabilir. Sayısal değerlerin hangi özelliği temsil ettiğine bakılmaksızın, seçimini diğer veri noktalarına yakınlığına göre yapar. Son olarak, veri noktasını benzer veri noktalarının bulunduğu sınıfı atar.

Seçilen veri kümesine uyan K değerini seçmek için, KNN algoritması farklı K değerleri ile defalarca çalıştırılır. Sonra, algoritma, yeni değerler için hassas tahminler yapma yeteneğini korurken karşılaşılan hata sayısını azaltan K'yi seçer.

- K'ye karar vermek, K-en yakın Komşular'ın en kritik kısmıdır.
- K değeri küçükse, gürültü sonuca daha fazla bağımlı olacaktır. Bu gibi durumlarda modelin aşırı uyumu çok fazladır.
- K'nin değeri ne kadar büyükse, KNN'nin arkasındaki prensibi yok edecektir.
- Çapraz doğrulamayı kullanarak K'nin optimum değeri bulunabilir.

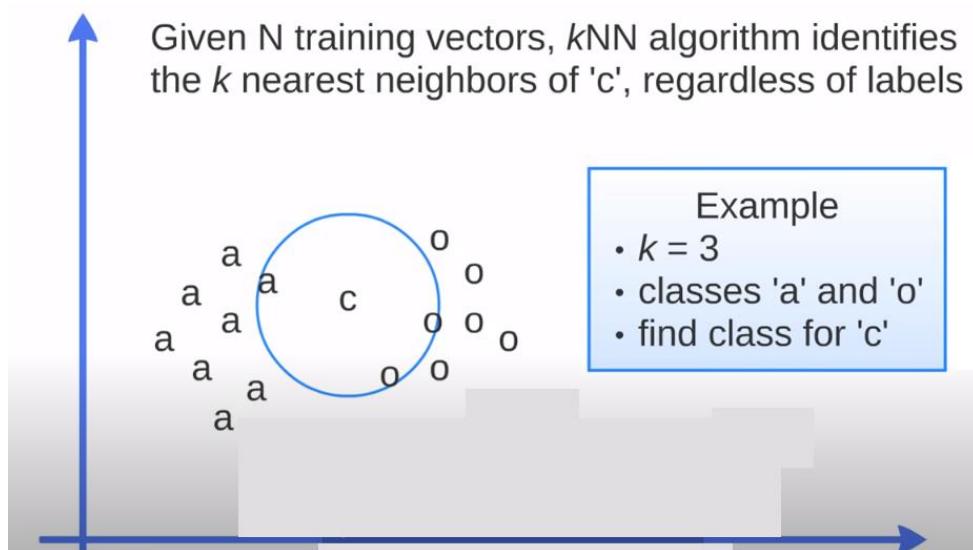


KNN algoritması sözde kod uygulaması

- İstenilen verileri yüklenir.
- k parametresi belirlenir. Bu parametre verilen bir noktaya en yakın komşuların sayısıdır. Örneğin: $k=2$ olsun. Bu durumda en yakın 2 komşuya göre sınıflandırma yapılacaktır.
- Örnek veri setine katılacak olan yeni verinin, mevcut verilere göre uzaklığı tek tek hesaplanır. İlgili uzaklık fonksiyonları yardımıyla.
- İlgili uzaklıklardan en yakın k komşu ele alınır. Öz nitelik değerlerine göre k komşu veya komşuların sınıfına atanır.
- Seçilen sınıf, tahmin edilmesi beklenen gözlem değerinin sınıfı olarak kabul edilir. Yani yeni veri etiketlenmiş (label) olur.

KNN, anlaşılması oldukça basit bir algoritmadır. Bunun başlıca nedeni, tahmin yapabilmek için bir modele ihtiyaç duymamasıdır. Bunun tersi, tahminini yapabilmek için tüm gözlemlerini hafızasında tutması gerektidir. Bu nedenle, girdi veri kümесinin boyutuna dikkat etmeniz gereklidir. Ayrıca, mesafenin hesaplanması için yöntemin seçimi ve komşuların K sayısı hemen belli olmayabilir. Kullanım durumunuz için tatmin edici bir sonuç elde etmek için birkaç kombinasyon denemeniz ve algoritmayı ayarlamamanız gerekebilir.

Örnek:



Öğrencinin Adı	Girdi Özellikleri		Çıktı Özneliği Başarı Durumu
	Sayfa Sayısı	Soru Sayısı	
Selin	2100	1500	BAŞARILI
Mert	2280	1600	BAŞARILI
Caner	800	400	BAŞARISIZ
Şenay	2400	1750	BAŞARILI
Efe	250	350	BAŞARISIZ
Burak	180	220	BAŞARISIZ
Esra	1800	1200	?

Öklid uzaklığı kullanılarak altı adet öğrencinin Esra adlı öğrenciye olan uzaklıkları:

Öğrencinin Adı	Sayfa Sayısı	Soru Sayısı	Başarı Durumu	Uzaklık Hesaplaması	Sonuç
Selin	2100	1500	BAŞARILI	$\sqrt{(2100 - 1800)^2 + (1500 - 1200)^2}$	424.2
Mert	2280	1600	BAŞARILI	$\sqrt{(2280 - 1800)^2 + (1600 - 1200)^2}$	624.8
Caner	800	400	BAŞARISIZ	$\sqrt{(800 - 1800)^2 + (400 - 1200)^2}$	1280
Şenay	2400	1750	BAŞARILI	$\sqrt{(2400 - 1800)^2 + (1750 - 1200)^2}$	813.9
Efe	250	350	BAŞARISIZ	$\sqrt{(250 - 1800)^2 + (350 - 1200)^2}$	1767
Burak	180	220	BAŞARISIZ	$\sqrt{(180 - 1800)^2 + (220 - 1200)^2}$	1893
Esra	1800	1200	?		

Örneğin K=3 seçersek ilk 3 öğrencinin başarı durumuna bakacağız.

Öğrencinin Adı	Başarı Durumu	Sonuç	Sıra No
Selin	BAŞARILI	424.2	1
Mert	BAŞARILI	624.8	2
Şenay	BAŞARILI	813.9	3
Caner	BAŞARISIZ	1280	4
Efe	BAŞARISIZ	1767	5
Burak	BAŞARISIZ	1893	6
Esra	?		

Sonuç BAŞARILI

Örneğin K=6 seçersek ilk 6 öğrencinin başarı durumuna bakacağız.

Öğrencinin Adı	Başarı Durumu	Sonuç	Sıra No
Selin	BAŞARILI	424.2	1
Mert	BAŞARILI	624.8	2
Şenay	BAŞARILI	813.9	3
Caner	BAŞARISIZ	1280	4
Efe	BAŞARISIZ	1767	5
Burak	BAŞARISIZ	1893	6
Esra	?		

Sonuç ?

Bu yüzden K parametresinin **çift** olarak seçilmesi tercih edilmemektedir.

Örnek:

KNN, eğitim seti yardımıyla veri noktasını belirli bir kategoriye sınıflandırmaya çalıştığımız parametrik olmayan denetimli bir öğrenme teknigidir. Basit bir deyişle, tüm eğitim durumlarının bilgilerini yakalar ve yeni durumları benzerliğe göre sınıflandırır.

Yeni bir örnek (x) için, en benzer K durum (komşular) için tüm eğitim seti aranarak ve bu K durumlar için çıktı değişkeni özetlenerek tahminler yapılır. Sınıflandırmada bu, mod (veya en yaygın) sınıf değeridir.

KNN algoritması nasıl çalışır?

Diyelim ki bazı müşterilerin boy, kilo ve tişört bedenleri var ve yeni bir müşterinin Tişört bedenini sadece sahip olduğumuz boy ve kilo bilgisine göre tahmin etmemiz gerekiyor. Boy, kilo ve tişört beden bilgilerini içeren veriler aşağıda gösterilmiştir.

Height (in cms)	Weight (in kgs)	T Shirt Size
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L

Adım 1: Uzaklık fonksiyonuna göre Benzerliği hesaplayın

Pek çok uzaklık fonksiyonu vardır ama en yaygın olarak kullanılan ölçü Ökliddir. Esas olarak veriler sürekli olduğunda kullanılır. Manhattan mesafesi de sürekli değişkenler için çok yaygındır.

Euclidean :

$$d(x, y) = \sqrt{\sum_{i=1}^m (x_i - y_i)^2}$$

Manhattan / city - block :

$$d(x, y) = \sum_{i=1}^m |x_i - y_i|$$

Distance Functions

Mesafe ölçümü kullanma fikri, yeni örnek ve eğitim durumları arasındaki mesafeyi (benzerliği) bulmak ve ardından boy ve ağırlık açısından yeni müşteriye en yakın k-müşterileri bulmaktadır.

'Monica' adlı yeni müşteri 161cm boyunda ve 61kg ağırlığındadır. İlk gözlem ile yeni gözlem (monica) arasındaki Öklid uzaklığı aşağıdaki gibidir:

$$=\text{SQRT}((161-158)^2+(61-58)^2)$$

Benzer şekilde, yeni vaka ile tüm eğitim vakalarının mesafesini hesaplayacağız ve mesafe açısından sıralamayı hesaplayacağız. En küçük mesafe değeri 1 olarak sıralanır ve en yakın komşu olarak kabul edilir.

Step 2 : Find K-Nearest Neighbors

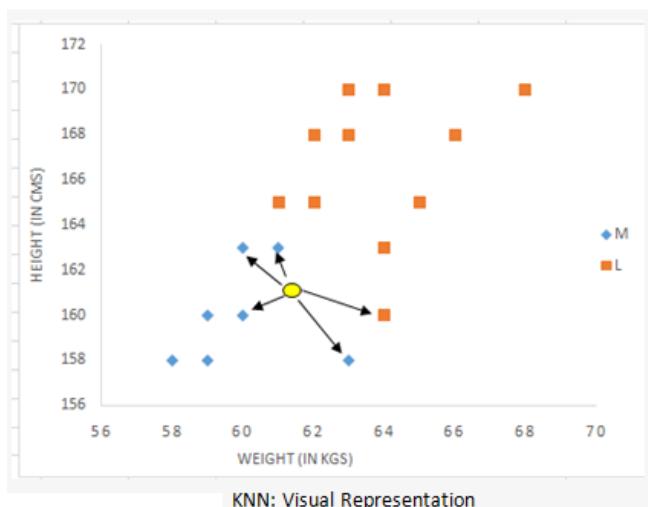
Let k be 5. Then the algorithm searches for the 5 customers closest to Monica, i.e. most similar to Monica in terms of attributes, and see what categories those 5 customers were in. If 4 of them had 'Medium T shirt sizes' and 1 had 'Large T shirt size' then your best guess for Monica is 'Medium T shirt. See the calculation shown in the snapshot below –

2. Adım : K-En Yakın Komşuları Bulunması

K=5 olsun. Ardından algoritma, özellikler açısından Monica'ya en yakın, yani Monica'ya en çok benzeyen 5 müşteriyi arar ve bu 5 müşterinin hangi kategorilerde olduğunu görür. 4 tanesi 'Orta T shirt bedenleri' ve 1 tanesi ise 'Büyük T gömlek bedeni' vardı, o zaman Monica için en iyi tahmininiz 'Orta T gömlek. Aşağıdaki anlık görüntüde gösterilen hesaplamaya bakın

	A	B	C	D	E
1	Height (in cms)	Weight (in kgs)	T Shirt Size	Distance	
2	158	58	M	4.2	
3	158	59	M	3.6	
4	158	63	M	3.6	
5	160	59	M	2.2	3
6	160	60	M	1.4	1
7	163	60	M	2.2	3
8	163	61	M	2.0	2
9	160	64	L	3.2	5
10	163	64	L	3.6	
11	165	61	L	4.0	
12	165	62	L	4.1	
13	165	65	L	5.7	
14	168	62	L	7.1	
15	168	63	L	7.3	
16	168	66	L	8.6	
17	170	63	L	9.2	
18	170	64	L	9.5	
19	170	68	L	11.4	
20					
21	161	61			

Aşağıdaki grafikte ikili bağımlı değişken (T-shirt bedeni) mavi ve turuncu renkte görüntülenmektedir. 'Orta boy tişört' mavi renkte ve 'Büyük boy tişört' turuncu renktedir. Yeni müşteri bilgileri sarı daire içinde sergilendir. Dört mavi vurgulanmış veri noktası ve bir turuncu vurgulanmış veri noktası sarı daireye yakındır. bu nedenle yeni vaka için tahmin, Orta T-shirt boyutu olan mavi vurgulu veri noktasıdır.



KNN Varsayımları:

1. Standardizasyon

Antrenman verilerindeki bağımsız değişkenler farklı birimlerde ölçüldüğünde, mesafeyi hesaplamadan önce değişkenleri standardize etmek önemlidir. Örneğin, bir değişken cm cinsinden yüksekliği temel alıyorsa ve diğer kg cinsinden ağırlığı temel alıyorsa, uzunluk mesafe hesaplamasını daha fazla etkileyecektir. Bunları karşılaştırılabilir hale getirmek için, aşağıdaki yöntemlerden herhangi biriyle yapılabilecekleri standartlaştırılmamız gereklidir:

$$X_s = \frac{X - \text{mean}}{\text{s. d.}}$$
$$X_s = \frac{X - \text{mean}}{\text{max} - \text{min}}$$
$$X_s = \frac{X - \text{min}}{\text{max} - \text{min}}$$

Standardization

Standardizasyondan önce yükseklik hakim olduğundan, standardizasyondan sonra 5. en yakın değer değişmiştir. Bu nedenle, K-en yakın komşu algoritmasını çalıştırmadan önce tahmin edicileri standart hale getirmek önemlidir.

	A	B	C	D	E
1	Height (in cms)	Weight (in kgs)	T Shirt Size	Distance	
2	-1.39	-1.64	M	1.3	
3	-1.39	-1.27	M	1.0	
4	-1.39	0.25	M	1.0	
5	-0.92	-1.27	M	0.8	4
6	-0.92	-0.89	M	0.4	1
7	-0.23	-0.89	M	0.6	3
8	-0.23	-0.51	M	0.5	2
9	-0.92	0.63	L	1.2	
10	-0.23	0.63	L	1.2	
11	0.23	-0.51	L	0.9	5
12	0.23	-0.13	L	1.0	
13	0.23	1.01	L	1.8	
14	0.92	-0.13	L	1.7	
15	0.92	0.25	L	1.8	
16	0.92	1.39	L	2.5	
17	1.39	0.25	L	2.2	
18	1.39	0.63	L	2.4	
19	1.39	2.15	L	3.4	
20					
21	-0.7	-0.5			

Knn after standardization

2. Aykırı Değer

Düşük k-değeri aykırı değerlere duyarlıdır ve daha yüksek bir K-değeri, daha fazla seçmenin tahmine karar vereceğini düşündüğü için aykırı değerlere karşı daha dirençlidir.

KNN neden parametrik değildir?

Parametrik olmayan, temel alınan veri dağılımı üzerinde herhangi bir varsayımda bulunmamak anlamına gelir. Parametrik olmayan yöntemler, modelde sabit sayıda parametreye sahip değildir. Benzer şekilde KNN'de model parametreleri aslında eğitim veri seti ile birlikte büyür - her eğitim durumunu modelde bir "parametre" olarak düşünebilirsiniz.

KNN ve K-ortalama: Birçok insan bu iki istatistiksel teknik - K-ortalama ve K-en yakın komşu arasında kafa karıştırır. Aşağıdaki farklardan bazılarına bakın:

- K-ortalama denetimsiz bir öğrenme tekniğidir (bağımlı değişken yoktur), KNN denetimli bir öğrenme algoritmasıdır (bağımlı değişken vardır)
- K-ortalama, veri noktalarını her kümedeki noktalar birbirine yakın olacak şekilde K-kümelerine ayırmaya çalışan bir kümeleme tekniğidir, K-en yakın komşu ise bir noktanın sınıflandırmasını belirlemeye çalışır, K sınıflandırmasını en yakın noktalara birleştirmeye çalışır.

KNN regresyon için kullanılabilir mi?

Evet, regresyon için K-en yakın komşu kullanılabilir. Başka bir deyişle, bağımlı değişken sürekli olduğunda K-en yakın komşu algoritması uygulanabilir. Bu durumda, tahmin edilen değer, en yakın k komşusunun değerlerinin ortalamasıdır.

KNN'nin Artıları ve Eksileri

Artıları:

- Anlaması kolay
- Veriler hakkında varsayıım yok
- Hem sınıflandırma hem de regresyona uygulanabilir
- Çok sınıfı problemlerde kolayca çalışır

Eksileri:

- Yoğun Bellek / Hesaplama açısından pahalı
- Veri ölçüğine duyarlı
- Nadir olay (çarpık) hedef değişkeni üzerinde iyi çalışmıyor
- Çok sayıda bağımsız değişken olduğunda mücadele
- Herhangi bir problem için, küçük bir k değeri, tahminlerde büyük bir varyansa yol açacaktır. Alternatif olarak, k değerini büyük bir değere ayarlamak, büyük bir model yanılığına yol açabilir.

KNN'de kategorik değişkenler nasıl ele alınır?

Kategorik bir değişkenden kukla değişkenler oluşturun ve orijinal kategorik değişken yerine bunları dahil edin. Regresyondan farklı olarak, $(k-1)$ yerine k kukla oluşturun. Örneğin, "Departman" adlı kategorik bir değişkenin 5 benzersiz düzeyi/kategorisi vardır. Böylece 5 kukla değişken oluşturacağız. Her kukla değişkenin departmanına karşı 1 ve 0'a sahiptir.

En iyi K değeri nasıl bulunur?

Çapraz doğrulama, optimal K değerini bulmanın akıllı bir yoludur. Model oluşturma sürecinden eğitim setinin bir alt kümesini dışında tutarak doğrulama hata oranını tahmin eder.

Çapraz doğrulama (diyelim ki 10 kat doğrulama), eğitim setinin rastgele olarak yaklaşık eşit büyülükte 10 gruba veya katlara bölünmesini içerir. Verilerin %90'ı modeli eğitmek için, kalan %10'u ise onu doğrulamak için kullanılır. Yanlış sınıflandırma oranı daha sonra %10 doğrulama verisi üzerinden hesaplanır. Bu prosedür 10 kez tekrarlanır. Farklı gözlem grupları, 10 defadan her biri bir doğrulama seti olarak ele alınır. Daha sonra ortalaması alınan doğrulama hatasının 10 tahminiyle sonuçlanır.

K-Nearest Neighbor(KNN) Algorithm for Machine Learning

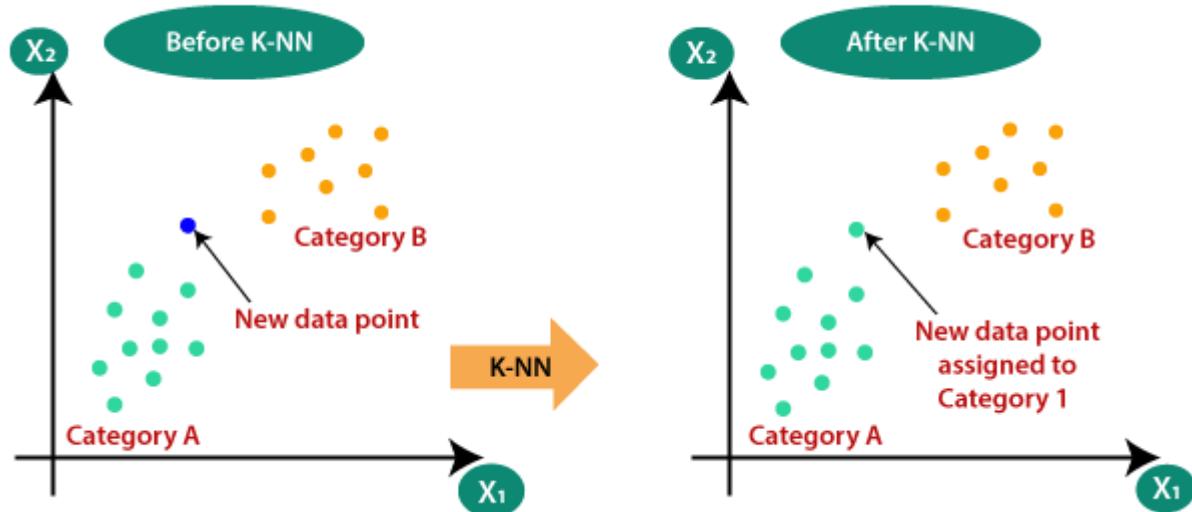
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suited category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

KNN Classifier



Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x_1 , so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

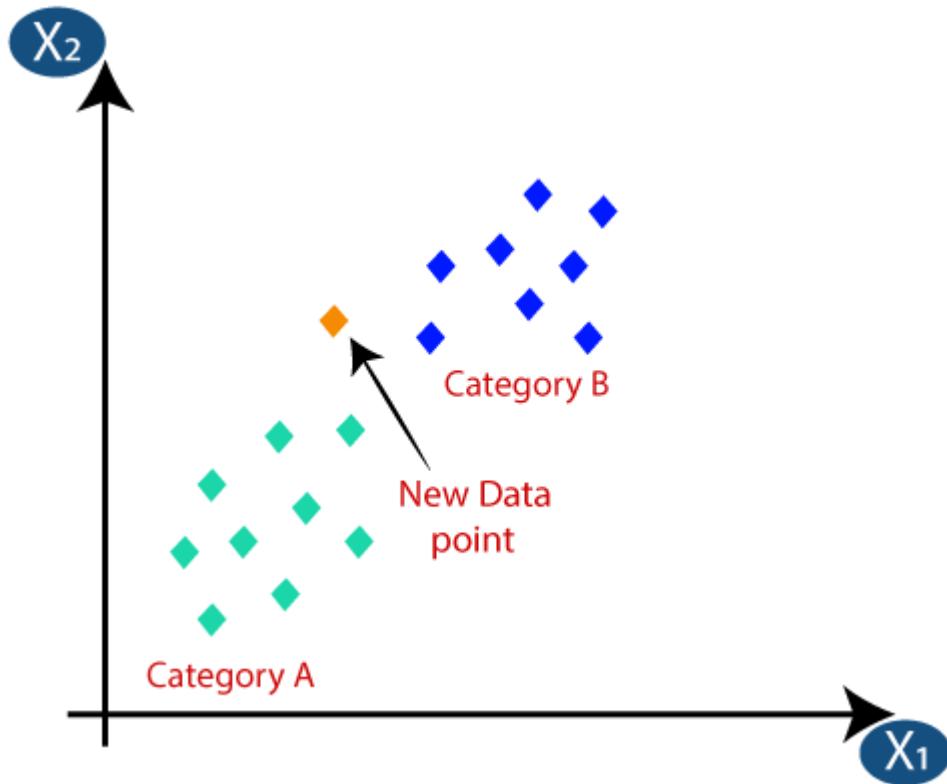


How does K-NN work?

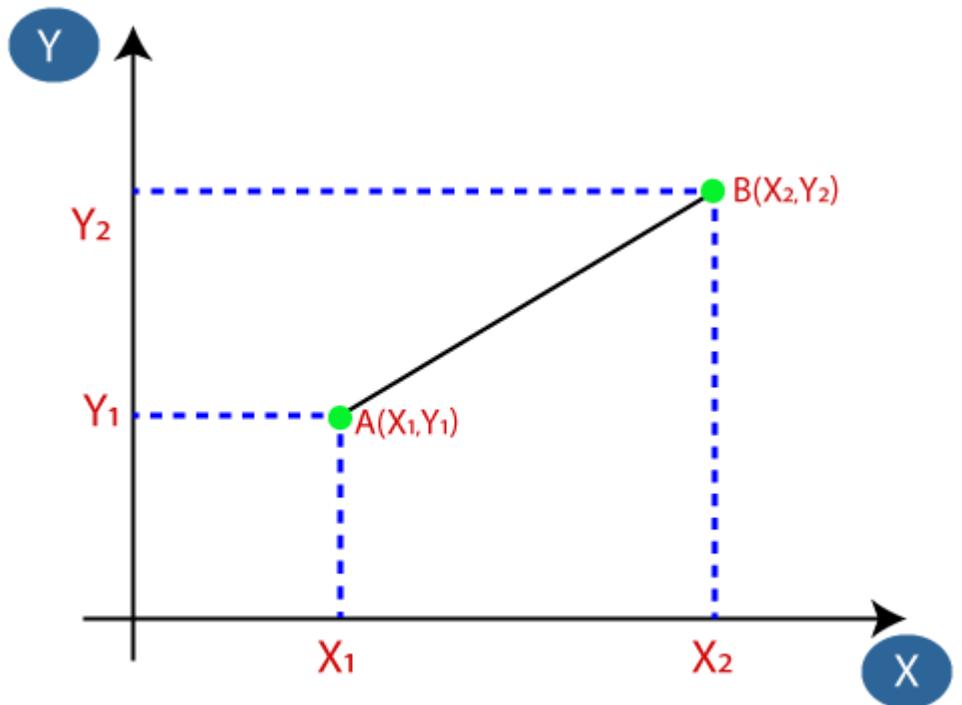
The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

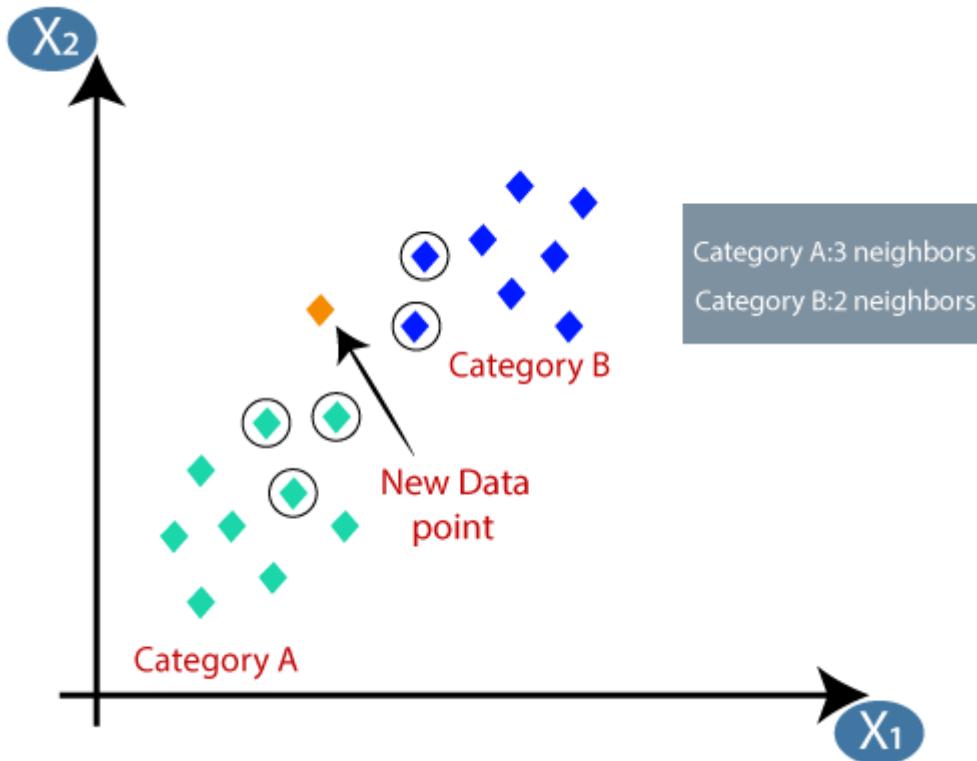


- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



$$\text{Euclidean Distance between } A_1 \text{ and } B_2 = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

9.3M

196

HTML Tutorial

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Python implementation of the KNN algorithm

To do the Python implementation of the K-NN algorithm, we will use the same problem and dataset which we have used in Logistic Regression. But here we will improve the performance of the model. Below is the problem description:

Problem for K-NN Algorithm: There is a Car manufacturer company that has manufactured a new SUV car. The company wants to give the ads to the users who are interested in buying that SUV. So for this problem, we have a dataset that contains multiple user's information through the social network. The dataset contains lots of information but the **Estimated Salary** and **Age** we will consider for the independent variable and the **Purchased variable** is for the dependent variable. Below is the dataset:

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

Steps to implement the K-NN algorithm:

- Data Pre-processing step
- Fitting the K-NN algorithm to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.

Data Pre-Processing Step:

The Data Pre-processing step will remain exactly the same as Logistic Regression. Below is the code for it:

1. # importing libraries

```

2. import numpy as nm
3. import matplotlib.pyplot as mtp
4. import pandas as pd
5. #importing datasets
6. data_set= pd.read_csv('user_data.csv')
7. #Extracting Independent and dependent Variable
8. x= data_set.iloc[:, [2,3]].values
9. y= data_set.iloc[:, 4].values
10. # Splitting the dataset into training and test set.
11. from sklearn.model_selection import train_test_split
12. x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0
   )
13. #feature Scaling
14. from sklearn.preprocessing import StandardScaler
15. st_x= StandardScaler()
16. x_train= st_x.fit_transform(x_train)
17. x_test= st_x.transform(x_test)

```

By executing the above code, our dataset is imported to our program and well pre-processed. After feature scaling our test dataset will look like:

The image shows two separate data preview windows. The left window is titled "x_test - NumPy array" and displays a 13x2 grid of numerical values. The right window is titled "y_test - NumPy array" and displays a 13x1 grid of categorical values. Both windows have standard window controls (minimize, maximize, close) at the top and buttons for "Format", "Resize", and "Background color" at the bottom. The "Save and Close" button is highlighted in blue.

	0	1
0	-0.804802	0.504964
1	-0.0125441	-0.567782
2	-0.309641	0.157046
3	-0.804802	0.273019
4	-0.309641	-0.567782
5	-1.1019	-1.43758
6	-0.70577	-1.58254
7	-0.210609	2.15757
8	-1.99319	-0.0459058
9	0.878746	-0.770734
10	-0.804802	-0.596776
11	-1.00287	-0.422817
12	-0.111576	-0.422817

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0

From the above output image, we can see that our data is successfully scaled.

- o **Fitting K-NN classifier to the Training data:**

Now we will fit the K-NN classifier to the training data. To do this we will import the **KNeighborsClassifier** class of **Sklearn Neighbors** library. After importing the

class, we will create the **Classifier** object of the class. The Parameter of this class will be

- **n_neighbors:** To define the required neighbors of the algorithm. Usually, it takes 5.
- **metric='minkowski':** This is the default parameter and it decides the distance between the points.
- **p=2:** It is equivalent to the standard Euclidean metric.

And then we will fit the classifier to the training data. Below is the code for it:

1. #Fitting K-NN classifier to the training set
2. from sklearn.neighbors **import** KNeighborsClassifier
3. classifier= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2)
4. classifier.fit(x_train, y_train)

Output: By executing the above code, we will get the output as:

Out[10]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                      metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                      weights='uniform')
```

- **Predicting the Test Result:** To predict the test set result, we will create a **y_pred** vector as we did in Logistic Regression. Below is the code for it:
 1. #Predicting the test set result
 2. y_pred= classifier.predict(x_test)

Output:

The output for the above code will be:

y_pred - NumPy array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	1
10	0
11	0
12	0

Format Resize Background color

Save and Close Close

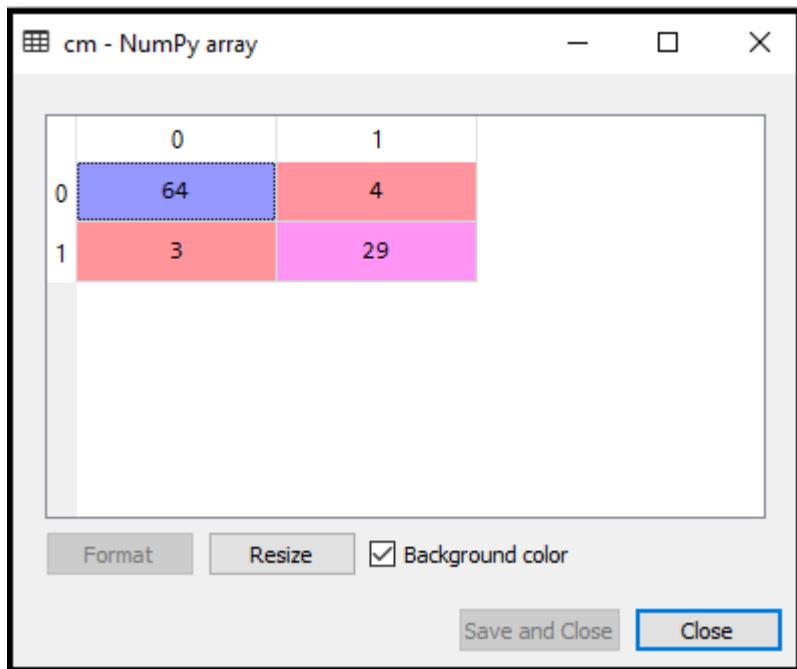
- o **Creating the Confusion Matrix:**

Now we will create the Confusion Matrix for our K-NN model to see the accuracy of the classifier. Below is the code for it:

1. #Creating the Confusion matrix
2. from sklearn.metrics import confusion_matrix
3. cm= confusion_matrix(y_test, y_pred)

In above code, we have imported the confusion_matrix function and called it using the variable cm.

Output: By executing the above code, we will get the matrix as below:



In the above image, we can see there are $64+29= 93$ correct predictions and $3+4= 7$ incorrect predictions, whereas, in Logistic Regression, there were 11 incorrect predictions. So we can say that the performance of the model is improved by using the K-NN algorithm.

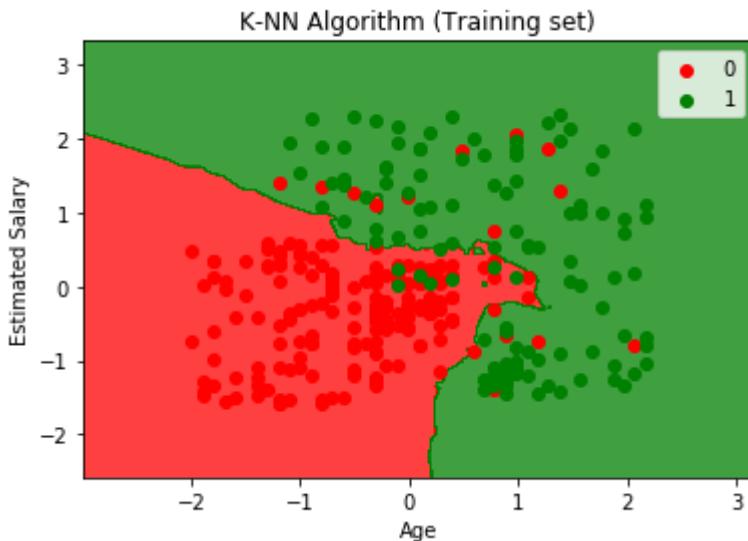
- **Visualizing the Training set result:**

Now, we will visualize the training set result for K-NN model. The code will remain same as we did in Logistic Regression, except the name of the graph. Below is the code for it:

1. #Visulaizing the trianing set result
2. from matplotlib.colors import ListedColormap
3. x_set, y_set = x_train, y_train
4. x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step = 0.01),
5. nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
7. alpha = 0.75, cmap = ListedColormap(('red','green')))
8. mtp.xlim(x1.min(), x1.max())
9. mtp.ylim(x2.min(), x2.max())
10. for i, j in enumerate(nm.unique(y_set)):
11. mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
12. c = ListedColormap(('red', 'green'))(i), label = j)
13. mtp.title('K-NN Algorithm (Training set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()

Output:

By executing the above code, we will get the below graph:



The output graph is different from the graph which we have occurred in Logistic Regression. It can be understood in the below points:

- As we can see the graph is showing the red point and green points. The green points are for Purchased(1) and Red Points for not Purchased(0) variable.
- The graph is showing an irregular boundary instead of showing any straight line or any curve because it is a K-NN algorithm, i.e., finding the nearest neighbor.
- The graph has classified users in the correct categories as most of the users who didn't buy the SUV are in the red region and users who bought the SUV are in the green region.
- The graph is showing good result but still, there are some green points in the red region and red points in the green region. But this is no big issue as by doing this model is prevented from overfitting issues.
- Hence our model is well trained.

○ Visualizing the Test set result:

After the training of the model, we will now test the result by putting a new dataset, i.e., Test dataset. Code remains the same except some minor changes: such as `x_train` and `y_train` will be replaced by `x_test` and `y_test`.

Below is the code for it:

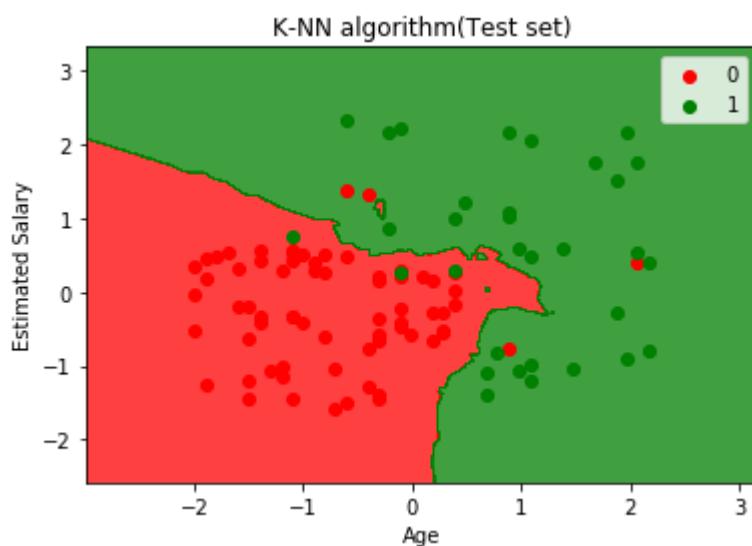
1. #Visualizing the test set result
2. from matplotlib.colors import ListedColormap
3. x_set, y_set = x_test, y_test
4. x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - * 1, stop = x_set[:, 0].max() + 1, step = 0.01), nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
5. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),

```

7. alpha = 0.75, cmap = ListedColormap(['red','green' )))
8. mtp.xlim(x1.min(), x1.max())
9. mtp.ylim(x2.min(), x2.max())
10. for i, j in enumerate(nm.unique(y_set)):
11.     mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
12.                  c = ListedColormap(['red', 'green'))(i), label = j)
13. mtp.title('K-NN algorithm(Test set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()

```

Output:



The above graph is showing the output for the test data set. As we can see in the graph, the predicted output is well good as most of the red points are in the red region and most of the green points are in the green region.

However, there are few green points in the red region and a few red points in the green region. So these are the incorrect observations that we have observed in the confusion matrix(7 Incorrect output).

Örnek:

4 sınıfa ayrılmış askerlere boy, ağırlık ve kafa ölçülerine göre kask dağıtılmaktadır. Yeni bir asker girişi yapılmış. K=8 seçilmiş. KNN yakınlık ölçütleri: G1'e yakınlık 4, G2'ye yakınlık 1, G3'e yakınlık 2 adet, G4'e yakınlık 1 adet ise yeni giriş yapan asker hangi gruptandır.

$$P(G1)=4/8$$

$$P(G2)=1/8$$

$$P(G3)=2/8$$

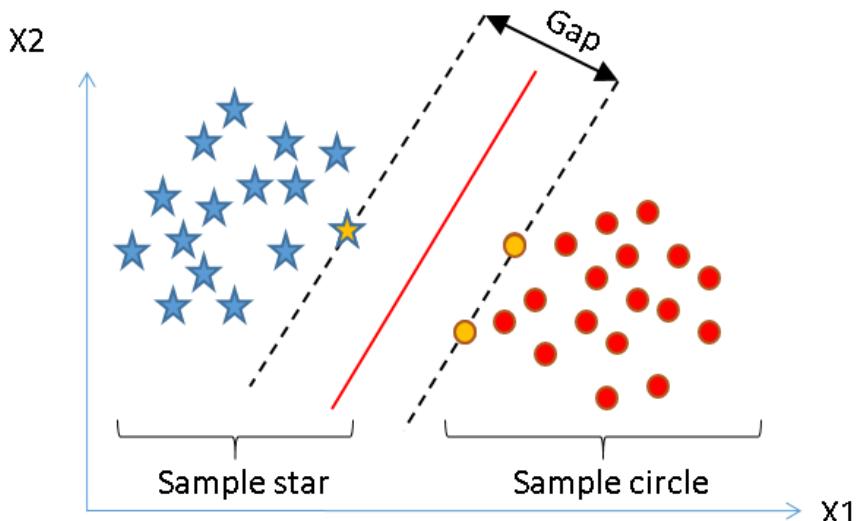
$$P(G4)=1/8$$

O halde asker grup-1'e aittir.

7.2.3. Destek Vektör Makineleri

Destek Vektör Makinesi, farklı sınıfları ayırmak ve sınır marjini en üst düzeye çıkarmak için karar sınırlarını bulmaktan sorumludur. Farklı sınıfların sınırları arasındaki boşluklar, çizgi ile çizgiye en yakın noktalar arasındaki (dik) mesafelerdir. DVM'de sınıflar arasındaki sınırlar çok önemlidir.

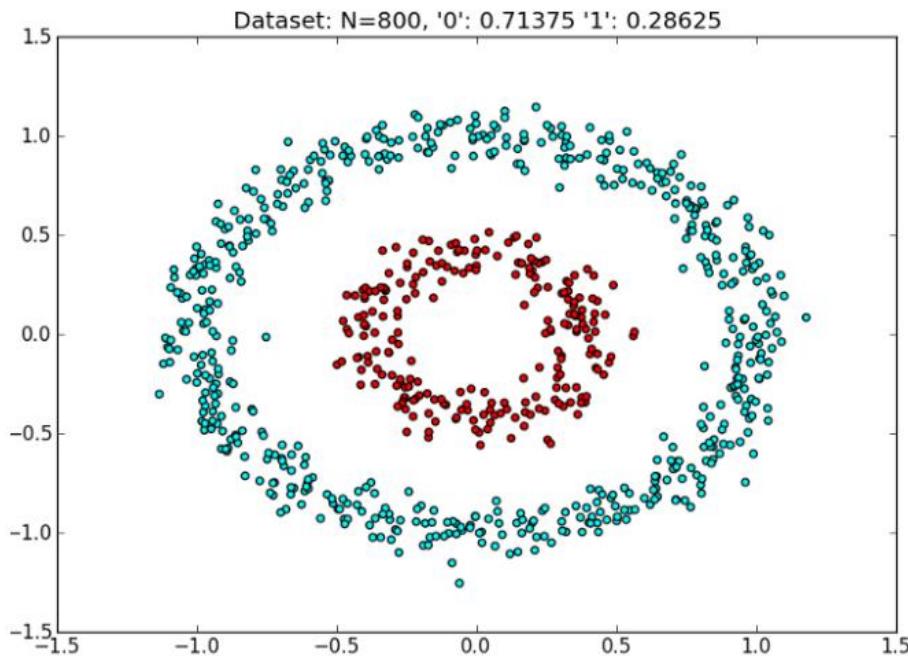
SVM, sınıflandırma için kullanılan denetimli algoritmalar sınıfında yer alır. 2 sınıf bir örnekle başlayalım: Verilen X1 ve X2 sınıfları için, 2 sınıfı en iyi, yani minimum hata ile ayıran karar sınırını bulmak istiyoruz. SVM bunu bir "Hiperplane" ile yapar. Şimdi bu hiperdüzlem 2 boyutlu veri olması durumunda tek bir doğru olabilir ve 3 boyutlu veri olması durumunda bir düzlem olabilir.



Destek Vektör Makineleri, hiper düzleme en yakın noktalar olan 'Destek Vektörleri' kavramını kullanır. Yukarıdaki örnekte, kırmızı çizgi, 2 sınıfı (Mavi yıldızlar ve Kırmızı daireler) ayıran karar sınırımızı gösterir ve tireli çizgiler, her iki sınıfın Destek Vektörleri arasında istediğimiz boşluğu, 'Marj'ımızı temsil eder. Sınırlar Önemlidir

Marj, Destek Vektörlerinin (dolayısıyla adı) yardımıyla tanımlanır. Örneğimizde, Sarı yıldızlar ve Sarı daireler, Marji tanımlayan Destek Vektörleridir. Boşluk ne kadar iyi olursa, sınıflandırıcı o kadar iyi çalışır. Bu nedenle destek vektörleri sınıflandırıcının geliştirilmesinde önemli bir rol oynamaktadır. Test verilerindeki her yeni noktası bu Marj'a göre sınıflandırılacaktır. Sağ tarafındaysa Kırmızı daire, aksi halde Mavi yıldız olarak sınıflandırılır.

En iyi yanı, SVM'nin doğrusal olmayan verileri de sınıflandırılamemesidir.



Doğrusal olmayan veriler söz konusu olduğunda işler biraz zorlaşır. Burada SVM 'Çekirdek hilesi' kullanır, doğrusal olmayan verileri daha yüksek boyutlara eşlemek için bir çekirdek işlevi kullanır, böylece doğrusal hale gelir ve orada karar sınırını bulur. Bir Çekirdek işlevi, ister doğrusal ister doğrusal olmayan veri olsun, SVM tarafından her zaman kullanılır, ancak ana işlevi, veriler mevcut haliyle ayrılmaz olduğunda devreye girer. Burada, Çekirdek işlevi, sınıflandırma sorununa boyutlar ekler.

Soruna bağlı olarak, farklı türde Çekirdek işlevleri kullanabilirsiniz:

- Doğrusal
- Polinom
- Radyal Temel Fonksiyonu
- Gauss
- Laplace

... ve daha fazlası. Doğru çekirdek işlevini seçmek, sınıflandırıcıyı oluşturmak için önemlidir.

Destek vektör ağları olarak da bilinen destek vektör makineleri, sınıflandırma ve regresyon için kullanılan ilgili denetimli öğrenme yöntemleri kümeleridir. Her biri iki kategoriden birine ait olarak işaretlenmiş bir dizi eğitim örneği verildiğinde, bir DVM eğitim algoritması yeni bir örneğin bir kategoriye mi yoksa diğerine mi düştüğünü tahmin eden bir model oluşturur. Bir DVM eğitim algoritması olasılık dışı, ikili, doğrusal bir sınıflandırıcıdır, ancak Platt Ölçeklendirme gibi yöntemler olasılıklı bir sınıflandırma ayarında DVM'yi kullanmak için

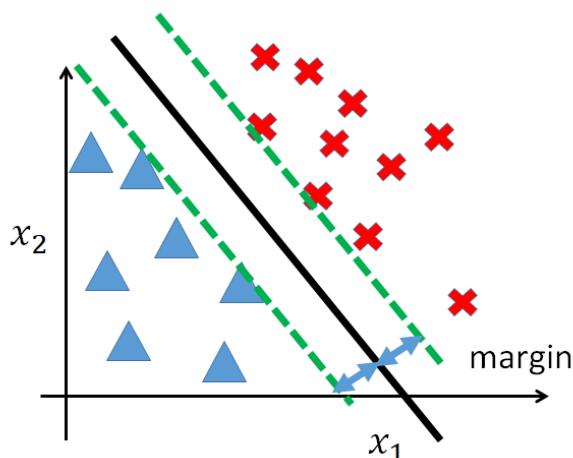
mevcuttur. Doğrusal sınıflandırmayı gerçekleştirmenin yanı sıra, DVM'ler, girdilerini yüksek boyutlu özellik alanlarına dolaylı olarak eşleyerek çekirdek hilesi adı verilen kullanarak doğrusal olmayan bir sınıflandırma gerçekleştirebilir.

SVM'nin (Destek Vektör Makinesi) kullanabileceği iki sınıflandırma yöntemi:

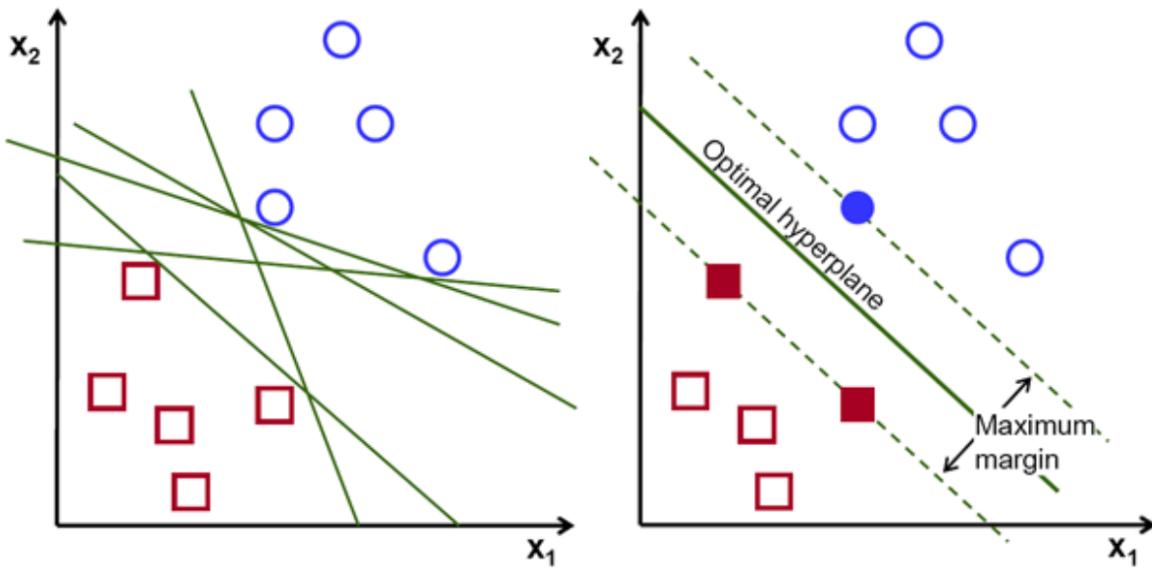
- ikili sınıflandırıcıları birleştirme
- Çok sınıfı öğrenmeyi dahil etmek için ikili programı değiştirme

Destek Vektör Makinesi algoritması oluştururken,

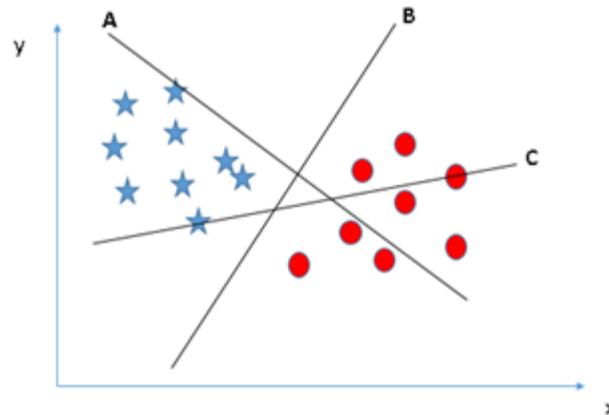
- Sınıflar arasındaki birbirine en yakın ikili seçilir. Bu noktalara destek vektörleri isimleri veriyoruz.
- Destek vektörlerinden geçecek şekilde doğrular çizilir. Bu doğrulara sınır çizgisi adı verilir.
- Her doğruya eşit uzakta çizilen doğruya karar doğrusu adı verilir. Karar doğrusuna hiper düzlem de denir.



- DVM'lerde sınıflar +1 ve -1 olarak etiketlenir. Karar doğrusunun üst kısmında kalan doğru denklemi, $wx+b=1$, altında kalan doğru denklemi ise $wx+b=-1$ olarak belirlenir. Karar doğrusu denklemi ise, $wx+b=0$ olur.

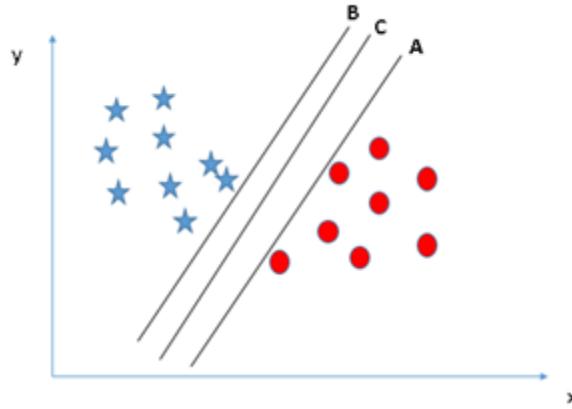


- **Identify the right hyper-plane (Scenario-1):** Here, we have three hyper-planes (A, B, and C). Now, identify the right hyper-plane to classify stars and circles.



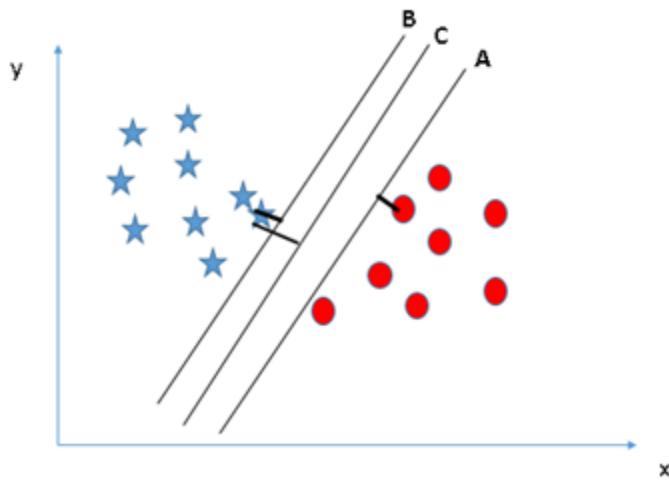
You need to remember a thumb rule to identify the right hyper-plane: "Select the hyper-plane which segregates the two classes better". In this scenario, hyper-plane "B" has excellently performed this job.

- **Identify the right hyper-plane (Scenario-2):** Here, we have three hyper-planes (A, B, and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?



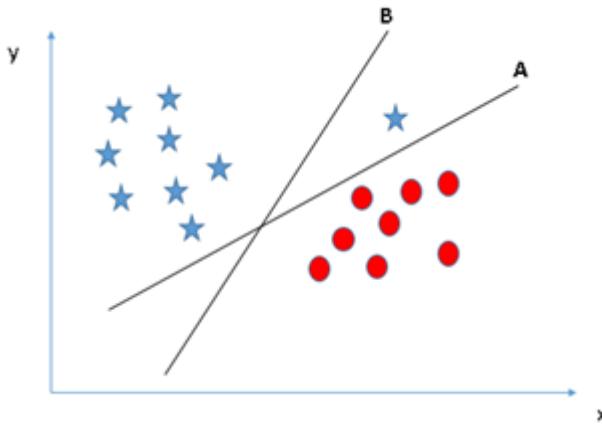
Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**.

Let's look at the below snapshot:



Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

- **Identify the right hyper-plane (Scenario-3):** Hint: Use the rules as discussed in previous section to identify the right hyper-plane

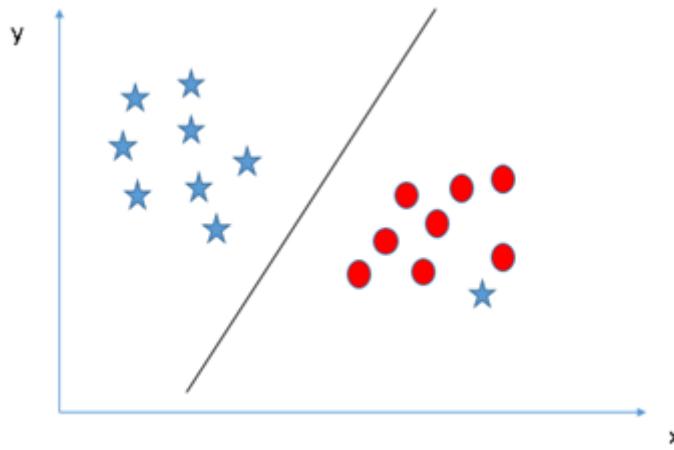


Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A**. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A**.

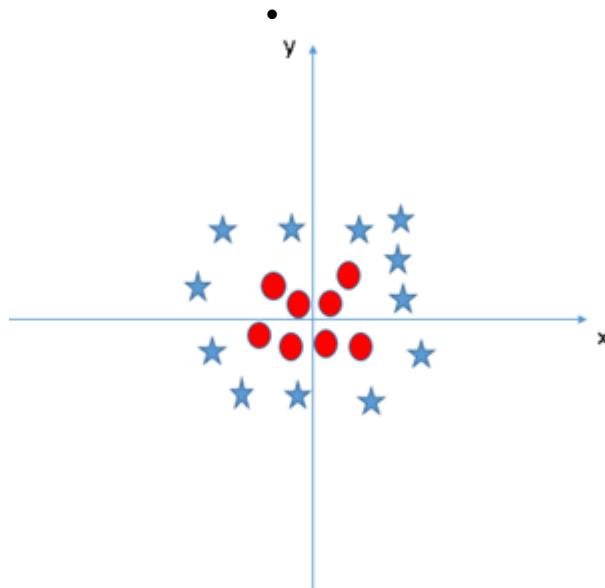
- **Can we classify two classes (Scenario-4)?:** Below, I am unable to segregate the two classes using a straight line, as one of the stars lies in the territory of other(circle) class as an outlier.



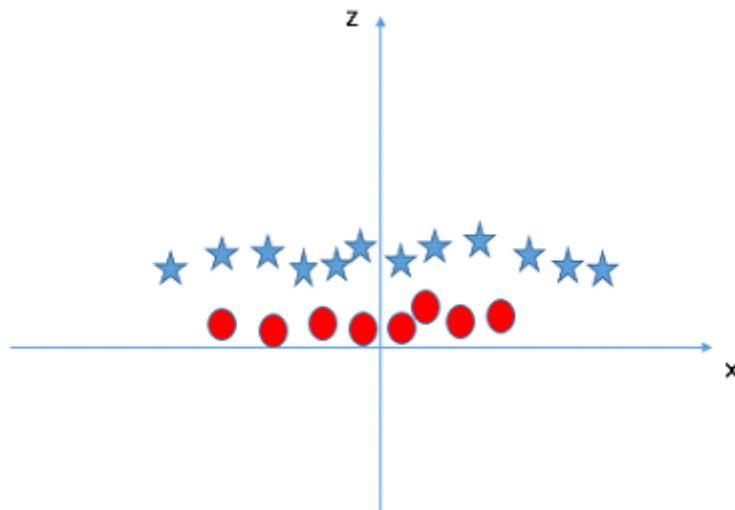
As I have already mentioned, one star at other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.



- **Find the hyper-plane to segregate to classes (Scenario-5):** In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.



- SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature $z=x^2+y^2$. Now, let's plot the data points on axis x and z:

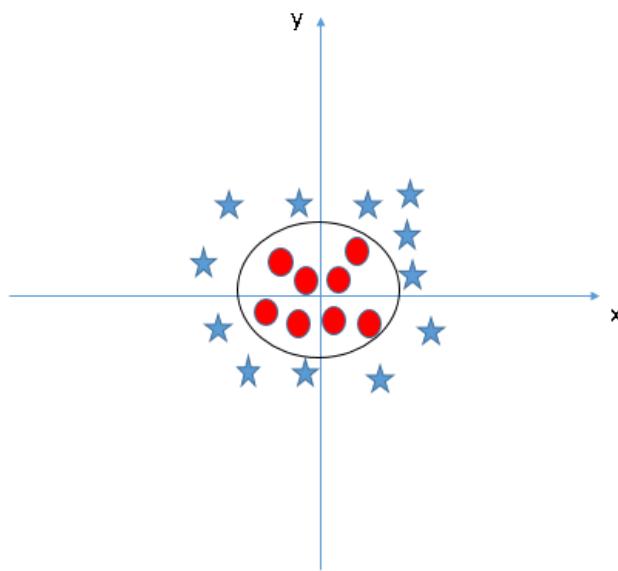


In above plot, points to consider are:

- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, the SVM algorithm has a technique called the [kernel trick](#). The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts not separable problem to separable problem. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you've defined.

When we look at the hyper-plane in original input space it looks like a circle:



Örnek:

İki sınıf arasında dairesel bir halka boşluğu bulunmaktadır. Halkanın iç yarı çapı 5cm, dış yarı çapı 10cm dir. Bir nokta halkanın içinde ve dış yarı çapına 1cm mesafede ise hangi yorum doğrudur?

Bu değer halkanın dışına yakın olan sınıfa aittir.

7.2.4. Naive Bayes

Bayes Ağı, bir dizi değişken arasındaki ilişkilerin olasılıklarını öğrenerek çıkarmış yapan bir makine öğrenmesi algoritmasıdır. Düşünce ya da oluşan kanı veya yönlendirilmiş olasılıklı bir modeldir. Örneğin, bir Bayes ağı, hastalıklar ve semptomlar arasındaki olasılık ilişkilerini temsil edebilir. Belirtiler verildiğinde, ağ çeşitli hastalıkların varlığının olasılıklarını hesaplamak için kullanılabilir.

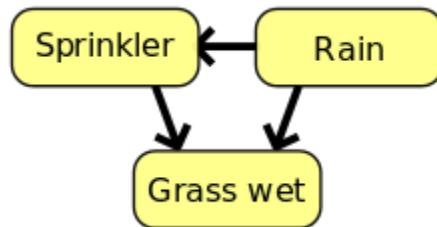
Konuşma sinyalleri veya protein dizileri gibi değişken dizilerini modelleyen Bayes ağlarına dinamik Bayes şebekeleri denir. Belirsizlik altında karar problemlerini temsil edebilen ve çözebilen Bayes ağlarının genellemelerine etki diyagramları denir.

Naïve Bayes'te sınıflandırıcı, lojistik regresyon gibi ayırt edici modellere göre daha hızlı birleşir, bu nedenle daha az eğitim verisine ihtiyacınız vardır. Ana avantajı, özellikler arasındaki etkileşimleri öğrenememesidir.

Bayesci mantık programı iki bileşenden oluşur. İlk bileşen mantıklı bir bileşendir; alanın niteliksel yapısını yakalayan bir dizi Bayes Cümlelerinden oluşur. İkinci bileşen nicelikseldir, alanla ilgili nicel bilgileri kodlar.

Yağmur, yağmurlama sisteminin etkinleştirilip etkinleştirilmeyeceğini etkiler ve hem yağmur hem de yağmurlama sistemi çimlerin ıslak olup olmadığını etkiler. Çim ıslak ise yağmu mu yağdı yoksa yağmurlama sistemi mi çalıştı.

Örneğin, Cebrail pikniğe gitmiş olsun, bu durumda Cebrail'in olduğu yer de yağmur yağmuyor.



Örnek:

Test altındaki sistemin giriş sayısı 5 çıkış sayısı 1 dir. Girişler ikili sayı sisteminde bit olarak uygulanmaktadır. Toplam test sayısı 20 cihaz ise, testten geçen cihaz sayısı 18 ise testten kalma olasılığı nedir?

Örnek:

Gün	Görünüş	Sıcaklık	Nem	Rüzgar	Play
1	Güneşli	Sıcak	Yüksek	Zayıf	Hayır
2	Güneşli	Sıcak	Yüksek	Kuvvetli	Hayır
3	Bulutlu	Sıcak	Yüksek	Zayıf	Evet
4	Yağmurlu	Hafif	Yüksek	Zayıf	Evet
5	Yağmurlu	Soğuk	Normal	Zayıf	Evet
6	Yağmurlu	Soğuk	Normal	Kuvvetli	Hayır
7	Bulutlu	Soğuk	Normal	Kuvvetli	Evet
8	Güneşli	Hafif	Yüksek	Zayıf	hayır
9	Güneşli	Soğuk	Normal	Zayıf	Evet
10	Yağmurlu	Hafif	Normal	Zayıf	Evet
11	Güneşli	Hafif	Normal	Kuvvetli	Evet
12	Bulutlu	Hafif	Yüksek	Kuvvetli	Evet
13	Bulutlu	Sıcak	Normal	Zayıf	Evet
14	Yağmurlu	Hafif	Yüksek	Kuvvetli	hayır
	Güneşli	Soğuk	Yüksek	Kuvvetli	?

Toplam Gün Sayısı=14

Evet=9

Hayır=5

$P(\text{Evet})=9/14$

$P(\text{Hayır})=5/14$

Havanın görünüş durumuna göre,

Havanın görünüşü güneşli olduğunda 2 gün oyun var. $P(H_{\text{Güneşli}} | \text{Evet})=2/9$

Havanın görünüşü güneşli olduğunda 3 gün oyun yok. $P(H_{\text{Güneşli}} | \text{Hayır})=3/5$

Toplam güneşli gün sayısı=2+3=5

Havanın görünüşü bulutlu olduğunda 4 gün oyun var. $P(H_{\text{Bulutlu}} | \text{Evet})=4/9$

Havanın görünüşü bulutlu olduğunda 0 gün oyun yok. $P(H_{\text{Bulutlu}} | \text{Hayır})=0/5$

Toplam bulutlu gün sayısı=4+0=4

Havanın görünüşü yağmurlu olduğunda 3 gün oyun var. $P(H_{\text{Yağmurlu}} | \text{Evet})=3/9$

Havanın görünüşü yağmurlu olduğunda 2 gün oyun yok. $P(H_{\text{Yağmurlu}} | \text{Hayır})=2/5$

Toplam yağmurlu gün sayısı=2+3=5

Toplam gün sayısı=5+4+5=14

Sıcaklık durumuna göre,

Sıcaklığın sıcak olduğunda 2 gün oyun var. $P(S_{\text{Sıcak}} | \text{Evet})=2/9$

Sıcaklığın sıcak olduğunda 2 gün oyun yok. $P(S_{\text{Sıcak}} | \text{Hayır})=2/5$

Toplam sıcak gün sayısı=2+2=4

Sıcaklığın hafif olduğunda 4 gün oyun var. $P(S_{\text{Hafif}} | \text{Evet})=4/9$

Sıcaklığın hafif olduğunda 2 gün oyun yok. $P(S_{\text{Hafif}} | \text{Hayır})=2/5$

Toplam hafif gün sayısı=4+2=6

Sıcaklığın soğuk olduğunda 3 gün oyun var. $P(S_{\text{Soğuk}} | \text{Evet})=3/9$

Sıcaklığın soğuk olduğunda 1 gün oyun yok. $P(S_{\text{Soğuk}} | \text{Hayır})=1/5$

Toplam soğuk gün sayısı=3+1=4

Toplam gün sayısı=4+6+4=14

Rüzgar durumuna göre,

Rüzgar zayıf olduğunda 6 gün oyun var. $P(R_{\text{Zayıf}} | \text{Evet})=6/9=2/3$

Rüzgar zayıf olduğunda 2 gün oyun yok. $P(R_{\text{Zayıf}} | \text{Hayır})=2/5$

Toplam rüzgar zayıf gün sayısı=6+2=8

Rüzgar Kuvvetli olduğunda 3 gün oyun var. $P(R_{\text{Kuvvetli}} | \text{Evet})=3/9=1/3$

Rüzgar Kuvvetli olduğunda 3 gün oyun yok. $P(R_{\text{Kuvvetli}} | \text{Hayır})=3/5$

Toplam rüzgar kuvvetli gün sayısı=3+3=6

Toplam gün sayısı=8+6=14

Nem durumuna göre,

Nem yüksek olduğunda 3 gün oyun var. $P(N_{\text{Yüksek}} | \text{Evet})=3/9$

Nem yüksek olduğunda 4 gün oyun yok. $P(N_{\text{Yüksek}} | \text{Hayır})=4/5$

Toplam nem yüksek gün sayısı=3+4=7

Nem normal olduğunda 6 gün oyun var. $P(N_{\text{Normal}} | \text{Evet})=6/9$

Nem normal olduğunda 1 gün oyun yok. $P(N_{\text{Normal}} | \text{Hayır})=1/5$

Toplam nem zayıf gün sayısı=6+1=7

Toplam gün sayısı=7+7=14

$X=\{\text{Güneşli}, \text{Soğuk}, \text{Yüksek}, \text{Kuvvetli}\}$ ise

$$P(X | \text{Evet}) = P(\text{Evet}) * P(H_{\text{Güneşli}} | \text{Evet}) * P(S_{\text{Soğuk}} | \text{Evet}) * P(R_{\text{Kuvvetli}} | \text{Evet}) * P(N_{\text{Yüksek}} | \text{Evet})$$

$$P(X \mid \text{Evet}) = (9/14) * (2/9) * (1/9) * (3/9) = (1/7) * (1/27) = 1/189 = 0.053$$

$$P(X \mid \text{Hayır}) = P(\text{Hayır}) * P(H\text{-}\text{Güneşli} \mid \text{Hayır}) * P(S\text{-}\text{Soğuk} \mid \text{Hayır}) * P(R\text{-}\text{Kuvvetli} \mid \text{Hayır}) * P(N\text{-}\text{Yüksek} \mid \text{Hayır})$$

$$P(X \mid \text{Hayır}) = (5/14) * (3/5) * (1/5) * (3/5) * (4/5) = (3/14)(12/125) = 18/875 = 0.0206$$

7.3. Sıra Desen Madenciliği

Sıra desen madenciliği, belirli bir sırayla gelen istatiksel olarak veri örnekleri arasındaki ilgili örüntüleri bulmaya çalışır. Müşterilerin teknoloji market alışverişini verilerine göre son 3 ayda sırasıyla önce bilgisayar sonra CD-ROM son olarak dijital kamera satın almaları, tıbbi tedaviler, doğal felaketler(deprem), DNA dizilişi ve gen yapısı sıralı örüntü madenciliği ile ilgilidir. Mesela Internet şubesinde yapılan işlemler bir sıraya göre yapıldığı için sıralı örüntü madenciliği içerisinde yer almaktadır. sıralı örüntü madenciliğinde arka arkaya yapılan işlemler göz önüne alınır.

PrefixSpan:

PrefixSpan (Prefix-projected Sequential pattern mining) algoritması, sıralı veri madenciliği için çok iyi bilinen bir algoritmadır. Ardışık desenleri, desen büyütme yöntemi ile çıkarır. Algoritma, küçük veri kümeleri için çok iyi performans gösteriyor.

PrefixSpan sık tekrarlanan işleri tekrar eder.

Örnek vererek anlatmak gerekiyse bir sürecin belirli zaman dilimlerinde nasıl tekrarlandığını inceler. Bu tekrarın hangi işlemler öncesi veya sonrası olduğu dikkate alınmaktadır. Algoritma yönelik manipülasyon bulmada kritik öneme sahiptir.

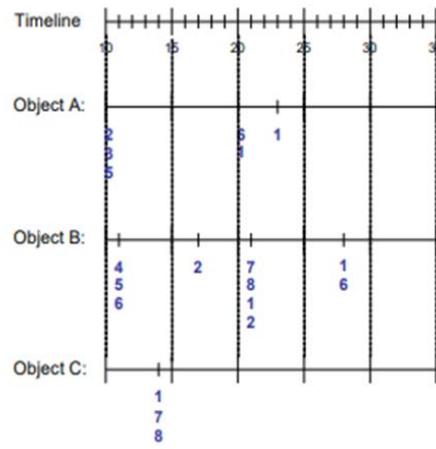
Süreç analizinde gelecekte oluşacak krizlerin izlerini aramada önem katmaktadır.

Örnek:

Aşağıdaki tabloda belli eylemler ve bu eylemlerin zaman dilimlerine göre gerçekleşme akışları verilmektedir. Veri madenciliğinin önemli adımlarından biri olan veri etiketleme süreci bu algoritmanın akışında rölyef oynamaktadır. Burada A – B – C eylemlerini bir internet bankacılığı süreçlerindeki havale(A), ödeme(B), giriş işlemleri(C) gibi etiketleme yaptıktan sonra süreci bir ID tanımlayarak sayısalştırmaktadırlar. Bu sayede algoritmanın temelinde yatan örüntü yakalama amacında hızlandırmış olmaktadır. Burada genel yaklaşımı anladıkten sonra, gerçek bir örnek üzerinde ilerleyebiliriz.

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7

Aşağıdaki şekil de görüldüğü gibi sıralı örüntü madenciliği yöntemine göre müşterilerin herhangi bir t anında sırayla yaptığı işlemleri, satın aldığı ürünleri, ziyaret ettiği web sayfalarını analiz ederek müşterinin davranışına uygun çıkarımlar yapılabilir. Bir sıralı veri analizi uygulamasının başarılı olması için destek (support) değerinin en küçük destek değerinden (minsupport) başka ifade ile eşik değerinden daha büyük olması gereklidir.



Şekil 3. Sıralı verilerin zaman diliminde gösterimi

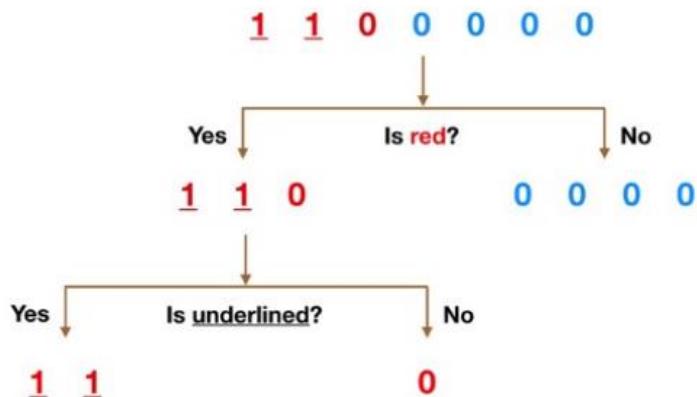
7.3.1. Rassal Orman Modeli Algoritması

Rastgele ormanlar, denetimli bir öğrenme algoritmasıdır. Hem sınıflandırma hem de regresyon için kullanılabilir.

Karar ağaçları:

Rastgele orman modelinin yapı taşları oldukları için karar ağaçlarının bilinmesi gerekmektedir. Oldukça sezgisel yaklaşımalar içerir. Çoğu insanın hayatlarının bir noktasında bilerek ya da bilmeyerek bir karar ağıacı kullandığına bahse girerim.

Bir karar ağaçının nasıl çalıştığını bir örnek üzerinden anlamak muhtemelen çok daha kolaydır.



Veri setimizin soldaki şemlin üstündeki sayılarından oluştuğunu hayal edin. İki 1 ve beş 0'ımız var (1'ler ve 0'lar sınıflarımızdır) ve özelliklerini kullanarak sınıfları ayırmak istiyoruz. Özellikler renklidir (kirmızıya karşı mavi) ve gözlemin altı çizili olup olmadığıdır. Peki bunu nasıl yapabiliz?

Renk, 0'lardan biri hariç tümü mavi olduğu için, ayrılması oldukça bariz bir özellik gibi görünüyor. Böylece "Kırmızı mı?" Sorusunu kullanabiliyoruz. ilk düğümümüzü ayırmak için. Bir ağaçtaki bir düğümü, yolun ikiye ayrıldığı nokta olarak düşünübilirsiniz - kriterleri karşılayan gözlemler Evet dalına ve Hayır dalına inmeyenler.

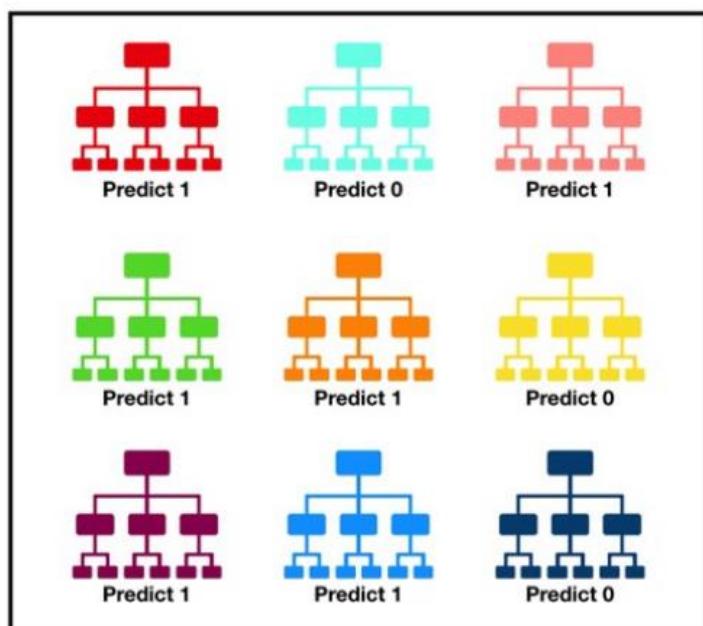
Hayır dalı (blues) artık 0'lardır, bu yüzden orada işimiz bitti, ancak Evet şubemiz yine de bölünebilir. Şimdi ikinci özelliği kullanıp "Altı çizili mi?" Diye sorabiliyoruz. ikinci bir bölme yapmak için.

Altı çizili iki 1, Evet alt dalına gider ve altı çizilmemiş 0, sağ alt daldan aşağı gider ve hepimiz işimiz biter. Karar ağacımız, verileri mükemmel bir şekilde bölmek için iki özelliği kullanabildi.

Açıkçası gerçek hayatı verilerimiz bu kadar net olmayacak, ancak bir karar ağacının kullandığı mantık aynı kalacak. Her düğümde sorulacak - Hangi özellik, eldeki gözlemleri, ortaya çıkan grupların olabildiğince farklı olacağı şekilde (ve ortaya çıkan her alt grubun üyeleri mümkün olduğunda birbirine benzeyecek şekilde) bölmeme izin verir?

Rastgele Orman Sınıflandırıcıları:

Rastgele orman, adından da anlaşılacağı gibi, bir topluluk olarak çalışan çok sayıda bireysel karar ağacından oluşur. Rastgele ormanın her bir ağaç bir sınıf tahmini verir ve en çok oyu alan sınıf, modelimizin öngörüsü haline gelir (aşağıdaki şekle bakın).



Tally: Six 1s and Three 0s

Rastgele ormanın ardından temel kavram basit ama güçlü bir kavramdır - kalabalıkların bilgeliği. Veri biliminde konuşursak, rastgele orman modelinin bu kadar iyi çalışmasının nedeni şudur: Bir komite olarak faaliyet gösteren çok sayıda görece ilişkisiz model (ağaç), münferit kurucu modellerin herhangi birinden daha iyi performans gösterecektir. Modeller arasındaki düşük korelasyon anahtardır. Tıpkı düşük korelasyonlu yatırımların (hisse senetleri ve tahviller gibi) bir araya gelerek parçalarının toplamından daha büyük bir portföy oluşturması gibi, ilişkisiz modeller, bireysel tahminlerin herhangi birinden daha doğru olan topluluk tahminleri üretебilir. Bu harika etkinin nedeni, ağaçların birbirlerini kendi hatalarından korumalarıdır (sürekli aynı yönde hata yapmadıkları sürece). Bazı ağaçlar yanlış olabilirken, diğer birçok ağaç haklı olacaktır, bu nedenle bir grup olarak ağaçlar doğru yönde hareket edebilecektir. Bu nedenle, rastgele ormanın iyi performans göstermesi için ön koşullar şunlardır: Özelliklerimizde bazı gerçek sinyaller olması gereklidir, böylece bu özellikler kullanılarak oluşturulan modeller rastgele tahmin etmekten daha iyi sonuç verir. Tek tek

ağaçların yaptığı tahminlerin (ve dolayısıyla hataların) birbirleriyle düşük korelasyonlara sahip olması gerekir.

İliksiz sonuçların neden bu kadar büyük olduğunu dair bir örnek:

İliksiz birçok modele sahip olmanın harika etkileri o kadar kritik bir kavramdır ki, şu oyunu oynadığımızı hayal edin: Bir sayı üretmek için tekdeze dağıtılmış rasgele sayı üretici kullanıyorum. Oluşturduğum sayı 40'tan büyük veya ona eşitse, kazanırsınız (yani% 60 zafer şansınız olur) ve size biraz para öderim. 40'in altındaysa ben kazanırım ve siz bana aynı miktarı ödersin. Şimdi size aşağıdaki seçenekleri sunuyorum. Şunlardan birini yapabiliriz:

Oyun 1 - 100 kez oynayın, her seferinde 1 \$ bahis yapın.

Oyun 2 - 10 kez oynayın, her seferinde 10 \$ bahis yapın.

Oyun 3 - bir kez oynayarak 100 \$ bahis yapın.

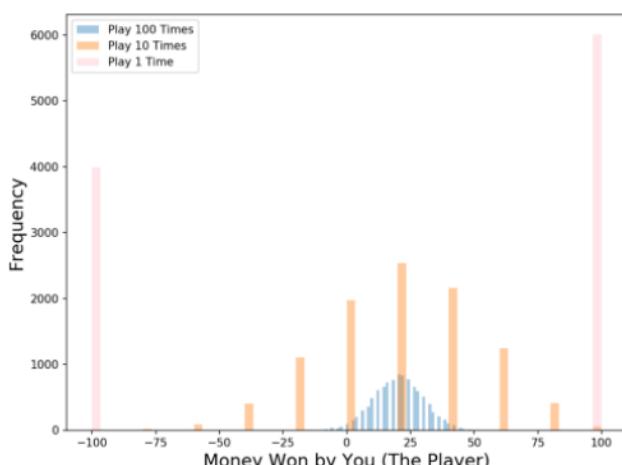
Hangisini seçerdin? Her oyunun beklenen değeri aynıdır:

$$\text{Beklenen Değer Oyunu 1} = (0.60 * 1 + 0.40 * -1) * 100 = 20$$

$$\text{Beklenen Değer Oyunu 2} = (0.60 * 10 + 0.40 * -10) * 10 = 20$$

$$\text{Beklenen Değer Oyunu 3} = 0.60 * 100 + 0.40 * -100 = 20$$

Dağılımlar ne olacak? Sonuçları bir Monte Carlo simülasyonu ile görselleştirelim (her oyun türü için 10.000 simülasyon çalıştıracağız; örneğin, 1. Oyundaki 100 oyunun 10.000 katını simüle edeceğiz). Soldaki tabloya bir göz atın - şimdi hangi oyunu seçerdiniz? Beklenen değerler aynı olsa bile, sonuç dağılımları, pozitif ve dardan (mavi) ikiliye (pembe) doğru büyük ölçüde farklıdır.



Oyun 1 (100 kez oynadığımız yer), biraz para kazanmak için en iyi şansı sunuyor - yürüttüğüm 10.000 simülasyondan% 97'sinde para kazanıyorsunuz! Oyun 2'de (10 kez oynadığımız yerde) simülasyonların% 63'ünde para kazanırsınız, ciddi bir düşüş (ve para

kaybetme olasılığınızda ciddi bir artış). Ve sadece bir kez oynadığımız 3. Oyun, beklendiği gibi simülasyonların% 60'ında para kazanıyorsunuz.

Dolayısıyla, oyunlar aynı beklenen değeri paylaşısa da, sonuç dağılımları tamamen farklıdır. 100 \$ 'lık bahsimizi farklı oyunlara ne kadar çok bölersek, para kazanacağımıza o kadar güvenebiliriz. Daha önce de belirtildiği gibi, bu işe yarar çünkü her oyun diğerlerinden bağımsızdır.

Rastgele orman aynıdır - her ağaç, önceki oyuncumuzdaki bir oyun gibidir. Daha fazla oynadığımızda para kazanma şansımızın nasıl arttığını gördük. Benzer şekilde, rastgele bir orman modeliyle, modelimizdeki ilişkisiz ağaçların sayısı ile doğru tahminler yapma şansımız artar.

Modellerin birbirini çeşitlendirmesini sağlamak:

Öyleyse rastgele orman, her bir ağaçın davranışının modeldeki diğer ağaçlardan herhangi birinin davranışıyla çok fazla ilişkili olmamasını nasıl sağlar?

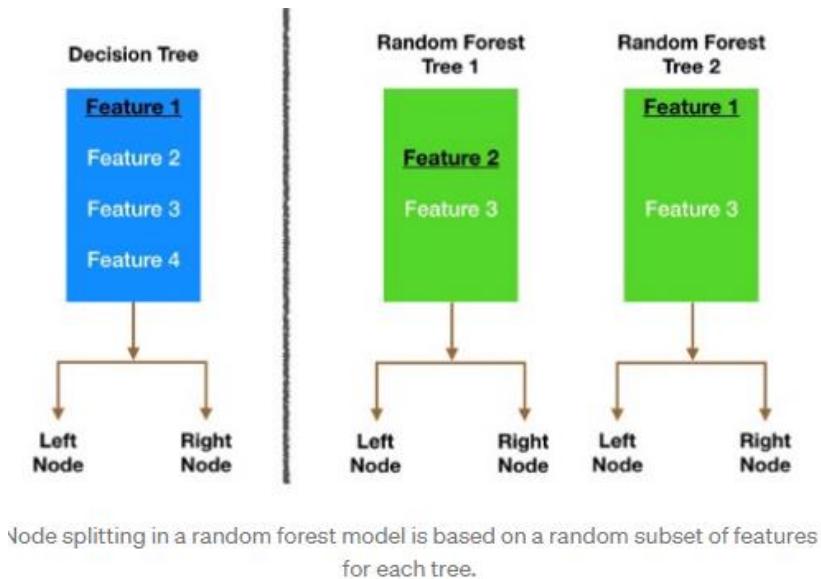
Aşağıdaki iki yöntemi kullanır:

Torbalama (Bootstrap Aggregation) - Karar ağaçları, eğitildikleri verilere karşı çok hassastır - eğitim setinde yapılan küçük değişiklikler, önemli ölçüde farklı ağaç yapılarına neden olabilir. Rastgele orman, her bir ağaçın veri kümesinden değiştirilerek rasgele örneklemesine izin vererek bundan yararlanır ve farklı ağaçlarla sonuçlanır. Bu işlem torbalama olarak bilinir.

Torbalama ile eğitim verilerini daha küçük parçalara ayırmadığımıza ve her ağaç farklı bir yığın üzerinde eğitmediğimize dikkat edin. Bunun yerine, N büyüklüğünde bir örneğimiz varsa, yine de her ağaçta N boyutunda bir eğitim seti besliyoruz (aksi belirtildiğince). Ancak orijinal eğitim verileri yerine, değiştirilmiş N boyutunda rastgele bir örnek alıyoruz. Örneğin, eğitim verilerimiz [1, 2, 3, 4, 5, 6] ise, ağaçlarımızdan birine aşağıdaki listeyi verebiliriz [1, 2, 2, 3, 6, 6]. Her iki listenin de altı uzunluğunda olduğuna ve "2" ile "6" nin ikisinin de ağaçımıza verdığımız rastgele seçilmiş eğitim verilerinde tekrarlandığına dikkat edin (çünkü değiştirme ile örneklemeye yapıyoruz).

Özellik Rastgeleliği:

Normal bir karar ağacında, bir düğümü bölme zamanı geldiğinde, mümkün olan her özelliğin göz önünde bulundururuz ve sağ düğümdekilerle sol düğümdeki gözlemler arasında en fazla ayrimı yaratmayı seçeriz. Bunun aksine, rastgele bir ormandaki her ağaç yalnızca rastgele bir özellik alt kümesinden seçim yapabilir. Bu, modeldeki ağaçlar arasında daha fazla çeşitliliği zorlar ve sonuçta ağaçlarda daha düşük korelasyon ve daha fazla çeşitlilik ile sonuçlanır.



Görsel bir örnek üzerinden geçelim - yukarıdaki resimde, geleneksel karar ağacı (mavi), düğümü nasıl böleceğine karar verirken dört özelliğin tümünden seçim yapabilir. Verileri olabildiğince ayrılmış gruplara böldüğü için Özellik 1 (siyah ve altı çizili) ile devam etmeye karar verir. Şimdi rastgele ormanımıza bir göz atalım. Bu örnekte ormanın iki ağacını inceleyeceğiz. Rastgele Orman Ağacı 1'i kontrol ettiğimizde, düğüm bölme kararı yalnızca Özellik 2 ve 3'ü (rastgele seçilir) dikkate alabileceğiğini görürüz. Geleneksel karar ağacımızdan (mavi olarak) Özelliğin bölme için en iyi özellik olduğunu biliyoruz, ancak Ağaç 1, Özellik 1'i göremediğinden, Özellik 2'ye (siyah ve altı çizili) gitmek zorunda kalır. Öte yandan Ağaç 2, yalnızca Özellik 1 ve 3'ü görebilir, bu nedenle Özellik 1'i seçebilir. Dolayısıyla rastgele ormanımızda, yalnızca farklı veri kümeleri üzerinde eğitilen (torbalama sayesinde) değil, aynı zamanda karar vermek için farklı özellikler kullanan ağaçlarla karşılaşıyoruz. Ve bu, sevgili okuyucum, birbirlerini tamponlayan ve hatalarından koruyan ilişkisiz ağaçlar yaratır.

Rastgele orman, birçok karar ağacından oluşan bir sınıflandırma algoritmasıdır. Komite tarafından tahmin edilmesi herhangi bir ağaçtan daha doğru olan ilişkisiz bir ağaç ormanı yaratmaya çalışmak için her bir ağaç inşa ederken torbalama kullanır ve rastgelelik özelliğini kullanır. Rastgele ormanınızın doğru sınıf tahminleri yapabilmesi için neye ihtiyacımız var? En azından bir miktar tahmin gücüne sahip özelliklere ihtiyacımız var. Sonuçta, eğer içine çöp koyarsak, o zaman çöpü dışarı atarız. Ormanın ağaçları ve daha da önemlisi tahminlerinin ilintisiz olması (veya en azından birbirleriyle düşük korelasyonlara sahip olması) gereklidir. Algoritmanın kendisi özellik rastgeleliği aracılığıyla bu düşük korelasyonları bizim için tasarlamaya çalışırken, seçtiğimiz özellikler ve seçtiğimiz hiper parametreler nihai korelasyonları da etkileyecektir.

8. Derin Öğrenme Algoritmaları

Derin öğrenme (aynı zamanda derin yapılandırılmış öğrenme, hiyerarşik öğrenme ya da derin makine öğrenmesi) bir veya daha fazla gizli katman içeren yapay sinir ağları ve benzeri makine öğrenme algoritmalarını kapsayan çalışma alanıdır. Yani en az bir adet yapay sinir ağının (YSA) kullanıldığı ve birçok algoritma ile, bilgisayarın eldeki verilerden yeni veriler elde etmesidir. Derin öğrenme gözetimli, yarı gözetimli veya gözetimsiz olarak gerçekleştirilebilir. Derin yapay sinir ağları pekiştirmeli öğrenme yaklaşımıyla da başarılı sonuçlar vermiştir.

Derin öğrenme, Yüksek bilgi işleme gücü ve büyük veri kümeleriyle birlikte katmanlı yapay sinir ağlarının güçlü matematiksel modelleri oluşturabildiği bir makine öğrenimi alt kümesidir. **Verinin yapısına göre hangi parametrelere ne ağırlık verileceğini kendisi keşfetmektedir.** Derin öğrenme, ham girdiden daha yüksek seviyedeki özellikleri aşamalı olarak çıkarmak için birden çok katman kullanan bir makine öğrenme algoritmaları sınıfıdır.

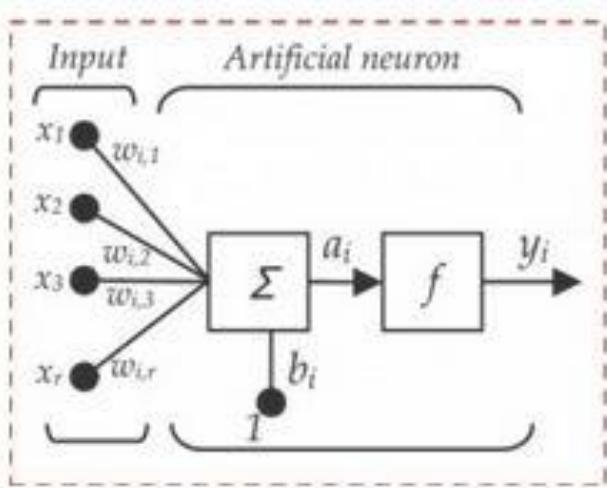
Modern derin öğrenme modellerinin çoğu, yapay sinir ağlarına, özellikle de Konvolüsyonel Sinir Ağlarına (CNN - Convolutional Neural Networks) dayanmaktadır, ancak aynı zamanda derin düşünce ağları ve derin düğümler gibi derin üretken modellerde katmansal olarak düzenlenmiş öneri formülleri veya gizli değişkenleri de içerebilirler. Yapay sinir ağları (YSA) biyolojik sistemlerde bilgi işleme ve dağıtılmış iletişim düğümlerinden esinlenmiştir. YSA'ların biyolojik beyinlerden çeşitli farklılıklarları vardır. Özellikle, sinir ağları statik ve sembolik olma eğilimindeyken, çoğu canlı organizmanın biyolojik beyni dinamik (plastik) ve analogdur.

Derin öğrenme, her seviye girdi verilerini biraz daha soyut ve bileşik bir temsile dönüştürmeyi öğrenir. Bir görüntü tanıma uygulamasında, ham girdi bir piksel matrisi olabilir; birinci temsili katman pikselleri soyutlayabilir ve kenarları kodlayabilir; ikinci katman kenar düzenlemelerini oluşturabilir ve kodlayabilir; üçüncü katman bir burnu ve gözleri kodlayabilir; ve dördüncü katman görüntünün bir yüz içerdığını tanıyalabilir. Daha da önemlisi, derin bir öğrenme süreci hangi özelliklerin hangi seviyeye en uygun şekilde yerleştirileceğini öğrenebilir. Elbette bu, elle ayarlama ihtiyacını tamamen ortadan kaldırır; örneğin, değişen sayıda katman ve katman boyutu farklı derecelerde soyutlama sağlayabilir.

Derin öğrenme, Yapay Zekada (AI) Makine Öğreniminin bir alt kümesidir. Bir yapay zeka, verileri işlemek, kalıplar oluşturmak ve kararlar almak için insan beyİNini taklit ettiğinde, bu Derin Öğrenmedir. Ham girdiden aşamalı olarak üst düzey özellikleri çıkarmak için birden çok katman kullanır. "Derin öğrenmedeki" "derin" kelimesi, verilerin dönüştürüldüğü katman sayısını ifade eder. Bu sayede yapılandırılmamış ve etiketsiz verilerden örüntüler öğrenmek mümkündür. Derin öğrenme algoritması, deneyimlerden nasıl öğrendiğimize benzer şekilde, her seferinde sonucunu iyileştiren bir görevi tekrar gerçekleştirir.

Arama motorlarında, sosyal medyada, e-ticaret platformlarında her saniye yaratılan muazzam miktarda veri, derin öğrenmeyi büyük bir potansiyele dönüştürdü. Buna ek olarak, bugün mevcut olan güçlü bilgi işlem gücü, derin öğrenmede kullanılan algoritmalar üzerinde büyük bir etki yarattı. Dahası, kendi kendine giden arabalar, AlphaGo, sesli asistanlar gibi yapay zekadaki atılımlar, derin öğrenme sayesinde mümkün.

Derin öğrenmede, her seviye girdi verilerini biraz daha soyut ve bileşik bir temsile dönüştürmeyi öğrenir. Bir görüntü tanımı uygulamasında, ham girdi bir piksel matrisi olabilir; birinci temsil katmanı pikselleri soyutlayabilir ve kenarları kodlayabilir; ikinci katman, kenar düzenlemelerini oluşturabilir ve kodlayabilir; üçüncü katman bir burnu ve gözleri kodlayabilir; ve dördüncü katman, görüntünün bir yüz içerdigini fark edebilir. Daha da önemlisi, derin öğrenme süreci hangi özelliklerin hangi seviyeye en uygun şekilde yerleştirileceğini kendi başına öğrenebilir. (Elbette bu, elle ayarlama ihtiyacını tamamen ortadan kaldırır; örneğin, değişen sayıda katman ve katman boyutu, farklı soyutlama dereceleri sağlayabilir.)



Nöronun farklı bileşenleri şu şekilde belirtilir:

- x_1, x_2, \dots, x_N : Bunlar nöronun girdileridir. Bunlar, giriş katmanındaki gerçek gözlemler veya gizli katmanlardan birinin ara değeri olabilir.
- w_1, w_2, \dots, w_N : Her girişin ağırlığı.
- b_i : Eğilim ya da Sapma birimleri olarak adlandırılır. Bunlar, her ağırlığa karşılık gelen aktivasyon fonksiyonunun girişine eklenen sabit değerlerdir. Kesme terimine benzer şekilde çalışır.
- a : Şu şekilde temsil edilebilen nöronun aktivasyonu olarak adlandırılır
- ve y : nöronun çıktısıdır

$$a = f(\sum_{i=0}^N w_i x_i)$$

Derin Öğrenme Algoritmaları:

Gözetimli öğrenme

Kümeleme

Boyut indirgeme

Yapılandırılmış tahmin

Anomali tespiti

Sınır ağları

Pekiştirmeli öğrenme

8.1. Anomali Tespiti:

Veri analizinde, anomali tespiti (aynı zamanda aykırı değer tespiti), verilerin çoğunuğundan önemli ölçüde farklılaşarak şüphe uyandıran nadir öğelerin, olayların veya gözlemlerin tanımlanmasıdır. Tipik olarak anormal öğeler, banka dolandırıcılığı, yapısal bir kusur, tıbbi sorunlar veya bir metindeki hatalar gibi bir tür soruna dönüşecektir. Anormallikler ayrıca aykırı değerler, yenilikler, gürültü, sapmalar ve istisnalar olarak da adlandırılmaktadır.

Özellikle, kötüye kullanım ve ağa izinsiz giriş tespiti bağlamında, ilginç nesneler genellikle nadir nesneler değil, beklenmedik etkinlik patlamalarıdır. Bu model, bir aykırı değerin nadir bir nesne olarak genel istatistiksel tanımına uymaz ve uygun şekilde bir araya getirilmediği sürece birçok aykırı değer algılama yöntemi (özellikle denetimsiz yöntemler) bu tür verilerde başarısız olmaktadır. Bunun yerine, bir küme analizi algoritması, bu modellerin oluşturduğu mikro kümeleri tespit edebilmektedir.

Üç geniş anomali tespit tekniği kategorisi mevcuttur[4]. Denetimsiz anomali tespit teknikleri, veri setindeki örneklerin çoğunuğunun normal olduğu varsayımlı altında, veri setinin geri kalanına en az uyan örnekleri arayarak etiketlenmemiş bir test veri setindeki anormallikleri tespit etmektedir. Denetimli anomali tespit teknikleri, "normal" ve "anormal" olarak etiketlenmiş bir veri seti gerektirir ve bir sınıflandırıcının eğitimini içermektedir (diğer birçok istatistiksel sınıflandırma probleminden temel fark, aykırı değer tespitinin doğal dengesiz doğasıdır). Yarı denetimli anomali tespit teknikleri, belirli bir normal eğitim veri setinden normal davranışını temsil eden bir model oluşturur ve ardından kullanılan model tarafından bir test örneğinin oluşturulma olasılığını test etmektedir.

8.2. Yapay Sinir Ağları

İnsan beyni:

İnsan beyninin nasıl çalıştığını daha yeni anlamaya başladık ; daha yolun başındayız. Akıllı makineler yapmanın bir yolu, insan beynini özellikle organizasyonel prensiplerini taklit etmeye çalışmaktadır.

Beyin son derece karmaşık, doğrusal olmayan ve paralel bir bilgisayardır ve yoğun şekilde bağlı 10^{11} nörondan oluşur (nöron başına $\sim 10^4$ bağlantı). Bir nöron, silikon mantık geçidine (10^{-9} sn) kıyasla çok daha yavaştır (10^{-3} sn), doğrudur, nöronlar arasındaki muazzam bağlantı, nispeten yavaş hızı oluşturur. Ancak, karmaşık algısal kararlara çok hızlı bir şekilde ulaşılır (birkaç yüz milisaniye içinde)

100 Adım kuralı: Bireysel nöronlar birkaç milisaniye içinde çalıştığından, hesaplamalar yaklaşık 100'den fazla seri adım içermez ve bir nörondan diğerine gönderilen bilgi çok küçüktür (birkaç bit)

Plastisite: Beynin nöral yapısının bir kısmı doğumda bulunurken, diğer kısımları çevreye uyum sağlamak için özellikle yaşamın erken dönemlerinde öğrenme yoluyla geliştirilir (yeni girdiler). Biyolojik Nöronlarda, farklı dallanma yapılarına sahip çeşitli farklı nöronlar (motor nöron, merkez üstü çevresel olmayan görsel hücreler...) mevcuttur. Ağın bağlantıları ve tek tek sinapsların güçleri ağıın işlevini oluşturur.

Beyindeki Biyolojik Nöronlar:

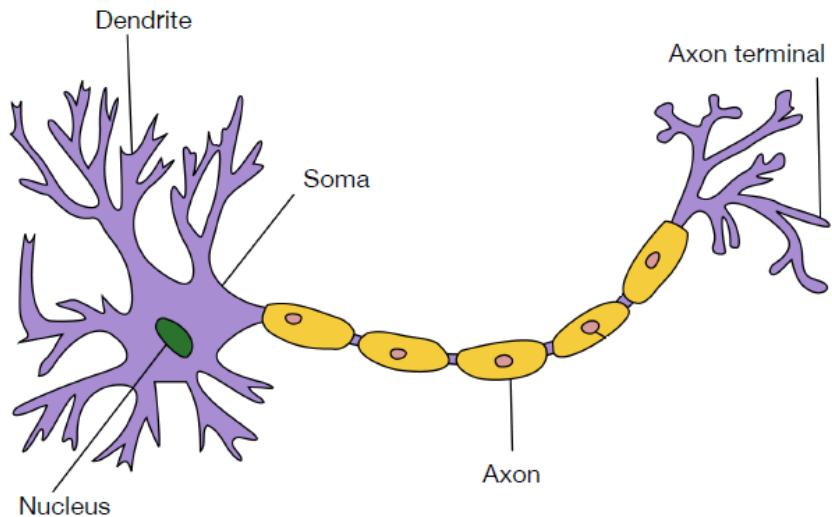
İnsan beyni yaklaşık 100 milyar nöronдан oluşur.

dendritler: hücrelere elektrik sinyalleri taşıyan sinir lifleri

hücre gövdesi: girdilerinin doğrusal olmayan bir işlevini hesaplar

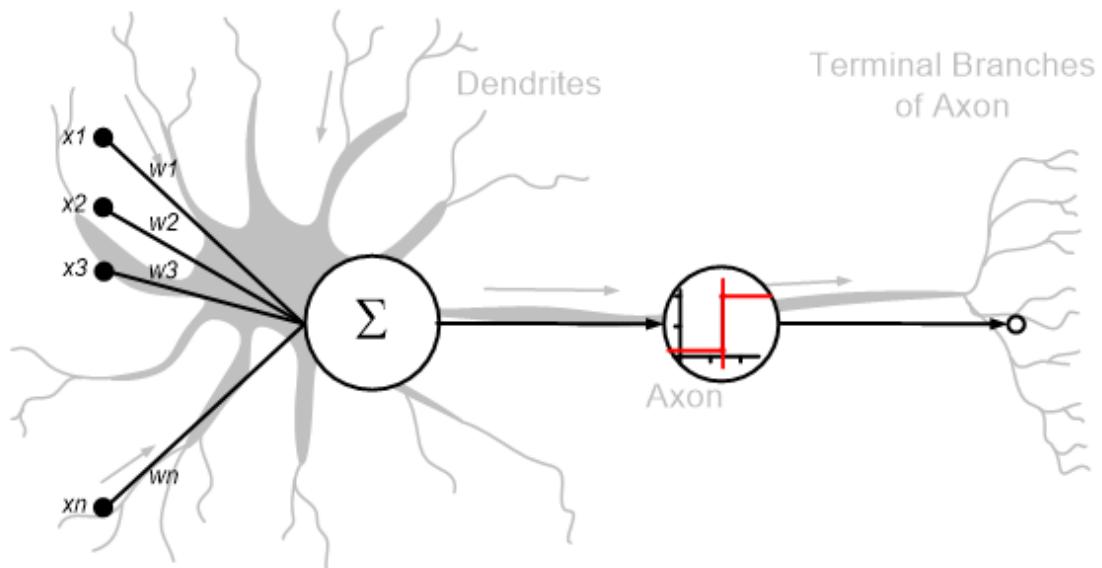
akson: elektrik sinyalini hücre gövdesinden diğer nöronlara taşıyan tek uzun lif. Nöronun aksonu, diğer birçok nöronun dendritlerine bağlıdır. Nöronlar, dendritlerden elektrik sinyalleri alır ve bunları aksona gönderir.

sinaps: gücü hücreye girişi etkileyen kimyasal bir bağlantıyi düzenleyen, bir hücrenin aksonu ile diğerinin dendriti arasındaki temas noktası.



İnsan beyninden ilham alan hesaplama model, Yapay Sinir Ağları:

- Basit işlem birimlerinden (nöronlar) oluşan büyük ölçüde paralel, dağıtılmış sistem
- Edinilen bilgiyi depolamak için nöronlar arasındaki sinaptik bağlantı güçleri kullanılır.
- Bilgi, ağ tarafından bir öğrenme süreci aracılığıyla çevresinden edinilir.



- Basit işlem birimlerinden (nöronlar) oluşan büyük ölçüde paralel, dağıtılmış sistem
- Edinilen bilgiyi depolamak için nöronlar arasındaki sinaptik bağlantı güçleri kullanılır.
- Bilgi, ağ tarafından bir öğrenme süreci aracılığıyla çevresinden edinilir.
- Örneklerden öğrenmek: etiketli veya etiketsiz
- Adaptivite: bir şeyler öğrenmek için bağlantı güçlerini değiştirmek
- Doğrusal olmama: doğrusal olmayan aktivasyon fonksiyonları gereklidir

- Hata toleransı: Nöronlardan veya bağlantılarından biri hasar görürse, tüm ağ hala oldukça iyi çalışıyor.
- Bu nedenle, aşağıdakilerle karakterize edilen sorunlar için klasik çözümlerden daha iyi alternatifler olabilirler: Yüksek boyutluluk, gürültülü, kesin olmayan veya eksik veriler; ve açıkça ifade edilmiş matematiksel bir çözüm veya algoritmanın olmaması

Beyin ve Yapay Sinir Ağları:

Benzerlikler:

- Nöronlar, nöronlar arasındaki bağlantılar
- Öğrenme = bağlantıların değişmesi, nöronların değişmesi değil
- Büyük paralel işleme

Ancak yapay sinir ağları çok daha basit:

- nöron içindeki hesaplama büyük ölçüde basitleştirildi
- ayrık zaman adımları
- tipik olarak çok sayıda uyarın içeren bir tür denetimli öğrenim

Bir sinir ağı, basit işlem birimlerinden (yapay nöronlar) oluşan büyük ölçüde paralel, dağıtılmış bir işlemcidir. Beyne iki açıdan benziyor:

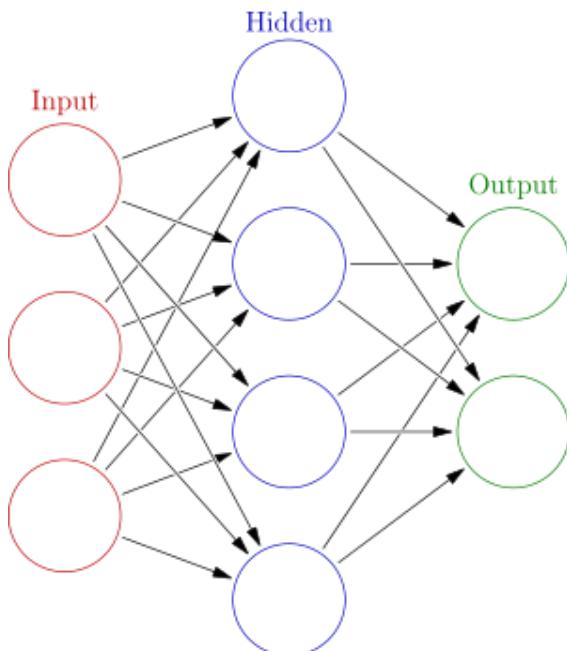
- Bilgi, ağ tarafından bir öğrenme süreci aracılığıyla çevresinden edinilir
- Edinilen bilgiyi depolamak için nöronlar arasındaki sinaptik bağlantı güçleri kullanılır.

Yapay sinir ağları (YSA) veya bağlantı sistemi, hayvan beyinlerini oluşturan biyolojik sinir ağlarından belirsiz bir şekilde esinlenen bilgisayar sistemleridir. Bu tür sistemler, genellikle herhangi bir görevde özgü kural ile programlanmadan, örnekleri dikkate alarak görevleri yerine getirmeyi "öğrenir".

YSA, biyolojik bir beyindeki nöronları gevşek bir şekilde modelleyen "yapay nöronlar" adı verilen bağlı birimler veya düğümler koleksiyonuna dayanan bir modeldir. Her bağlantı, biyolojik bir beyindeki sinapslar gibi, bir yapay nörondan diğerine bilgi, bir "sinyal" iletebilir. Bir sinyal alan yapay bir nöron, onu işleyebilir ve daha sonra ona bağlı ek yapay nöronları işaret edebilir. Ortak YSA uygulamalarında, yapay nöronlar arasındaki bağlantıdaki sinyal gerçek bir sayıdır ve her bir yapay nöronun çıkışını, girdilerinin toplamının doğrusal olmayan bir fonksiyonu tarafından hesaplanır. Yapay nöronlar arasındaki bağlantılar "kenarlar" denir. Yapay nöronlar ve kenarlar tipik olarak öğrenme ilerledikçe ayarlanan bir ağırlığa sahiptir. Ağırlık, bir bağlantıdaki sinyalin gücünü arttırır veya azaltır. Yapay nöronlar, sadece toplam sinyal bu eşği geçtiğinde sinyalin gönderileceği bir eşik değerine sahip olabilir. Tipik olarak, yapay nöronlar katmanlar halinde toplanır. Farklı katmanlar, girdilerinde farklı türde dönüşümler gerçekleştirilebilir. Sinyaller, muhtemelen katmanları birden çok kez geçtikten sonra, ilk katmandan (giriş katmanı) son katmana (çıkış katmanı) gider.

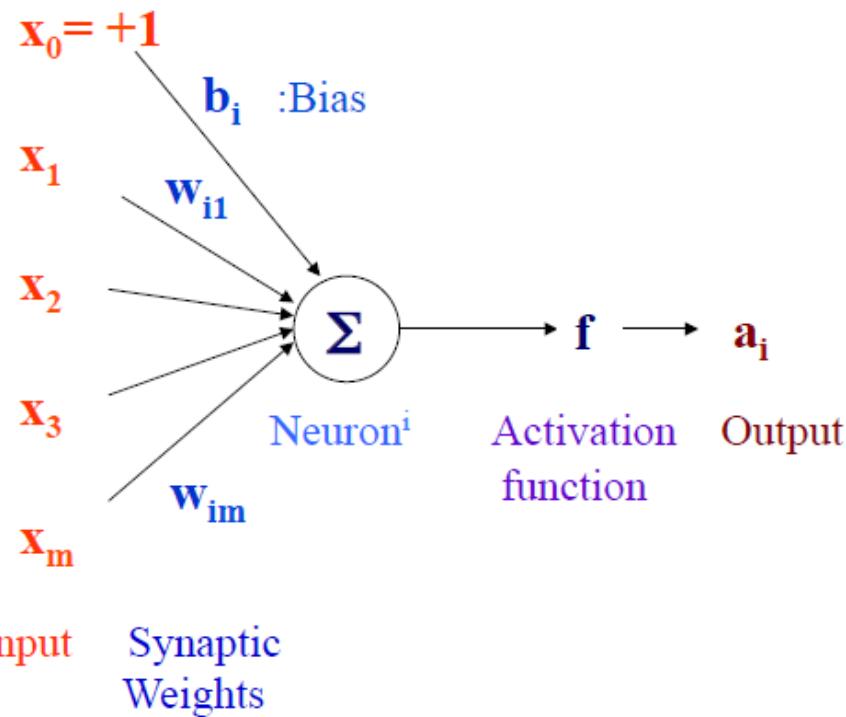
YSA yaklaşımının asıl amacı, problemleri bir insan beyninde olduğu gibi çözmekti. Bununla birlikte, zamanla dikkat, belirli görevlerin yerine getirilmesine ve biyolojiden sapmalara yol açtı. Yapay sinir ağları, bilgisayar görme, konuşma tanıma, makine çevirisii, sosyal ağ filtreleme, oyun tahtası ve video oyunları ve tıbbi teşhis gibi çeşitli görevlerde kullanılmıştır.

Derin öğrenme yapay bir sinir ağında birden fazla gizli katmandan oluşur. Bu yaklaşım, insan beyninin ışığı ve sesi görme ve işitme biçimini modellemeye çalışır. Derin öğrenmenin bazı başarılı uygulamaları bilgisayarla görme ve konuşma tanımadır.



Yapay bir sinir ağı, bir beyindeki geniş nöron ağına benzer şekilde birbirine bağlı bir düğüm grubudur. Burada, her dairesel düğüm bir yapay nöronu temsil eder ve bir ok, bir yapay nöronun çıkışından diğerinin girişine bir bağlantıyı temsil eder.

Yapay Nöron Modeli



Eğitim – Sapma (Bias):

Bias: eşik değer, eğilim ya da sapma değeri.

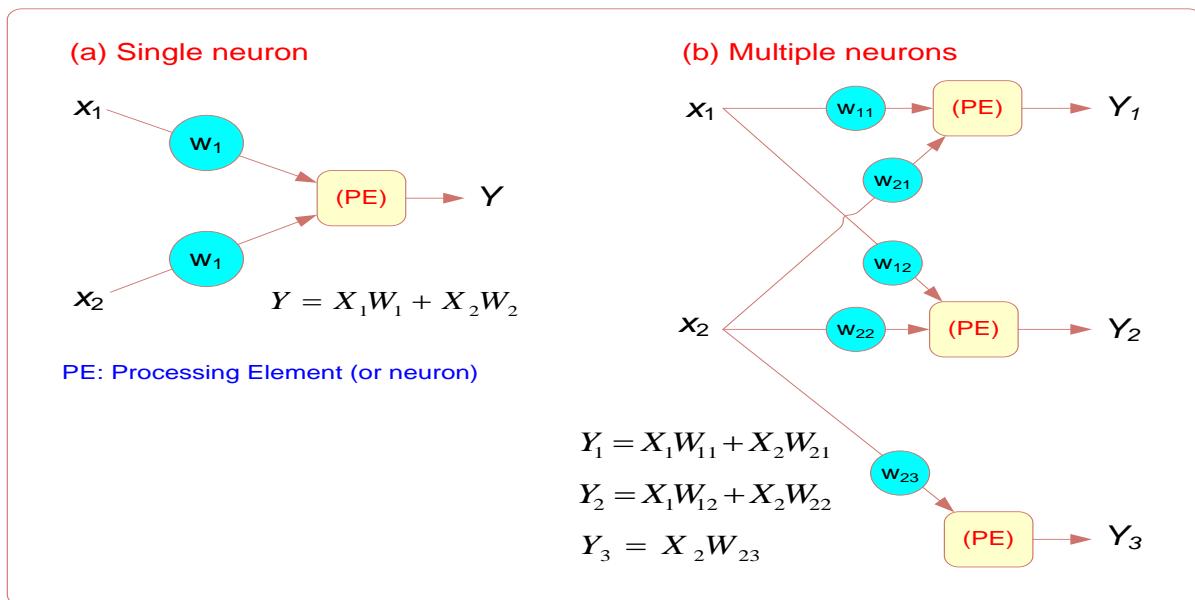
Yapay bir nöron: Girdisinin ağırlıklı toplamını hesaplar (net girdisi denir). Sapmasını ekler. Bu değeri bir aktivasyon işlevinden geçirir. Nöronun çıkışı sıfırın üzerindeyse tetiklendiğini, "ateşlendiğini" (yani aktif hale geldiğini) söylüyoruz. Sapma, sabit +1.0 girdisine kenetlenen başka bir ağırlık olarak dahil edilebilir. Bu ekstra serbest değişken (eğilim ya da sapma), nöronu daha güçlü hale getirir.

n

$$a_i = f(n_i) = f(\sum_{j=1}^n w_{ij}x_j + b_i)$$

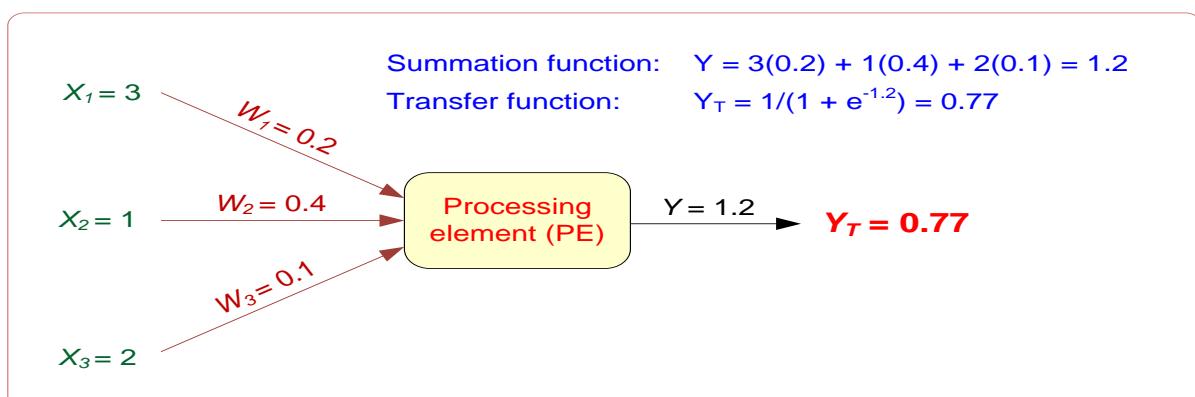
Yapay Sinir Ağları (YSA) Unsurları:

- İşleme öğesi (Processing element - PE)
- Ağ Mimarisi
 - Gömülü Katmanlar
 - Paralel İşlem
- Ağ Bilgisi İşleme
 - Girişler
 - Çıışlar
 - Baııltı Ağırılıkları (Connection weights)
 - Toplama İşlevi (Summation function)



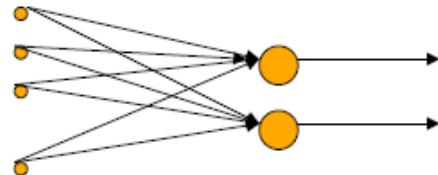
Transformation (Transfer) Function:

- Linear function
- Sigmoid (logical activation) function [0 1]
- Tangent Hyperbolic function [-1 1]



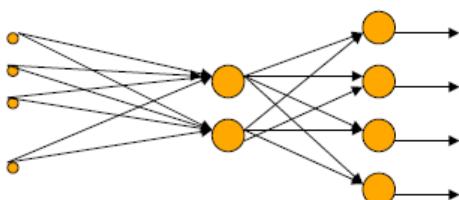
Farklı Ağ Topolojileri:

- **Tek katmanlı ileri beslemeli ağlar:** Çıktı katmanına yansıyan girdi katmanı.



Input Output
layer layer

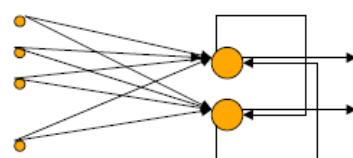
- **Çok katmanlı ileri beslemeli ağlar:** Bir veya daha fazla gömülü katman. Girdi projeleri yalnızca önceki katmanlardan bir katmana yansıtılır. Tipik olarak, yalnızca bir katmandan diğerine bağlanırlar.



2-layer or
1-hidden layer
fully connected
network

Input Hidden Output
layer layer layer

- **Tekrarlayan ağlar:** Bazı girişlerinin bazı çıkışlarına (ayrık zaman) bağlı olduğu geri beslemeli bir ağ.



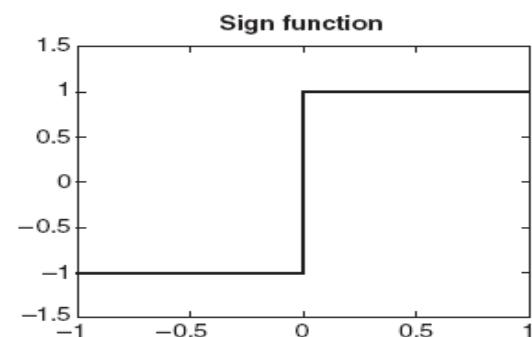
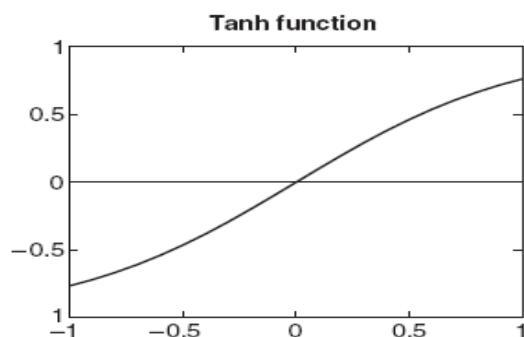
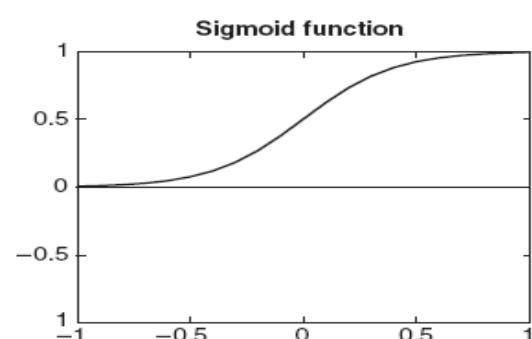
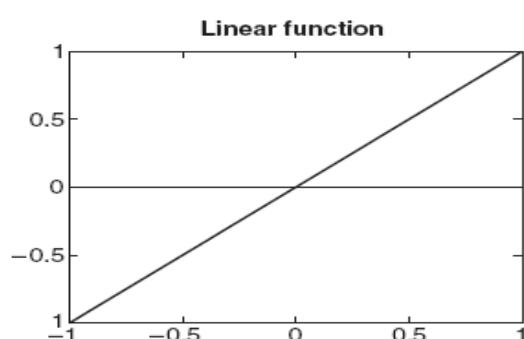
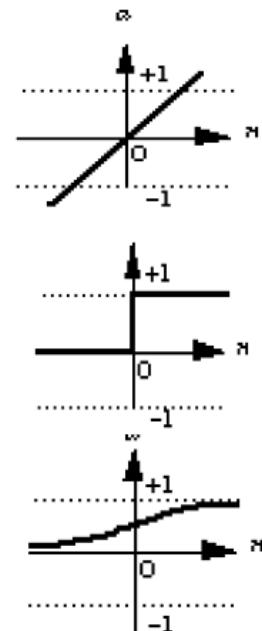
Input Output
layer layer

Aktivasyon fonksiyonları:

Nöronun çıkışının genliğini sınırladığı için limit işlevi olarak da adlandırılır.

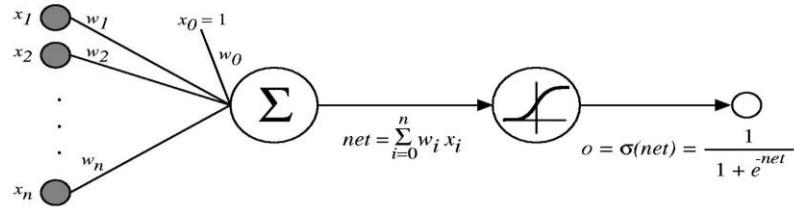
Many types of activation functions are used:

- linear: $a = f(n) = n$
- threshold: $a = \begin{cases} 1 & \text{if } n \geq 0 \\ 0 & \text{if } n < 0 \end{cases}$ (hardlimiting)
- sigmoid: $a = 1/(1+e^{-n})$
- ...



Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins
Log-Sigmoid	$a = \frac{1}{1 + e^{-n}}$		logsig
Hyperbolic Tangent Sigmoid	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$		tansig
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n$		poslin
Competitive	$a = 1 \quad \text{neuron with max } n$ $a = 0 \quad \text{all other neurons}$		compet

Sigmoid unit:

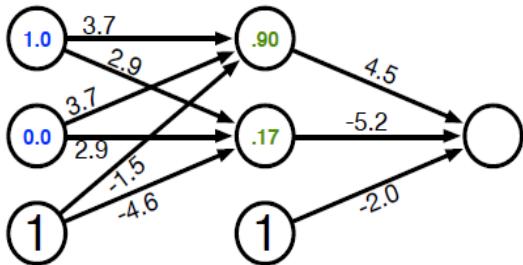


$\sigma(x)$ is the sigmoid function

$$\frac{1}{1 + e^{-x}}$$

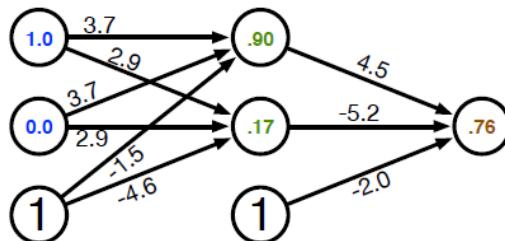
Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

Örnek: Basit Sinir Ağı



$$\text{sigmoid}(1.0 \times 3.7 + 0.0 \times 3.7 + 1 \times -1.5) = \text{sigmoid}(2.2) = \frac{1}{1 + e^{-2.2}} = 0.90$$

$$\text{sigmoid}(1.0 \times 2.9 + 0.0 \times 2.9 + 1 \times -4.5) = \text{sigmoid}(-1.6) = \frac{1}{1 + e^{1.6}} = 0.17$$

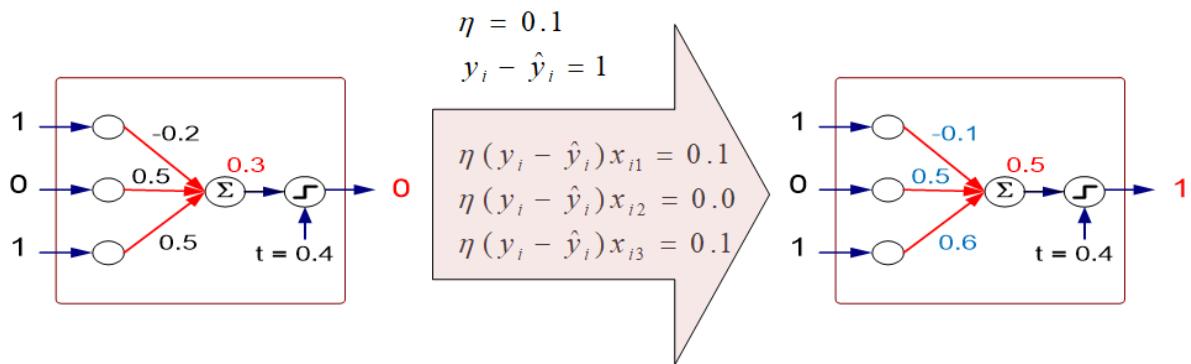


- Output

$$\text{sigmoid}(.90 \times 4.5 + .17 \times -5.2 + 1 \times -2.0) = \text{sigmoid}(1.17) = \frac{1}{1 + e^{-1.17}} = 0.76$$

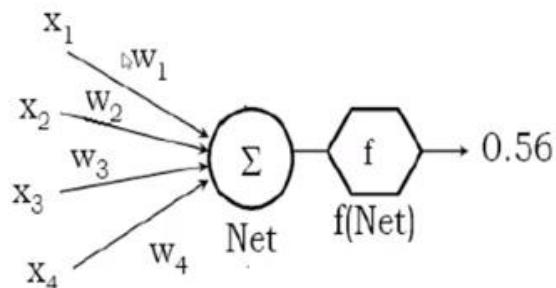
Bir numuneyi işleme örneği:

X₁	X₂	X₃	Y
1	0	1	1



Örnek:

<u>Girişler</u>	<u>Ağırlıklar</u>
$x_1 = 0.5$	$w_1 = -0.2$
$x_2 = 0.6$	$w_2 = 0.6$
$x_3 = 0.2$	$w_3 = 0.2$
$x_4 = 0.7$	$w_4 = -0.1$



Hücrenin net girdisi;

$$\text{Net} = \sum x_i w_i \quad i=1 \dots 4$$

$$\text{Net} = 0.5 * (-0.2) + 0.6 * 0.6 + 0.2 * 0.2 + 0.7 * (-0.1)$$

$$\text{Net} = 0.23$$

• **Sigmoid aktivasyon fonksiyonuna göre hücrenin çıkışı;**

$$f(\text{Net}) = 1 / (1 + e^{-\text{Net}})$$

$$f(\text{Net}) = 0.56$$

8.3. Pekiştirmeli öğrenme

Pekiştirmeli öğrenme, davranışçılıktan esinlenen, öznelerin bir ortamda en yüksek ödül miktarına ulaşabilmesi için hangi eylemleri yapması gereğiyle ilgilenen bir makine öğrenmesi yaklaşımıdır. Bu problem, genelligidenden ötürü oyun kuramı, kontrol kuramı, yüneylem araştırması, bilgi kuramı, benzetim tabanlı eniyileme ve istatistik gibi birçok diğer dalda da çalışılmaktadır.

Makine öğrenmesinde, ortam genellikle bir Markov karar süreci (MKS) olarak modellenir, bu bağlamda birçok pekiştirmeli öğrenme algoritması dinamik programlama tekniklerini kullanır. Pekiştirmeli öğrenme algoritmalarının klasik tekniklerden farkı, MKS hakkında ön bilgiye ihtiyaç duymamaları ve kesin yöntemlerin verimsiz kaldığı büyük MKS'ler için kullanılmalıdır.

Pekiştirmeli öğrenme, doğru girdi/çıktı eşleşmelerinin verilmemesi ve optimal olmayan eylemlerin dışarıdan düzeltilememesi yönleriyle gözetimli öğrenmeden ayrıılır. Dahası, pekiştirmeli öğrenmede bilinmeyen uzayda keşif (İngilizce: exploration) ile mevcut bilgiden istifade (İngilizce: exploitation) arasında bir denge kurma söz konusudur.

Types of Algorithms used in Deep Learning

Here is the list of top 10 most popular deep learning algorithms:

1. Convolutional Neural Networks (CNNs)
2. Long Short Term Memory Networks (LSTMs)
3. Recurrent Neural Networks (RNNs)
4. Generative Adversarial Networks (GANs)
5. Radial Basis Function Networks (RBFNs)
6. Multilayer Perceptrons (MLPs)
7. Self Organizing Maps (SOMs)
8. Deep Belief Networks (DBNs)
9. Restricted Boltzmann Machines(RBMs)
10. Autoencoders

Deep learning algorithms work with almost any kind of data and require large amounts of computing power and information to solve complicated issues. Now, let us, deep-dive, into the top 10 deep learning algorithms.

1. Convolutional Neural Networks (CNNs)

[CNN](#)'s, also known as ConvNets, consist of multiple layers and are mainly used for image processing and object detection. Yann LeCun developed the first CNN in 1988 when it was called LeNet. It was used for recognizing characters like ZIP codes and digits.

CNN's are widely used to identify satellite images, process medical images, forecast time series, and detect anomalies.

How Do CNNs Work?

CNN's have multiple layers that process and extract features from data:

Convolution Layer

- CNN has a convolution layer that has several filters to perform the convolution operation.

Rectified Linear Unit (ReLU)

- CNN's have a ReLU layer to perform operations on elements. The output is a rectified feature map.

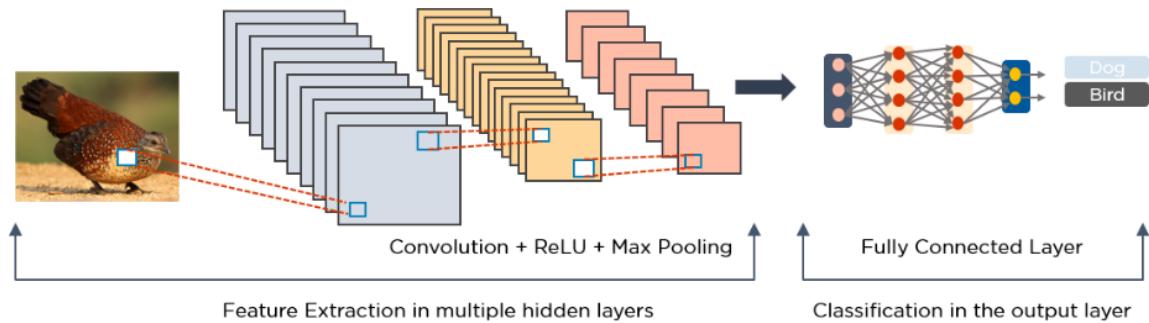
Pooling Layer

- The rectified feature map next feeds into a pooling layer. Pooling is a down-sampling operation that reduces the dimensions of the feature map.
- The pooling layer then converts the resulting two-dimensional arrays from the pooled feature map into a single, long, continuous, linear vector by flattening it.

Fully Connected Layer

- A fully connected layer forms when the flattened matrix from the pooling layer is fed as an input, which classifies and identifies the images.

Below is an example of an image processed via CNN.



2. Long Short Term Memory Networks (LSTMs)

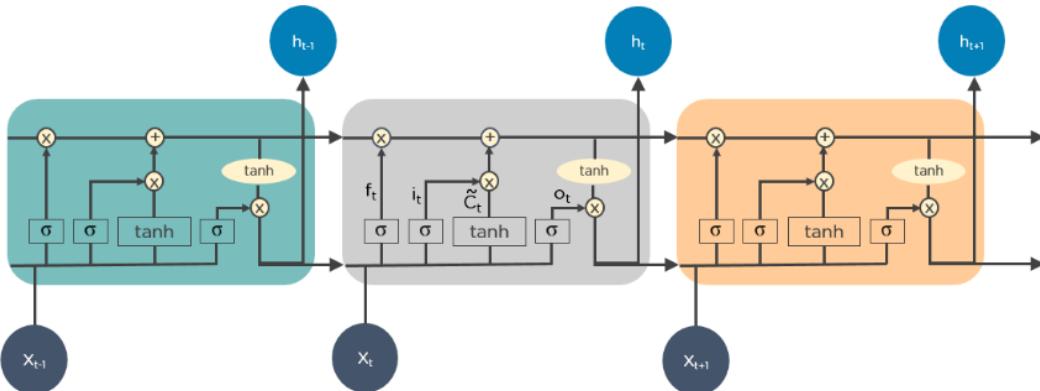
LSTMs are a type of Recurrent Neural Network (RNN) that can learn and memorize long-term dependencies. Recalling past information for long periods is the default behavior.

LSTMs retain information over time. They are useful in time-series prediction because they remember previous inputs. LSTMs have a chain-like structure where four interacting layers communicate in a unique way. Besides time-series predictions, LSTMs are typically used for speech recognition, music composition, and pharmaceutical development.

How Do LSTMs Work?

- First, they forget irrelevant parts of the previous state
- Next, they selectively update the cell-state values
- Finally, the output of certain parts of the cell state

Below is a diagram of how LSTMs operate:

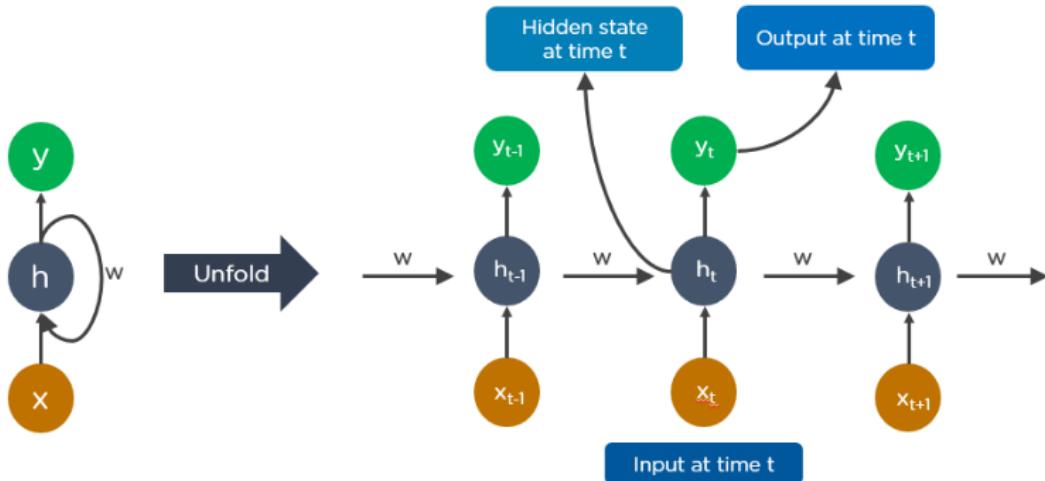


3. Recurrent Neural Networks (RNNs)

[RNNs](#) have connections that form directed cycles, which allow the outputs from the LSTM to be fed as inputs to the current phase.

The output from the LSTM becomes an input to the current phase and can memorize previous inputs due to its internal memory. RNNs are commonly used for image captioning, time-series analysis, natural-language processing, handwriting recognition, and machine translation.

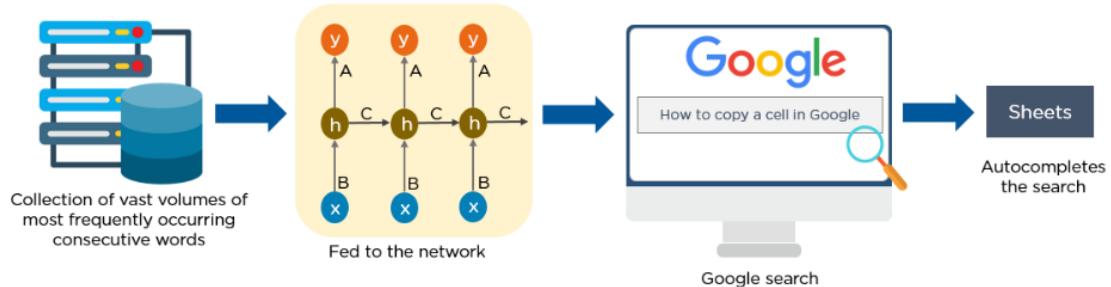
An unfolded RNN looks like this:



How Do RNNs work?

- The output at time $t-1$ feeds into the input at time t .
- Similarly, the output at time t feeds into the input at time $t+1$.
- RNNs can process inputs of any length.
- The computation accounts for historical information, and the model size does not increase with the input size.

Here is an example of how Google's autocompleting feature works:



4. Generative Adversarial Networks (GANs)

GANs are generative deep learning algorithms that create new data instances that resemble the training data. GAN has two components: a generator, which learns to generate fake data, and a discriminator, which learns from that false information.

The usage of GANs has increased over a period of time. They can be used to improve astronomical images and simulate gravitational lensing for dark-matter research. Video game developers use GANs to upscale low-resolution, 2D textures in old video games by recreating them in 4K or higher resolutions via image training.

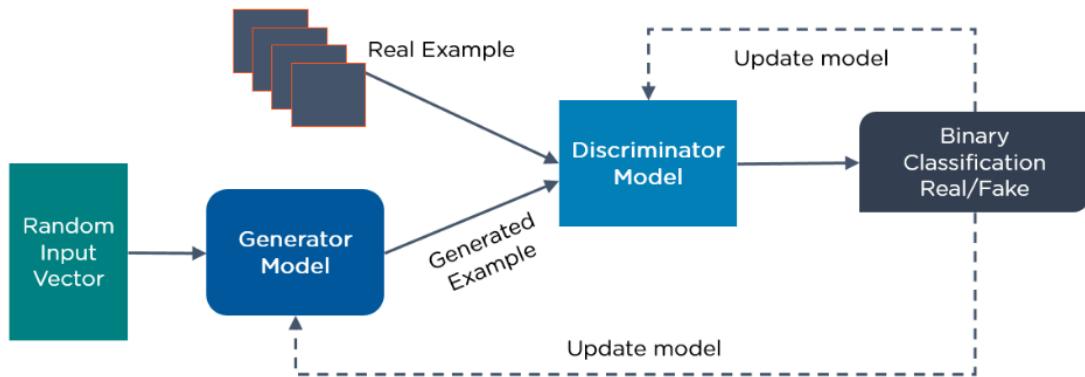
GANs help generate realistic images and cartoon characters, create photographs of human faces, and render 3D objects.

How Do GANs work?

- The discriminator learns to distinguish between the generator's fake data and the real sample data.
- During the initial training, the generator produces fake data, and the discriminator quickly learns to tell that it's false.

- The GAN sends the results to the generator and the discriminator to update the model.

Below is a diagram of how GANs operate:



Master deep learning concepts and the TensorFlow open-source framework with the [Deep Learning Training Course](#). Get skilled today!

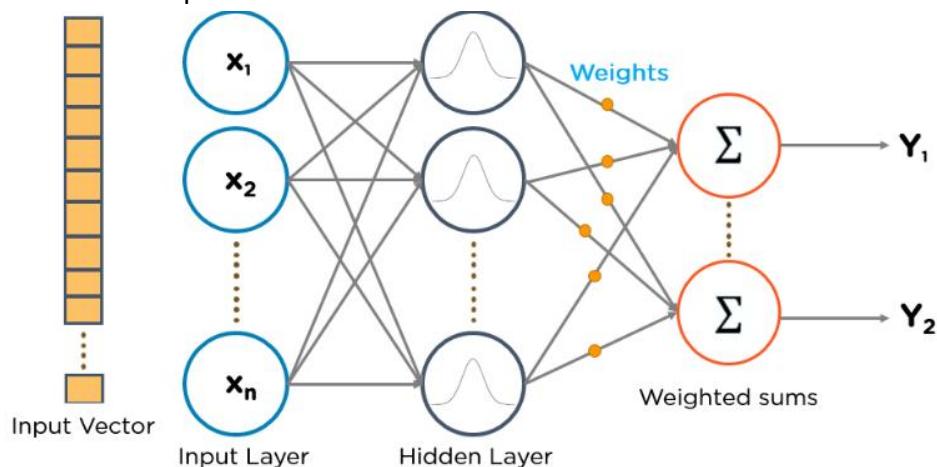
5. Radial Basis Function Networks (RBFNs)

RBFNs are special types of feedforward neural networks that use radial basis functions as activation functions. They have an input layer, a hidden layer, and an output layer and are mostly used for classification, regression, and time-series prediction.

How Do RBFNs Work?

- RBFNs perform classification by measuring the input's similarity to examples from the training set.
- RBFNs have an input vector that feeds to the input layer. They have a layer of RBF neurons.
- The function finds the weighted sum of the inputs, and the output layer has one node per category or class of data.
- The neurons in the hidden layer contain the Gaussian transfer functions, which have outputs that are inversely proportional to the distance from the neuron's center.
- The network's output is a linear combination of the input's radial-basis functions and the neuron's parameters.

See this example of an RBFN:



6. Multilayer Perceptrons (MLPs)

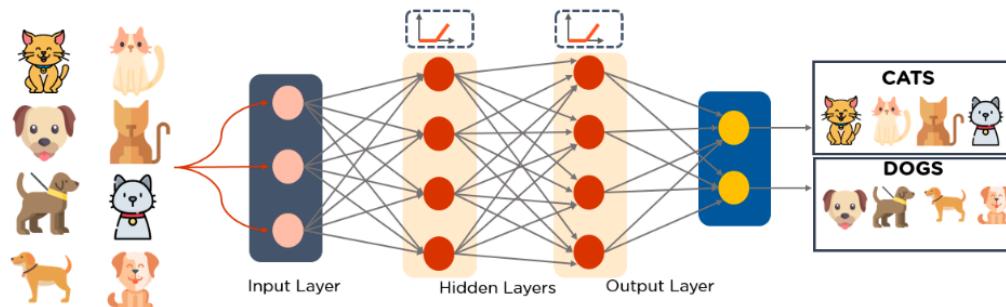
MLPs are an excellent place to start learning about deep learning technology.

MLPs belong to the class of feedforward neural networks with multiple layers of perceptrons that have activation functions. MLPs consist of an input layer and an output layer that are fully connected. They have the same number of input and output layers but may have multiple hidden layers and can be used to build speech-recognition, image-recognition, and machine-translation software.

How Do MLPs Work?

- MLPs feed the data to the input layer of the network. The layers of neurons connect in a graph so that the signal passes in one direction.
- MLPs compute the input with the weights that exist between the input layer and the hidden layers.
- MLPs use activation functions to determine which nodes to fire. Activation functions include ReLUs, sigmoid functions, and tanh.
- MLPs train the model to understand the correlation and learn the dependencies between the independent and the target variables from a training data set.

Below is an example of an MLP. The diagram computes weights and bias and applies suitable activation functions to classify images of cats and dogs.



7. Self Organizing Maps (SOMs)

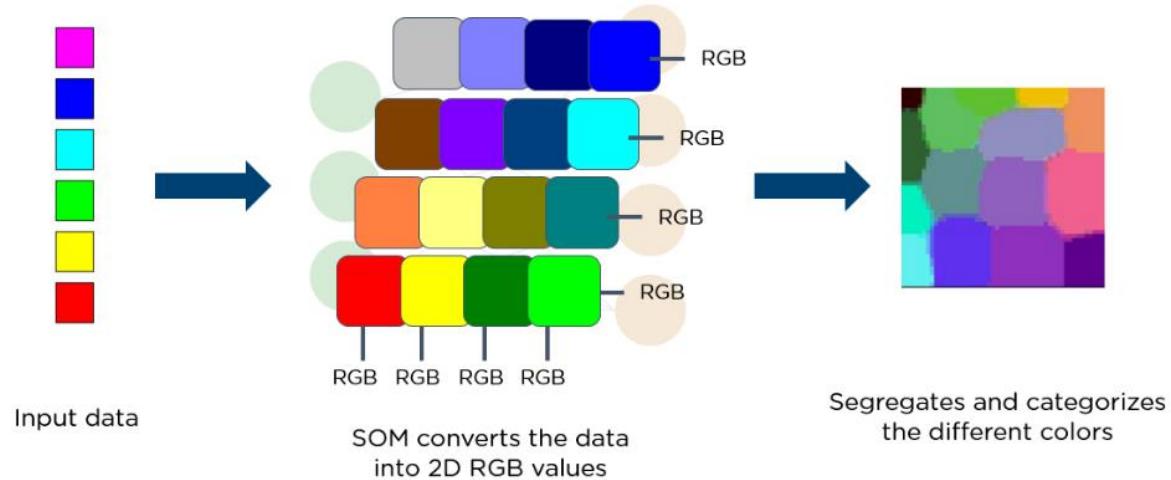
Professor Teuvo Kohonen invented SOMs, which enable [data visualization](#) to reduce the dimensions of data through self-organizing artificial neural networks.

Data visualization attempts to solve the problem that humans cannot easily visualize high-dimensional data. SOMs are created to help users understand this high-dimensional information.

How Do SOMs Work?

- SOMs initialize weights for each node and choose a vector at random from the training data.
- SOMs examine every node to find which weights are the most likely input vector. The winning node is called the Best Matching Unit (BMU).
- SOMs discover the BMU's neighborhood, and the amount of neighbors lessens over time.
- SOMs award a winning weight to the sample vector. The closer a node is to a BMU, the more its weight changes..
- The further the neighbor is from the BMU, the less it learns. SOMs repeat step two for N iterations.

Below, see a diagram of an input vector of different colors. This data feeds to a SOM, which then converts the data into 2D RGB values. Finally, it separates and categorizes the different colors.



8. Deep Belief Networks (DBNs)

DBNs are generative models that consist of multiple layers of stochastic, latent variables. The latent variables have binary values and are often called hidden units.

DBNs are a stack of Boltzmann Machines with connections between the layers, and each RBM layer communicates with both the previous and subsequent layers. Deep Belief Networks (DBNs) are used for image-recognition, video-recognition, and motion-capture data.

Deep Learning Course (with TensorFlow & Keras)

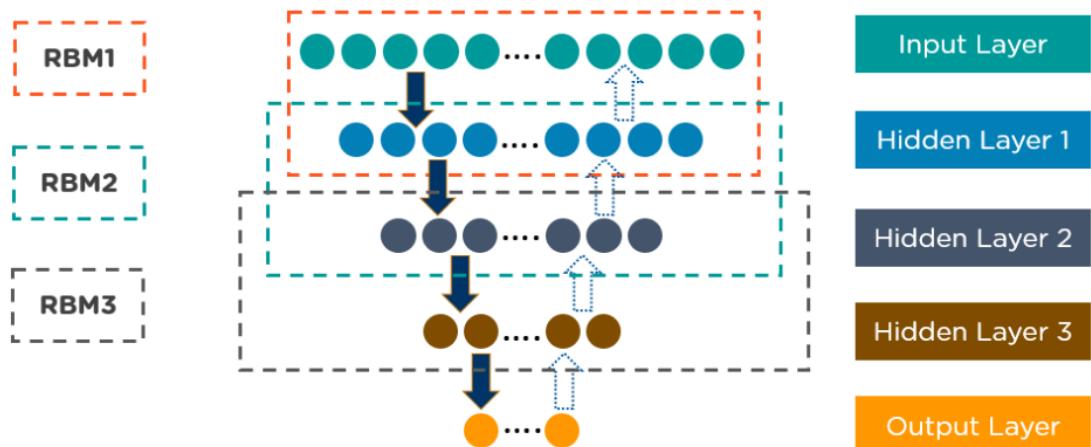
Master the Deep Learning Concepts and Models [VIEW COURSE](#)



How Do DBNs Work?

- Greedy learning algorithms train DBNs. The greedy learning algorithm uses a layer-by-layer approach for learning the top-down, generative weights.
- DBNs run the steps of Gibbs sampling on the top two hidden layers. This stage draws a sample from the RBM defined by the top two hidden layers.
- DBNs draw a sample from the visible units using a single pass of ancestral sampling through the rest of the model.
- DBNs learn that the values of the latent variables in every layer can be inferred by a single, bottom-up pass.

Below is an example of DBN architecture:



Build deep learning models in TensorFlow and learn the TensorFlow open-source framework with the [Deep Learning Course \(with Keras &TensorFlow\)](#). Enroll now!

9. Restricted Boltzmann Machines (RBMs)

Developed by Geoffrey Hinton, RBMs are stochastic neural networks that can learn from a probability distribution over a set of inputs.

This deep learning algorithm is used for dimensionality reduction, classification, regression, collaborative filtering, feature learning, and topic modeling. RBMs constitute the building blocks of DBNs.

RBM^s consist of two layers:

- Visible units
- Hidden units

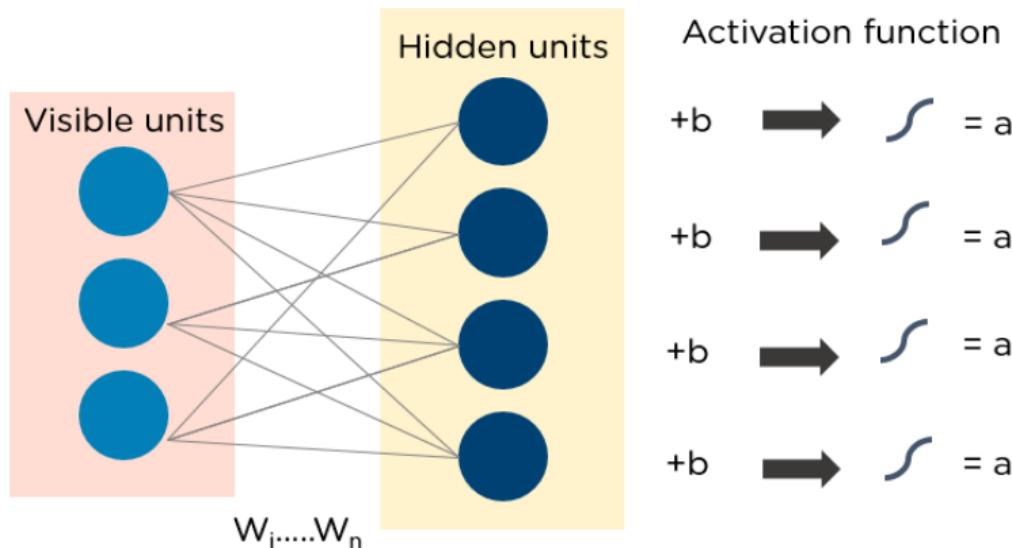
Each visible unit is connected to all hidden units. RBMs have a bias unit that is connected to all the visible units and the hidden units, and they have no output nodes.

How Do RBMs Work?

RBM^s have two phases: forward pass and backward pass.

- RBMs accept the inputs and translate them into a set of numbers that encodes the inputs in the forward pass.
- RBMs combine every input with individual weight and one overall bias. The algorithm passes the output to the hidden layer.
- In the backward pass, RBMs take that set of numbers and translate them to form the reconstructed inputs.
- RBMs combine each activation with individual weight and overall bias and pass the output to the visible layer for reconstruction.
- At the visible layer, the RBM compares the reconstruction with the original input to analyze the quality of the result.

Below is a diagram of how RBMs function:



10. Autoencoders

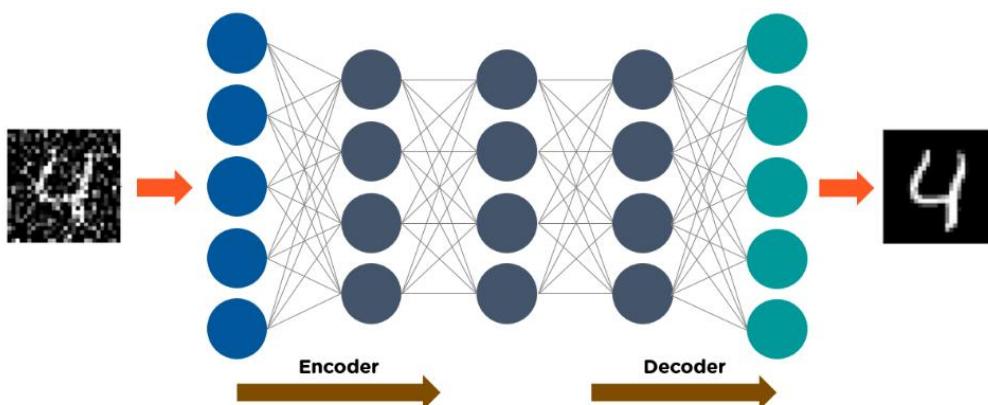
[Autoencoders](#) are a specific type of feedforward neural network in which the input and output are identical. Geoffrey Hinton designed autoencoders in the 1980s to solve unsupervised learning problems. They are trained neural networks that replicate the data from the input layer to the output layer. Autoencoders are used for purposes such as pharmaceutical discovery, popularity prediction, and image processing.

How Do Autoencoders Work?

An autoencoder consists of three main components: the encoder, the code, and the decoder.

- Autoencoders are structured to receive an input and transform it into a different representation. They then attempt to reconstruct the original input as accurately as possible.
- When an image of a digit is not clearly visible, it feeds to an autoencoder neural network.
- Autoencoders first encode the image, then reduce the size of the input into a smaller representation.
- Finally, the autoencoder decodes the image to generate the reconstructed image.

The following image demonstrates how autoencoders operate:



Conclusion

Deep learning has evolved over the past five years, and deep learning algorithms have become widely popular in many industries. If you are looking to get into the exciting career of data science and want to learn how to work with deep learning algorithms, check out our [AI and ML courses](#) training today.

If you have deep learning algorithm questions after reading this article, please leave them in the comments section, and Simplilearn's team of experts will return with answers shortly.

9. Genetik Algoritmalar

Genetik algoritma (GA), belirli bir soruna iyi çözümler bulma umuduyla yeni genotipler üretmek için mutasyon ve çaprazlama gibi yöntemleri kullanarak doğal seleksiyon sürecini taklit eden bir arama algoritması ve sezgisel tekniktir. Makine öğrenmesinde, 1980'ler ve 1990'larda genetik algoritmalar kullanılmıştır. Tersine, genetik ve evrimsel algoritmaların performansını artırmak için makine öğrenme teknikleri kullanılmıştır.

Genetik Algoritma yaklaşımının ortaya çıkıştı 1970 'lerin başında olmuştur . 1975 'te John Holland'ın makine öğrenmesi üzerine yaptığı çalışmalarla canlılardaki evrimden ve değişimden etkilenerken, bu genetik evrim sürecini bilgisayar ortamına aktarması ve böylece bir tek mekanik yapının öğrenme yeteneğini geliştirmek yerine, çok sayıdaki böyle yapıların tamamını “ çoğalma, değişim gibigenetik süreçler sonunda üstün yeni bireylerin elde edilebileceğini gösteren çalışmasından çıkan sonuçların yayınlanmasından sonra geliştirdiği yöntemin adı “Genetik Algoritmalar” olarak tanınmıştır.

Genetik Algoritmalar, süreçleri model olarak kullanan problem çözme teknikleridir. Öğrenen Makina Algoritmaları ise tecrübe ile ortaya çıkan yeni duruma ya da belirsizliğe ilişkin parameterlerin otomatik olarak belirlendiği matematiksel modellerdir.

Genetik Algoritmalar mümkün olan çözümlerin bir popülasyonu üzerinde işlem yapan olasılıklı (stochastic) arama algoritmalarıdır. Geleneksel programlama teknikleriyle çözülmesi güç olan, özellikle sınıflandırma ve çok boyutlu optimizasyon problemleri, bunların yardımıyla daha kolay ve hızlı olarak çözüme ulaşmaktadır. Genetik algoritmalar doğada geçerli olan en iyinin yaşaması kuralına dayanarak sürekli iyileşen çözümler üretir. Bunun için “iyi”nin ne olduğunu belirleyen bir uygunluk fonksiyonu ve yeni çözümler üretmek için yeniden kopyalama, değiştirme gibi operatörleri kullanır. Görevler için programın ölçülen performansı tecrübe ile artıyor ise bu program tecrübe ile öğreniyor denilebilir. Genetik algoritmalar robot kontrolü için kural kümelerinin öğrenilmesi ve yapay sinir ağları için topoloji ve öğrenme parametrelerinin optimize edilmesi için kullanılmaktadır.

Göçmen kuşlar, her yıl çıktııkları yolculukta adreslerini hiç şaşırmadan, hiçbir duraklarını atlamanın, yüzyıllardır sürekli hareket halindeler; üstelik pusulasız ve haritasız... Dünyadaki on binlerce kuş türünün sadece sekiz bin kadarı her yıl binlerce kilometre kat ederek nesillerini sürdürmeye çalışır. Kimileri sürüler halinde, kimileri küçük gruplar halinde ömürlerinin sonuna kadar kendilerini bu uzun yolculuğa adarlar. Bu kuşların tek bir ortak amaçları vardır: Üremek ve beslenmek. Göçmen kuşların uzun uçuşlarına iç güdüsel olarak hazırlandıklarını da söyleyebiliriz. Öyle ki, genlerinde uzak diyalara göç etme nitelikleri bulunan hayvanat bahçelerinde, ya da kafeste beslenen kuşlar göç vakti geldiklerinde kiper kiper oldukları gözlenir.

Göçmen kuşların her yıl binlerce kilometre süren yolculukları sırasında yönlerini kaybolmadan nasıl tayin edebildikleri üzerine yapılan araştırmalarda ortaya sadece mantıksal tahminler atılmış durumda. Örneğin, kuşların tanık kara parçalarını ezberledikleri, sürekli sahil şeridini takip ettikleri gibi var sayımlar bir yere kadar doğru ama uzun deniz yolculukları yapanlar bu varsayımları yerle bir ediyor. Diğer taraftan gece uçuşu yapanlar içinse yıldızların onlara yol gösterdiği düşünülüyor. Bir diğer var sayımla ise göçmen kuşların, dünyanın manyetik alan çizgilerini takip ederek kaybolmadan yönlerini bulabildikleri şeklinde. Modern çağın nimetlerinden biri olan GPS sayesinde işaretlenen kimi göçmen kuşların izledikleri rotalar her yıl tipa tip aynı. Hatta inanılması güç ama, her yıl mola verdikleri yerler bile şaşmıyor.

Göçmen kuşların hayranlıkla izlenebilecek bir başka özellikleri de havada çizdikleri V şeklindeki düzen olsa gerek. Öyle ya, ancak bu şekilde onları diğer sürülerden net bir şekilde ayırt edebiliyoruz. Bu uçuş düzeninin iki önemli avantajı var. İlki, en öndeği lider kuşu görebilmek. İkincisi ve tabi ki daha önemli olanı ise hemen öndeği kuşun yarattığı hava akımından yararlanarak daha az enerji harcayarak uçmak. Kuşların bu şekilde uçmaları sayesinde tahminen yüzde 20'lik bir enerji tasarrufu sağladıklarına inanılıyor. Eğer havada bu tür bir V oluşumu yakalarsanız, kuşların yerlerini sürekli değiştirdiğine de tanık olacaksınız. Özellikle de en önde uçarak en fazla yorulan lider kuşa eşlik edenlerin onun yerini nasıl almaya çalıştıklarını göreceksiniz.

Geleneksel programlama teknikleriyle çözülmesi güç olan, özellikle sınıflandırma ve çok boyutlu optimizasyon problemleri, bunların yardımıyla daha kolay ve hızlı olarak çözüme ulaştırılmaktadır. Genetik algoritmalar doğada geçerli olan en iyinin yaşaması kuralına dayanarak sürekli iyileşen çözümler üretir. Bunun için "iyi"nin ne olduğunu belirleyen bir uygunluk fonksiyonu ve yeni çözümler üretmek için yeniden kopyalama, değiştirme gibi operatörleri kullanır. Görevler için programın ölçülen performansı tecrübe ile artıyor ise bu program tecrübe ile öğreniyor denilebilir. Genetik algoritmalar robot kontrolü için kural kümelerinin öğrenilmesi ve yapay sinir ağları için topoloji ve öğrenme parametrelerinin optimize edilmesi için kullanılmaktadır.

Genetik algoritmalar, evrim teorisinin dayandığı temel prensiplerinden olan doğal seçelim ile en iyi bireylerin hayatı kalması ilkesini taklit eden bir tekniktir. Burada yapılan, en iyi çözümün pek çok çözüm seçeneği içinden arama yapılarak belirlenmesidir. Rassal arama teknikleri ile eldeki mevcut çözümlerden hareketle en iyi çözüme ulaşılmasına çalışılmaktadır. Basit bir genetik algoritmanın işlem adımları; problemin olası çözümlerinin dizilere (kromozomlar) kodlanarak çözüm yığının oluşturulması, kromozomların çözüme yaklaşma başarısının uygunluk fonksiyonu ile değerlendirilmesi, genetik parametrelerin belirlenmesi, seçim stratejisi ve mekanizmaları, genetik operatörler ve durdurma kriteri olarak sıralanabilir.

Genetik algoritmalar, bilinen yöntemlerle çözülemeyen veya çözüm süresi problemin büyüklüğüne göre oldukça fazla olan problemlerde, kesin sonuca çok yakın sonuçlar verebilen bir yöntemdir. Bu özelliği ile, NP (Nonpolynomial-Polinom olmayan) problemler yanında gezgin satıcı, karesel atama, yerleşim, atölye çizelgeleme, mekanik öğrenme, üretim planlama, elektronik, finansman ve hücresel üretim gibi konularda uygulanmaktadır.

GEN: Kendi başına anlamı olan ve genetik bilgi taşıyan en küçük genetik birimdir.

- Bir gen A, B gibi bir karakter olabileceği gibi 0 veya 1 ile ifade edilen bir bit veya bit dizisi olabilir. Örneğin bir cismin x koordinatındaki yerini gösteren bir gen 101 şeklinde ifade edilebilir

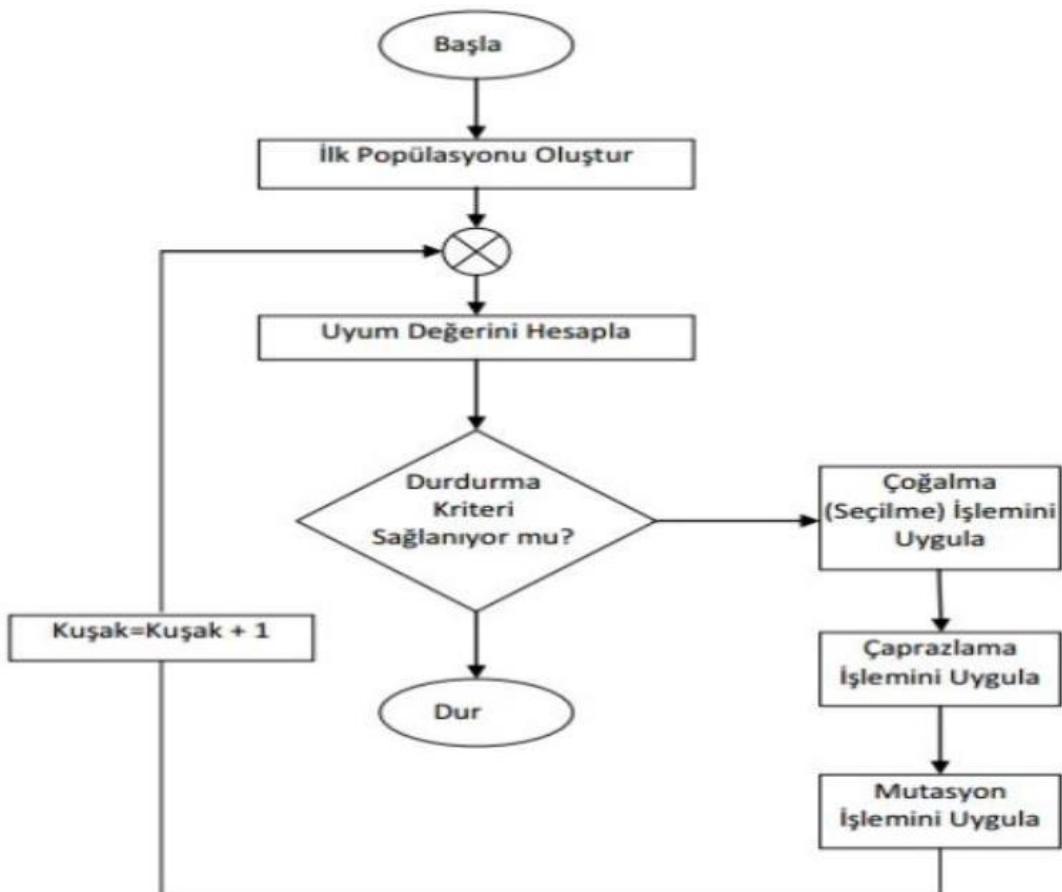
KROMOZOM: Bir ya da birden fazla genin bir araya gelmesiyle oluşurlar Probleme ait tüm bilgileri içerirler. Kromozomlar toplumdaki bireyler yada üyelere karşılık gelirler. Ele alınan problemde alternatif çözüm adayıdır.

Örneğin, kromozom bir problemde açı, boyut ve koordinat değişkenlerinden veya bir dikdörtgen prizmasının ölçülerinden (genişlik, derinlik) oluşabilir 001 101 111 1 5 7 değerleri kromozomu oluşturan genlerdir. Genetik algoritma işlemlerinde kromozomları kullandığı için kromozom tanımları çok iyi ifade edilmelidir.

POPÜLASYON: Kromozomlar veya bireyler topluluğudur. Popülasyon üzerinde durulan problem için alternatif çözümler kümesidir. Aynı anda bir popülasyonda ki birey sayısı sabit ve probleme göre kullanıcı tarafından belirlenir. Zayıf olan bireylerin yerini kuvvetli yeniler almaktadır.

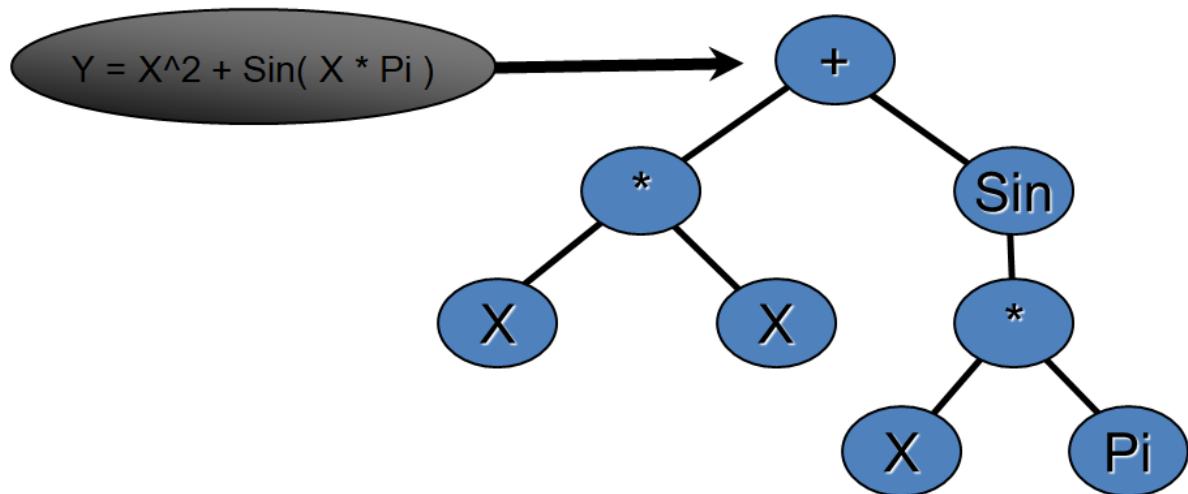
Genetik Algoritmalar Nasıl Çalışır?

1. Adım: Olası çözümlerin kodlandığı bir çözüm grubu oluşturulur. Çözüm grubuna biyolojideki benzerliği nedeniyle populasyon çözümlerin kodları da kromozom olarak adlandırılır. Bu adıma populasyonda bulunan birey sayısı belirleyerek başlanır. Bu sayı için bir standart yoktur. Genel olarak önerilen 100 - 300 aralığında bir büyülüktür. Büyüklük seçiminde yapılan işlemlerin karmaşıklığı ve aramanın derinliği önemlidir. Popülasyon bu işlemden sonra rasgele oluşturulur.
2. Adım: Her kromozomun ne kadar iyi olduğu bulunur Kromozomların ne kadar iyi olduğunu bulan fonksiyona uygunluk fonksiyonu denir Bu fonksiyon işletilerek kromozomların uygunlıklarının bulunmasına ise hesaplama(evalution adı verilir Bu fonksiyon genetik algoritmanın beynini oluşturmaktadır GA da probleme özel çalışan tek kısım bu fonksiyondır. Coğu zaman GA'nın başarısı bu fonksiyonun verimli ve hassas olmasına bağlı olmaktadır



Genetik algoritmanın akış diyagramı.

Genetic Programming Background:



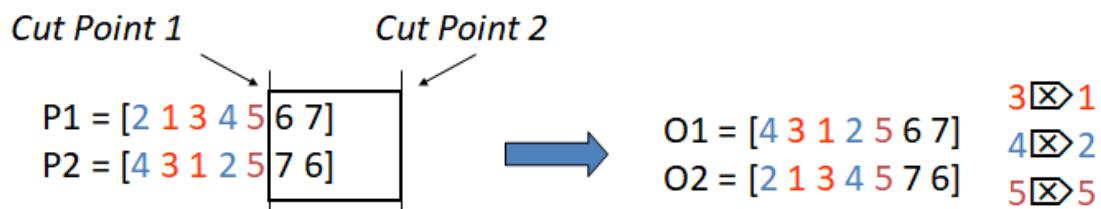
Crossover: bir üst programdaki bir makinedeki işlem sırasını başka bir ana makinedeki başka bir makinedeki işlem dizisiyle birleştirir.

Example 1.

Cut Point



Example 2. Partially Mapped Crossover



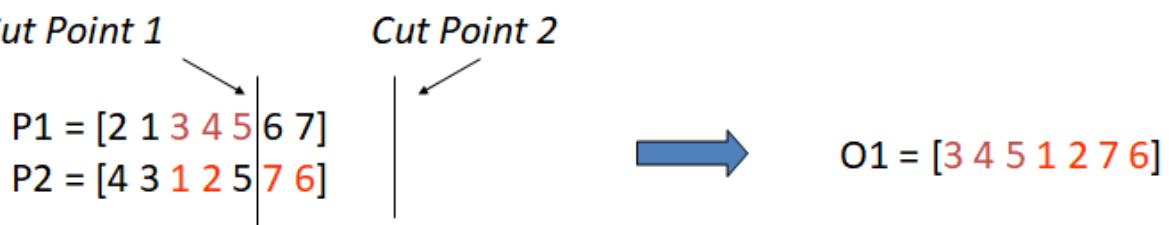
Örnek: P1'den alınan işlerin mutlak konumlarını ve P2'den alınanların göreceli konumlarını korur.

Cut Point 1



Örnek: Örnek 3'e benzer ancak 2 geçiş noktası vardır.

Cut Point 1



Örnek:

Aşağıdaki ifadeyi karşılayan GA'yı kullanarak a ve b'nin optimal değerlerini tahmin edelim.

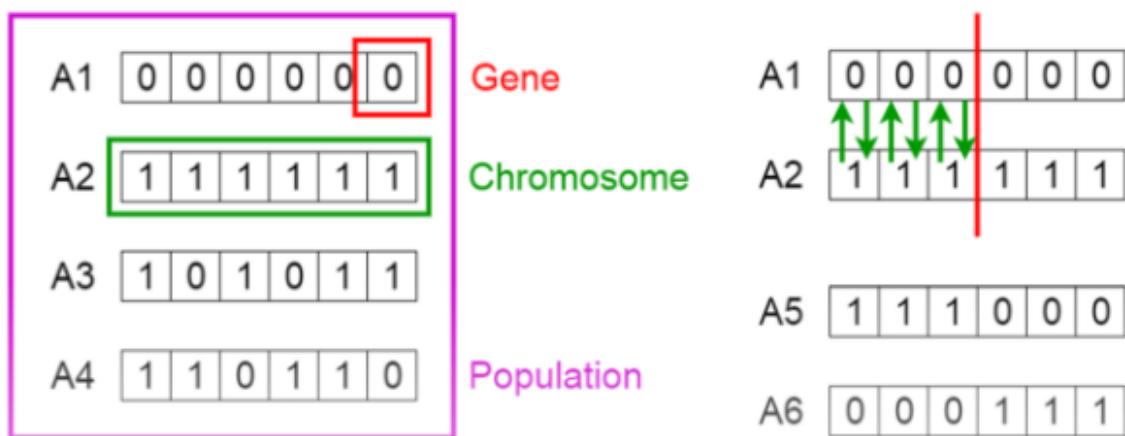
$$2a^2 + b = 57$$

Herhangi bir optimizasyon problemi, bir amaç fonksiyonuyla başlar. Yukarıdaki denklem şu şekilde yazılabilir:

$$f(a,b) = 2a^2 + b - 57$$

Fonksiyonun değerinin 0 olduğu anlaşılmaktadır. Bu fonksiyon bizim amaç fonksiyonumuzdur ve amaç a ve b değerlerini tahmin etmektir, öyle ki objektif fonksiyonun değeri sıfıra indirilir.

Örnek:



Örnek:

Adil bir jetonu 60 kez atıyoruz ve aşağıdaki ilk popülasyonu elde ediyoruz:

$s_1 = 1111010101$	$f(s_1) = 7$
$s_2 = 0111000101$	$f(s_2) = 5$
$s_3 = 1110110101$	$f(s_3) = 7$
$s_4 = 0100010011$	$f(s_4) = 4$
$s_5 = 1110111101$	$f(s_5) = 8$
$s_6 = 0100110000$	$f(s_6) = 3$

Seçimi yaptıktan sonra aşağıdaki popülasyonu elde ettiğimizi varsayalım

$s_1' = 1111010101$	(s_1)
$s_2' = 1110110101$	(s_3)
$s_3' = 1110111101$	(s_5)
$s_4' = 0111000101$	(s_2)
$s_5' = 0100010011$	(s_4)
$s_6' = 1110111101$	(s_5)

Sonra crossover için dizeleri eşleştiriyoruz. Her çift için crossover olasılığına (örneğin 0.6) göre crossover yapıp yapmamaya karar veririz. Sadece çiftler için (s_1' , s_2') ve (s_5' , s_6') crossover yapmaya karar verdığımızı varsayıyalım. Her çift için rastgele bir geçiş noktası çıkarırız, örneğin birinci için 2 ve ikinci için 5.

Before crossover:

$$s_1' = 11\color{blue}{1}01010101$$

$$s_2' = 11\color{orange}{1}011010101$$

$$s_5' = 0100010011$$

$$s_6' = 11101\color{orange}11101$$

After crossover:

$$s_1'' = 11\color{blue}{1}011010101$$

$$s_2'' = 11\color{red}{1}101010101$$

$$s_5'' = 010001\color{blue}{1}1101$$

$$s_6'' = 11101\color{red}10011$$

Son adım, rastgele mutasyon uygulamaktır: yeni popülasyona kopyalayacağımız her bit için küçük bir hata olasılığına izin veriyoruz (örneğin 0.1)

Mutasyonu uygulamadan önce:

$$s_1'' = 1110110101$$

$$s_2^{\prime\prime} = 1111010101$$

$$s_3^{\prime\prime} = 1110111101$$

$$s_4^{\prime\prime} = 0111000101$$

$$s_5^{\prime\prime} = 0100011101$$

$$s_6^{\prime\prime} = 1110110011$$

After applying mutation:

$$s_1^{\prime\prime\prime} = 1110100101 \quad f(s_1^{\prime\prime\prime}) = 6$$

$$s_2^{\prime\prime\prime} = 1111110100 \quad f(s_2^{\prime\prime\prime}) = 7$$

$$s_3^{\prime\prime\prime} = 1110101111 \quad f(s_3^{\prime\prime\prime}) = 8$$

$$s_4^{\prime\prime\prime} = 0111000101 \quad f(s_4^{\prime\prime\prime}) = 5$$

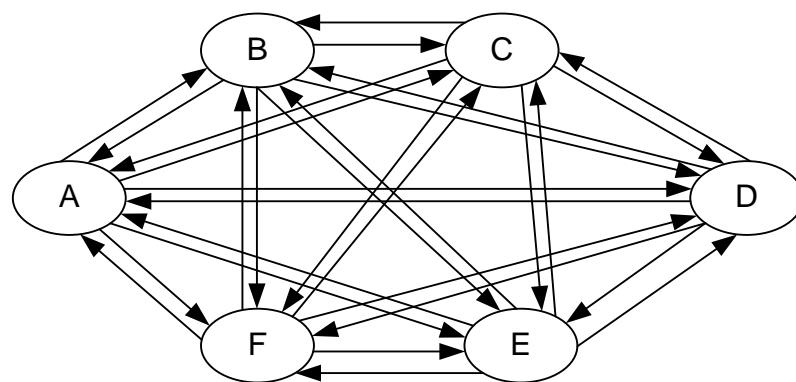
$$s_5^{\prime\prime\prime} = 0100011101 \quad f(s_5^{\prime\prime\prime}) = 5$$

$$s_6^{\prime\prime\prime} = 1110110001 \quad f(s_6^{\prime\prime\prime}) = 6$$

Bir nesilde, toplam popülasyon uygunluğu 34'ten 37'ye değişti, böylece ~% 9 arttı. Bu noktada, bir durdurma kriteri karşılanana kadar aynı süreci yeniden yaşıyoruz.

Örnek:

A, B, C, D, E, F olmak üzere 6 durumum olsun. Her bir durumdan(genden) diğer bütün durumları (genlere) geçiş mümkün olsun. Tüm durumlar bir defa muhakkak ziyaret edilecek. O halde $5! = 120$ tur mümkündür. Tur, her zaman A'dan başlayacak bütün durumlar dolaşılacak sonunda yeniden A'ya dönecek. Durumlar arasında rasgele bir rota oluşturalım. ABDFECA.



Bu problem Genetik Algoritma ile çözülürken,

- Uygunluk fonksiyonu seçilir. Uygunluk fonksiyonu mesafeler olacak, mesafeyi en aza indirilmesi gerekiyor.
- Poülasyon büyülüüğü, kaç çözümden oluşacak, mesela 10 seçilebilir.
- Ebeveyn seçimi, çözüm uzayındaki farklı durumları kullanarak yeni bir rota oluşturulacak. (Rulet tekerliği, sıralı yöntem)
- Çaprazlama, iki turdan yeni bir tur oluşturulurken çaprazlama iki noktalı mı yoksa tek noktalı mı?
- Mutasyon oranı, yeni tur oluşturulduktan sonra bir veya iki durumun (gen) rasgele değiştirilmesi
- Bitirme Şartı, iterasyon sayısının belirli bir sayıya ulaşması mı, uygunluk fonksiyonun belirli bir değerinin altında veya üstünde olması mı karar verilir.

Öncelikle durumlar (genler) arasında rasgele 2 çözüm turu oluşturalım. 6 durum arasında oluşturulacak tur sayısı $= 5! = 120$ farklı tur oluşturulabilir.

Tur-1: A B D F E C A

Tur-2: A C E D F B A

Bu iki tur kendi aralınlarda çaprazlamaya tabi tutulur. Çaprazlama oranı %50 olsun ve tek noktalı olarak yapılsın.

Tur-1: A B D F E C A

Tur-2: A C E D F B A

Yeni tur oluşturulurken birinci turdan ilk kısmı ikinci turdan ise ikinci kısmı alınır. İkinci kısmında olanlar birinci kısmında var ise alınmaz, yerine ikinci turundan başından itibaren olmayanlar alınır.

Tur-3: A B D D F B A = A B D F C E A olur.

Mutasyonu yaparken tek noktalı bir durumun (genin) değiştirilsin. Yeni oluşan Turda rasgele bir durum (gen) değiştirilsin. Sözgelimi B'yi değiştirelim ve E olsun. Yeni tur (kromozom),

Tur-4: A E D F C B A olur. E durumu iki adet olamayacağı için sağ tarafındaki E B ile değiştirilir. Böylece mutasyondan sonraki turda elde edilmiş olur.

Bu aşamadan sonra uygunluk fonksiyonu hesaplanır. Popülasyon içerisindeki diğer çözümlerden daha kısa mesafe ise diğer çözümlerden daha iyi çözüm olarak kabul edilir. Eğer değilse öldürülür.

Adımları sıralarsak,

- 10 tane başlangıç çözümünü oluşturulur.
- Herbir çözüm için uygunluk fonksiyonu değerini hesaplanır.
- Başlangıç çözümlerinden iki tane ebeveyn seçilir.
- Ebeveynler üzerinde çaprazlama yapılır.
- Mutasyona tabi tutulur.
- Yeni bireyin uygunluk fonksiyon değeri hesaplanır.
- Yeni birey öldürülecek mi yoksa popülasyona katılacak mı karar vermek için seleksiyon işlemi yapılır.
- Bitirme şartı sağlandı mı diye kontrol edilir, sağlandıysa en iyi çözüm al ve çıkış.

Yararlanılan Kaynaklar

1. Machine Learning (ML) Algorithms For Beginners with Code Examples in Python.
<https://medium.com/towards-artificial-intelligence/machine-learning-algorithms-for-beginners-with-python-code-examples-ml-19c6afd60daa>
2. https://www.tutorialspoint.com/machine_learning_with_python/knn_algorithm_finding_nearest_neighbors.htm
3. <https://brilliant.org/wiki/feature-vector/>
4. <https://stanford.edu/~shervine/l/tr/teaching/cs-229/>
5. <https://stanford.edu/~shervine/l/tr/teaching/cs-221/>
6. <https://stanford.edu/~shervine/l/tr/teaching/cs-230/>