4. YAPAY SINIR AGLARI

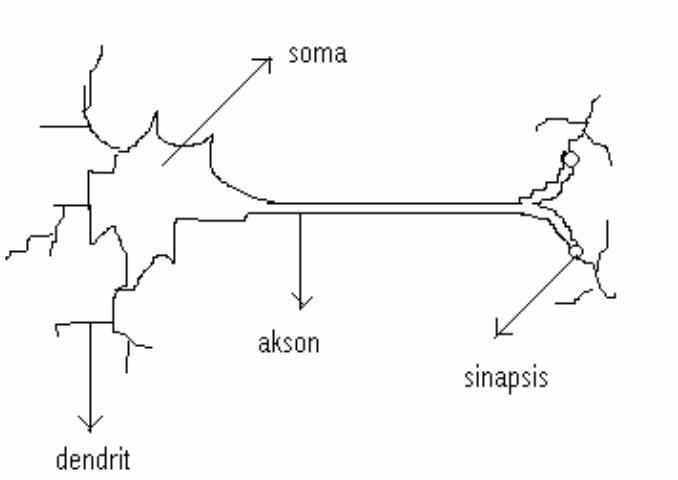
Donald Hebb (1949) bugünün sinir agi teorisinin babasi olarak bilinmektedir. Nörolog olan Hebb, beynin nasil ögrendigi ile ilgili çalismalar yapmistir. Çalismalarina beynin en temel birimi olan sinir hücresini ele alarak baslamistir. Iki sinir hücresinin birbirleriyle nasil bir koralasyon sergilediklerini incelemis ve sinir agi teorisini bu temel üzerine oturtmustur. Bu temel kuskusuz tek gerçek degildir. Çünkü beynin nasil bir çalisma sergiledigi su an dahi teoriler yardimiyla açiklanmaktadir. Ancak Hebb'in yardimiyla bu fikir ile yola çikilmis ve günümüzdeki yüzlerce ayri teoriyle genis bir yelpazeye hitap eder hale gelmistir. Su an gerçek yasamda kullanilan ve basari orani %99'lar ile ifade edilegelen bir sürü yapay sinir agi modeli mevcuttur. Tüm gelistirilen modeller bilgisayar dünyasinda "çözümsüz" veya "np karmasik" olarak nitelendirilen problemlerin çözümünü hedeflemekte ve hatta bir kismini basariyla çözmektedir.(Fausett L.,1994)

4.1 Yapay Sinir Agi Nedir?

Yapay sinir agi; insan beyninin sinir hücrelerinden olusmus katmanli ve paralel olan yapisinin tüm fonksiyonlariyla beraber sayisal dünyada gerçeklenmeye çalisilan modellenmesidir. Sayisal dünya ile belirtilmek istenen donanim ve yazilimdir. Bir baska ifadeyle yapay sinir agi hem donanimsal olarak hemde yazilim ile modellenebilir. Bu baglamda, yapay sinir aglari ilk elektronik devreler yardimiyla kurulmaya çalisilmis ancak bu girisim kendini yavas yavas yazilim sahasina birakmistir. Böylesi bir kisitlanmanin sebebi; elektronik devrelerin esnek ve dinamik olarak degistirilememesi ve birbirinden farkli olan ünitelerin biraraya getirilememesi olarak ortaya konmaktadir.

Yazilim yardimiyla daha kolay kurulabilen yapay sinir aglari, yine yazilimsal olarak çalistirilabilmesi de rahat olabilecek modellerdir. Ancak elektronik devrelerle kurulan yapay sinir agi modelleri dogal olarak yazilim ile kurulan modellere kiyasla daha hizli sonuca ulasabilecektir. Bu sebepten dolayi, yapay sinir aglari günümüzde yazilimsal olarak kurulup, çalistirilip, test edilmekte ve gerekli tüm degisiklikler ve dinamik güncellemeler yapilmakta, ardindan sonuçlara göre karar verilmektedir. Eger elde edilen sonuçlarin basarisi %99'lar ifade edilebiliniyorsa, o zaman gerekli görüldügü takdirde model elektronik devreler üzerine aktarilmaya çalismaktadir. Böylece yapay sinir agi modelleri, gerçek yasama uygulanmak üzere fiziksel bir platform üzerinde hazir hale getirilmis olmaktadir.

Buraya kadar, yapay sinir aginin donanim ve yazilim sahasiyla olan iliskisi gündeme getirilmistir. Simdi ise, yapay sinir aginin yapisindan ve onu olusturan elemanlardan söz edilecektir. Bu yapiyi anlayabilmek için öncelikle biyolojik sinir hücresinden bahsedilmesi gerekmektedir.(Sekil 21.)



Sekil 21. Biyolojik sinir hücre yapisi(Fausset L.,1994)

Dendrit : Görevi diger sinir hücrelerinden iletilen sinyalleri, sinir hücresinin çekirdegine iletme ktedir. Bu yapi basit gibi görünse de günümüzde dendritlerin görevlerinin daha kompleks oldugu yolunda söylemler hakim olan görüstür. Hücrenin çekirdegi ile herbir dendrit arasinda farkli bir iletisim söz konusudur. Bu sebeple bazi dendritlerin etkilesimde agirlikli (dominand) pay sahibi, digerlerinin de pasif (resesif) oldugu gözlenmektedir. Bu ise disaridan alinan sinyallerde seçicilik gibi önemli bir olgunun sinir hücresi tarafindan gerçeklestirilmesi anlamini tasimaktadir.

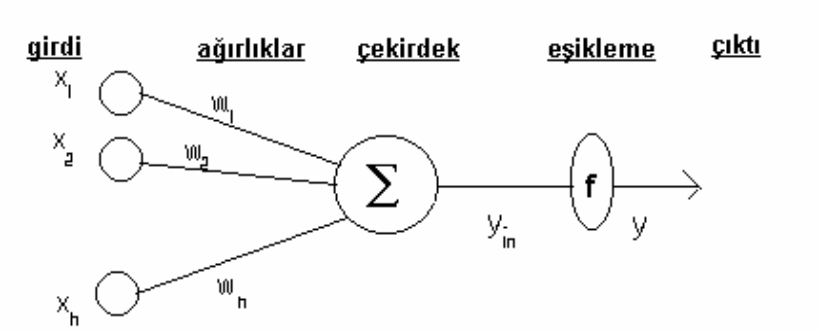
Soma : Dendritler yoluyla iletilen tüm sinyalleri alip toplayan merkezdir. Biyolojik olarak hücre çekirdegi (nükleus) olarak da bilinen yapidir. Çekirdek gelen toplam sinyali diger sinir hücrelerine göndermek üzere, bilgiyi aksona iletir.

Akson: Hücre çekirdeginden aldigi toplam bilgiyi bir sonraki sinir hücresine dagitmakla görevlidir. Ancak akson bu toplam sinyalin ön islemden geçirilmeden diger sinir hücresine aktarilmasina engel olur. Çünkü akson ucunda sinapsis denilen birimlere bilgiyi aktarir.

Sinapsis : Aksondan gelen toplam bilgiyi ön islemden geçirdikten sonra diger sinir hücrelerinin dendritlerine iletmekle görevlidir. Sinapsisin ön islem ile gerçeklestirdigi görev çok önem tasimaktadir. Bu ön islem gelen toplam sinyalin, belli bir esik degerine göre degistirilmesinden ibarettir. Böylece toplam sinyal oldugu gibi degil, belli bir araliga indirgenerek diger sinir hücrelerine iletilmis olunur. Bu açidan, her gelen toplam sinyal ile dendrite iletilen sinyal arasinda bir korelasyon (iliski) olusturulur. Buradan yola çikilarak

"ögrenme" isleminin sinapsislerde gerçeklestigi fikri ortaya atilmis ve bu hipotez, günümüz yapay sinir agi dünyasi için teori haline dönüsmüstür. Yapay sinir agi modelleri üzerinde "ögrenme" bu teoriye dayanilarak, sinapsisler ve dendritler arasinda yer alan agirlik katsayilarinin güncellenmesi olarak algilanmaktadir.

Yukarida biyolojik olan sinir hücresinin elemanlari tanitilmaktadir. Öyleyse yapay sinir agi hücre modeline geçis yapmak gerekmektedir. Yapay sinir hücresi, gerçek biyolojik hücreyle ayni ilkelere dayandirilmaya çalisilmistir. (Fausett L.,1994)



Sekil 22. Yapay sinir hücre yapisi

Sekil 22.'de görünen yapay sinir hücresinin dendritleri x_n ve herbir dendritin agirlik katsayisi (önemlilik derecesi) w_n ile belirtilmistir. Böylece x_n girdi sinyallerini, w_n ise o sinyallerin agirlik katsayilarinin degerlerini tasimaktadir. Çekirdek ise tüm girdi sinyallerinin agirlikli toplamlarini elde etmektedir. Tüm bu toplam sinyal y_{in} ile gösterilmis ve sinapsise esiklenme fonksiyonuna girdi olarak yönlendirilmistir. Sinapsis üzerindeki esikleme fonksiyonundan çikan sonuç sinyali y ile belirtilmis ve diger hücreye beslenmek üzere yönlendirilmistir.

Yapay sinir hücresinin görevi kisaca; x_n girdi örüntüsüne karsilik y çiktisi sinyalini olusturmak ve bu sinyali diger hücrelere iletmektir. Her x_n ile y arasindaki korelasyonu temsil eden w_n agirliklari, her yeni girdi örüntüsü ve çikti sinyaline göre tekrar ayarlanir. Bu ayarlama süreci ögrenme olarak adlandirilir. Ögrenmenin tamamlandiginin belirtilebilmesi için; girdi örüntüleri, w_n agirliklarindaki degisim stabilize olana dek sistemi beslemektedir. Stabilizasyon (duraganlik) saglandigi zaman hücre ögrenmesini tamamlamistir.

Yapay sinir aglari; görevi yukaridaki biçimde belirtilen yapay sinir hücrelerinin birlesiminden olusan katmanli yapinin tümü olarak nitelendirilir. Böylece "m" adet yapay sinir hücresinin katmanli yapisiyla yapay sinir agi modeli kurulmus olunmaktadir.

4.2 Yapay Sinir Aglarinda Ögrenme

Önceki bölümde yapay sinir agi yapis inin islevi kisaca anltilmistir. Bu temel üzerine, yapay sinir hücresinin ögrenme sürecinin açiklanmasi gerekmektedir. Bir yapay sinir hücresi nasil ögrenir ? Ögrenme süreci neye göre belirlenir ? gibi sorularin cevaplari bu bölümde verilmeye çalisilacaktir.

Ögrenmenin ilk adimi aktivasyon olarak nitelendirilebilir. Sinir hücresine giren sinyallerin toplami o hücreyi aktif hale getirebilecek bir degere sahip midir ya da degil midir ? Cevap olarak su verilmektedir: eger toplam sinyal hücreyi atesleyebilecek, esik degerini atalatabilecek kadar yüksek ise o hücre aktiftir (y=1) aksi durumda o hücre pasiftir (y=0). Bu sorunun cevabi yardimiyla yapay sinir hücresinin siniflandirma yapabildigi sonucuna ulasilacaktir. Çok basit anlamda; girdi örüntülerine 1 ya da 0 cevabini vererek siniflandirma yapabilen böyle bir hücre, hangi girdi örüntüsüne 1 hangi örüntüye 0 diyecegi hakkinda karar vermis sayilmaktadir. "Karar vermek" ve "siniflandirmak", ögrenme sürecinin temel yapi taslarini olusturmaktadir.(Fausett L.,1994)

Bir yapay sinir hücresi neye göre girdi örüntüsüne 0 ya da 1 demektedir? Bu noktada sinir hücresinin agirliklari olarak adlandirilan ve Sekil 22. 'de belirtilen w_n degerleri devreye girmektedir. Disaridan alinan herbir girdi örüntüsü (girdi sinyali) ile her defasinda ayarlanan bu agirliklar ögrenmenin gerçeklestiginin temsilcisidirler. Matematiksel bir ifadeyle w_n agirliklari, tüm girdi örüntülerini en iyi temsil etmeye çalisan ve tüm girdi örüntülerinin uzakliklar toplaminin minimum oldugu regresyon egrisinin temsil edildigi geometrik seklin en belirleyici noktalarini olusturmaktadir. Bu sayede girilen örüntüye en dogru olan cevap verilebilmektedir.

Örnegin; x_1=10, x_2=18, x_3=24, x_4=6 olarak belirtilen 4 girdili bir hücrenin baslangiç agirliklari söyle varsayilsin; $w_1=w_2=w_3=w_4$=0,4. Bir baska girdi kombinasyonu da söyle verilsin: x_1=11, x_2=18, x_3=20, x_4=2. Eger hücre iki örüntüyede 1 cevabi vermis ise agirliklari örnegin su sekilde degisime ugratacaktir, x1 yükseldigi için w_1=0,45; x_2 degismedigi için w_2=0,4; x_3 düstügü için w_3=0,35; x_4 düstügü için w_4=0,35. Böylece hücre her iki girdi ile karsilastiginda 1 cevabi verecek sekilde agirliklari (korelasyon matrisi degerleri) ayarladi. Diger bir ifadeyle iki girdi örüntüsünü ögrendi.

Bu teorik olarak belirtilen açiklamanin formülize edilmis hali asagidaki gibidir;

$$y_{in} = \sum_{i=1}^{p} x_i * w_i \qquad (1)$$

(1)' de (Fausett L., 1994) x_i sinyali kendine ait olan katsayi ile çarpilarak toplan sinyale eklenmektedir. Y_{in} seklinde toplanan deger çekirdek tarafindan akson kullanilarak sinapsise gönderilir. Sinapsis gelen toplam sinyal degerini esikleyerek çikti degerini vermektedir.

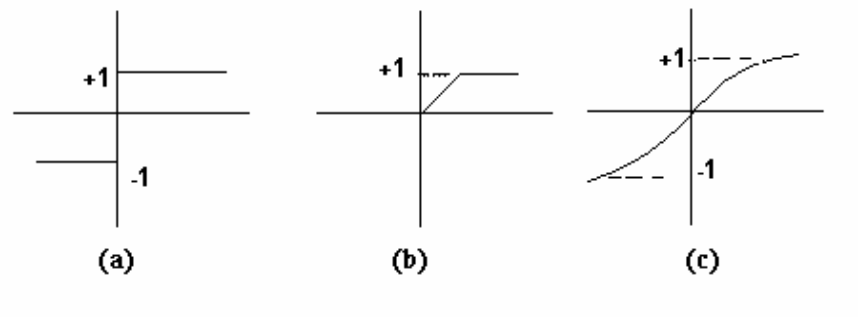
$$y = f(y_{in}) \tag{2}$$

(2)' de "f" olarak betimlenen foksiyon matematiksel herhangi bir fonksiyona denk olabilir. Ancak yapay sinir agi modellerinde temel alinan 3 tip fonksiyon vardir(Sekil 23.)(Lipmann R.P.,1987):

a. Hard Limiter fonksiyonu : Bu fonksiyon girdi örüntülerinin degerlerine göre ayrik (discrete) sonuç elde etmek için kullanilir. Bir baska ifadeyle girdi ya +1 sonucu verir ya da -1. Baska bir ihtimal söz konusu degildir. Böylece kesin bir limit alinmasi saglanmistir.
b. Threshold fonksiyonu: Bu fonksiyonda Hard Limiter fonksiyonuna benzemekle beraber, girdi örüntüsünün toplam degerine belli bir esik degerine kadar dogrusal (lineer) artan degerlerle cevap vermektedir. Üst limite ulasildiginda ise (esik degeri, threshold) artik cevap ayrik olarak yine kesinlik göstermektedir. Artan bir egilim göstermez.
c. Sigmoid fonksiyonu: Bu fonksiyonda girdi örüntüsüne devam eden, süreklilik gösteren (continuos) cevaplar verilmektedir. Cevaplar kesinlikle ayrik degildir. Bu sebeple sigmoid fonksiyonu yaygin bir kullanima sahiptir. Çünkü hassas degerlendirmelerin kullanilacagi problemler için uygulanmasi en uygun olan fonksiyonu temsil etmektedir. Sigmoid fonksiyonu yerine yine süreklilik arz eden tanjant fonksiyonlarida ya da benzeri fonksiyonlarda kullanilabilir. Önemli olan fonksiyonun türevinin alinabilecek bir fonksiyon olmasidir.

$$w_{i,yeni}(t+1) = w_{i,eski}(t) + (m * \left[d(t) - f(y_{in})\right] * x_i(t)) \tag{3}$$

(3)' de(Fausett L., 1994) esikleme sonucu elde edilen cevap ile $f(y_{in})$ olmasi beklenen deger olan d(t) arasindaki yanilma payi μ ögrenme katsayisi ile ve girdi sinyaliyle çarpilip yeni agirligi belirlemek için eski agirlik ile toplanir. Bu formül yardimiyla, yapay sinir hücresinin ögrenmek için isleme koydugu güncelleme süreci açiklanmis olmaktadir.



Sekil 23. A) hard limiter B) Threshold C)Sigmoid

Sinir hücrelerinin yapay modelleri için, ögrenmenin"w" üzerinde gerçeklesti teorisi yapay sinir aglari modelleri için kabul edilmis bir teori olup bu çalismada da teori aynen korunacaktir. Bu açiklamanin isigi altinda, ögrenme sürecinin kaç degisik yöntemle yapilabileceginin vurgulanmasi gerekmektedir.

Ögrenme süreci 2 ana prensip gözetimi altinda gerçeklestirilmektedir; "ögretmenli ögrenme(supervised learning)" ve "kendi kendine ögrenme(unsupervised learning)" (Fausett L., 1994).

Ögretmenli ögrenmede (Supervised learning), yapay sinir aginin disaridan etki ile egitilmesi söz konusudur. Bu tip bir ögretimde, girilen $x_i(t)$ degerlerinin ne tür bir çikti vermesi gerektigi önceden bilinmekte (d(t)) ve yapay sinir agi agirliklari bu korelasyona göre güncellenmektedir. Bu ögretimde temel bilinen "d(t)" ile yapay sinir aginin verdigi "y" sonucu arasindaki hatanin "w_i" agirliklarina ögretilmesidir. Kabaca bir örneklendirme yapilmak istenirse; insani tarif ederken su özellikler dikkate alinsin: boy, kile, saç rengi, ayakkabi numarasi. Bu kriterlere göre bir ögrencinin futbol oynayabilir ya da oynayamaz seklinde ayrimi yapilmak istensin. Bir ögretmenin verdigi bilgiler isiginda her ögrenciye "oynayabilir", "oynayamaz" bilgisi atansin. Eger yapay sinir agimiz 100 kisilik böyle bir grubu ögrenseydi ve ardindan 100 kisinin içinden herhangi biri yapay sinir agina boyu, kilosu, saç rengi, ayakkabi numarasi verilerek sorulsaydi, yapay sinir agi o ögrencinin oynayip oynayamayacaginin cevabini verecekti. Çünkü yapay sinir agi her bir özellik için (boy kilo vs…) agirlik güncellemesini (3) nolu formule göre yapacak ve ögretmenin gösterdigi her ögrenci ve ona ilistirilen karari arasindaki korelasyonu "w" agirliklari üzerinde güncelleye güncelleye ögretecekti. Ag burada ögrenmesini bir ögretmenle yapmistir.

Kendi kendine ögrenme(Self, Unsupervised Learning) ise yapay sinir aginin disaridan herhangi bir etki olmaksizin aldigi bilgileri kendi içerisinde kiyaslama yaparak siniflandirmasi ile olusan ögrenme sürecini belirtmektedir. Bunu yapabilmek için yapay sinir agi ilk aldigi örnegi (ya da örnekleri, kaç sinifa ayirmak istiyorsa o kadar örnek alinabilir) bir sinif olarak ilan eder. Ve geriden gelecek tüm girdi örüntülerini o sinifa

benzetmeye çalisir. Bu sekilde, tüm girdi örüntülerini kendi aralarinda benzeyip benzememelerine göre ayirt edecektir. Elbette burada siniflandirma yaparken hatali bazi cevaplar üretmek kaçinilmazdir. Ancak girdi örüntüleri sisteme çok defalar ögrenim için beslenecek olunursa, elde edilen sonuç %4 gibi bir yanilma oranina kadar düsecektir. Yine kabaca örneklendirmek gerekirse, 100 kisilik bir sinifin boy, kilo, saç rengi ve ayakkabi numarasi yine girdi örüntülerinin degerlerini olustursun. Yapay sinir agi ilk gelen kisiyi bir sinif olarak ilan edecek ve ardindan sisteme beslenen tüm girdileri bu buna benziyor ya da benzemiyor diye ayirt edecektir. Bu ayrimi yapabilmesi için 100 kisilik grubun degerleri 1000 kere sisteme tekrar tekrar beslendigi takdirde, yapay sinir agi en sonunda ögrenmesini tamamlayacak ve 2 sinif ögrenmis olacaktir. 1.girdiye benzeyenler ve benzemeyenler. Adini koyamadigi bu siniflar yadrimiyla "hanimlar" ve "beyler" gibi olarak rahatlikla sinif ayrilmis daha dogrusu bu iki sinif basariyla ögrenilmis olunacaktir. Böyle bir ag modeline yabanci bir kisi degerleri verilirse, sinir agi bu kisiye dogru cinsiyetini söylecektir.

Yapay sinir aglari temelde bu iki ögrenme metodundan ya birini ya da hibrid denilen karma modeli kullanir. Ancak ögrenmede "w" ile gösterilen agirliklarin güncellenmesinin bu metodlara göre yapilmasi, ögrenme ile ilgili baska etkin rol oynayan etmenlerin var olmadigi anlamina gelmemektedir. Güncellemede kullanilan formül (3)'te yer alan "µ" ile sembolize edilmis bir ögrenme katsayisi da ögrenmenin sürecini etkileyen bir faktördür.

"µ" katsayisi ögrenmenin süresinin ve dogrulugunun iliskisini düzenleyen önemli bir degiskendir. Uzayda bir merkez düsünülürse ve bir baslangiç noktasindan interpolasyon yöntemiyle merkeze yaklasilmak istense, merkeze dogru atilacak her adimin büyüklük degeri önem tasir. Örnegin çikis noktasindan merkeze dogru 5cm atlayarak gidilmeye çalisilsa merkez belki de 3 adim sonrasinda ulasilmanin ötesinde üstünden atlanip geçilmis bile olacaktir. Eger adim uzakligi 1cm seklinde belirlenecek olunursa, merkeze 14 adim sonra tam olarak ulasilmis olunacaktir. Örnekten görüldügü gibi, "µ" degiskeninin degeri büyük alinirsa ögrenme kaba ve kisa süreli, küçük alinirsa hassas ve uzun süreli olacaktir. Bu noktada "µ" degeri mümkün oldugunca optimum tutulmalidir, ne büyük ne de çok küçük alinmalidir. Çünkü büyük alinan deger yüzünden istenilen sonuca ulasilamayacak, çok küçük alindigi takdirde ise istenen hedefe çok düsük bir performans ile ulasacaktir. Bunun için yapay sinir aglarinin ögrenme algoritmalarinda genelde belli bir iterasyon (döngü sayisi) geçtikten sonra, "µ" degeri azaltilir. Böylelikle, baslangiçta algoritma büyük adimlar atarak hedefe dogru yönelir, belli bir süre sonra adimlarini küçültmeye baslar. Bu sayede algoritma hizli bir biçimde sonuca, hassas bir yaklasimla ulasir. Bunun gerçeklestirilmesi için "µ" degeri belli bir sabir katsayi ile çarpilmaktadir.(Kung S.Y.,1993)

$$m = m * sabit \qquad (4)$$

Ögrenmenin baska bir kritik ögesi verilerin büyüklügü ve iterasyon sayisidir. Yapay sinir aginin çözüme ulasabilmesi için problemin karakteristigini yakalayabilecek kadar girdi örüntüsüne (veri) sahip olmasi gerekir. Örnegin Türkiye'nin yagmur yagisini tahminleyebilmek için 30 günlük verilerin kullanilmasi, o

problemin çözüme ulasabilmesini engelleyen bir etken faktör olacaktir. Bu yüzden problemin çözümünden önce kullanilacak veriler üzerinde istatistiksel bir yorumlamanin yapilmis olmasi, yapay sinir aglarinin o verileri kullanarak basariya ulasmasi için yararli olacaktir. Kullanilacak veri popülasyonun büyüklügü her probleme göre degisik olabilir, bu nedenle yapay sinir aglarinin basariya ulasmasi için gereken iterasyon (döngü) sayisi her bir probleme göre degisebilir. Bunun yanisira yapay sinir aginin problemi daha iyi ögrenebilmesi, cevap üretirken daha hassas derecelerde dogru cevap üretebilmesi için problemin kendi içinde de iterasyon sayisi degistirilebilir. Örnegin, Türkiye'nin yagisinin modellenebilmesi için 10 senelik veri, yapay sinir agina bir kere beslenirse verecegi cevap çok kaba ve yuvarlamali olacak iken, 1000 kere ayni veriyi beslemek ile daha hassas derecelere kadar ögrenmesi saglanmis olacaktir. Bu noktada iterasyon sayisi problemden probleme ve problemin kendi içindeki hassasiyete göre degisebilecektir. Çok fazla döngü performansi azaltip sonuca ulasmayi uzun bir zamana yayarken, az sayida döngü de sonuca ulasmayi engelleyecek kadar kaba sonuçlar üretmeye neden olacaktir. Bu yüzden yapay sinir aglari, her bir problem için uygulanirken iterasyon sayisi deneme yanilma yöntemiyle tesbit edilmeye çalisilir.

Ögrenmenin üzerinde etkin rol oynayan bir baska faktör, o yapay sinir agi üzerinde kullanilan katman sayisidir. Burada kisaca su söylenebilir. Modelden modele katman sayisi degisiklik gösterse de, su an yaygin olan görüse göre; 3 katmandan olusan bir yapay sinir agi modelinin en karmasik problemlere dahi yeterli oldugudur(Ünal N., 1994). Ancak bu bir kisitlama degildir, 2 katmanli ya da 4 katmanli yapay sinir agi modelleriyle problemler çözülemez anlami içermemektedir. Bu 3 katman sirasiyla; girdi, sakli, çikti katmani olarak adlandirilirlar. 5 katmanli bir yapida 1.katman girdi, 5.katman çikti ve arada kalan katmanlar sakli katmanlar olarak adlandirilirlar.

Ögrenmede etkili olan bir diger faktör, her katman üzerinde kullanilan sinir hücresinin sayisidir. Örnegin Sekil 24.'te girdi katmaninda 30, sakli katmanda 6 ve çikti katmaninda 8 tane sinir hücresi yer almaktadir. Ve buradaki sayilar probleme giris için kullanilan tamamen kullanicinin kurgusuyla belirlenmis sayilari ifade etmektedir. Bu yüzden her katmanda olmasi gereken hücre sayisi gibi sabit bir belirleme söz konusu olmayip ayni iterasyon sayisi gibi bu da deneme yanilma yöntemiyle tesbit edilmektedir. Ancak yine en iyileme açisindan su vurgulanabilir; çok sayida yapay sinir hücresi yapinin karmasik ve kompleks fonksiyonlarla çalisip düsük performans sergilemesine, az sayida hücre ise problemin ögrenilmemesine neden olacaktir.

Tüm buraya kadar ögrenmede etkin olan unsurlarin yapay sinir aglari için tasidigi önemin aktarilmasi yer almistir. Bunlara ek olarak, ögrenmeyle dogrudan iliskili olan kapasiteden söz etmek gerekmektedir. Ögrenmede bahsedilen yukaridaki tüm etmenler kullanilarak yapay sinir aginin ne kadar ögrendigi tesbit edilir. Bunun ölçütü olarak ögretilen populasyondan tek tek girdi örüntüleri alinarak yapay sinir agina sorulur. Eger yapay sinir agi bunlari dogru olarak cevaplandirabiliyorsa, bu dogru cevaplarin orani tutulur. Isletim sistemi

teorisindeki “hit ratio, vurus yüzdesi” mantiginin aynisi yapay sinir aglari üzerinde kapasiteyi belirtmektedir. Örnegin Hopfield modelinde, 100 bitlik girdi örüntüleri kullaniliyorsa, Hopfield kapasitesi 15 adet girdi örüntüsünü tutar denmektedir. Bir baska deyisle, Hoprfiel modelinde 16. Girdi örüntüsünün yapay sinir agina ögretilmesi mümkün degildir. Buna kiyasla, backpropagation modelinde ise %99 ulasan ögrenme kapasitesine ulasilmaktadir.(Fausett L.,1994)

4.3 Sorgulama

Yapay sinir agi modellerinin temel iki islevi vardir. Ögrenmek ve ögrendiginin sorgulamasi yapildiginda buna yanit vermek. Ancak dikkat edilmesi gereken nokta, yapay sinir agi modellerine daha önce hiç ögrenmedigi bir örüntünün sorulmasiyla, yapay sinir aginin bu örüntüyü daha önce ögrendigi örüntülerden birine benzeterek cevap vermeye çalismasidir. Bu çalisma prensibi sayesinde, yapay sinir agi modelleri hata-toleransli “ fault tolerant ” mekanizmalara olarak bilinirler.

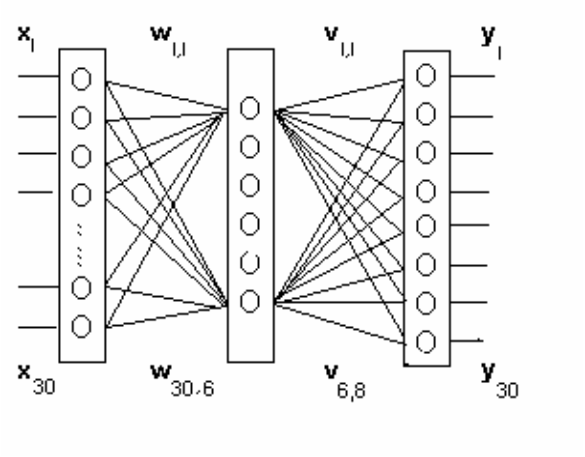
Hata-toleransli yapisina örnek vermek gerekirse, yüz tanima ile ugrasan yapay sinir agi modelleri vardir. Bu sistem yüz tiplerini ögrenmeye yönelik çalismaktadir, ve çok iterasyon yaparak bir kisinin yüzündeki bazi karakteristikleri ögrenmek suretiyle, insan yüzünü tanimaya çalismaktadir. Eger bu sistem bir kisinin yüzünü ögrendiyse, yüzün bir parçasinin kapatilarak sorgulanmasi, sistemin yine dogru cevap vermesini etkilemeyebilir. Iste girdi örüntüsünün belli bir miktari yok olsa dahi, yapay sinir aginin o girdi örüntüsüne dogru cevap veriyor olmasi, yapay sinir aglarinin hataya olan toleransli yaklasimini göstermektedir.

Sorgulama esnasinda kullanilan çok basit teknikler vardir. Bu tekniklerden bazilari; Hamming Distance (HD), Eucledean Distance (ED), Partitioned Generalized Eucledean Distance (PGED) olarak isimlendirilen uzaklik ölçüm teknikleridir(Cinsdikici M.,vd.,1997,ISCIS XII). Girilen örüntünün saklanan(ögrenilmis) olan örüntülerle matematiksel olarak uzakliklarini bulmak için kullanilan bu teknikler sayesinde, girdi örüntüsüne en yakin olan ögrenilmis örüntü cevap olarak yapay sinir agi tarafindan disari verilmektedir.

Sorgulamada baska bir metodolojide (1) ve (2) formüllerinin uygulanip sonuçta kimlerin “y” çiktisinda aktif, kimlerin ise pasif olduklarina göre girdi-ögrenilmis ikilisi arasinda yakinlik karari verilmesidir. Burada yapay sinir aginin kendi aktiflik fonksiyonu disinda HD, ED, PGED gibi harici bir foksiyon kullanilmadigi için, bu tip sorgulama yapan yapay sinir aglari modelleri sorgulama sonucuna daha hizli ulasmaktadirlar. (Cinsdikici M., vd., 1997)

Yapay sinir aglarinin hepsi ayri ayri sorgulama teknigi kullaniyor olabilirler. Bazilari ise bahsi geçen sorgulama tekniklerinin hibrid(ikili) seklinde birlikte kullanimini da uyguluyor olabilirler. Bu noktada sorgulama

tekniginin belirlenmesi durumu modeli gelistiren kisinin tercihine ve problemin kararkteristigine birakilmistir. Ancak basit bir genelleme ile su vurgulanabilir; HD, ED, PGED gibi matematiksel uzaklik hesaplama fonksiyonlari görüntü isleme problemlerinde, örüntü tanima problemlerinde sik kullanilmaktadir. Optimizasyon (en iyileme) problemlerinde ise (1) ve (2) formüllerinin birarada kullanilarak hücre aktivasyonuna göre sorgulamanin saglandigi görülmektedir.



Sekil 24. Girdi katmani 30 hücreli, sakli katmani 6 hücreli, çikti katmani 8 hücreli bir sinir agi.

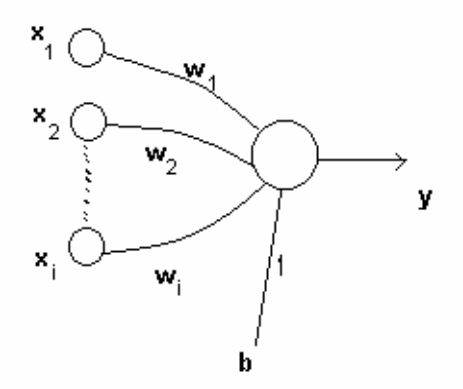
4.4 Temel Yapay Sinir Agi Modelleri

Daha önceki bölümlerde yapay sinir aglari için gereken temel elemanlar ile ögrenme ve sorgulama sistemi irdelenmistir. Bu bölümde ise varolan yapay sinir agi modellerinin bu tez kapsami içinde önem tasiyanlari açiklanacaktir.

4.4.1 Hebb Net

Yapay sinir agi modellerinin ilk bilinen tipidir. Ögrenme metodolojisinde Hebb kuralini kullandigi için bu adi almistir. Hebb'in koymus oldugu kural çok basittir; eger iki hücre ayni aktivasyonu gösteriyorsa arasindaki bagin kuvvetlenmesi gerekir. Eger ikisinin aktivasyonu farkli ise aralarindaki bag degismez ya da azaltilmalidir. (Fausett L.,1994)

Yapisi itibariyle ileri beslemeli olan bu model, tek katmanli bir yapi ile olusturulmustur. Bu yapi girdi hücreleri ve onlarin bagli oldugu çikti hücresinden ibarettir. Girdi hücrelerinin birbirleri arasinda baglantilari söz konusu degildir. Hebb net'in mimarisi Sekil 25.'te gösterildigi gibidir.



Sekil 25. Hebb net'in mimarisi

Ögrenme süreci daha öncede belirtildigi gibi hücreler arasindaki iliskinin kuvvetlendirilip, azaltilmasiyla saglanmaktadir. "w" agirliklari bagli oldugu iki hücrede ayni aktiflikte ise arttirilir, farkli ise azaltilir yahut girdi örüntü tipine göre degistirilmez. Burada girdi örüntüsünün kullandigi veri tipi de önemlidir. Eger girdi örüntüsü olarak ikili (binary, 0 ve 1) tipi kullaniliyorsa, o zaman "w" ler farkli aktiflikteyken degisime (azaltilmaya) ugratilmazlar. Ancak kullanilan örüntü iki kutuplu (bipolar) ise o zaman "w" ler farkli aktifliklerde azaltilirlar. Buna göre ögrenme formülü asagida verilmistir.

$$w_i(yeni) = w_i(eski) + x_i * y \qquad (5)$$

Önemle vurgulanmasi gereken husus ise (5) formülünde yer alan "$x_i * y$" ifadesinden yola çikilarak bir hata ile karsilasildigidir. Eger binary (0,1) tipli örüntüler kullaniliyorsa "x_i ve y"'nin aktivasyonlari zitlik gösteriyorsa "w_i"'de degismeyecektir. Bu dogrudur. Ancak x_i ve y ikisi birden pasif aktivasyonda ise (0,0) o zaman "w_i" yine degismemektedir. Oysaki "w_i"nin kevvetlendirilmesi gerekmektedir. Bu sebeple girdi örüntülerinin bipolar (iki kutuplu) alinmasi Hebb net için en dogru olan girdi örüntüsü veri tipi olacaktir.

Hebb net'in algoritmasi asagidaki gibidir; (Fausett L.,1994)

Adim 0: Tüm agirliklari baslangiç degerlerine ata

$w_i = 0$ i=1,2,3,…..n

Adim 1: Herbir girdi örüntüsü (s) ve o örüntünün istenen sonucu (t) gereken deger ikilisi için

Adim 2-4 arasini tekrarla.

Adim 2: Herbir girdi elemanini girdi örüntüsüden al:

$x_i=s_i$ i=1,2,3,…..n

Adim 3: Çimasi gereken sonu ata

y=t

Adim 4: Agirliklari ve denge unsurunu ayarla

$w_i(yeni)=w_i(eski) + x_i*y$ (i=1,2,3…..n)

b(yeni)=b(eski) + y

Hebb net'in basit yapisi güncel olarak basit siniflandirma problemlerinde kullanilmaktadir. Siniflandirma rol oynayan "y" (1) formülündeki gibi hesaplandiginda çikan sonuç ögrenmenin tamamlandigini ve girdinin hangi sinifa ait oldugunu belirtir. En basit örnekleri AND ve OR kapilarinin Hebb net ile simüle edilmesidir. Örnegin AND için ;

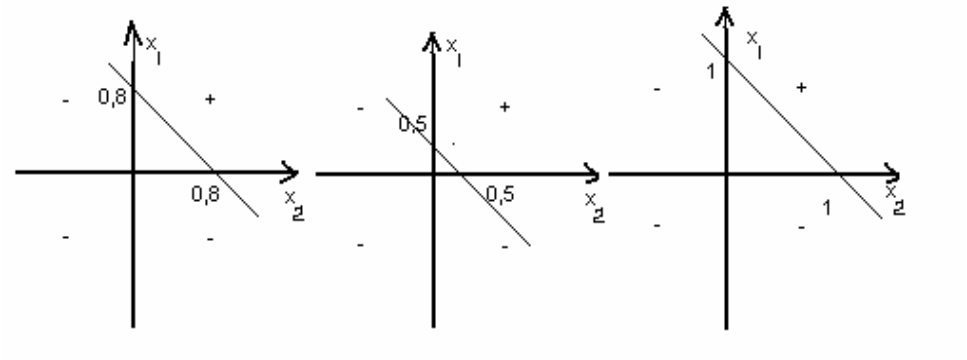
s_1=(1,1) t_1=1 b=1,

s_2=(1,1) t_2=1 b=1,

s_3=(1,1) t_3=1 b=1,

s_4=(1,1) t_4=1 b=1,

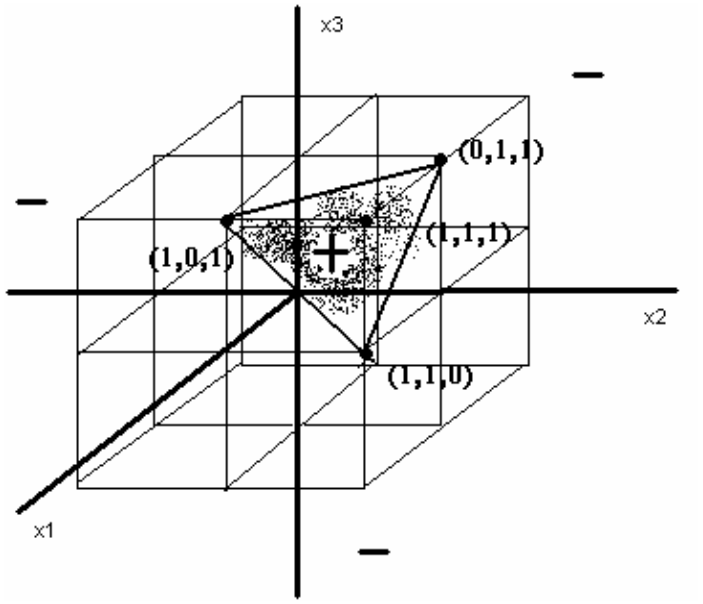
Bu girdilerin için Hebb net ögrenmesini yaptiginda elde edilen agirliklar w_1=2, w_2=2, b=-2. Hebb net bu durumdayken (-1,1) sorulsa cevap olarak (-1) deger alinacaktir. Bu ise 2 boyutlu uzayda girdi örüntülerini bir dogru ile iki parçaya ayirmak bir baska degisle siniflandirma yapmasi anlamina gelmektedir. Hebb net'in sayesinde (-1) ve (+1) olan girdiler vardir. Bu Sekil 26.'daki açikça ifadesini bulmaktadir.



Sekil 26. AND kapisinin modellenmesi ile elde edilebilecek 2 boyutlu düzlem ve (+1) (-1) siniflari.

Sekil 26.'dan anlasilacagi üzere Hebb net (+1,+1) (-1,+1) (+1.-1) (-1,-1)'i dogru biçimde siniflandiracaktir. Çünkü bir tek çikti hücresi bu sistemde (-1), (+1) cevabi verebilecektir. Ancak dogrunun düzlemdeki, konumu paralel olarak iterasyon sayisina bagli olarak degisiklik gösterebilir. Önemli olan Hebb net'in AND kapisini basariyla benzetilebildigidir. (Fausett L.,1994)

Eger bizim girdi örüntümüz 3 girdili (x_1,x_2,x_3) olsaydi ve Hebb net'ten yine AND kapisini simüle edilmesi istenseydi, Hebb net 3 boyutlu girdi düzlemini yine basarili bir biçimde 2 sinifa ayirabilecekti. Bu üç boyutlu düzlemin 2 ayri sinifa ayrilmasi isleminin sonucu ise Sekil 27.'de belirtilmistir.



Sekil 27. Hebb net ile 3 girdili AND kapisinin simüle edilmis(siniflandirilmis) hali.

Sekil 27.'de de rahatlikla görülecegi üzere 3 boyutlu bir girdi kümesi, bir araci düzlem tayin edilerek yine 2 parçaya bölünebilmistir. Kübün (1,1,1) noktasi ile üçgen düzlem arasi kalan hacimli bölge (+1) sinifini, geri kalan büyük hacimli bölge ise (-1) sinifini belirtecek sekilde basarili bir siniflandirma yapilmistir.

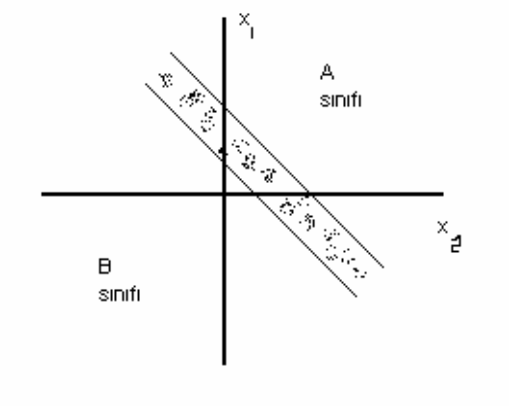
4.4.2 Perceptron

Hebb net modelinden sonra ortaya konan perceptron modeli, bugünkü yapay sinir aglari için önemli bir temel olusturmaktadir. Çünkü Hebb net'in ögrenme kavrami yerine simdi esikleme fonksiyonundan geçirilme kavrami gelmektedir. Bu model yardimiyla, ögrenmede esiklemenin rolü kabul edilmektedir. Sekil 23.'te

yer alan herhangi bir fonksiyonun kullanilmasini öneren bu model, temelde kendine “hard limiter” tipindeki esikleme fonksiyonunu referans olarak kabul etmektedir. Perceptron modeli de yine Hebb netin mimarisini kullanmaktadir.(Sekil 25.) (Fausett L.,1994)

Perceptron modelinde en önemli faktör esik degeridir. Saptanacak olan esik degeri problemin karakteristigine göre belirlenebilir. Esik degeri yardimiyla perceptron modeli basarili bir siniflandirma yapabilmektedir. Hebb net’te yer almayan bir kavram olan “kararsiz (geçis) bölge” diye adlandirilan çözüm bölgesinide önemseyen perceptron, hebb nete göre siniflandirmayi daha dikkatli yerine getirmektedir. Bilindigi gibi Hebb net çözüm uzayini 2 parçaya bir bicak gibi bölmekte ve bölümlemeyi saglayan dogrunun (ya da düzlemin) bir yanina A sinifi(+1) diger yanina B sinifi(-1) seklinde esikleme yapmaktadir. Oysa ki perceptron yine çözüm uzayini iki ayri parçaya ayirmakla beraber, A ve B sinifina ait özelliklerin karma seklini üzerinde tasiyan nötr sahayi da ögrenebilme yetenegine sahiptir.

Perceptron yapisinda kullanilan döngü (iterasyon) sayisinin yükseltilmesi sonucunda çözüm için daha kararli bir yapiya ulasilmaktadir. Bu sayede A sinifi ve B sinifi daha net konumlandirilabilinecek, arada kalan nötr bölge (kararsiz bölge) daha inceltilebilecektir. Sekil 28.’de perceptron hücresinin belli bir iterasyon sonunda 2 boyutlu girdi (x1,x2) örüntülerinin ögrenilmesi sonucunda elde edilen siniflandirmayi görmekteyiz.



Sekil 28. Perceptron modelinin kullanilmasiyla elde edilen siniflandirma.

Perceptron yapay sinir agi modelinin algoritmasi asagidaki gibidir, (Fausset L., 1994)

Adim 0: Tüm agirliklara baslangiç degeri ata.

$w_i=0$, $b=0$

Ögrenme katsayisini küçük bir degerden baslat.

$0 < \mu \leq 1$

Adim 1: Belli bir iterasyon sayisi kadar 2-6. Adimlar arasini tekrarla

Adim 2: Her girdi örüntüsü(s) ve ona karsilik beklenen sonuç(t) için 3-5. Adimlari tekrarla

Adim 3: Girdi degerlerini girdi örüntüsünden al.

$x_i = s_i$

Adim 4: Perceptrona gelen toplam sinyali hesapla

$y_{in} = b + \Sigma x_i w_i$

Aktivasyonu hesapla

$y = f(y_{in}) = 1$ (eger $y_{in} > \Phi$) ya da 0 (eger $-\Phi \leq y_{in} \leq \Phi$) ya da -1 ($y_{in} < -\Phi$)

Adim 5: Eger hata varsa bu hatayi ögren buna göre agirliklari güncelle;

Eger ($y \neq t$) ise

$w_i(yeni) = w_i(eski) + \mu * t * x_i$

$b(yeni) = b(eski) + \mu * t$

Eger ($y = t$) ise

O zaman hata yoktur. Degisime gerek kalmamistir.

Adim 6: Döngü sonunu kontrol et.

Görüldügü üzere perceptron algoritmasinda μ katsayisi kullanilmistir. Katsayinin kullanimi için bu çalismanin "yapay sinir aglarinda ögrenme" basligina bakiniz. Burada μ degeri baslangiç olarak 0.6 gibi bir deger olabilir. Bu sabir sayi bir çok algoritmanin baslangiç için kullandigi alisilagelmis bir sayidir. Ancak bu kabullenme bir zorunluluk degildir. μ degeri 0 ile 1 arasi küçük herhangi bir deger olabilir.(Fuasset L.,1994)

Yine algoritmada Φ ile gösterilen esik degeri üzerine "yapay sinir aglarinda ögrenme" basligina bakilmasi açiklayici olacaktir. Bu deger problemden probleme farkli alinabilir.

Perceptronun ögrenme algoritmasi "hata" üzerine dayandirilmis bir ögrenme metodolojisidir. Eger sistem hata ile karsilasiyorsa ögrenmektedir. Ne kadar hata yapilirsa o kadar ögrenme söz konusudur. Baska bir ifadeyle perceptron hatalari ögrene ögrene, sis temi en az hataya indirgemeyi hedeflemektedir. Güncel yasamda beynin bir ögrenme stili olarak bu yaklasimi kullanmaktadir(Lippman R.P.,1987). Örnegin bir ögrenci hep dogru yaparsa bu onda ögrenmeden çok aliskanlik halini alan bir mekanizmaya dogru gidisi saglarken, beyin bir daha o isle mesgul olmaz ve refleks haline gelen islemi artik omurilik gibi yan sistemler götürürler. Oysa ki beyin o hatayi düzeltmeye çalisacak, ve her hatali durumda hatayi biraz daha azaltmaya çalisacaktir. Bu ise beynin hata yapara k ögrenmesini saglamaktadir.

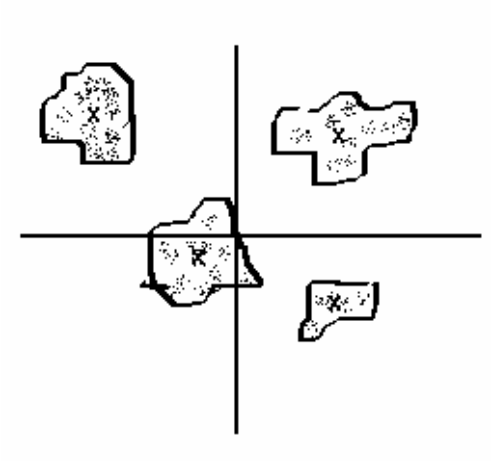
4.4.3 Khonen'in Self Organizing Feature Map (SOM) Modeli

Khonen'in sundugu ileribeslemeli olan bu model (Khonen T.,1984) daha önceden sunulmayan ve sunuldugu andan itibaren büyük ilgi odagi haline gelen bir mimari olmustur. Çünkü Khonen beynin çalisma

stilinin gerçekte bir “associative memory” yani “iliskisel bellek” karakteristiginde oldugunu vurgulamistir. Khonen bu teorisini iki tipte karsimiza çikan iliskisel bellek tipleriyle pekistirmistir: otomatik iliskili(autoassociative), zit iliskili (heteroassociative). Iliskisel bellek tiplerinde temel yapi, iki nesneyi birbiriyle iliskilendirerek ögrenmektir.

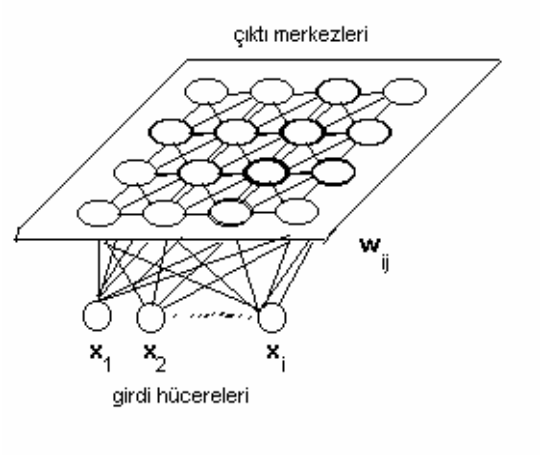
Khonen’in modeline dayanak olan bu iki tip iliskisel bellek modeli, iki nesne arasindaki bagin kuvvetliliginden yararlanmaktadirlar. Ögrenme algoritmalari ise “ögretmenli ögrenme” (supervised learning) temeline dayanmaktaydi. Bu ögrenme modeline güncel yasamdan örnek getirmek gerekirse; muz meyvesi ile kokusunun iliskisi, limonla eksimsi tad iliskisi yeterli olacaktir. Iliskisel belleklerin çalisma prensipleri olarak, örnekte verilen limon girdi örüntüsü ve çikmasi gereken sonuç ise eksilik olacaktir. Ve yapi bu iki ayri nesneyi (meyve - tad) iliskilendirerek ögrenmis olacaktir.

Khonen modeli için çikis noktasi olarak bu temeli kullandi. Ancak “self” yani kendi kendine ögrenmeyi algoritmasi için temel olarak benimsedi. Ayrica Khonen kurdugu yapay sinir agi modellerinde çok önemli bir baska kavramdan daha söz etti: “komsuluk”. SOM yapisinda “cluster” adi verilen grup merkezleri vardi. Eger popülasyonda iyi temsilciler seçilebilirse, o popülasyon seçilmis olan temsilcilerle ifade edilebilirdi. Ya da bir baska deyisle “m” adet elemandan olusan popülasyon (girdi uzayi) “n” degisik sinifa ayrilabilirdi. “n” degisik sinif ise grup merkezleri tarafindan temsil edilebilirdi. Sekilsel olarak bunun ifade edilmesi Sekil 29.’da yer almaktadir. (Kung. S.Y., 1993)

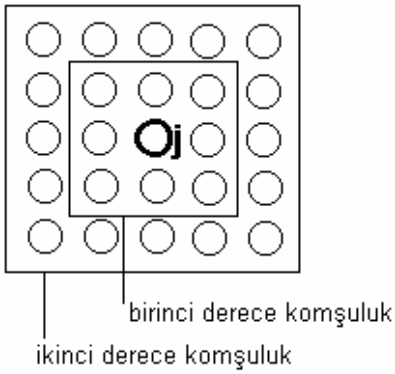


Sekil 29.: Khonen’in SOM modeliyle uzayda birçok merkez noktasi mevcuttur.

Khonen gelistirdigi yapay sinir agi modeli için Sekil 30.’daki mimariyi temel aldi:



Sekil 30: Khonen'in 2 katmanli (girdi, çikti) SOM ileribeslemeli ag yapisi.



Sekil 31: Khonen çikti katmanindaki merkezleri güncellerken komsuluk derecelerini de hesaba katmistir.

Khonen gelistirdigi mimaride, tamamen kendi kendine ögrenmeyi hedefledigi için ele aldigi "k" adetlik girdi popülasyonunun ilk "n" adedini "n" cluster (öbek, merkez noktasi) olarak yerlestirir. Ve tüm popülasyon elemanlarini bu merkez noktalarina göre ögrenir. Ögrenmek için alinan girdi, tüm cluster'lar ile kiyaslanir ve girdi hangi cluster'a (gruba) yakin görünüyorsa oraya adapte edilir (güncellenir). Ve her nokta yakin buldugu grup merkezini kendine dogru çekerek grup merkezlerinin daha iyi grup temsilcisi olabilmesini saglamaya çalisirlar. Eger bu islem 100, 1000, 10000 iterasyon ile tekrarlanirsa, görünen sonuç; cluster olarak belirlenen grup merkezlerinin popülasyonu en iyi temsil eden grup merkezleri oldugu görülecektir. Bir baska deyisle, her grup merkezi örnek uzayin ayri ayri siniflarini temsil edecektir. Bu sayede girdi örüntüleri kendilerini ait olduklari sinifin merkezine yakin bulacaklardir. Elbette ki "komsuluk" kavrami sayesinde birbirine yakin olan siniflar yine birbirleriyle iliskili hareket edeceklerdir.

Khonen'in bu ögretisi ve gelistirdigi yapi çok basarili olmustur. Günümüzde birçok ayri problemi çözmek için kullanilir hale gelen Khonen'in SOM yapisi algoritmasinin kolayligi ilede dikkat çekmektedir. Algoritmada önemli olan girdi örüntüsünün hangi cluster'a daha yakin oldugunun tesbiti için kullanilan matematiksel uzaklik fonksiyonlaridir. Bu uzaklik ölçümü için HD, ED ya da PGED gibi fonksiyonlardan herhangi biri rahatlikla kullanilabilir. Algoritma bu fonksiyonlari kullanarak hangi grup merkezinin girdiye daha yakin oldugunu buldugu an bulunan cluster "winner, kazanan" olarak isaretlenir. Ve "winner take all, kazanan hepsini alir" prensibiyle kazananin agirlik degerleri "w_i" güncellenir. Elbette bu islem yapilirken "winner" olarak isaretlenen merkezin komsularida daha az bir kuvvetle "winner" olan merkeze dogru çekilir. Bu da komsularin birbirine yakinliginin devam etmesi için gereklidir.

SOM yapisinin algoritmasi asagida belirtilmistir;(Khonen T.,1984)

Adim 0: Agirliklara baslangiç degerlerinin rastgele küçük sayilarla atamasini yap.

$0 < w_i < 1$

Ögrenme katsayisinida ata

$0 < \mu < 1$

Adim 1: Yeterli iterasyona ulasilincaya dek 2-8. Adimlar arasini tekrarla

Adim 2: Herbir girdi örüntüsü için 3-5. Adimlar arasini tekrarla

Adim 3: Herbir cluster (j) için uzaklik hesapla

$$uzaklik(j) = \Sigma (w_{ij} - x_i)^2$$

(eger ED fonksiyonu ise bu HD,PGED de olabilir)

Adim 4: Minimum uzakliga sahip olan j. cluster'i bul.

$index = \min\{j, uzaklik(j)\}$

Adim 5: index'in gösterdigi cluster'in ve komsularinin agirliklarini güncelle.

$$w_{i,index}(yeni) = w_{i,index}(eski) + \mu * (x_{i,index} - w_{i,index}(eski))$$

Komsulari için ise "r" uzaklik katsayisi gözeterek güncelleme yap.

Adim 6: Ögrenme katsayisini hassaslastir.

$\mu = \mu * 0.6$ (0.6 rastgele bir sayidir)

Adim 7: Komsuluk derecesini azalt

Adim 8: Durusu kontrol et.

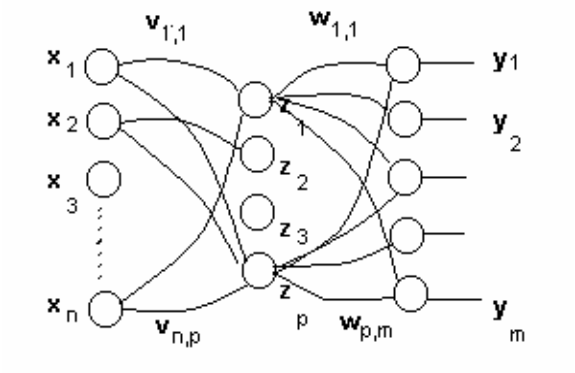
4.4.4 Backpropagation Net

Backpropagation net modeli(Lipmann R.P., 1987), yapay sinir aglari içerisinde hemen hemen her probleme rahatlikla uyarlanabilir bir yapiyi olusturmaktadir. Backpropagation geribeslemeli bir ögrenme mekanizmasi kullanir. Burada yapay sinir aginin yapisi ileribeslemeli olmasina karsin, hatanin geriye dogru ket vurmasindan kaynaklanan bir ögrenme oldugu için geribesleme söz konusu edilmektedir. Ve bu yapay sinir agi modeli ögretmenli ögrenme stratejisini (supervised learning) kullanmaktadir(Lipmann R.P., 1987).

Backpropagation ögrenmede sürekli (continuous) girdi tipini kullanir. Aktivasyon için ise türevi alinabilecek bir fonksiyonu isleme sokmaktadir. Genellikle sigmoid fonksiyonunu kullanan backpropagation (Sekil 23.c) ögrenme fonksiyonu olarakta Delta kuralini kullanir.(6)(Fausett L., 1984)

$$w_{i,j}(new) = w_{i,j}(old) + (m * [t - f(y_{in})] * f'(y_{in})) \qquad (6)$$

(6)'ya göre türevi alinabilinen bir fonksiyon kullanilmasi backpropagation tipi bir yapay sinir agi modeli için önem tasimaktadir. Çünkü türev bir egri üzerinde degisim olarak tarif edilmektedir. Bir baska deyisle; hatanin minimize edilmesi demek, hatanin türevinin 0 olmasi anlamina gelmektedir. Bu yüzden baclpropagation hatanin türevini "wij" agirliklari üzerinde ögrenmekte bu sayede hatalar her bir iterasyon (döngü) sonunda 0'a dogru yaklasmaktadir. Backpropagation bu sebepten dolayi basarili kullanim sahasina sahip olup mimarisi Sekil 32.'deki gibidir. (Lipmann R.P.,1987)



Sekil 32.: Standart 3 katmanli (x girdi, z sakli, y çikti) Backpropagation agi yapisi.

Backpropagation mimarisi için kullanilan algoritma, ilk adimda "y"'ler üzerinde olusan hatayi sakli ve çikti katmani arasinda yer alan "$w_{p,m}$"'ler üzerine yansitirlar. Ancak güncelleme hemen gerçeklestirilmez. "$w_{p,m}$" lerin güncelleme yapilmamis olan hallerinde olusan hatalar ise girdi ve sakli katman arasinda yer alan "$v_{p,n}$"'ler üzerine yansitilir. "$w_{p,m}$" ve "$v_{p,n}$"'ler ayni anda güncellenerek backpropagation algoritmasinin paralel

bir yapi arz ederek ögrenmesi saglanmaktadir. Bu paralel yapi saysinde backpropagation mimarisinin performansi diger yapilara oranla daha yüksek olacaktir. Backpropagation algoritmasinda kullanilan sigmoid fonksiyonu ve bu fonksiyonun türevi (7) ve (8)'nolu formüllerde verilmistir. (Fausett L., 1984)

$$f_{sigmoid}(x) = (2/(1+\exp(-x))) - 1 \tag{7}$$

$$f'_{sigmoid}(x) = 1/2 * \left|1 + f_{sigmoid}(x)\right| * \left|1 - f_{sigmoid}(x)\right| \tag{8}$$

Backpropagation algoritmasi asagidaki gibidir; (Lipmann R.P.,1987)

Adim 0: Agirliklara baslangiç degeri ata.

$0 < w_{jk} < 1$, $0 < v_{ij} < 1$

Adim 1: Enerji stabilize olan dek 2-9.Adimlar arasini tekrarla

Adim 2: Herbir girdi örüntüsü (s) ve ona ait olan çikti degeri (t) için 3-8. arasini tekrarla.

<u>Ileribeslemeli Kisim:</u>

Adim 3: Girdi sinyalini al ve üst katmana ilet.

$x_i = s_i$

Adim 4: Herbir gizli katman için toplam sinyalleri hesapla.

$z_{in\_j} = \Sigma\, x_i * v_{ij}$

Herbir gizli katman için çikis degeri hesapla.

$z_j = f_{sigmoid}(z_{in\_j})$

Adim 5: Herbir çikti katmani için toplam sinyalleri hesapla.

$y_{in\_k} = \Sigma\, z_j * w_{jk}$

Herbir çikti hücresinin çikis degerini hesapla.

$y_k = f_{sigmoid}(y_{in\_k})$

<u>Hatanin Geribeslenmesi Kismi</u>:

Adim 6: Çikti katmaninda olusan hatayi hesapla

$\delta_k = (t_k - y_k) * f'_{sigmoid}(y_{in\_k})$

Çikti-sakli katmanlari arasi yapilacak agirlik degisimini bul.

$\Delta w_{jk} = \mu * \delta_k * z_j$

Adim 7: Herbir sakli katman üzerindeki hatayi hesapla

$\delta_{in\_j} = \Sigma\, \delta_k * w_{jk}$

Girdi katmanina yansitilacak hatayi hesapla

$\delta_j = \delta_{in\_j} * f'_{sigmoid}(z_{in\_j})$

Girdi-sakli katmanlari arasi yapilacak agirlik degisimini bul

$$\Delta v_{ij} = \mu * \delta_j * x_i$$

Adim 8: Simdi tüm agirliklari ayni anda güncelle.

$$w_{jk}(yeni) = w_{jk}(eski) + \Delta w_{jk}$$

$$v_{ij}(yeni) = v_{ij}(eski) + \Delta v_{ij}$$

Adim 9: iterasyonu kontrol et. Bitir.

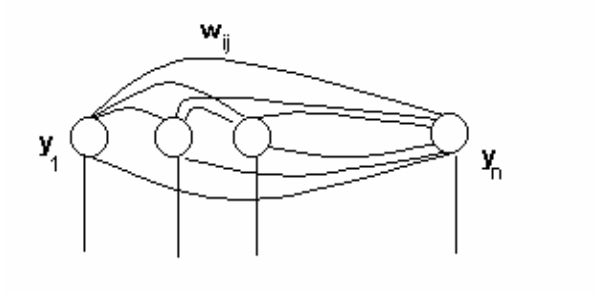
4.4.5 Hopfield Yapay Sinir Agi Modeli

Yapay sinir aglarinin en karma modelini teskil eden Hopfield yapay sinir agi "recurrent" ya da "recursive" yani tekrar beslemeli bir yapiya sahiptir. Bu özelligi ile Hopfield modeli diger yapay sinir agi modellerinden ayrilmaktadir.(Hopfield J.J, et all, 1985) Tekrar besleme kabiliyeti sayesinde girdi örüntüsü Hopfield mimarisine verildiginde, mimari isleme bir baslangiç enerjisi ile baslar. Bu baslangiç konumundan itibaren yapi, girdi örüntüsünü bir baska girdi örüntüsüne(daha önce ögrendigi) dogru yönlendirmeye baslar. Bu yönlendirme sürecinde girdi örüntüsünde yapilan her ufak degisimin ardindan enerji tekrar tekrar hesaplanarak girdi örüntüsünün morfolojik dönüsümünün kontrolü saglanir. Girdi örüntüsünün bir baska ögrenilmis olan girdi örüntüsüne benzetilme islemi (morfolojisi), enerji stabilize olana dek sürer. Enerji stabilizasyonu ise enerjinin minimuma ulastigi ve degismedigi yerdir.

Girdi örüntüsünün tekrar besleme yardimiyla enerjisini düsüre düsüre yerel minimuma ulasmasi (enerjinin artik düstügü ve degismedi süreç) olayina "convergence" denir. "Convergence" pozisyonunda olan girdi örüntüsünün morfolojik haline ise "converged vector" adi verilir. Yani elde edilen cevap vektörüdür. Bu açidan matematiksel olarak interpolasyon islemine benzeyen bir operasyon olan bu ögrenme süreci sayesinde Hopfield sinir agi modeli optimizasyon problemlerinin kaçinilmaz referans modeli olmustur.(Cavalieri S., et all, 1994).

Bu tez çalismasina referans model teskil eden MAREN (Öztürk Y., 1995) yapay sinir agi modeli de referans olarak Hopfield yapay sinir agini ve onun enerji fonksiyonunu almistir.

Mimari olarak Hopfield tek katmanli bir yapiya sahiptir. Bu tek katman hem girdi hem de çikti katmani olarak görev yapar. Simetrik olarak tüm sinir hücreleri birbirine baglanmistir. Sadece hücrelerin kendi üzerlerine baglantilari mevcut degildir. Sekil 33.'te Hopfield modelinin mimarisi açik biçimde görülmektedir.



Sekil 33: Hopfield Net Modeli

Hopfield modelinin ögrenme sürecinde kullandigi kural -en basit kural olarak bilinen- Hebb kuralidir. (5)'nolu formülde yer aldigi gibi agirliklarin ögrenilmesi iki sinir hücresi arasinda ki degerin ayni olup olmadigina baglidir. Girdi örüntülerinde binary(ikili) degerlerden ziyade bipolar(iki kutuplu) degerlerin kullanilmasi daha dogru olan yaklasimdir. Buna göre her girilen örüntü için ögrenme söyle ifade edilir;

$$W = \{w_{ij}\} \quad \text{ve} \quad w_{ij} = \sum_{p} s_i(p) * s_j(p) \qquad , i \neq j \tag{10}$$

(10) nolu formülde(Fausett L., 1984) Hopfield modelinin ögrenme formülü yer almaktadir. Ögrenme Hopfield'da çok basit olan Hebb kurali ile saglanmaktadir. Yapilan "m" tane (p=m) girdi için(herbir girdi "n" hücreden olusmaktadir. 'i=j=1,2,....n.') agirlik matrisini olusturmaktadir.'W'. Örnegin y_1 hücresi -1, y_3 hücresi -1 iken, $w_{13} = w_{13} + (-1)*(-1) = w_{13} + 1$. Tüm girdi örüntüleri kullanilarak olusturuldugunda "w_{ij}" üzerinde ögrenmenin tamamlanmis oldugu belirtilir.

Hopfield modelinin ögrenim süreci tamamlandiktan sonra asil önemli olan, girdi örüntüsünün sisteme beslendiginde istenen sonuca ulasip ulasilamayacaginin (convergence) kontrolünü yapacak enerji fonksiyonunun kurulmasidir. Hopfield; Lyapunov fonksiyonu ile sistemin enerjisinin hedefe ulasabilecegi (convergence) ve o hedefte stabilize halinde kalabileceginin kontrolünü yine bu fonksiyon yardimiyla saglayabilecegini ispat etmistir. Hopfield modelinin en önemli parçasi olan enerji fonksiyonu (11) nolu formülde belirtilmistir. (Fausett L., 1984)

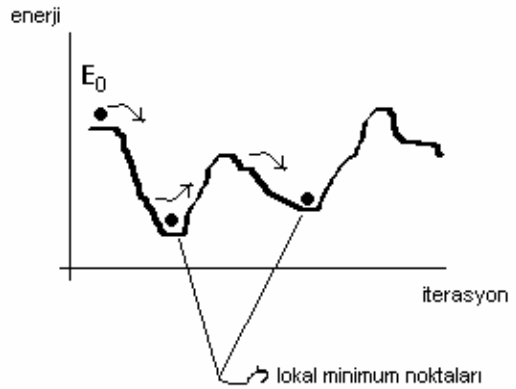
$$E = -(1/2) * \sum_{i \neq j} \sum_{j} y_i * y_j * w_{ij} - \sum_{i} x_i * y_i + \sum_{j} j_i * y_i \tag{11}$$

Enerji formülüne dikkat edilecek olunursa, burada her bir hücrenin aktifliginin kontrolünün yapildigi ve aktif olanlarin agirlik degerlerinin enerjiye katildigi görülecektir. Bu noktada önemli olan girdi örüntüsünün

Δy_i kadar bir degisiklige ugratildiginda (yönlendirildiginde) enerji degisiminin ne olacaginin hesaplanmasidir. Enerjide olusan bu degisim asagidaki formül ile belirtilmistir:

$$\Delta E = -\left[\sum_j (y_j * w_{ij}) + x_i - \Phi_i\right] * \Delta y_j \qquad (12)$$

Eger ΔE (enerjideki degisim miltari) bir önceki enerjiyi daha da düsürecek yönde ilerliyorsa, bu girdi örüntüsünün yerel (local) minumuma dogru haraket ettigini gösterir. Bu ilerleyis Sekil 34.'te enerji grafigi ile gösterilmistir.



Sekil 34: Hopfield modelinde enerji degisim grafigi.

Sistem yerel minimumu yakaladigi an stabilize (uzun müddet degisim geçirmeden) kalabiliyorsa sistem o girdi için cevaba ulasmis demektir. Hopfield modelinin bu baglamda algoritmasi ise asagida sunulmaktadir;(Hopfield J.J. et all, 1985)

Adim 0: Agirliklarin degerlerini ögren. (Hebb kurali kullanarak)

$w_{ij} = \Sigma_p s_i(p) * s_j(p)$, $i \neq j$ iken

Enerji lokal minimumu yakalayana kadar 1-7.Adimlar arasini tekrarla

Adim 1: Herbir girdi vektörü için 2-6.Adimlar arasini tekrarla.

Adim 2: Disaridan gelen girdi vektörünü sisteme besle

$y_i = x_i$

Adim 3: Herbir sinir hücresi için 4-6. Adimlar arasini tekrarla

Adim 4: Her hücrenin girdi toplam sinyalini bul

$y_{in\_i} = x_i + \Sigma_j (y_j * w_{ij})$

Adim 5: Aktivasyonu hesapla

y_i= 1 (eger $y_{in\_i} > \Phi_i$), y_i (eger $y_{in\_i} = \Phi_i$), 0 (eger $y_{in\_i} < \Phi_i$)

Adim 6: Çiktilari diger hücreler dagit.

Adim 7: Enerji fonksiyonu hesapla, "converge" noktasindaysa dur, degilse rastgele bir "y_i" degerini

Δy_i kadar degistir.

4.4.6 Maren

Maren yapay sinir agi tipi çikis olarak Hopfield modeline tamamen benzemektedir. Ancak maren'in ögrenme ve enerji fonksiyonu Hopfield'a ait olan fonksiyondan biraz farklidir. Maren mimarisi Hopfield mimarisinin aynisi olup tüm hücreler birbirine bagli (fully connected) ve hücrelerin kendilerine baglantilari mevcut degildir. Maren'in agirliklari notasyon olarak "T_{ij}" ile gösterilir. Ögrenme süreci ise asagida ki formül ile anilir,(Öztürk Y.,1995)

$$T_{ij} = \sum_{s=1}^{p} \sum_{i=1}^{n} \sum_{j=1}^{n} sign(v_i^s - v_j^s), ve, T_{ji} = -T_{ij} \qquad (13)$$

(13)'te kullanilan signum fonksiyonu(sign) söyle tanimlanmistir, sign(x)=1 (eger x>0), -1 (eger x<0), 0 (eger x=0). Bu açiklamayla maren, Hopfield modelinden ayrilmistir. Çünkü Hopfield Hebb kuralini kullanirken, maren büyüklük küçüklük iliskisini ögrenmektedir. Bu sayede maren yardimiyla hücreler arasi iliski rahatlikla tutulmaktadir. Eger $v_i > v_j$ (i.hücre, j.hücreden büyük degere sahipse) ise aradaki bagi kuvvetlendir, eger $v_i < v_j$ (i.hücre, j.hücreden küçük degere sahipse) ise aradaki agirligi zayiflat.

Bununla beraber marenin enerji fonksiyonu asagidaki iki fonksiyonun biriyle temsil edilmektedir. (14) ve (15) birbirinin aynisi olup girdi örüntülerinin ayri ayri gösterimini tasimasi açisindan 2 ari formül gibi belirtilmistir: (Öztürk Y.,1995)

$$E = \sum_{i=1}^{n} \sum_{j=1}^{n} T_{ij} * sign(v_i - v_j) \qquad (14)$$

$$E = \sum_{s=1}^{p} \sum_{i=1}^{n} \sum_{j=1}^{n} sign(v_i^s - v_j^s) * sign(v_i - v_j) \qquad (15)$$

Yukaridaki enerji fonksiyonu (15) alabilecigi en yüksek deger (maksimum) asagidaki formülle belirtilmistir. (Öztürk Y.,1995)

$$E_{\max} = N^2 - N, E_{\min} = 0 \qquad (16)$$

Enerji fonksiyonu marenin yapisinda maksimuma dogru götürülmek istenmektedir. Bu Hopfield modelinde minimuma dogru gidis seklinde olmaktadir. Eger enerji degisimi artis gösteriyorsa girdi örüntüsü o yöne dogru yönlendirilir. Eger enerjide düsüs izleniyorsa o zaman girdi örüntüsünde o yöne dogru bir yönlendirme yapilmaz. Yönelndirme islemi için Δ gibi bir aralik belirlenir. Ve Δ sayisi rastgele seçilen hücrelerden birinin degerine eklenerek eklenmis halin enerjisi hesaplanir. Yine ayni hücrenin eski degeri Δ kadar azlatilarak, azaltilmis halin enerjisi hesaplanir. Her iki durumun enerjisi kiyaslanir. Eger Δ kadarlik degisim yapilmadan önceki enerjiden daha büyük bir enerji elde edilmisse hangi enerji büyük ise girdi örüntüsü o enerji dogrultusunda Δ kadar degistirilmis ya da diger ifadeyle yönlendirilmis olunur.

Bu algoritma enerjinin maksimumda stabilize olusuna kadar devam ettirilir. Eger sistem belli bir maksimum enerji degeri yakalamissa cevap bulunmus demektir. Bu noktada "v_i" ile gösterilen girdi örüntüsü ve enerji arasindaki iliski asagidaki tablo ile ifade edilir(Çizelge 1.) ve bir sonraki adim bu tabloya göre tesbit edilir.

Çizelge 1 .: Marende Δ degisimiyle v_i ve enerjiler arasindaki iliski

ΔE_+	ΔE_-	sign($E_+ - E_-$)	V_i ve E_0 (sakli enerji)
<0	<0	önemsiz	degisim yok. $E_0=E_0$
>0	<0	önemsiz	$V_i=(V_i+\Delta)$, $E_0=E_+$
<0	>0	önemsiz	$V_i=(V_i-\Delta)$, $E_0=E_-$
>0	>0	+1	$V_i=(V_i+\Delta)$, $E_0=E_+$
>0	>0	-1	$V_i=(V_i-\Delta)$, $E_0=E_-$
>0	>0	0	$V_i=(V_i+\Delta)$, $E_0=E_+$ ya da $V_i=(V_i-\Delta)$, $E_0=E_-$

Çizelge 1. baz alinarak marenin algoritmasi asagida sunulmustur; (Öztürk Y.,1995)

Adim 0: Agirliklari (13)'ü kullanarak ögren.

$T_{ij} = \Sigma_s \Sigma_i \Sigma_j \ \text{sign}(V^s_i - V^s_j)$

Adim 1: Baslangiç enerjisini hesapla

$E_0 = \Sigma_i \Sigma_j \ T_{ij} * \text{sign}(V_i - V_j)$

Adim 2: Enerji stabilize olana kadar (maksimumda kalana dek) 3-8.Adimlari tekrarla

Adim 3: Belli bir iterasyon sayisi kadar 4-7.Adimlari tekrarla

Negatif V_i = Pozitif V_i = V_i

Adim 4: Rastgele seçilmis t. Hücrenin degerini "Δ" kadar degistir.

Negatif V_t = Negatif V_t - Δ

Pozitif V_t = Pozitif V_t - Δ

Adim 5: Enerji degisimlerini hesapla.

$E_+ = \Sigma_i \Sigma_j T_{ij}$ * sign(Pozitif V_i - Pozitif V_j)

$E_- = \Sigma_i \Sigma_j T_{ij}$ * sign(Negatif V_i - Negatif V_j)

Adim 6: Enerji farklarini hesapla.

$\Delta E_+ = E_+ - E_0$

$\Delta E_- = E_- - E_0$

Adim 7: E_0 ve V_i 'yi Çizelge 1.'e göre degistir.

Adim 8: Enerji fonksiyonunu kontrol et.

4.5 Yapay Sinir Aglarinin Kullanim Sahalari

Gerçek yasamda yapay sinir aglarinin kullanimi gün geçtikçe karsimiza daha sik çikmaya baslamistir. Günümüz problemleri oldukça karisik ve siradan algoritmalarla çözümlenemeyecek derecede zeki algoritmalara ihtiyaç duyar hale gelmistir. Problemi olusturan sebeplerin dinamik olarak degisimi klasik algoritmalarin çözüme baslamadan tikanmasina yol açmistir.

Bir baska boyutla bakilacak olunursa günümüz problemleri hataya toleransli yaklasimda bulunabilecek algoritmalara gereksinim duymaktadir. Örnegin bilgisayar agi üzerinde ki trafigin yogunlugu artinca paketler için bir limit konacaktir. Bu limiti algilayan algoritmalar, kesin olarak paket göderimini durduracak yahut kesin bir zaman dilimi uygulamasina giderek paketleri o zaman dilimlerinde göndererek trafigin yogunlugunu kontrol altina almaya çalisacaktir. Oysa ki yapay sinir aglari yardimiyla agin trafik yogunlugu sürekli dinlenmekte, trafigin dinamik degerine bakilarak paket gönderimi üzerine karar verilebilmektedir. Böylece dinamik ve hatalara açik durumlara toleransli yaklasabilen yapay sinir aglari bu özelligi yüzünden kullanim alani olarak yayginlasmaya baslamistir.

Degisen durumlardan kaynaklanan dinamik degerlerin akisinin kontrolü "np complete" olarak nitelendirilen karmasik ortamlari dogurmaktadir. Örnegin yönlendirme(routing) yapan klasik bir algoritma, trafigin yogunlasmasi ya da iletisim hatti üzerindeki düğümlerden birinin devre disi kalmasi gibi sebeplerden dolayi agin yapisinda meydana gelen dinamik degisimi algilamakta güçlük çekecek, algiladigi zaman ise

yönlendirme tablolarini güncelleyemeyecektir. Oysa bu tezin çalisma konusu olarak iletilecegi üzere yönlendirme(routing) her dinamik degisimde bile rahatlikla yönlendirme tablolarini güncelleyebilecektir.

Karar verme mekanizmasi olarak kullanilan yapay sinir aglari bu özellikleri yüzünden de tercih edilmektedir. Iyi bir siniflandirma karakteristigine sahip olan yapay sinir agi modelleri, sorgulanan bilginin hangi sinifa ait oldugunun cevabini vereceginden o girdi üzerinde karar vermis olacaktir. Örnegin A hastaliginin olup olmadigina dair egitilen bir yapay sinir agi modeli, bir hastanin sikayetleri girildiginde doktorun görevini üstlenebilecek ve kisiye A sinifi hastaligi oldugunu ve buna göre tedavi olmasi için gerekenleri karar verebilecektir. Bu tip kullanim sahasina ülkemizde ki akademik çalisma yapan tip fakültelerinde artik rastlanabilinmektedir. (Görüntü Anlama ve Analizi sempozyumu, 1996).

Yukaridaki özellikleriyle yapay sinir aglari günümüzde; sinyal isleme, kontrol mekanizmalari, görüntü isleme, örüntü tanima, tip, ses algilama gibi bir çok sahada kendini kanitlamistir. Herbirine küçük birer örnek vermek gerekecek olursa; sinyal islemede, sinyallerin gönderildigi frekans araliklarinda olusan gürültülerin esiklenmesinde ve temizlenmesinde yapay sinir aglari kullanilmaktadir. Kontrol mekanizmalari sahasinda stepper motorlarin kumanda edilisi, haraketlerinin kontrolü yine yapay sinir aglari modelleriyle saglanmaktadir. Yine kontrol mekanizmalarina örnek olarak tezin kapsaminda olan, bilgisayar aglarinda baglanti kurulmasi için servis kalitesi parametrelerinin kontrolünün saglanmasi yine yapay sinir aglari ile temin edilmektedir. Görüntü isleme alanindan bir örnek ise, fotografin renklerinin görsel kalitesinde kayip olmadan sahip oldugu 16 milyon rengin 256 renge düsürülmesinde yine yapay sinir aginin basarili kullanimi görülmektedir(Cinsdikici M.,vd., 1997,TAINN97). Örüntü tanima sahasinda, karakter tanima, yüz tanima, parmak izi tanima problemlerinin çözümleri yine yapay sinir aglariyla saglanmaktadir. Tip alaninda, hastalik teshisi ve her hastanin özelliklerine göre tedavi seçimi uygulamalarinda yine yapay sinir aglari karsimiza çikmaktadir. Ses algilama konusunda, özellikle güvenlik sistemleri ve ses kontrollü cihazlar ya da yazilim gelistirilmesinde yine kullanimina rastladigimiz modellerdir.

Bu baglamda yapay sinir aglari yarinin “cyborg” leri için temel çözümler sunmaktadir. Ileride ses taniyan, görüntü algilayan, kendi kendine karar verebilen cybörgler yapay sinir aglari olmadan kullanilmadan hayata geçirilemeyecektir.

5. ATM PROBLEMLERI ÇÖZÜMLERI

Bu bölümde daha önce anlatilan ATM aglarinda karsilasilan problemler ve onlarin klasik çözümlerine deginilecek, ardindan yapay sinir aglariyla çözümleri verilecektir.

5.1 ATM Problemleri Klasik Çözümleri

Baglanti Kurumu Kontrolü (Call Admission Control): ATM aglarinda kullanici bir baglanti kurma isteminde bulundugu zaman, ag üzerinde yeni bir baglantinin saglanip saglanmayacaginin belirlenmesi gerekmektedir. Belirleme islemi ag üzerindeki kaynaklarin kullanimlariyla dogrudan iliskilidir. Klasik yaklasimlar ile çözüm tüm agin istatistiksel yorumunun elde edilmesi ve servisin kalitesinin analizinin yapilmasiyla gerçeklestirilmektedir. Tüm bu analizi yapmak neredeyse imkansizdir. Çünkü bu analiz ATM aginda dinamik olarak degisen ve çok genis bir yalpazeye yayilmis birden fazla servisin incelenmesi anlamina gelmektedir. QOS parametrelerinin ve trafik yogunlugunun degerleri gözönüne alinarak, aga yeni bir baglanti kurulma kararinin - henüz kullanici arabiriminde iken (UNI tarafi)- karar verilebilmesi için daha etkin, sonuca daha çabuk ulasacak karar verme mekanizmasina sahip bir çözüm sunulmalidir.

Baglanti saglanmasi kontrolü; kullanici tarafindan belirlenecek baglanti servis kalitesi degerlerinin ATM agi üzerindeki mevcut degerlerle kiyaslanip, baglanti kurulup kurulmamasi üzerine karar verilmesinden olusmaktadir. ATM aglari için belirlenen servislerin mevcut bit hizi (ABR, available bit rate) tipi hizmet alan baglantilarda kullanilabilen bu mekanizma yardimiyla baglanti onayinin alinmasi bastan saglanacaktir. (Neves J.,et all, 1995)

Öyleyse ATM üzerinde baglanti kontrolü için dinamik yük dengesine göre karar vermeye çalisan ve her degisimi aninda degerlendirme altina alan çözüm sunulmalidir. Baglanti kurumu için güncel yasamda kullanilan klasik yöntemler bulundugu dügümün(anahtar) komsulari için aldigi trafik yogunluk degerine bakar. Hat belli bir yogunlugu geçmis ise kullaniciya paket gönderimi yaptirilmaz. Bunun süresi ise belli bir zamanla kisitlanmistir. Bu tip yöntemler aloha, slotted aloha veya pure aloha metodunun türevleridir. Hatti dinleyerek paketlerin belli bir zaman dilimi tutulup o zaman diliminin ardindan birakilmalarina izin verilir. Broadcast yapilar üzerinde uygulanan bu tip algoritmalarin benzerleri pp (point to point) tipli yapilarda da mevcttur. ATM aglari için kullanici-ag (UNI, User to Network Interface) tarafinda bunu yapmak için klasik metodlara basvurmak paket göderiminin statik bir yapi ile çözümlenmesi anlamina gelmektedir. Bu statiklik degisen durum kosullarinda yetersiz kalacaktir.