



**T.C.
İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**



YÜKSEK LİSANS TEZİ

DERİN ÖĞRENME UYGULAMALARI

Cihan TİKEN

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Danışman

Yrd.Doç.Dr. Tolga ENSARİ

Haziran, 2015

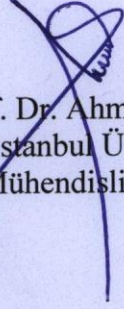
İSTANBUL

Bu çalışma 19/06/2015 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği programında Yüksek Lisans Tezi olarak kabul edilmiştir.

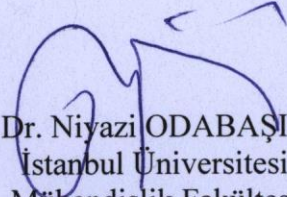
Tez Jürisi:



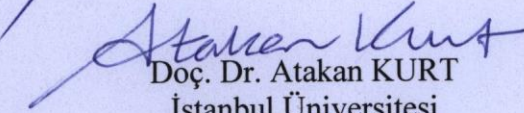
Yrd. Doç. Dr. Tolga ENSARİ (Danışman)
İstanbul Üniversitesi
Mühendislik Fakültesi



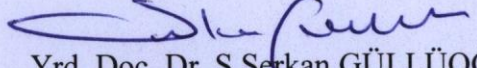
Prof. Dr. Ahmet SERTBAŞ
İstanbul Üniversitesi
Mühendislik Fakültesi



Doç. Dr. Niyazi ODABAŞIOĞLU
İstanbul Üniversitesi
Mühendislik Fakültesi



Doç. Dr. Atakan KURT
İstanbul Üniversitesi
Mühendislik Fakültesi



Yrd. Doç. Dr. S. Serkan GÜLLÜOĞLU
İstanbul Arel Üniversitesi
Mühendislik-Mimarlık Fakültesi

ÖNSÖZ

Yüksek lisans tez çalışmam boyunca zaman gözetmeksizin değerli bilgi ve birikimi yardımıyla zorlandığım anlarda önümü açan ve hiçbir zaman desteğini esirgemeyen tez danışmanım Yrd.Doç.Dr. Tolga ENSARİ hocama teşekkürü bir borç bilirim.

Eğitim hayatım boyunca her türlü maddi ve manevi destekleriyle benimle beraber olan çektiğim sıkıntıları aşmamda bana yardımcı olan başta merhum annem, ağabeyim Av. Rüstem TİKEN ve ablam Perihan TİKEN olmak üzere sevgili aileme ve bütün dostlarıma tüm kalbimle teşekkür ederim.

Haziran, 2015

Cihan TİKEN

İÇİNDEKİLER

Sayfa No

ÖNSÖZ.....	i
İÇİNDEKİLER	ii
ŞEKİL LİSTESİ.....	v
SİMGE VE KISALTMA LİSTESİ	vii
ÖZET.....	viii
SUMMARY	ix
1. GİRİŞ.....	1
2. GENEL KISIMLAR	3
2.1. YAPAY SİNİR AĞLAR	3
2.1.1. Yapay Sinir Ağlarının Özellikleri	4
2.1.1.1. Doğrusal Olmama	4
2.1.1.2. Öğrenme	4
2.1.1.3. Adaptasyon	4
2.1.1.4. Genelleme	4
2.1.1.5. Hata Toleransı.....	4
2.1.1.6. VSLI Uygulanabilirlik	4
2.1.1.7. Analiz ve Tasarım Birliği	5
2.1.2. Yapay Sinir Ağlarının Avantaj ve Dezavantajları	5
2.1.2.1. Avantajları.....	5
2.1.2.2. Dezavantajları	5
2.1.3. Yapay Sinir Ağlarının Uygulama Alanları	6
2.1.3.1. YSA'nın Kullanıldığı Sektörler	6
2.1.4. Yapay Sinir Ağlarda Öğrenme.....	7
2.1.4.1. Öğretmenli Öğrenme	7
2.1.4.2. Öğretmensiz Öğrenme	8
2.1.4.3. Destekleyici Öğrenme	9
2.1.4.4. Karma Öğrenmeler	9
2.1.5. Yapay Sinir Ağlarda Öğrenme Kuralları	9

2.1.5.1. Çevrimiçi Öğrenme Kuralları	9
2.1.5.2. Çevrimdışı Öğrenme Kuralları	9
2.1.5.3. Hebb Kuralı	10
2.1.5.4. Delta Kuralı	10
2.1.5.5. Kohonen Kuralı	11
2.1.5.6. Hopfield Kuralı	12
2.1.6. Yapay Sinir Ağ Türleri	13
2.1.6.1. İleri Beslemeli YSA	13
2.1.6.2. Geri Beslemeli YSA	14
3. MALZEME VE YÖNTEM	16
3.1. DERİN ÖĞRENME	16
3.1.1. Öz-Öğretim Öğrenmesinden Derin Öğrenmeye	16
3.1.2. Boltzmann Makinesi	18
3.1.3. Boltzmann Makine'sinde Öğrenme	19
3.1.3.1. Gibbs Örneklemesi	20
3.1.3.2. Gizli Birimler Olmadan Öğrenme	20
3.1.3.3. Gizli Birimler ile Öğrenme	21
3.1.4. Boltzmann Makinesi Türleri	21
3.1.4.1. Üst Düzey Boltzmann Makineler	21
3.1.4.2. Şartlı Boltzmann Makineler	22
3.1.4.3. Ortalama Boltzmann Makineler	22
3.1.4.4. İkili Olmayan Birimler (Non-Binary Units)	23
3.1.4.5. Öğrenme Hızı	23
3.1.5. Kısıtlı Boltzmann Makineleri	23
3.1.5.1. Kısıtlı Boltzmann Makinelerin Eğitimi	25
3.1.5.2. Karşılaştırmalı Ayırışma Öğrenme Metodu	26
3.1.5.3. İleri Markov-Chain Monte-Carlo (MCMC) Örnekleme Metodu Tabanlı Öğrenme	28
3.1.5.4. Diğer Yaklaşımlar	28
3.2. AUTOENCODERS	29
3.2.1. Autoencoder Mimarisi	30
3.2.2. Eğitim Yöntemleri	32
3.2.2.1. Geri Yayılım	32
3.2.2.2. Polak Ribiere Eşlenik Gradyan Geri Yayılımı	32
3.2.2.3. Yığın Autoencoder	33

4. BULGULAR	37
4.1. KARAKTER TANIMA.....	37
4.2. KENAR BELİRLEME	40
5. TARTIŞMA VE SONUÇ	42
KAYNAKLAR	43
ÖZGEÇMİŞ.....	46

ŞEKİL LİSTESİ

Sayfa No

Şekil 1.1: Derin ağ katmansal yapısı.	2
Şekil 2.1: Yapay Sinir Ağlarının temel gösterimi.	3
Şekil 2.2: Öğretmenli öğrenme modelinin işleyiş diyagramı [14].	8
Şekil 2.3: Öğretmensiz öğrenme modelinin işleyiş diyagramı [14].	8
Şekil 2.4: Destekleyici öğrenme mekanizması [15].	9
Şekil 2.5: Kohonen öğrenme kuralı.	12
Şekil 2.6: Hopfield öğrenme kuralı.	13
Şekil 2.7: Üç katmanlı ileri beslemeli YSA yapısı.	14
Şekil 2.8: Geri beslemeli Yapay Sinir Ağı.	15
Şekil 3.1: Verilen x örneğinden gizli katman kullanılarak a özellik çıkarımı yapılması.	16
Şekil 3.2: Lojistik regresyon birimini gösteren şema.	17
Şekil 3.3: Verilen x örneğinden gizli katman kullanılarak a özellik çıkarımı aşaması ve lojistik regresyon birimini birlikte gösteren şema.	17
Şekil 3.4: Öz-Öğretimli öğrenme örnekleri.	18
Şekil 3.5: Kısıtlı Boltzmann Makine ile Boltzmann Makine arasındaki fark.	24
Şekil 3.6: RBM ' de paralelleştirilmiş yapılar.	25
Şekil 3.7: CD öğrenme yöntemi.	26
Şekil 3.8: Model Dağılımı ve CD öğrenme yöntemi karşılaştırma.	27
Şekil 3.9: Autoencoder'ın mimari yapısını gösteren şema.	30
Şekil 3.10: Bir autoencoder düğümüne ait girdiler ve oluşan eşik.	31
Şekil 3.11: Verinin ileri beslemesi ve hatanın geri yayılımı [25].	32
Şekil 3.12: AE'ye eklenmiş tek gizli katmanlı AE.	33
Şekil 3.13: MNIST veri kümesine ait bir kesit.	36

Şekil 4.1: Eğitim setinde test imgesi başına düşen ortalama karesel oluşturma (reconstruction) hatası [27].	38
Şekil 4.2: MNIST veri kümesinin Temel Bileşen Analizi (PCA-Principal Component Analysis) ile sınıflandırılması sonuçları.	39
Şekil 4.3: MNIST veri kümesinin 784-1.000-500-250-2 yapısındaki AE içeren derin öğrenme yaklaşımıyla sınıflandırılması sonuçları [27].	39
Şekil 4.4: MNIST karakter veri seti (AE çıkışı).	40
Şekil 4.5: MNIST karakter veri setine uygulanan Prewitt kenar belirleme yöntemi.	40
Şekil 4.6: MNIST karakter veri setine uygulanan Sobel kenar belirleme yöntemi.	40
Şekil 4.7: MNIST karakter veri setine uygulanan Roberts kenar belirleme yöntemi.	41

SİMGE VE KISALTMA LİSTESİ

Simgeler

Açıklama

$(s_i, s_j)_{\text{model}}$	: Modelin ortak dağıtımını üzerinden beklenen değer
α	: Öğrenme katsayısı veya öğrenme oranı
w_{jk}	: Nöron u_k 'den nöron u_j 'ye olan ağırlık
μ	: Eğitim oranı

Kısaltmalar

Açıklama

BM	: Boltzmann Machine (Boltzmann Makinesi)
CD	: Contrastive Divergence (Karşılaştırmalı Iraksama)
HMC	: Hybrid Monte Carlo (Melez Monte Carlo)
ECG	: Elektrokardiogram (Kardiyoloji Tetkik Yöntemi)
EEG	: Elektroensefalografi (Epilepsi Tetkik Yöntemi)
L-BFGS	: Left Broyden-Fletcher-Goldfarb-Shanno
MCMC	: Markov-Chain Monte-Carlo (Markov-Zinciri Monte-Carlo)
MR	: Magnetic Resonance (Manyetik Rezonans)
PCA	: Principal Component Analysis (Temel Bileşen Analizi)
PCD	: Persistent Contrastive Divergence (Kalıcı Karşılaştırmalı Iraksama)
PT	: Parallel Tempering (Paralel Tavlama)
RBM	: Restricted Boltzmann Machine (Kısıtlı Boltzmann Makinesi)
RM	: Ratio Matching (Eşleştirme Oranı)
VSLI	: Very Large Scale Integrated (Çok Büyük Ölçekli Tümleşik Devreler)
YSA	: Yapay Sinir Ağlar

ÖZET

YÜKSEK LİSANS TEZİ

DERİN ÖĞRENME UYGULAMALARI

Cihan TİKEN

İstanbul Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman : Yrd. Doç. Dr. Tolga ENSARİ

Derin Öğrenme (Deep Learning) ve Derin İnanç Ağları (Deep Belief Networks), makine öğrenmesinde ve örüntü tanıma alanlarında son yıllarda en çok ilgi duyulan konulardan biri olmuştur. Derin inanç ağları üst üste eklenmiş ve birbirine bağlı Kısıtlı Boltzmann Makinelerinden (Restricted Boltzmann Machine) meydana gelmektedir. Kısıtlı Boltzmann makineleri, enerji tabanlı olasılıksal yapay sinir ağlarıdır. Derin inanç ağları pek çok gizli katman içermektedir ve Autoencoder mimarisi kullanılarak ince ayar yapıldıktan sonra optimize edilmektedir. Autoencoder ise girdi uzayını yeni bir uzaya dönüştüren bir yapıdır. Bu tez çalışmasında, MNIST veri kümesi kullanılarak karakter tanıma işlemi yapılmıştır. Ardından, Autoencoder çıkışı görüntülerin kenar belirleme uygulamaları yapılmıştır. Kenar belirleme, hem makine öğrenmesi hem de bilgisayarlı görü işlemlerinde önemli bir yere sahiptir. Kenar belirleme işlemlerinde Sobel, Prewitt ve Roberts kenar belirleme yöntemleri kullanılmıştır. Ayrıca, tez çalışmasında kenar belirleme konusu literatürde ilk defa derin öğrenme yapısı kullanılarak ele alınmıştır. Uygun derin inanç ağları kullanılarak, bazı saklı özellikler (feature) incelenmiştir.

Haziran 2015, 57 Sayfa.

Anahtar kelimeler: Derin Öğrenme, Derin İnanç Ağları, Kenar Belirleme, Bilgisayarlı Görü.

SUMMARY

M.Sc. THESIS

DEEP LEARNING APPLICATIONS

Cihan TİKEN

İstanbul University

Institute of Graduate Studies in Science and Engineering

Department of Computer Engineering

Supervisor: Asst. Prof. Dr. Tolga ENSARİ

Deep learning and deep belief networks (DBNs) are two of the most used topics in machine learning and pattern recognition areas, in recent years. DBNs consist of stacked Restricted Boltzmann Machine (RBM) structure and these stacked RBMs are connected to each other. RBM is energy based stochastic neural network. DBNs have many hidden layers and they are optimized after fine-tuning process with Autoencoder (AE) architecture. AE transforms input space to a new space. In this thesis, character recognition is handled by using MNIST dataset. Then AE output images used in edge detection applications. Edge detection is also one of the important issues in machine learning and computer vision areas. Some of classical edge detection techniques such as Sobel, Prewitt and Roberts used in applications. In this thesis, we propose deep learning based edge detection method. Our proposal has not been discussed before in the literature related with DBNs. Some hidden features are examined with suitable DBNs architectures.

June 2015, 57 Pages.

Keywords: Deep Learning, Deep Belief Networks, Edge Detection, Computer Vision.

1. GİRİŞ

Akıllı sistemler geliřtirmek adına günümüze gelinceye kadar pek çok ilerlemeler kaydedilmiřtir. Arařtırmacıların çoęu biyolojik sinir hücrelerinden esinlenmiřlerdir. Farklı disiplinlerden arařtırmacılar kontrol, iliřkisel bellek, örüntü tanıma, optimizasyon, tahmin gibi çeřitli alanlardaki sorunları çözebilmek adına yapay sinir aęları tasarlamıřlardır. Bu alanlarda karřılařılan sorunların çözümleri için pek çok geleneksel yöntem önerilmiřtir. Önerilen bu yöntemlerin bazıları kısıtlanmıř ortamlarda iyi sonuçlar vermesine raęmen, hiçbiri etki ettięi alanın dıřında yeterli bir verim saęlamamıřtır [1]. Bařka bir deyiřle etki alanı dıřında bařarı elde edebilecek esneklięe ve performansa sahip olamamıřtır. Yapay sinir aęları bu alanlarda oldukça etkileyici alternatifler sunmaktadır.

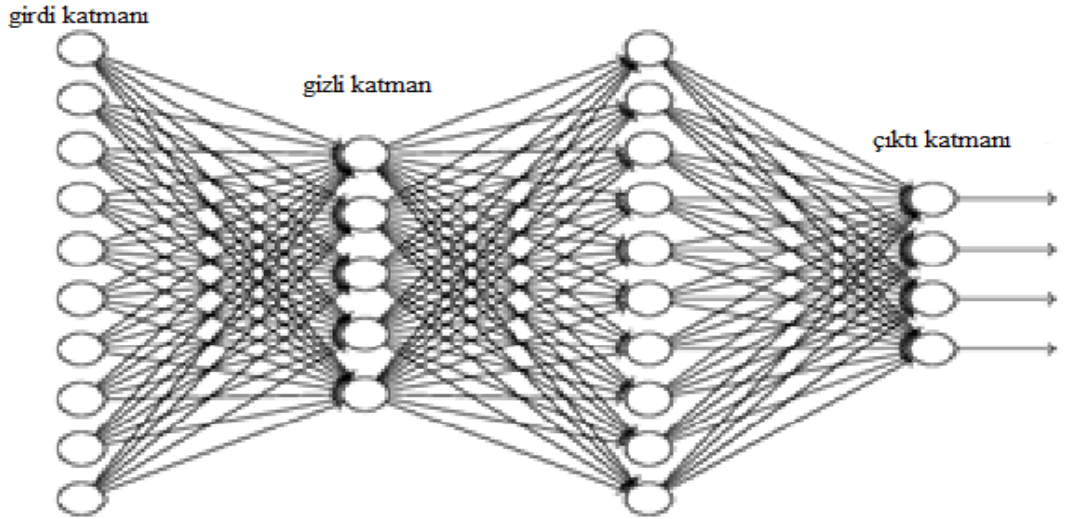
Yapay sinir aęları insan beyni modern paralel bilgisayarların sahip olmadıęı öğrenme yeteneęi, genelleřtirme yeteneęi, doęal içerięe baęlı bilgi iřleme, hata toleransı ve düşük enerji tüketimi gibi karakteristik özelliklere sahiptir [2]. Biyolojik sinir aęlarından esinlenerek geliřtirilen yapay sinir aęlarının bu karakteristik özelliklerin pek çoęunu icra etmesi beklenmektedir. Modern bilgisayarlar sayısal hesaplamalarda çok iyiye insan beyninin bir insan kalabalıęı içerisinde hiç zorlanmadan birini tanıması gibi karmařık algısal problemleri çözmeye yeteneęinin oldukça gerisindedir [3].

Derin öğrenme, makine öğrenmesi ailesinin geliřmekte olan bir üyesidir ve son yıllarda oldukça popüler bir durumdur. Derin öğrenme oldukça büyük ve karmařık olasılıklı model sistemlerinin eęitimi ve bu sistemlere öğretim yeteneęinin kazandırılması hususunda çok etkili bir yöntemdir [4]. Özellikle derin yapay sinir aęlarında kullanılan geleneksel yaklařımlara göre etkileyici sonuçlar vermektedir. Derin sinir aęları, verileri iřleyen birimlerden oluřan oldukça fazla sayıda doęrusal olmayan katmanlardan oluřmaktadır [5]. Olasılıklı model yapısında mevcut verilerin eęitimi esnasında katmanlı denetimsiz ön eęitim yöntemi kullanılmaktadır. Derin aędaki katmanlar düşük seviyeli özelliklerden yüksek seviyeli özelliklere doęru ilerleyen hiyerarřik bir yapıya sahiptir [6]. Derin sinir aęları, katlamalı derin sinir aęları ve derin inanç aęları derin

öğrenmenin çeşitli bazı mimarileridir. Bu mimari türleri doğal dil işleme, ses tanıma, bilgisayarlı görme, otomatik konuşma tanıma gibi farklı alanlarda kullanılmaktadır [7].

Derin sinir ağları genellikle üst üste yığılmış pek çok RBM'den meydana gelmektedir. RBM yapıları birbirlerine bağlıdır. Bir RBM'nin gizli katmanı bir diğer RBM'nin görünür katmanı olacak şekilde bir bağlantı söz konusudur. Bir RBM'nin katmanlı ön eğitimi veriler için daha doğru bir model bulmayı kolaylaştırmaktadır [8]. Araştırmacılar tarafından yapılan birçok çalışma çok aşamalı öğrenme yapılarının geleneksel öğrenme algoritmalarına göre daha iyi sonuçlar verdiğini kanıtlamaktadır.

Bir RBM'nin eğitilmesi oldukça zor olduğundan RBM'nin eğitilmesi için yararlı bir yöntemin olması önemlidir. Öğrenme parametreleri dikkatlice belirli veri setlerine ve RBM yapılarına uygun bir şekilde ayarlanmalıdır. Aksi halde veri dağılımı modellemesini gerçekleyen öğrenme algoritması kolayca başarısız olabilir. Öğrenme sırasında azalan olasılık değerleri bu sorunu gözler önüne sermektedir. Bu sorunlar RBM yapılarının daha sofistike makine öğrenme görevlerinde kullanılma oranında düşüşe neden olmaktadır [9].



Şekil 1.1: Derin ağ katmansal yapısı.

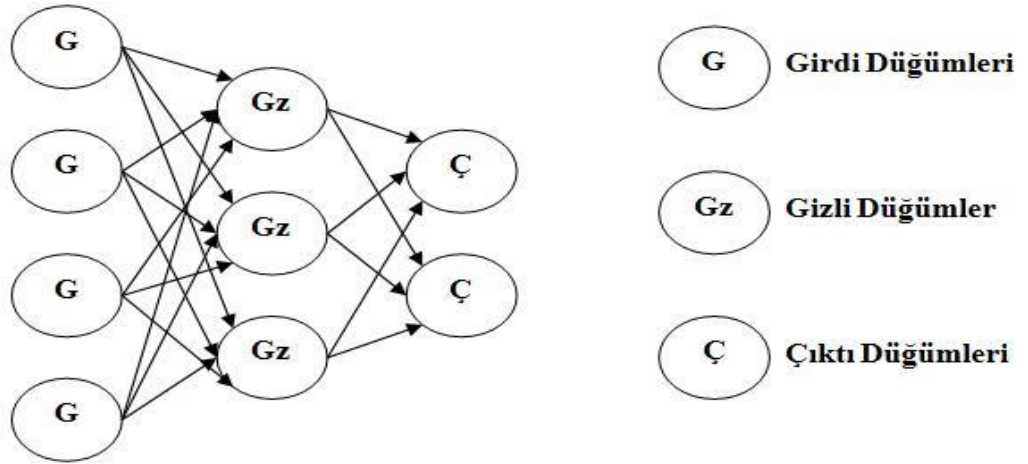
2. GENEL KISIMLAR

2.1. YAPAY SİNİR AĞLAR

YSA, insan beyninin bir özelliği olan yeni bilgiler türetebilme, oluşturabilme, keşfedebilme gibi kabiliyetleri öğrenme yöntemiyle otomatik olarak gerçekleyen bilgisayar tabanlı bir sistemdir. Bu kabiliyetleri bilinen programlama dilleriyle oluşturmak imkânsıza yakın olduğundan YSA bu işi kolaylaştırmak için geliştirilmiştir.

İnsan beyninin sınıflandırma, optimizasyon, ilişkilendirme, öğrenme gibi işlevleri YSA tarafından başarılı bir şekilde gerçekleştirilerek bir çok probleme oldukça faydalı çözümler üretmektedir.

Yapay Sinir Ağları birbirine bağlı ve paralel bir düzen içinde çalışan düğümlerden oluşmaktadır, bu düğümlerin birbirine bağlanmasıyla aşağıdaki gibi bir ağ yapısı oluşmaktadır.



Şekil 2.1: Yapay Sinir Ağlarının temel gösterimi.

Düğümleri birbirine bağlayan bağlantıların her birinin bir değeri vardır. Bu değerler öğrenme yoluyla elde edilen ve ağın yapısı sayesinde farklı düğümlere iletebilmektedir.

Yapay Sinir Ağının temel vazifesi, girdilere karşılık çıktı üretmektir, bunun yapılabilmesi için YSA girdiler ile ilişkili örneklerle çalıştırılır ve genelleme yapabilme yeteneği kazandırılır ve örneklerle benzer girdilere karşılık çıktılar üretilir[10].

2.1.1. Yapay Sinir Ağlarının Özellikleri

2.1.1.1. Doğrusal Olmama

YSA'nın temel işlem birimi olan sinir hücresi doğrusal değildir bundan dolayı sinir hücrelerinin bir araya gelmesiyle oluşan genel yapı da doğal olarak doğrusal değildir.

2.1.1.2. Öğrenme

YSA sisteme giren ve çıkan veriler arasında bir eşleştirme yapabilir. Bu eşleştirme bir tür haritalandırma olarak görülebilir. YSA'nın istenilen davranışı gösterebilmesi için bu doğrultuda ayarlanması gerekmektedir. Bu da sinir hücreleri arasında doğru bağlantı ve uygun ağırlıkların belirlenmesiyle mümkündür.

2.1.1.3. Adaptasyon

Çevre şartlarından kaynaklanan değişimlere bağlı olarak YSA, bağlantı ağırlıklarını şartlara göre ayarlayabilme yeteneğine sahiptir.

2.1.1.4. Genelleme

YSA'daki her bir nöron, diğer nöronlarda meydana gelen değişikliklerden etkilenebilmektedir. YSA eğitim esnasında karşılaşmadığı örneklerle de istenen tepkiyi verebilir.

2.1.1.5. Hata Toleransı

YSA, birtakım farklı yöntemlerde olduğu gibi küçük hatalardan etkilenmez. Mantıksal ve fiziksel hata toleransı fazladır. YSA'daki bazı bağlantıların işlevselliğini yitirmesi YSA'nın üreteceği çıktıyı çok fazla etkilemez. YSA'da veriler tek bir yerde toplu halde değildir bütün sisteme dağılmış halde bulunur bundan dolayı hata toleransı yüksektir.

2.1.1.6. VSLI Uygulanabilirlik

Yapay sinir ağları kullanılarak oldukça büyük ölçekli bütünleşik devre uygulamaları yapılabilir.

2.1.1.7. Analiz ve Tasarım Birliđi

Yapay sinir ađlarının temel iřlem birimi olan hücrenin yapısal özellikleri bütün YSA yapılarında hemen hemen aynıdır. Böylece deđişik uygulama alanlarına sahip YSA modelleri birbiriyle benzerlik gösteren teorileri ve algoritmaları uygulayabilirler. Bu özellik problemlerin YSA kullanılarak çözümünde büyük bir kolaylık sağlamaktadır[11].

2.1.2. Yapay Sinir Ađlarının Avantaj ve Dezavantajları

2.1.2.1. Avantajları

- Daha önce karşılařmadıđı örnekleri sistemden geçirerek onlar hakkında bir çıktı üretebilme yetisine sahiptir.
- Yapay Sinir Ađları doğrusal deđildir bu durumu bir avantaj olarak kullanıp oldukça karmařık problemleri doğrusal çözüm tekniklerinden daha doğru çözerler. Doğrusal olmayan problemleri matematiksel olarak çözmek zordur.
- Ađdaki ađırlık katsayıları deđiřtiđinde problemin çözümü için YSA kendini bu duruma adapte edebilme özelliđine sahiptir.
- Geleneksel çözüm teknikleri sistemde meydana gelebilecek aksaklık ve hatalara karşı çok hassastır. Küçük bir hata bile yanlış sonuçlar elde etmeye sebebiyet vermektedir. Fakat yapay sinir ađlarda durum bundan çok farklıdır çünkü YSA'da sistemdeki bir veya birkaç nöron hasar görse bile geleneksel yöntemlere nazaran oldukça az hasar görür.
- Yapay sinir ađları eldeki verilerden faydalanarak gizli iliřkileri ortaya çıkarabilmektedir. Ađın bu özelliđi uygulama açısından çok önemlidir.
- Yapay sinir ađları daha önce karşılařtıđı örneklerden faydalanarak öğrenebilmektedir, bir defa eđitildikten sonra yeni gördüđü bir veri kümesine cevap verebilmektedir.
- Yapay sinir ađları matematiksel bir gösterime gereksinim duymazlar.

2.1.2.2. Dezavantajları

- Farklı sistemlere uyarlanması kolay deđildir.
- Eđitilmeleri uzun zaman alabilmektedir bundan dolayı zaman ve para açısından maliyeti yüksektir.

- Bir problemin çözümünde hata yapabilir ve istenilen çözümü bulamayabilirler. Bunun nedeni ağ eğitilirken herhangi bir fonksiyon kullanılmamasıdır. Bazı durumlarda fonksiyon bulunsa bile yeterli veri bulunamayabilir.
- Bazı durumlarda ağın verdiği sonuçları değerlendirmek oldukça zordur bunun nedeni de sistemin içinde ne olduğu bilinmediğindendir.
- Ağın kalitesi ve kapasitesi, uygulama esnasındaki hızı ile doğru orantılıdır. Düğümlerin sayısındaki artış sistemin çalışma zamanının artmasına sebep olmaktadır [12].

2.1.3. Yapay Sinir Ağlarının Uygulama Alanları

Yapay Sinir Ağları başta tahmin, modelleme, sınıflandırma uygulamaları olmak üzere pek çok alanda kullanılmaktadır. YSA'nın uygulandığı başarılı çalışmalar incelendiğinde YSA modellerinin daha çok matematiksel bir çözümün mümkün olmadığı ve herhangi bir algoritmanın bulunmadığı kesin olmayan, karmaşık, gürültülü, çok boyutlu ve sadece örneklerin bulunduğu problemlerin çözümünde kullanıldığı görülmektedir. Yapay sinir ağlarının gerçekleştirdiği fonksiyonlar örüntü eşleştirme, örüntü tanıma, optimizasyon, veri sıkıştırma, sınıflandırma, ilişkilendirme, sinyal filtreleme, doğrusal olmayan sistem modelleme ve sinyal işleme olarak sıralanabilir.

2.1.3.1. YSA'nın Kullanıldığı Sektörler

- Eğlence: Özel efektler, animasyonlar vs.
- Elektronik: Çip bozulma analizi, kod sırası öngörüsü, doğrusal olmayan modelleme vs.
- Bankacılık: Müşteri analizi, kredi uygulamaları geliştirilmesi, bütçe yatırım tahminleri vs.
- Finans: Pazar performans analizi, bütçe kestirimi, kıymet biçme, hedef belirleme vs.
- Sigortacılık: Uygulama politikası geliştirme, ürün optimizasyonu vs.
- Uzay: Otomatik pilot uygulamaları, uçuş simülasyonları vs.
- Robotik: Uzaktan kumandalı sistemler, kaldıraç robotlar, görsel sistemler, yörünge kontrol vs.
- Dil: Dil tercüme, sözcük tanıma, yazı ve konuşma çevrimi vs.

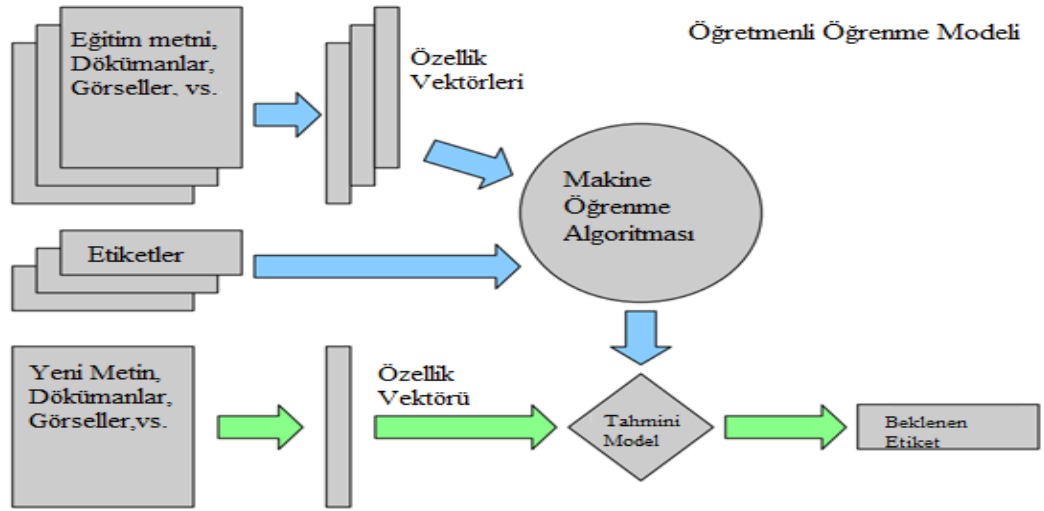
- Sağlık: EEG, ECG, MR, ilaç etkileri analizi, kan analizi sınıflandırma, kalp krizi teşhis ve analizi, göğüs kanseri teşhis ve tedavisi vs.
- Güvenlik: Parmak izi tanıma, retina tarama, yüz eşleştirme, kredi kartı hileleri saptama vs.
- Petrokimya: Arama, verim analizi vs.
- Telekomünikasyon: Eko ve gürültü sönmülendirilmesi, filtreleme, ses ve görüntü işleme, görüntü ve veri karşılaştırma, trafik yoğunluğunun kontrolü ve anahtarlama vs.
- Üretim: Kalite kontrol, ürün tasarımı, üretim işlem kontrolü, dayanıklılık kontrolü, makine yıpranmalarının tespiti vs.
- Savunma: Radar, sensör sonar sistemleri, sinyal işleme, görüntü işleme, hedef seçme, silah yönlendirme vs.
- Otomotiv: Rehber, yol koşullarına göre sürüş analizi vs [13].

2.1.4. Yapay Sinir Ağlarda Öğrenme

Yapay sinir ağları sistemlerinin çalışma prensipleri örneklere dayanmaktadır. YSA yapıları eğitilirken farklı yöntemler kullanılmaktadır. Bu farklılıklar kullanılan öğrenme algoritmalarının birbirinden farklı olmasından kaynaklanmaktadır.

2.1.4.1. Öğretmenli Öğrenme

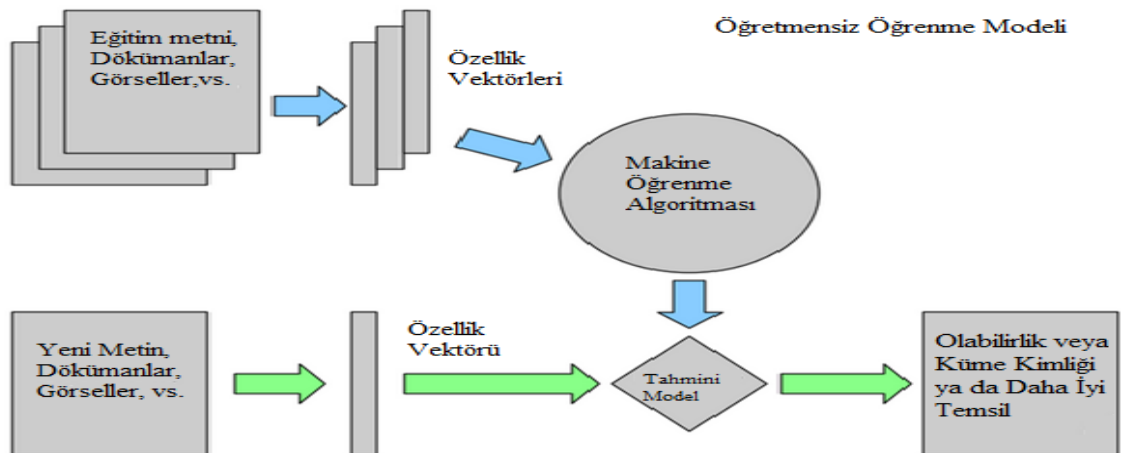
Bu öğrenme şeklinde ağın olayı kavraması için bir eğiticiden yardım almaktadır. Öğretmen ağa giren bazı belirli girdiler için oluşturulması gereken çıktı değerlerini ağa verir. Sistemin yapması gereken iş sisteme öğretmen tarafından veriler çıktıları ve girdileri haritalandırmaktır. Sistemdeki girdiler ve çıktılar arasındaki ilişkiler düğümler arasındaki ağırlıklar vasıtasıyla öğrenilmektedir.



Şekil 2.2: Öğretmenli öğrenme modelinin işleyiş diyagramı [14].

2.1.4.2. Öğretmensiz Öğrenme

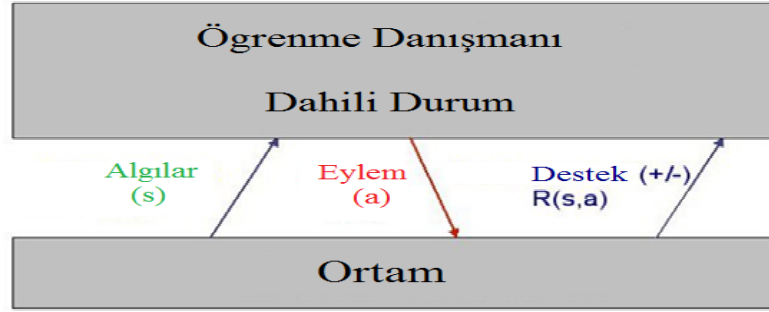
Öğrenmenin bu türünde ise ağız öğrenmesi için bir öğretmen ya da diğerk adıyla bir eğitici bulunmamaktadır. Bu öğrenme şeklinde sisteme sadece girdiler verilir ve bunlardan çıktılar üretilmesi beklenir. Bu tür öğrenme genellikle sınıflandırma problemleri için kullanılır. Sistem öğrenme işlemini bitirdikten sonra oluşturulan çıktıların anlamlandırılması için kullanıcı tarafından çıktıları etiketlendirme işleme yapılmalıdır.



Şekil 2.3: Öğretmensiz öğrenme modelinin işleyiş diyagramı [14].

2.1.4.3. Destekleyici Öğrenme

Bu öğrenme türünde Öğretmenli Öğrenme’de olduğu gibi bir öğretmen ağa yardımcı olmaktadır. Fakat bu sistemde öğretmen girdilere karşılık gelmesi gereken çıktıları sisteme vermek yerine sistemin bir çıktı üretmesini bekler ve üretilen çıktının doğruluğunu ve ya yanlışlığını gösteren bir sinyal üretir. Ağ öğretmenden gelen doğru ve ya yanlış sinyali doğrultusunda hareket ederek öğrenme sürecine devam eder.



Şekil 2.4: Destekleyici öğrenme mekanizması [15].

2.1.4.4. Karma Öğrenmeler

Daha önce bahsettiğimiz öğrenme yöntemlerinin tek bir sistemde beraber kullanıldıkları ağlardır. Hem öğretmenli hem de öğretmensiz öğrenmenin kısmen kullanıldığı sistemlerdir.

2.1.5. Yapay Sinir Ağlarda Öğrenme Kuralları

Yapay Sinir Ağlarda ağın eğitimi belli bir kurala göre yürümektedir, bunu yaparken hangi öğrenme tekniğini kullandığı önemli değildir. Öğrenme kuralları iki başlık altında incelenebilir.

2.1.5.1. Çevrimiçi Öğrenme Kuralları

Bu tip kurallar gerçek zamanlı olarak çalışır. Sistemde bir taraftan öngörülen işlemleri gerçekleştirirken diğer taraftan da öğrenmeye devam etmektedir.

2.1.5.2. Çevrimdışı Öğrenme Kuralları

Bu tip öğrenme stratejisinde ağ bir yandan işlevini yürütürken diğer taraftan da eğitilmesi söz konusu değildir [16].

2.1.5.3. Hebb Kuralı

En eski ve en ünlü öğrenme algoritması olan Hebb öğrenme algoritması Kanadalı psikolog Donald Hebb tarafından 1949 yılında biyolojik temele dayalı olarak geliştirilmiştir. Bu algoritmanın mantığı oldukça basittir. Eğer A nöronu bir başka B nöronundan bir girdi alıyorsa ve eğer her ikisi de aktifse (A) ve (B) nöronları arasındaki ağırlıklar artar.

$$\Delta w_{jk} = \alpha \cdot y_j \cdot x_k \quad (2.1)$$

Bu formülde $\Delta w_{jk} = w_{jk}^{(t+1)} - w_{jk}^{(t)}$ ve $\Delta > 0$ dır. (t = zaman).

Burada w_{jk} nöron u_k 'den nöron u_j 'ye olan ağırlık, y_j nöronun çıktısı ve x_k ise u_k nöronunun çıktısıdır. Burada kullanılan α ise pek çok öğrenme algoritmasında kullanılan “öğrenme katsayısı” veya “öğrenme oranı” olarak adlandırılmaktadır. Öğrenme oranı 0 ile 1 arası değer alır ve bu değer de ağın öğrenme hızını belirlemektedir. Öğrenme katsayısının değeri büyük olduğu zaman sistemde hızlı öğrenme, küçük olduğu zamanda daha yavaş öğrenme işlemi gerçekleştirilmektedir. Eğer ağın öğrenme hızı yüksekse bu durum YSA'nın “genelleme” kapasitesini azaltmaktadır. Genelleme yeteneği ağına verilen eksik ve gürültülü verilerden doğru çıktılar üretebilmesi bakımından son derece önemlidir. Hebb algoritmasında ağ eğitilmeden önce ağırlıklarının 0 olması gerektiğidir [17].

2.1.5.4. Delta Kuralı

Delta kuralı en çok kullanılan öğrenme kurallarından biridir. Bu kural Hebb kuralının daha geliştirilmiş halidir. Gerçekleşen çıktı ile beklenen çıktı arasındaki farkı (delta) minimuma indirmek ve istenilen sonuca daha da yaklaşmak için ağırlıkların sürekli güncellenmesi mantığına dayanmaktadır. Ağ hatasının karesini minimize etmek adına ağırlıklar sürekli güncellenmektedir. İlk katmana gidene kadar hata bir önceki katmana geri çoğaltılır. Hatayı geriye doğru yaymak vasıtasıyla düzeltilmesini amaçlamaktadır. Yapılan hata eğimli iniş adı verilen süreçle minimuma indirgenmeye çalışılır. Bu kural yapay sinir ağlarda Hebb kuralı ile birlikte sistemdeki hata miktarının belirlenmesinde en çok kullanılan öğrenme kuralıdır. En küçük ortalama kareler yöntemi de uzayda bulunan iki farklı değişkenin birbirine yaklaştırılması mantığına dayanmaktadır.

$$\Delta w_{ij} = 2\mu x_i (\text{beklenen çıktı} - \text{gerçekleşen çıktı})_j \quad (2.2)$$

Yukarıdaki formülde, ağdaki düğümleri birbirine bağlayan bağlantılardaki ağırlık değerlerindeki değişim miktarı delta kuralı baz alınarak hesaplanmaktadır. Kurala göre bağlantının ağırlığındaki değişim, eğitim oranı olan μ , söz konusu bağlantının giriş değeri x_i ve söz konusu bağlantıda beklenen ve gerçekleşen çıktı değerleri arasındaki farkın çarpımına eşittir [18].

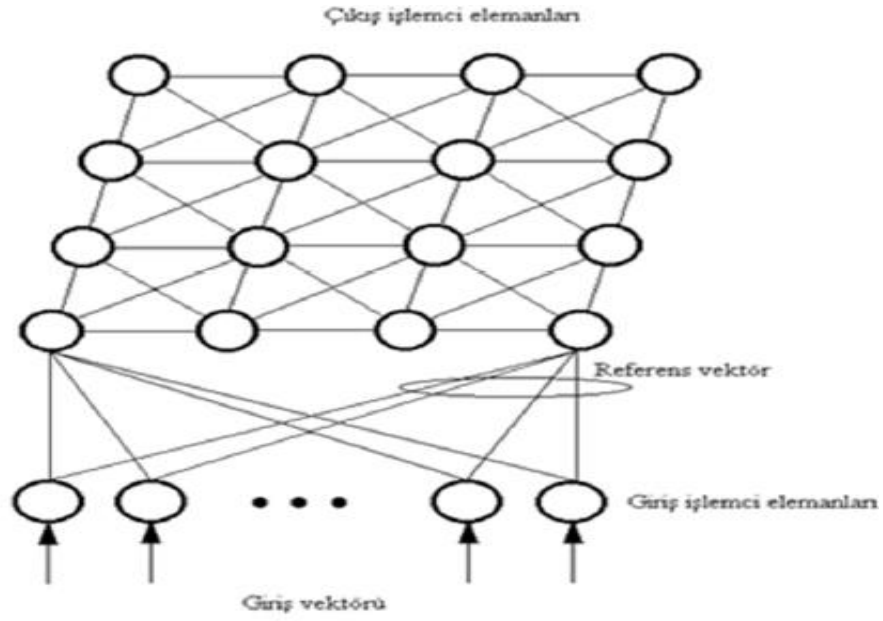
2.1.5.5. Kohonen Kuralı

Kohonen öğrenme kuralında, giriş ve çıkış olmak iki tabakadan oluşan Kohonen tabakasındaki işlem elemanlarının birbiriyle yarışması prensibine dayanır. Çıkış tabakasındaki düğümler(işlem elemanları) iki boyutlu aralıklarda bir düzenle yer almaktadırlar. Çıkış tabakasındaki tüm düğümler giriş tabakasındaki düğümlerle bağlantılıdır. Bağlantıların ağırlıkları kullanılarak çıkış düğüm elemanının referans vektörünün elemanları oluşturulur.

Kohonen ağınnın öğrenme aşamaları aşağıda sıralanmış halde verilmiştir:

- Çıkış düğüm elemanlarının bütün referans vektörlerinin rastgele küçük değerlere çekilmesi,
- Bir giriş deseninin alınması,
- Bu giriş desenine en yakın referans vektörüne sahip düğümü kazanan çıkış düğümü olarak belirlenmesi (giriş vektörü ve referans vektörü arasındaki öklit uzaklığı genellikle uzaklık ölçüsü olarak alınır),

Kazanan düğümü ve ona komşu olan diğer düğümlerin referans vektörlerinin güncellenmesi, yapılan bu işlem referans vektörlerini giriş vektörüne yakınlaştırır. Bu ayarlama işlemi en fazla kazanan düğüme yapılır ve bu düğümden uzaklaştıkça yapılan ayarlama işlemi azalmaktadır. Öğrenme işlemi ilerledikçe birbirine komşu düğümlerin sayısı azalmaktadır ve öğrenme işlemi sonlandığında sadece kazanan düğümün referans vektörü ayarlanmaktadır [19].



Şekil 2.5: Kohonen öğrenme kuralı.

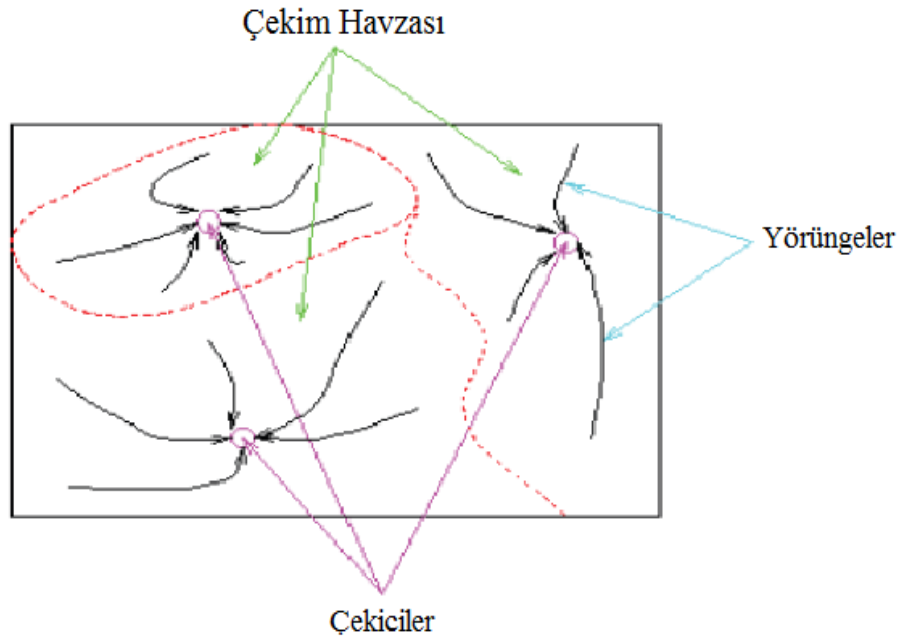
2.1.5.6. Hopfield Kuralı

Hopfield kuralının uygulandığı Hopfield ağı, diğer yapay sinir ağ modellerinden farklıdır. Bu farkın nedeni tek bir katmana sahip olmasıdır yani giriş ve çıkış katmanı aynı katmandadır. Tek katmandan oluştuğu için düğüm sayısı sınırlıdır ve dolayısıyla sınırlı bir ağ kapasitesi vardır. Hopfield ağındaki her bir düğüm iki duruma sahiptir bunlar açık (+1) ve kapalıdır (-1). Ağdaki x ve y düğüm çifti a_{xy} ağırlığıyla birbirine bağlıdır. a_{xy} ağırlığı x düğümünün çıkış değerinin y hücresine olan etkisini belirtmektedir.

Hopfield ağında yapılan işlemler iki aşamada gerçekleşmektedir bunlar depolama ve geri çağırma. Birinci aşama olan depolama aşamasında ağı eğitimi için n boyutlu karakter veya desenler hafızada depolanırlar. Daha sonra test işlemi için ağda yapılmak istenen örüntü tanılamayı gerçekleştirmek adına ağı hafızasında depolanan veriler geri çağırılarak ağa girdi olarak verilen bir takım gürültülü karakterler veya desenler ile karşılaştırılarak bazı ipuçlarını da kullanarak bu karakter veya desenleri eşleştirir ve gerekirse bozuk ve gürültülü kısımlar düzeltilir. Gürültülü ve ipuçları kullanılan örüntüler ağ tarafından değerlendirilir ve enerji açısından çukur olan bölge tespit ederek

örüntüyü bu yöne doğru kaydırır. Bu yakınsama işlemi tekrar edilerek verilen örüntünün enerji açısından çukur olan hangi bölgeye yakınsadığını tespit eder.

Bu öğrenme kuralı kullanılarak pek çok resmin depolandığı bir ağ sistemine, ipucu olarak ya gürültülü bir resim ve ya bir resmin bir kısmı girdi olarak verilir ve Hopfield ağının çalışma prensibi doğrultusunda iterasyonlar gerçekleştirilerek bu ipucu girdinin sistemde daha önce depolanan hangi resmin parçası olduğunu tespit edilir [20].



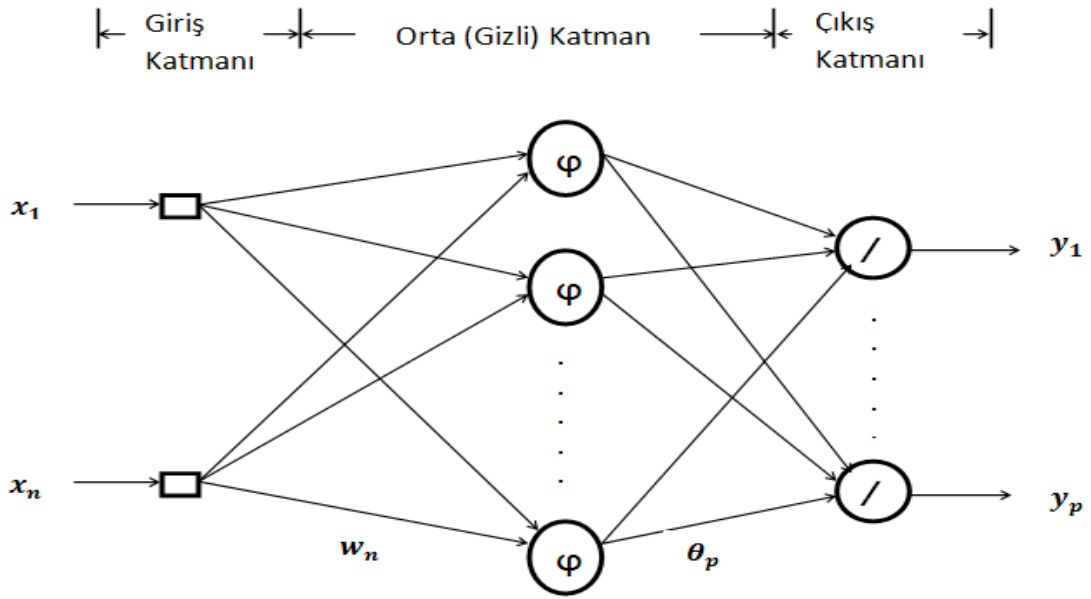
Şekil 2.6: Hopfield öğrenme kuralı.

2.1.6. Yapay Sinir Ağ Türleri

2.1.6.1. İleri Beslemeli YSA

Bu tür YSA'da hücreler katmanlar şeklinde tasarlanmaktadır. Bir katmandaki çıkış değerleri hücreleri birbirine bağlayan bağlantı ağırlıkları üzerinden diğer katmana girdi değerleri olarak verilmektedir. Giriş katmanı dış ortamdan aldığı verileri herhangi bir değişiklik uygulamadan orta katmana iletmektedir. Bilgiler orta ve çıkış katmanında gerekli işlemlerden geçirilerek son halini alır. Böylece ileri beslemeli yapay sinir ağlar doğrusal olmayan statik bir işlev gerçekleştirir.

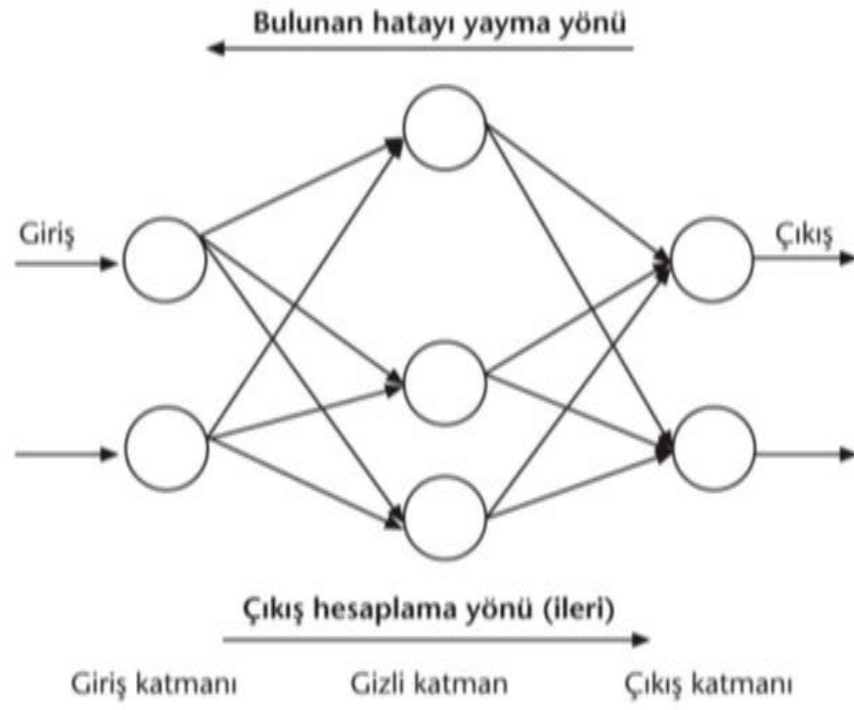
İleri beslemeli üç katmanlı bir YSA'nın orta katmanında yeteri kadar hücre bulunması durumunda herhangi bir fonksiyonu istenilen doğrulukta yaklaştırabileceği belirlenmiştir. En çok bilinen geriye yayılım öğrenme algoritması bu tip ağların eğitimde kullanılmaktadır ve çoğu zaman bu tür ağlara geriye yayılım ağları da denilmektedir. İleri beslemeli yapay sinir ağları nesne tanıma ve sinyal işlemenin yanı sıra sistemlerin tanılanması ve denetiminde de yaygın olarak kullanılmaktadır. Aşağıdaki şekilde üç katmanlı bir ileri beslemeli bir YSA yapısı verilmiştir.



Şekil 2.7: Üç katmanlı ileri beslemeli YSA yapısı.

2.1.6.2. Geri Beslemeli YSA

Geri beslemeli Yapay Sinir Ağlarda sistemdeki hücrelerden en az bir hücrenin çıkış elemanı ya kendisine ya da diğer hücelere giriş elemanı olarak verilmektedir. YSA' lar da geri besleme genellikle bir geciktirme elemanı üzerinden yapılır. Geri besleme aynı katmandaki hücreler arasında olabileceği gibi farklı katmandaki hücreler arasında da olabilir. Sistem bu yapıyla doğrusal olmayan dinamik bir davranış sergilemiş olmaktadır. Sistemlere uygulanan geri besleme yapısının yapılışına göre farklı davranışlar sergileyen ve farklı yapıda olan YSA yapıları oluşturulabilmektedir.



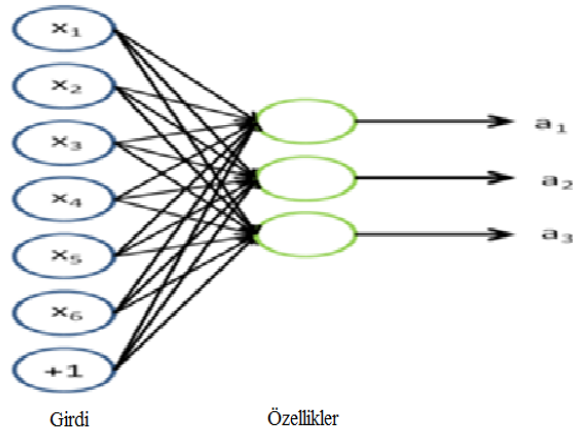
Şekil 2.8: Geri beslemeli Yapay Sinir Ağı.

3. MALZEME VE YÖNTEM

3.1. DERİN ÖĞRENME

3.1.1. Öz-Öğretim Öğrenmesinden Derin Öğrenmeye

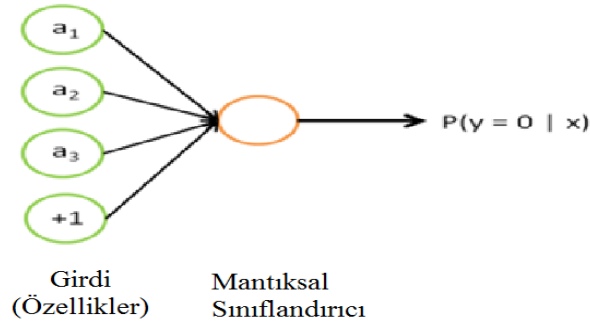
Öz-Öğretim öğrenmesinde ilk olarak etiketlenmemiş veriler üzerinden seyrek bir Autoencoder (AE) eğitilir. Daha sonra verilen X örneğinde, gizli katman kullanılarak a özelliklerini çıkarımı yapılır. Bu durumun şeması da aşağıdaki gibidir.



Şekil 3.1: Verilen x örneğinden gizli katman kullanılarak a özellik çıkarımı yapılması.

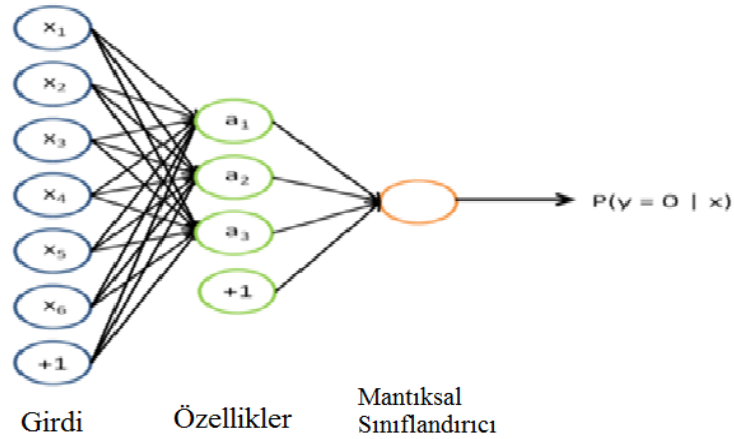
Burada amacımız y etiketlerini tahmin etmek olan bir sınıflandırma problemini çözmektir. M sayıdaki etiketlenmiş eğitim seti örneklerimiz mevcuttur:

$\{ (x_1^{(1)}, y^{(1)}), (x_1^{(2)}, y^{(2)}), \dots, (x_1^{(m)}, y^{(m)}) \}$. Daha öncede belirtildiği gibi orijinal özelliği (x^i) , seyrek AE'dan üretilen a^i ile yer değiştirilebilir. Bu adımı gösteren lojistik regresyon birimi aşağıda turuncu ile gösterilmiştir.



Şekil 3.2: Lojistik regresyon birimini gösteren şema.

Bu yöntemi kullanarak öğrendiğimiz genel sınıflandırıcıları düşünelim, özellikle bizim sınıflandırıcının bir test örneğinden (π) yeni bir tahmin $p(y = 1 | x)$ eşleştirmesi yaparken kullandığı fonksiyonu düşünelim. Yukarıdaki iki şekli üst üste koyarak bu fonksiyonun bir gösterimini elde edebiliriz.



Şekil 3.3: Verilen x örneğinden gizli katman kullanılarak a özellik çıkarımı aşaması ve lojistik regresyon birimini birlikte gösteren şema.

Bu modelin parametreleri iki aşamada eğitilmiştir: Birinci aşamada ağırlıklar hesaplanırken x girdilerinden a gizli birimini aktifleştirmek için gizli birim seyrek AE'nin bir parçası olarak eğitilmiştir. İkinci aşamada ağırlıkları a gizli birimi elamanlarından y çıktısına eşleştirmek için lojistik regresyon kullanılarak eğitilmiştir. Sonuç olarak sınıflandırıcımızın son halini bir bütün olarak büyük bir YSA olarak değerlendirebiliriz. Modelimizin başlangıçta eğitilmiş (birinci aşama ve ikinci aşamada

eğitilen) bir dizi parametreye sahip olmak, modelimizdeki eğitim hatalarını daha da azaltmak için bize yardımcı olacaktır. Özellikle mevcut parametrelere ince ayar yaparak (gradyan düşüşü yada L-BFGS uygulayarak) etiketli eğitim seti örneklerindeki eğitim hatalarını azaltabiliriz $\{ (x_1^{(1)}, y^{(1)}), (x_1^{(2)}, y^{(2)}), \dots, (x_1^{(m)}, y^{(m)}) \}$. İnce ayar yapılırken bazen orijinal denetimsiz özellik öğrenme adımları (AE yada lojistik sınıflandırıcı) ön eğitim olarak adlandırılır. İnce ayarın etkisi, etiketlenmiş verilerde ağırlık değiştirmek $w(1)$ için kullanılabilir olmasıdır. Böylece gizli birim tarafından özellik çıkarımı yapılan a özelliklerine de gerekli ayarlamalar yapılabilir.

Peki ince ayarı ne zaman kullanmalıyız? İnce ayar genellikle büyük boyutta etiketli eğitim setlerinde kullanılır. Bu büyüklükteki bir eğitim setinde ince ayar yapmak sınıflandırıcınız performansını önemli ölçüde artırır. Eğer büyük etiketsiz bir veri setiniz (denetimsiz özellik öğrenme / ön-eğitim) varsa ve bu etiketsiz veri setine nispeten küçük bir etiketli eğitim setiniz varsa ince ayar yapmanın çok faydalı olacağı pek olası değildir [21].



Öz-Öğretimli Öğrenme

Şekil 3.4: Öz-Öğretimli öğrenme örnekleri.

3.1.2. Boltzmann Makinesi

Boltzmann Makinesi, G. E. Hinton, T. J. Sejnowski ve D. H. Ackley tarafından tanıtılan bir enerji modelidir. Topolojik anlamda Boltzmann Makinesi ikili rastgele birimlerden oluşan yönsüz bir grafiktir. Birbirine bağlı birimler arasında ağırlıklar mevcuttur ve her bir birimin bir eşik (bias) değeri vardır. Sapmalar ve ağırlıklar, ikili birimleri üzerine bir dağıtım belirlenebilen bir modelin parametreleridirler. İkili birimlerle yapılandırılmış bir modelin sahip olduğu toplam enerjiyi s ile gösterirsek bunun hesaplanması aşağıdaki gibidir.

$$E(s) = \sum_i^{|S|} \sum_{j < i} w_{i,j} s_i s_j + \sum_i^{|S|} b_i s_i \quad (3.1)$$

Burada s_i , i ikili biriminin etkinleştirmesini gösterir. $w_{i,j}$, i birimi ve j birimini birbirine bağlayan ağırlığı göstermektedir. b_i 'de i birimine ait eşik değerini göstermektedir. Ağırlıklar ve eşikler gerçek değerleri belirtmektedir. Mevcut durumu verilen modelin herhangi bir biriminin (s_i) aktif olasılıkları tarafından tanımlanmış geçiş olasılıkları, model yapılandırılmaları, ya da durumlar Markov zinciri üzerinden tanımlanır:

$$P(s_i|s) = \frac{1}{1 + \exp(-b_i - \sum_{j \neq i} w_{i,j} s_j)} \quad (3.2)$$

Ağırlıkları ve eşikler sabit olarak verilmiş model durumları için tam koşullu dağıtımı kullanmak mümkündür, bu durumu örneklemede ise Gibbs örnekleme yöntemi kullanılabilir. Bu şekilde örneklenebilen ortak dağıtım şekilleri modelin minimum enerji durumlarına karşılık gelmektedir. Tam yapılandırılmış bir modelin olasılığı yine o modelin bir enerji fonksiyonu şeklinde tanımlanır:

$$P(s) = \frac{E(s)}{\sum_{t \in S} E(t)} \quad (3.3)$$

3.1.3. Boltzmann Makine'sinde Öğrenme

Boltzmann Makinesi yapılandırmaların bazı verilerin sahip olduğu düşük enerji haritalarını anlamlı bir gösterime büründürüldüğü bir ilişkisel bellek olarak kabul edilebilir. Model üzerinden bazı ikili verilerin elde edilebilirliğini en üst seviyeye çıkarmak için model parametrelerini güncelleyebilecek bir araca ihtiyaç duyulmaktadır; öyle ki veri gösterimleri modelin düşük enerji ayarlarını sıkı bir şekilde bağlayabilecektir.

Veriler modelin sahip olduğu düğümlerin bir alt kümesi üzerinden tanımlanabilir, geriye kalan düğümlerde verilerin gizli değişkenleri işlevini görmektedir. Hinton'ın çalışmaları şunu gösterdi ki gradyan alçalması, değişen parametrelere göre kayıt olasılığı gradyanı üzerinden kullanılabilir. Gradyan'ın bu şekilde uygulanması verilerin tanımlandığı düğümler üzerinden lokal minimuma sahip parametrelerle temsil edilmesini sağlar. Böylece bağlantı ağırlıklarının sahip olduğu öğrenme sinyalleri aşağıdaki gibi olur:

$$\frac{\partial L}{\partial w_{i,j}} = (s_i, s_j)_{\text{data}} - (s_i, s_j)_{\text{model}} \quad (3.4)$$

Burada $(\cdot)_{data}$, gizli olmayan birimler üzerinden modeli dağılımı ile ilgili verilen veri için beklenen değerdir, $(s_i, s_j)_{model}$ ise modelin ortak dağıtımını üzerinden beklenen değerdir. Ancak bu değerlerin tahminini yapmak için modelin gerçek ortak dağıtımını örneklemeye yardımcı olacak bir yol gereklidir.

3.1.3.1. Gibbs Örneklemesi

Markov Zinciri Monte Carlo yöntemleri, kapalı formda ifade edilmemiş dağılımları örneklemek için bir yol sağlamaktadır. Ancak, tam koşullu dağılımın kapalı formu her bir değişken için varsa eğer daha sonra bu değişkenler tekrarlı bir şekilde örneklenebilirler. Gibbs örnekleme yöntemi tüm değişkenleri sırasıyla o değişkenlerin koşullu dağıtımını örnekleyerek ilerler. Ağ'ın ortak dağılımdan bu şekilde elde edilen örnekler Markov zinciri yakınsandıktan sonra çizilecektir.

Kısıtlı Boltzmann Makinesi gibi simetrik bağlantıları olan iki tabakalı bir ağda, belirli bir katmandaki her düğümün Markov paketi (blanket) sadece komşu katmandaki tüm düğümleri içeren kümesidir. Bütün bir tabakanın sadece bir kez örneklendiği böyle bir ağ göz önüne alındığında tıkalı Gibbs metodunun kullanılması uygun olacaktır. Tıkalı Gibbs örneklemesini kullanmak ortak dağıtım için kullanılan Markov zincirinin yakınsamasını önemli ölçüde hızlandırır [22].

3.1.3.2. Gizli Birimler Olmadan Öğrenme

Eğitim seti verilmiş durum vektörleri (veriler) göz önüne alındığında öğrenme aşaması söz konusu durum vektörlerini iyileştirirken bağlantı ağırlıkları ve bias (parametreleri) kullanmaktadır. Aslında buradaki asıl amaç yüksek olasılıklı eğitim vektörlerine sahip bir Boltzman dağıtımını tanımlayan bağlantı ağırlıklarını ve bias'ları tanımlamak. (3.5) denklemi göz önüne alınarak ve (3.6) denklemi de kullanılarak aşağıdaki gibi gösterilebilir.

$$\frac{\partial E(v)}{\partial w_{ij}} = -s_{vi}s_{vj} \quad (3.5)$$

$$P(v) = \frac{e^{-E(v)}}{\sum_u e^{-E(u)}} \quad (3.6)$$

$$\langle \partial \log P(v) \partial w_{ij} \rangle_{data} = \langle s_i s_j \rangle_{data} - \langle s_i s_j \rangle_{model} \quad (3.7)$$

Burada $\langle \cdot \rangle_{\text{data}}$ veri dağıtımından beklenen değeri, $\langle \cdot \rangle$ model Boltzman makinesinin 1 derece sıcaklıkta denge dağıtımındaki durum vektörlerini örneklemesinden beklenen değeri ifade etmektedir. Kayıt (log) olasılığındaki gradyan artışını gerçekleştirmek için Boltzman Makinesi, denge dağılımı örneklendirilirken gözlenebilir veri oluşturacaktır.

3.1.3.3. Gizli Birimler ile Öğrenme

Eğer Boltzmann Makinesi durumları gözlemlenebilir bazı görünür birimler ve örnekler gözlemlenmesine rağmen durumları belirlenemeyen bazı gizli birimlerden oluşuyorsa öğrenme çok daha ilginç bir hale gelir. Gizli değişkenler olarak hareket eden gizli birimler, Boltzmann Makinesinin, görünür birimler ile doğrudan bir etkileşime girerek modellenemeyen görünür durum vektörlerinin dağılımını modellemesine olanak sağlar. Boltzmann makinelerinin şaşırtıcı olan bir özelliği ise gizli birimler kullanılsa bile öğrenme kuralının değişmeden kalmasıdır. Bu durum verilerdeki üst düzey yapıları yakalayabilen ikili (binary) özelliklerin öğrenilmesini mümkün kılmaktadır. Gizli birimlerle birlikte, beklenen $\langle s_i s_j \rangle_{\text{data}}$, bütün veri vektörlerinin ötesinde, bir veri vektörü gözlemlenebilir birimlere bağlandığında beklenen $s_i s_j$ değerinin ortalamasıdır ve aynı zamanda gizli birimler, bağlanmış veri vektörleri ile dengelenene kadar sürekli güncellenirler.

$$\frac{\partial \log P(v)}{\partial w_{ij}} \quad (3.8)$$

Bu öğrenme kuralı oldukça basittir çünkü (3.8) formülü ağdaki diğer bütün bağlantı ağırlıklarına bağlıdır ve onlara göre değişebilmektedir. Neyse ki, (3.7) denklemindeki iki korelasyon arasında mevcut yerel fark w_{ij} 'ye diğer ağırlıklar hakkında bilmesi gereken her şeyi söylemektedir. Bu hata türevlerinin açık bir şekilde yayılmasını gereksiz kılmaktadır tıpkı geri yayılım algoritmasında olduğu gibi.

3.1.4. Boltzmann Makinesi Türleri

3.1.4.1. Üst Düzey Boltzmann Makineler

Stokastik (rassal) dinamikler ve öğrenme kuralı daha karmaşık enerji fonksiyonları barındırabilmektedir.

$$E(v) = - \sum_i S_{vi} b_i - \sum_{i < j} S_{vi} S_{vj} w_{ij} \quad (3.9)$$

$$z_i = b_i + \sum_j S_j S_k w_{ij} \quad (3.10)$$

Örneğin (3.9) formülündeki ikinci dereceden enerji fonksiyonu, normal terimi $S_i S_j S_k w_{ijk}$ olan bir enerji fonksiyonuyla değiştirilebilir. i biriminin güncelleme kuralında da kullanılan girdilerin toplamı daha sonra (3.10) ile değiştirilmelidir. Öğrenme kuralındaki tek değişiklik $S_i S_j$ ' in yerine $S_i S_j S_k$ gelmesidir.

3.1.4.2. Şartlı Boltzmann Makineler

Boltzmann makineleri veri vektörlerinin dağılımını modellemektedir, fakat koşullu dağılımları modellemek için de basit bir uzantı bulunmaktadır. Görünür ve gizli birimler arasındaki tek fark, $\langle S_i S_j \rangle_{\text{data}}$ örneklenirken görünür birimler kenetlenmişken (clamped) gizli birimler ise kenetlenmemiştir. $\langle S_i S_j \rangle_{\text{model}}$ örneklenirken eğer görünür birimlerin bir alt kümesi de kenetlenmiş ise bu altküme bir input gibi kalan diğer görünür birimlerde output gibi hareket ederler. Aynı öğrenme kuralı geçerlidir, fakat bu defa gözlenen çıktı vektörlerinin kayıt (log) olasılıklarını girdi (input) vektörleri üzerinden koşullu bir şekilde maksimize eder.

3.1.4.3. Ortalama Boltzmann Makineler

Rassal (stokastik) ikili durumlara sahip birimler kullanmak yerine, Hopfield ağında olduğu gibi 0 ile 1 arasında gerçek değerli duruma sahip deterministik “mean field” birimlerini kullanmak mümkündür.

$$\text{prob}(s_i = 1) = \frac{1}{1 + e^{-z_i}} \quad (3.11)$$

(3.11) denklemi, mevcut durumu verilen diğer birimler bulunurken bir birimin “ideal” değerli durumunu hesaplamak için kullanılır ve gerçek değer farkın bazı fraksiyonları tarafından ideal değere doğru yaklaştırılır. Eğer bu fraksiyon küçükse, tüm birimleri paralel olarak güncellenebilir. Aynı öğrenme kuralı rassal (stokastik), ikili verilerin deterministik gerçek verilerle değiştirilmesiyle de uygulanabilir. Fakat öğrenme algoritmasını doğrulamak oldukça zordur ayrıca “mean field” ağı, çoklu-model dağılımlarını modellerken bazı sorunlarla karşılaşmaktadır.

3.1.4.4. İkili Olmayan Birimler (Non-Binary Units)

Boltzmann makinelerde kullanılan ikiden fazla ayrık değere sahip olan ikili rassal (stokastik) birimler, "softmax" birimler olarak genelleştirilebilir. Çıktı değerlerinin girdi değerlerinin basit toplamı artı Gaussian gürültüsü olan Gaussian birimleri, binom birimler, Poisson birimler ve üstel aileye mensup diğer tip birimler bu birimlere örnek olarak verilebilir. Ayarlanabilir parametrelerin kayıt (log) olasılıkları üzerinde doğrusal etkileri olduğu gerçeği düşünülerek yukarıda sayılan birim türleri karakterize edilir. Öğrenme işlemi için gerekli olan gradyan'ın genel formu yalnızca gerekli (yeterli) istatistiklerde görünür birimlere kenetlenmiş verilerden kaynaklanan basit değişiklidir.

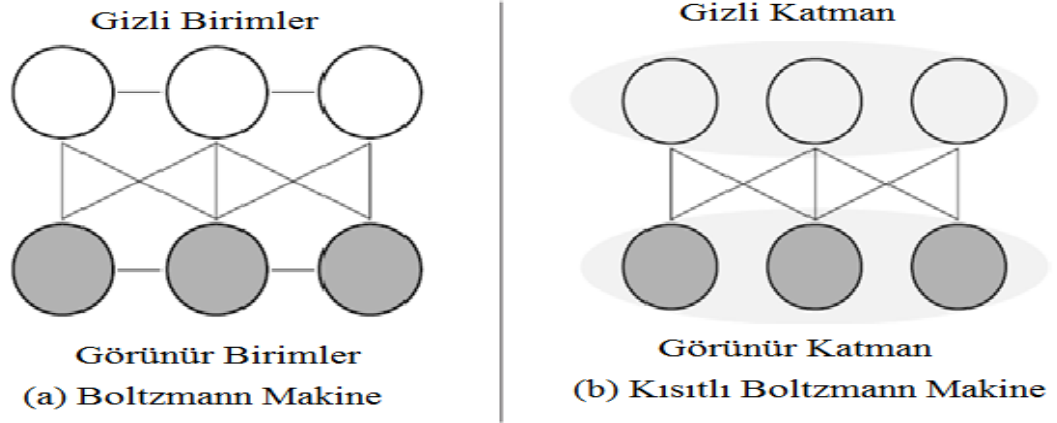
3.1.4.5. Öğrenme Hızı

Çok fazla gizli katmandan kaynaklı Boltzmann makinelerde öğrenme işlemi oldukça yavaştır çünkü büyük ağların denge dağılımına doğru yaklaşması uzun zaman almaktadır, özellikle bağlantı ağırlıklarının büyük ve denge dağılımlarının çok modelli olduğu durumlarda. Genellikle bu tür durumlarda görünür birimler kenetlenmemiş olarak bulunurlar. Denge dağılımının örnekleri elde edilebilir olsalar dahi beklenen iki örnek arasındaki farktan dolayı öğrenme sinyali gürültülüdür.

Ancak bağlantının kısıtlanması, öğrenme algoritmasının basitleştirilmesi ve bir seferde tek gizli katmanlı öğrenme seçenekleri kullanılarak bu zorlukların üstesinden gelinebilir [23].

3.1.5. Kısıtlı Boltzmann Makineleri

Genel Boltzmann makinelerinde uygulanan verimsiz örnekleme gibi pratik sınırlamaların üstesinden gelmek için, Smolensky (1986) tarafından yapısal olarak Boltzmann makinelerinin sınırlandırılmış bir versiyonu olan Restricted Boltzmann Machine (RBM) önerilmiştir. RBM görünür birimler (nöronlar) ve gizli nöronlar arasında yönlü bağlantılar kaldırarak oluşturulmuştur. Dolayısıyla, görünür bir birim (nöron) sadece gizli birimlere bağlanmış olabilirken aynı şekilde de gizli bir birimde sadece görünür bir birime bağlı olabilmektedir. RBM'nin yapısı bir birine bağlantılarla bağlı iki katman olarak görülebilir. RBM ve BM arasındaki fark aşağıdaki şekilde görülmektedir.



Şekil 3.5: Kısıtlı Boltzmann Makine ile Boltzmann Makine arasındaki fark.

Dayatılan kısıtlamanın temsil gücünü azalttığını düşünmemize neden olabilir, fakat Le Roux & Bengio (2008), RBM'nin herhangi bir ayrık değerli olasılık dağılımı modelleyebilecek bir yakınlştırıcı (approximator) olduğunu gösterdiler. Yapıya dayatılan kısıtlama söz konusu olduğundan, enerji ve durum olasılığı ona uygun şekilde değiştirilmelidir.

$$E(v, h|\theta) = -v^T W h - b^T v - c^T h \quad (3.12)$$

$$P(v, h|\theta) = \frac{1}{Z(\theta)} \exp\{-E(v, h|\theta)\} \quad (3.13)$$

Burada $\theta = (W, b, c)$ parametresi b ve c eşik değerlerini de (bias) içermektedir.

Her gizli nöron (birim) verilen diğer görünür birimlerden bağımsız olduğundan, görünür birimleri dışarıda tutup görünür birimlerin normalize edilmemiş olasılıklarını elde etmek mümkündür. Buna göre görünür bir birimin durum olasılığını belirten v , aşağıdaki gibi hesaplanmaktadır,

$$P(v|\theta) = \frac{1}{Z(\theta)} \exp(b^T v) \prod_{j=1}^{n_h} (1 + \exp(c_j + \sum_{i=1}^{n_v} W_{ij} V_i)) \quad (3.14)$$

Burada n_v ve n_h sırasıyla görünür ve gizli birimlerin sayısal olarak adedini belirtmektedir.

3.1.5.1. Kısıtlı Boltzmann Makinelerin Eğitimi

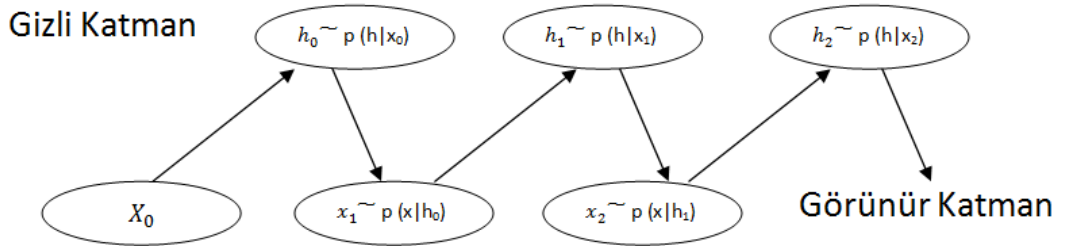
RBM öğrenme kuralları aşağıdaki gibi hesaplanmaktadır,

$$w_{ij} \leftarrow w_{ij} + n_w [\langle v_i h_j \rangle_d - \langle v_i h_j \rangle_m] \quad (3.15)$$

$$b_i \leftarrow b_i + n_b [\langle v_i \rangle_d - \langle v_i \rangle_m] \quad (3.16)$$

$$c_j \leftarrow c_j + n_c [\langle h_j \rangle_d - \langle h_j \rangle_m] \quad (3.17)$$

Öğrenme oranlarının seçiminde titiz teorik bir arka plan çalışması olmamasından dolayı genellikle her iki eşik değerini (bias) öğrenmek için nispeten daha küçük öğrenme oranları kullanılır. RBM, BM'den türetilmiş özel bir yapı olduğundan öğrenme işlemi için BM'de kullanılan Gibbs örneklemesinin kullanılması mümkündür. Kısıtlı yapısı sayesinde Gibbs örnekleme daha verimli olarak kullanılabilir, verilen katman ister görünür ister gizli katman olsun, diğer katmanlardaki birimlerden karşılıklı olarak bağımsız hale gelir.



Şekil 3.6: RBM ' de paralelleştirilmiş yapılar.

Bu katman odaklı örnekleme yöntemi paralelleştirilmiş modern yapıların tamamıyla kullanılmasına olanak sağlamaktadır.

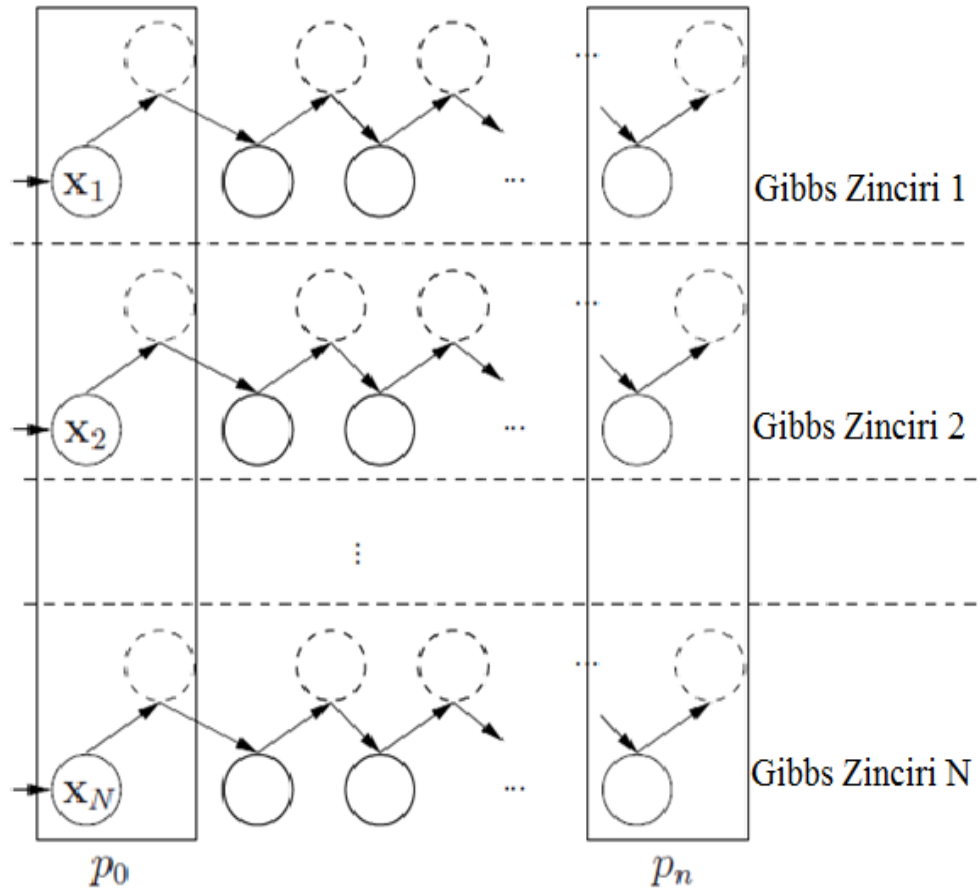
Bununla birlikte, RBM'deki nöronların sayısı arttıkça, RBM tarafından temsil edilen olasılık dağılımının daha derli toplu açıklanması için Gibbs örnekleme yöntemi tarafından daha fazla sayıda örnek toplanmak zorundadır. Dahası, Gibbs örnekleme yöntemi doğası gereği, örnekler dağıtımının bazı kısımlarını gözden kaçırabilirler. Bu zorlukların üstesinden gelmek için pek çok yöntem önerilmiştir.

3.1.5.2. Karşılaştırmalı Ayırışma Öğrenme Metodu

Hinton (2002) tarafından karşılaştırmalı ayırışma öğrenme metodu uzman ürün modellerin eğitimini gerçekleştirmek için önerilen popüler bir metottur. (3.14) denklemi RBM' in uzman ürün modelinin özel bir durumu olduğunu işaret etmektedir ve Contrastive Divergence (CD) yöntemi bir RBM'nin eğitimi için kolaylıkla kullanılabilir. CD öğrenme yöntemi, gerçek gradyan değerine yaklaşmak için (3.13) dağılımından gelmesi beklenen değer yerine eğitim örnekleri tarafından tanımlanan deneysel dağılımdan beklenen Gibbs örneklemesinin n kere uygulanmasıyla oluşan P_n değerini kullanır.

Bağlantı ağırlıkları için oluşan formül aşağıdaki gibidir:

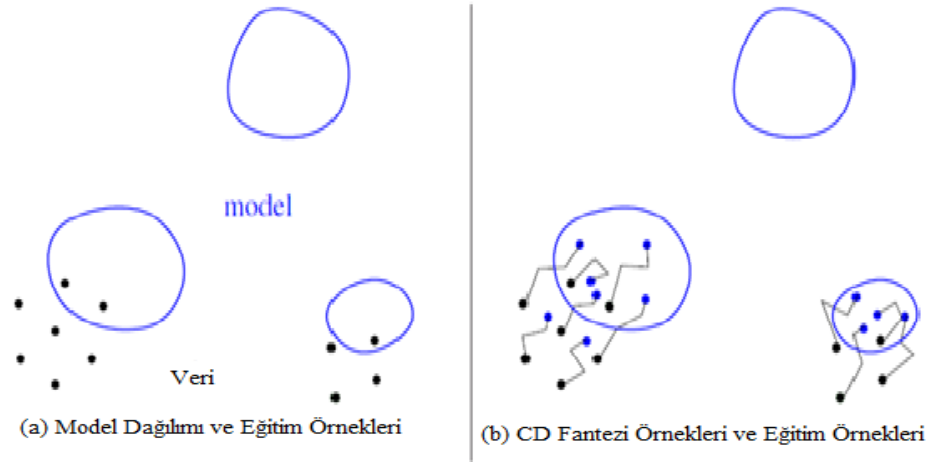
$$w_{ij} \leftarrow w_{ij} + n[\langle x_i h_j \rangle_{P_0} - \langle x_i h_j \rangle_{P_n}] \quad (3.18)$$



Şekil 3.7: CD öğrenme yöntemi.

$n=0$ durumu, pozitif aşamada kullanılan $P(h | \{v^{(t)}\}, \theta)$ deneysel dağılımını üretir, halbuki $n=\infty$ durumu negatif aşamadaki $P(x | \theta)$ gerçek dağılımı üretmektedir. Aslında gradyan'ın yönü gerçek gradyan ile birebir aynı olmadığı gerçeği beklendiğinden, CD öğrenme yöntemi biased (kayıt) olarak bilinmektedir. Bununla birlikte CD öğrenme yönteminin pratikte iyi çalıştığı görülmüştür. CD öğrenme yönteminin iyi bir tarafı, veri dağılımının çoklu model olduğu durumlarda, her veri örneğinden başlanarak zincirin çalıştırılması, negatif aşamaya yaklaşan örneklerin farklı modlardan temsilcilerinin olmasını garanti eder. CD öğrenme yönteminin bu avantajı aynı zamanda bir dezavantajdır.

P_n 'den gelen örneklerin bütün durum uzayını açıkladığı söylenemez. Bu nedenle model dağılımındaki bazı modlar incelenememiştir ve öğrenme işlemi yakınsandıktan sonra bile model dağılımı eğitim seti tarafından tanımlanmış veri dağılımlarında bulunmayan modlar içerebilmektedir.



Şekil 3.8: Model Dağılımı ve CD öğrenme yöntemi karşılaştırma.

Bu problemin üstesinden gelebilmek için CD öğrenme yöntemine dayanan farklı yaklaşımlar önerilmiştir. Bu yaklaşımlar arasından Kalıcı Karşılaştırmalı Iraksama (PCD), CD öğrenme yönteminin en basit uzantılı olanıdır.

Her gradyan güncelleme adımında, CD öğrenme yöntemi eğitim veri setinden başlayarak Gibbs örneklemesini uygularken, PCD öğrenme yöntemi örnekleme işlemine

en son gradyan güncellemesinden elde edilen model örneklerini örnekleyerek başlar. Böylelikle model örneklerinin model dağıtımında olan ve eğitim setindeki örneklerle benzemeyen modları bulması beklenmektedir.

Ancak, PCD öğrenme yöntemi öğrenme işlemi devam ederken model dağılımdaki modların tespit edilmesinde hala zorlanmaktadır. Bunun nedeni Gibbs örnekleme yönteminin zayıf bir karıştırma yapmasıdır, Gibbs yöntemi örnekleme yaparken arka arkaya gradyan güncellemesi yapmak için yüksek korelasyonlu örnekler üretmektedir. Eğer öğrenme işlemi yapılırken dikkati olunmaz ve öğrenme oranı takvimi, bağlantı ağırlıklarında erime ve denge faktörlerinin elle seçilmesi durumu oluşursa, bu davranış CD öğrenme yöntemine dayalı yaklaşımların olasılık sapmasından muzdarip olmasına neden olur. Son zamanlarda PCD öğrenme yönteminin de dışında çok fazla CD öğrenme yaklaşımları ileri sürülmüştür. Örneğin, Tieleman & Hinton (2009) tarafından daha iyi model örnekleri elde edebilmeyi sağlayan hızlı ağırlıkların iyileştirilmesiyle geliştirilen hızlı PCD öğrenme yöntemi önerilmiştir.

3.1.5.3. İleri Markov-Chain Monte-Carlo (MCMC) Örnekleme Metodu Tabanlı Öğrenme

Gradyan yön yaklaşımı yerine, basit Gibbs örnekleme yönteminin dışında ve daha gelişmiş MCMC örnekleme yöntemleri uygulamak mümkündür. Gibbs örnekleme yöntemine bir alternatif paralel tavlama (parallel tempering PT) örnekleme yöntemidir. Bu yöntem son zamanlarda RBM'leri eğtmek için Gibbs örnekleme yöntemi yerine önerilen bir yöntemdir(Desjardins et al. (2010) and Cho et al. (2010)). PT örnekleme yöntemine ek olarak gelişmiş MCMC örnekleme yöntemlerine dayanan diğer yaklaşımlarda önerilmiştir. Örneğin, temperli geçiş dayalı stokastik yaklaşım prosedürü bunlardan biridir. Bu işlem son zamanlarda Salakhutdinov (2009) tarafından önerilmiştir. Farklı sıcaklıklara sahip birden çok Gibbs örneklemesini kullanan bir yöntemdir. Aynı zamanda daha gelişmiş RBM yapılarını (üç yollu faktöre edilmiş RBM yapıları gibi) eğtmek için veya enerji tabanlı modelleri eğtmek için melez bir Monte Carlo algoritması (HMC) da başarıyla uygulanmıştır.

3.1.5.4. Diğer Yaklaşımlar

Şimdiye kadar sunulan yaklaşımlar MCMC örnekleme yöntemi kullanılarak stokastik yaklaşıma dayalı yaklaşımlar idiler. Ancak RBM'leri eğtmek için başka yaklaşımlarda

mevcuttur. Bu yaklaşımlardan biri olasılık fonksiyonunu sözde olasılığa yaklaştırma yöntemidir ve bu nedenle eğitilen RBM yapıları sözde olasılığın maksimize edilmesi seviyesine ulaşırlar. Kayıt – sözde – olasılığı fonksiyonuna verilen $\{v^{(t)}\}_{t=1}^N$ veri seti aşağıdaki şekilde tanımlanmaktadır:

$$f_{PL}(\theta) = \frac{1}{N} \sum_{t=1}^N \sum_{i=1}^d \log P(v_i^{(t)} | v_{\setminus i}^{(t)}) \quad (3.19)$$

Burada $x_{\setminus i}$ daha önce de olduğu gibi $[x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_d]^T$ vektörünü ifade etmektedir. Son zamanlarda geliştirilen bir başka yaklaşımda Hyvärinen (2007) tarafından önerilen oran eşleştirme (ratio matching (RM)) yöntemidir. Olasılık yerine, RM olasılık oranlarını göz önünde bulundurur. Söz konusu olasılık oranları, verilen bir gözlemin olasılığı ile tek bir i değişkene sahip gözlem vektörünün olasılığı arasındaki orandır. i değişkeni de aşağıdaki formülünden çevrilmektedir.

$$\frac{P(v)}{P(v_{\rightarrow i})} = \frac{P^*(v)}{P^*(v_{\rightarrow i})} \quad (3.20)$$

Ve model oranı ve model dağılımındaki oranla aynı orandır. RM öğrenme yöntemi verileri ve model oranını zorlayarak olabilecek en yakın değere yaklaştırmaya çalışır. Oran, sabitin normalizasyonunun hesaplamayı gerektirmediği için RM oldukça faydalıdır.

Ayrıca son zamanlarda önerilen genelleştirilmiş skor eşleştirme yöntemi de RBM'yi eğitmekte kullanılabilir. Bu yaklaşımlar Marlin (2010)'in rastsal yaklaşımı ve birbirleriyle karşılaştırılmıştır. Ancak bu öğrenme yöntemleri gözlemlenen sistemin boyutunun büyüklüğünden kaynaklanan hesaplama karmaşıklığından muzdariptir ve bu yöntemler MCMC öğrenme yöntemine dayalı rastsal yaklaşıma önemli bir katkı sunamamaktadırlar [24].

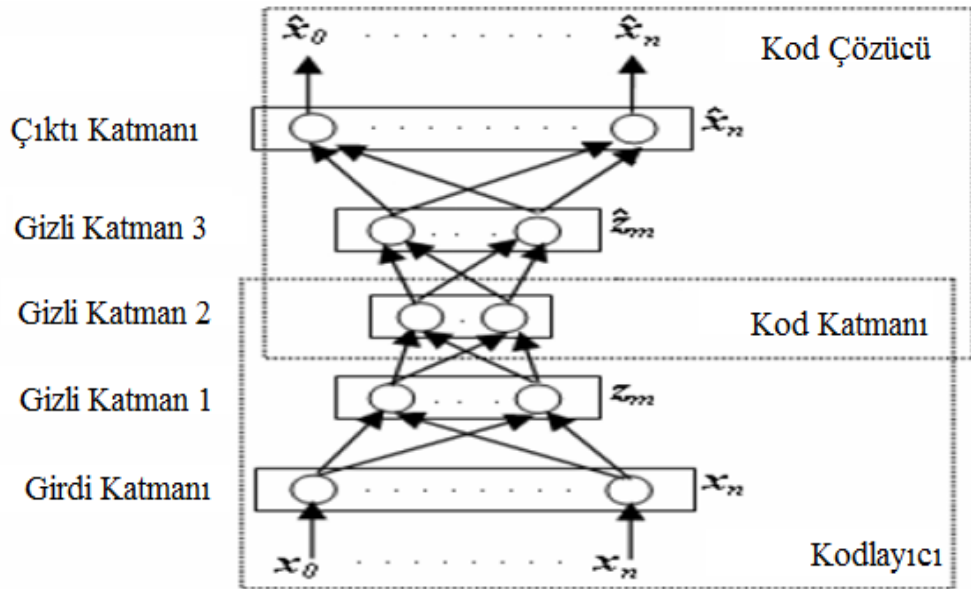
3.2. AUTOENCODERS

Autoencoder ağları birden fazla gizli katmanı olan ileri beslemeli sinir ağlarıdır. Bu ağlar girdi verilerini çıkış katmanında yeniden yapılandırmaya çalışırlar. Çıkış katmanındaki hedef veriler ile girdi verileri aynıdır, bu nedenle çıkış katmanının boyutu ile giriş katmanının boyutu aynıdır. AE'de tıpkı geri yayılım algoritması gibi gradyan alçalma yöntemini kullanarak eğitilir. Bir AE'de gizli katmanın boyutu, giriş verilerinin

boyutundan daha küçük olduğundan, girdi verilerinin boyutu gizli katmanda kısıtlanarak daha küçük bir alanda tekrar boyutlandırılır. Gizli katmandan çıkan veriler daha sonra çıkış katmanında tekrar orijinal veriye dönüştürülür. AE veri ve kod alanı arasında çift yönlü haritalandırma yapabilmektedir. Bir AE’de fazla sayıda gizli katman kullanılarak büyük boyutlara sahip girdi verileri çok daha az bir kod alanına kısıtlanabilir. Bununla birlikte birden fazla gizli katmana sahip bir sinir ağını eğitmek oldukça can sıkıcı olabilmektedir ve ayrıca iyi sonuçlar elde etmekte zorlaşmaktadır. Bunun nedeni de çok alt seviyelerdeki gizli katmanların çok zor optimize edilmesidir.

3.2.1. Autoencoder Mimarisi

AE ileri beslemeli bir yapıya sahiptir, bu sinir ağının bir veya birden çok gizli katmanı olabilmektedir. AE girdi çıkış katmanında girdi değerlerini yeniden oluşturmaya çalışır. AE ile bilinen geleneksel yapay sinir ağları arasındaki en temel fark çıkış katmanının boyutudur. Bir AE’de çıkış katmanının boyutu ile giriş katmanının boyutu birebir aynıdır. Bununla birlikte bir AE’de bulunan gizli katmanların sayısı ne kadar olursa olsun girdi katmanı sayısı ve çıkış katmanı sayısından her zaman için daha azdır.

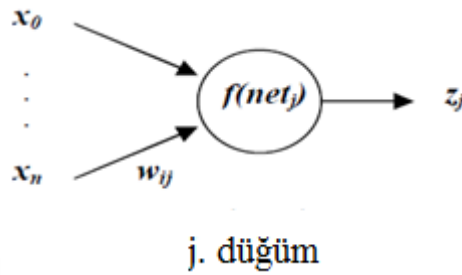


Şekil 3.9: Autoencoder’ın mimari yapısını gösteren şema.

Yukarıdaki şekilde de görüldüğü gibi bir AE, ‘encoder’ ve ‘decoder’ denilen iki kısmın bir araya gelmesiyle oluşmaktadır.

“Encoder” kısmı çok boyutlu girdi bilgilerini küçük boyutlara indirgerken, “Decoder” kısmı da diğer kısım tarafından küçük boyutlara indirgenmiş veriyi tekrar orijinal yapısına döndürür. AE doğrusal olmayan haritalandırma için sigmoid aktivasyon fonksiyonlarını kullanır. Sigmoid aktivasyon fonksiyonları kullanıldığında AE, PCA’ya benzer performans ortaya koyacaktır. Ağlar orijinal veri ile yeniden yapılandırılmış veri arasındaki ortalama hata farkını minimize edebilmek için eğitilebilirler. Hata türevlerinin geri yayılımını yapmak için Zincir kuralı (Chain Rule) uygulanarak önce “Decoder” daha sonra da “Encoder” kısımlarından geçirilerek istenilen gradyan değeri kolayca elde edilebilir.

AE’deki bir düğüme ait bir gizli katmanın n tane girdi elemanı ve bir eşik’i (bias) girdi olarak aldığını düşünelim



Şekil 3.10: Bir autoencoder düğümüne ait girdiler ve oluşan eşik.

Burada ki x_0 eşik değeri ve -1’e ve gizli katmandaki toplam bağlantı ağırlığı da

$$net_j = \sum_{i=0}^n w_{ij} x_i \quad (3.21)$$

w_{ij} ise x_i ile bağlantılı j . düğümünde bulunan gizli katmanın bağlantı ağırlığını göstermektedir. Gizli bir düğümünde çıktısı $z_j = f(net_j)$ ve $f(s) = \frac{1}{1+e^{-\beta s}}$ en son gizli katmanda oluşan çıktılar, çıktı katmanına girdi olarak gönderilir, böylece çıktı katmanındaki en sonuncu düğümünün çıktısı $\hat{x}_l = f(net_l)$ şeklinde elde edilir.

$$net_l = \sum_{j=0}^m w_{jl} z_j \quad (3.22)$$

3.2.2. Eğitim Yöntemleri

3.2.2.1. Geri Yayılım

AE'nin $\hat{x}_l$ çıktıları ile beklenen x çıktıları karşılaştırılır burada amaç hataların kareleri toplamını elde etmektir. Düğümlerin bağlantı ağırlıkları ağırlık değişikliklere göre ayarlanır. Çıktı katmanını için bağlantı ağırlıklarında meydana gelen değişiklikler aşağıdaki gibi iki farklı yolla hesaplanabilir:

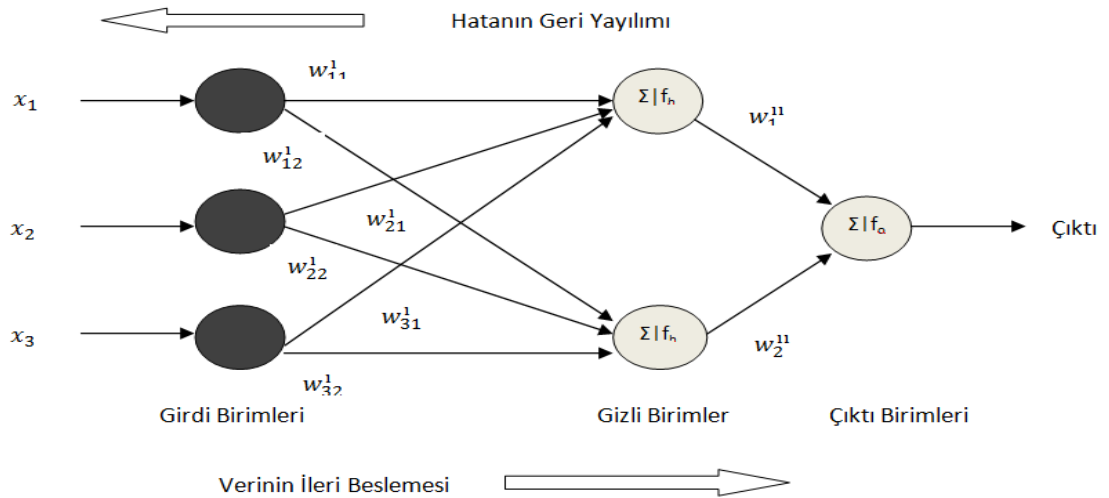
$$\Delta w_{jl} = \eta(x_l - \hat{x}_l) \lambda_0 f(net_l)(1 - f(net_l))z_j \quad (3.23)$$

$$\Delta w_{jl} = \eta(x_l - \hat{x}_l) f'(net_l)z_j \quad (3.24)$$

Gizli katmandaki bağlantı ağırlıklarındaki değişikliklerde aşağıdaki gibi hesaplanır.

$$\Delta w_{jl} = \eta(\sum_{l=0}^n (x_l - \hat{x}_l) f'(net_l) w_{jl}) f'(net_l) x_i \quad (3.25)$$

burada η öğrenme oranını göstermektedir.



Şekil 3.11: Verinin ileri beslemesi ve hatanın geri yayılımı [25].

3.2.2.2. Polak Ribiere Eşlenik Gradyan Geri Yayılımı

Eşlenik geri yayılım, geri yayılım türlerinden biridir. Geleneksel geri yayılım yöntemleriyle karşılaştırıldığında eğitim süresini belli bir ivme ile hızlandırmaktadır. Konjügasyon'un çizgi arama fonksiyonu, hata fonksiyonundaki minimum hata noktasını bulmak için kullanılır. Arama yönü de aşağıdaki formüle göre yapılmaktadır:

$$p_k = -g_k + \beta_k p_{k-1} \quad (3.26)$$

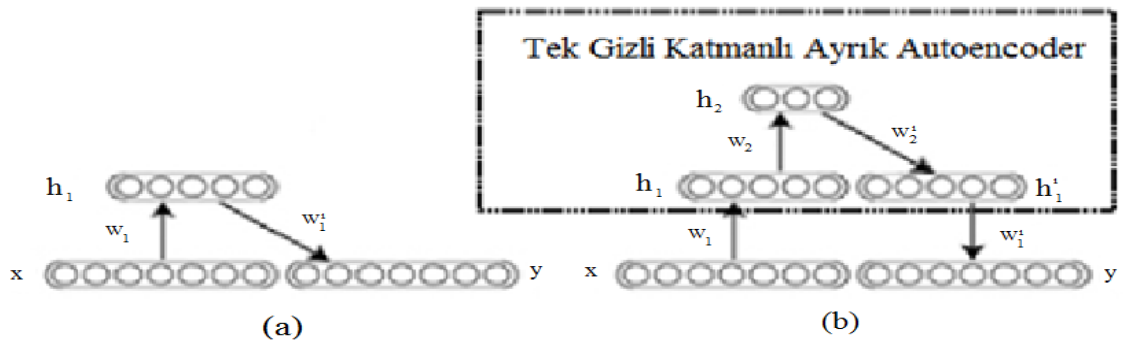
Burada g_k yeni gradyanı p_{k-1} 'da eski arama yönünü belirtmektedir.

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{g_{k-1}^T g_{k-1}} \quad (3.27)$$

β_k parametresi farklı yollarla hesaplanabilir. Polak Ribiere yöntemi için yukarıdaki gibi hesaplanmaktadır.

3.2.2.3. Yığın Autoencoder

AE şekline bakıldığında (çok katmanlı) görüldüğü üzere, birden fazla sayıda gizli katmana sahip bir AE'de bağlantı ağırlıklarını optimize etmek oldukça zordur. Başlangıçta bağlantı ağırlıklarının büyük olduğu durumlarda zayıf bir local minimum değeri bulunurken, başlangıçta bağlantı ağırlıklarının değerlerinin küçük olduğu durumlarda ise ilk tabakalardaki gradyan değeri çok zayıf çıkmaktadır bu da eğitim aşamasını imkânsızla yakın yapmaktadır. Çok katmanlı AE yapılarını eğitmek için Greedy Layer-Wise Training adlı bir yöntemde mevcuttur. Bu yöntemde eğitim aşamalara bölünmüştür. Birinci aşamada AE'nin 3 katmanda oluştuğu varsayılmaktadır, girdi katmanı x , çıktı katmanı y ve gizli katmanda h_1 olarak adlandırılmaktadır, W_1 ve W'_1 bağlantı ağırlıkları da geri yayılım yöntemi kullanılarak eğitilir. İkinci aşamada bir gizli katman (h_2) bir girdi katmanı (h_1) ve birde çıktı katmanından (h'_1) oluşan yapı autoencoder'a bağlanmıştır.



Şekil 3.12: AE'ye eklenmiş tek gizli katmanlı AE.

Bir AE'nin girdi ve çıktı katmanının büyüklüğü ve değerleri aynı olduğundan girdi ve çıktı katmanı birebir aynıdır. h_2 gizli katmanı yeni bir AE'ye eklenmiş yeni bir gizli katmandır. W_2 ve W_2^l bağlantı ağırlıkları geri yayılım kullanılarak eğitilebilir. h_2 gizli katmana sahip ağ eğitildikten sonra global minimuma yakınsamak için AE'nin bütün ağırlıkları birlikte eğitilir. Bu tür AE yapıları yığın autoassociatorlar olarak adlandırılırlar. Yakınsamayı gerçekleştirmek için bu yöntem kullanılarak birden fazla tek gizli katmana sahip derin AE yapıları eğitilebilir [26].

Yapay sinir ağları insan beyninin eldeki bilgileri işleme şekline ilham alınarak oluşturulan bir bilgi işleme tekniği olarak bilinmektedir. YSA'nın çalışma şekli biyolojik sinir hücrelerinin çalışma şekline benzetilmiştir. Biyolojik sinir hücrelerine benzetilen bu yapı nöronlar (düğüm) içerirler ve bu nöronlar çeşitli şekillerde birbirlerine bağlanarak bir ağ oluştururlar. Oluşan bu ağlar veriler arasında ilişki kurma ve öğrenme yeteneklerine sahiptirler. YSA, insanın doğal özellikleri olan gözlem yapabilme ve öğrenme gibi yetenekleriyle çözdüğü problemlere bilgisayar ortamında bir çözüm üretmektedir. Tez çalışmasında, önceki yapısı YSA'na dayanan, derin öğrenme (yeni nesil YSA'lar) metodu kullanılmıştır.

Derin Öğrenme makine öğrenme araştırmalarında en popüler yöntemlerden biridir. Y. Bengio, G. Hinton, M. Ranzato, R. Salakhutdinov ve diğer meslektaşları tarafından yeni nesil yapay sinir ağları olarak sunulmuştur. Derin inanç ağlarının tanıtılmasıyla hata geri yayılımından kaynaklanan dezavantajların üstesinden gelinmiştir. Kapsamlı ve karmaşık olasılık modellerin eğitim ve test aşamalarında oldukça etkili sonuçlar vermektedir. Özellikle derin sinir ağlarında geleneksel yaklaşımlara göre etkileyici sonuçlar vermektedir. Derin sinir ağları doğrusal olmayan çok sayıda katmandan oluşur ve bu katmanlarda çok sayıda işlem birimi içermektedir. Modelin eğitimi esnasında katmanlı denetimsiz ön eğitim yöntemi kullanılır. Katmanlar düşük seviyeden üst düzey özelliklere doğru bir hiyerarşik yapıya sahiptir. Derin sinir ağları, katlamalı derin sinir ağları ve derin inanç ağları derin öğrenmenin bazı farklı mimari modelleridir. Bu mimariler otomatik konuşma tanıma, doğal dil işleme, bilgisayarlı görü gibi farklı alanlarda kullanılmaktadır.

Tez çalışmasında, öncelikle karakter tanıma işlemi yapabilmek için MNIST veri kümesi [29] kullanılmış ve bunun üzerinde derin öğrenme uygulaması yapılmıştır. Sonrasında ise kenar belirleme uygulaması yapılmıştır. Kenar belirleme, görüntüdeki noktaların parlaklıklarının çok keskin bir şekilde değişti noktaları belirlenmesini amaçlayan bir dizi işlemler bütünü olarak tanımlanmaktadır. Kenarlar görüntüdeki parlaklığın keskin bir biçimde değiştiği noktalardır ve genellikle insanların görüntünün içeriğini anlaması için yeterlidir ve vektörel bir büyüklüktür. Kenar bilgileri bir pikselin kendisine komşu diğer piksellerin bilgileriyle karşılaştırıldığında ortaya çıkmaktadır.

Dijital bir görüntüde parlaklıkta meydana gelen keskin değişiklikleri tespit etmedeki amaç önemli olayları yakalamaktır. Bazı yöntemler sadece alıştırma bilgileri döndürürken diğerleri söz konusu kenarın nerelerde mevcut olduğunu veya olmadığını döndürür.

Kenar belirleme yöntemleri şunlardır: Sobel yöntemi, orijinal görüntüden dönüştürülmüş bir 3x3 kernel maskesi kullanmaktadır bu görüntüyü yatay ve dikey yönlerde inceleyerek kenarları belirlemektedir. Canny kenar belirleme yöntemi ise Gaussian filtrelemesini kullanarak güçlü kenarlarla ilişkisi olmayan zayıf kenarları baskılayarak kenarların ortaya çıkmasını sağlar. Prewitt kenar operatörü ise fonksiyonun yaklaşık gradyan değerinin hesaplar bunu yaparken de yatak ve dikey kernelleri kullanır. Roberts kenar belirleme yöntemi ise görüntünün gradyan değerine yaklaşık değeri hesaplamak için ayırık farklılaşma yöntemini kullanır. Ayırık farklılaşma da birbirine çapraz piksel değerlerinin farklarının kareleri toplamını hesaplayarak gerçekleştirmektedir.

Karakter tanıma uygulamasında kullandığımız MNIST veri kümesi, farklı kişiler tarafından rakamların el yazısıyla yazılmış halde bulunduğu ve farklı kenar belirleme yöntemlerinin eğitiminde kullanılan çok geniş bir veri tabanıdır. Bu veri tabanı aynı zamanda makine öğrenme yöntemlerinin eğitimi ve testi için de literatürde geniş bir biçimde kullanılmaktadır. Bu geniş veri tabanı NIST veri setinin orijinal verilerinin karıştırılmasıyla oluşmuştur.

Veri tabanı, 60.000 eğitim görüntüsü ve 10.000 adet test görüntüsüne sahiptir. Bu eğitim görüntülerinin yarısı ve test görüntülerinin yarısı NIST veri setinden alınmıştır.

Bazı arařtırmacılar, yapay sinir aęlardan oluřan bir komite kullanarak MNIST veri seti üzerinde insan performansına yakın bir başarı elde etmiřlerdir.



řekil 3.13: MNIST veri kümesine ait bir kesit.

Yukarıda MNIST veri kümesine ait olan 10 farklı el yazısıyla yazılmış rakamlar kümesi görölmektedir. Bu řekilde toplanan el yazması rakamlarla veri tabanı oluřturulmuřtur [29]. Ayrıca, yapılan bütün uygulamalar için Matlab programlama dili kullanılmıştır.

4. BULGULAR

4.1. KARAKTER TANIMA

Bu bölümde, Matlab programlama dili kullanılarak MNIST veri kümesi üzerinde karakter tanıma uygulaması gerçekleştirilmiştir. Ön eğitim olarak her bir RBM'nin ön eğitimini hızlandırmak için bütün veri kümeleri küçük gruplara bölünmüştür. Küçük grupların boyutuna tam bölünemeyen veri kümeleri geriye kalan veri vektörleriyle en son gruba dâhil edilmiştir. Her veri kümesindeki her bir gizli katman tüm eğitim kümesinden 50 çevrim yaptırılarak ön eğitim gerçekleştirilmiştir [27, 28].

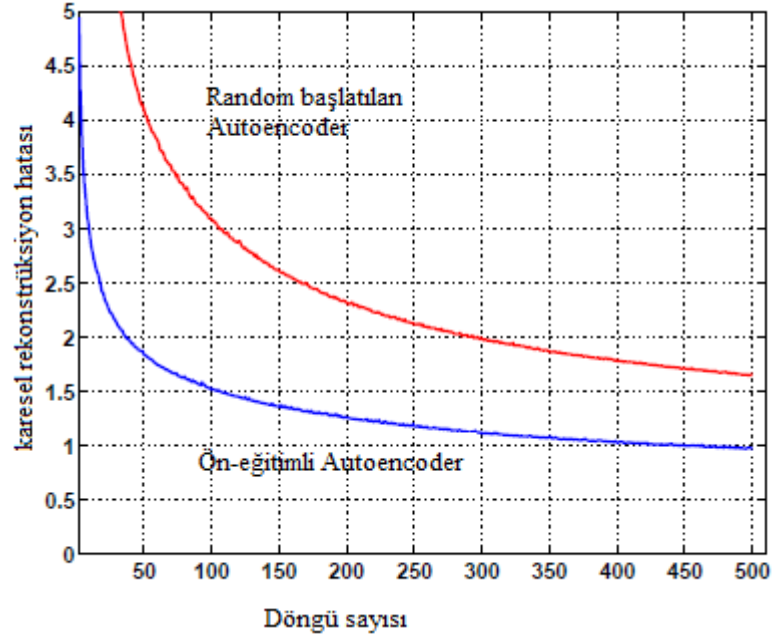
$$p_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (4.1)$$

Ağırlıklar, her bir küçük grup (4.1) formülü ve 0,1 öğrenme oranı (learning rate) kullanılarak güncellenmiştir. Ağırlıklar 0 ortalama ve 0,1 standart sapma kullanılarak normal dağılımdan örneklenen küçük rastgele değerlerle başlatılmıştır. Kullanılan Matlab kodu [28] referansından alınmıştır [27, 28].

İnce ayar (fine tuning) yaparken, 1.000'den fazla veri vektörü içeren gruplar için eşlenik gradyan metodu kullanılmıştır. Optimizasyon için ise Carl Rasmussen'in "minimize" kodu kullanılmıştır [27, 28]. Yeterli sayıda çevrimi hesaplamak için her bir AE'nin ince ayarı eğitim kümesinin bir parçası kullanılarak yapılmıştır, eğitim setinden kalan kısımda ise performans test etmek için kullanılmıştır. Daha sonra ince ayar işlemi bütün eğitim seti üzerine uygulanmıştır [27].

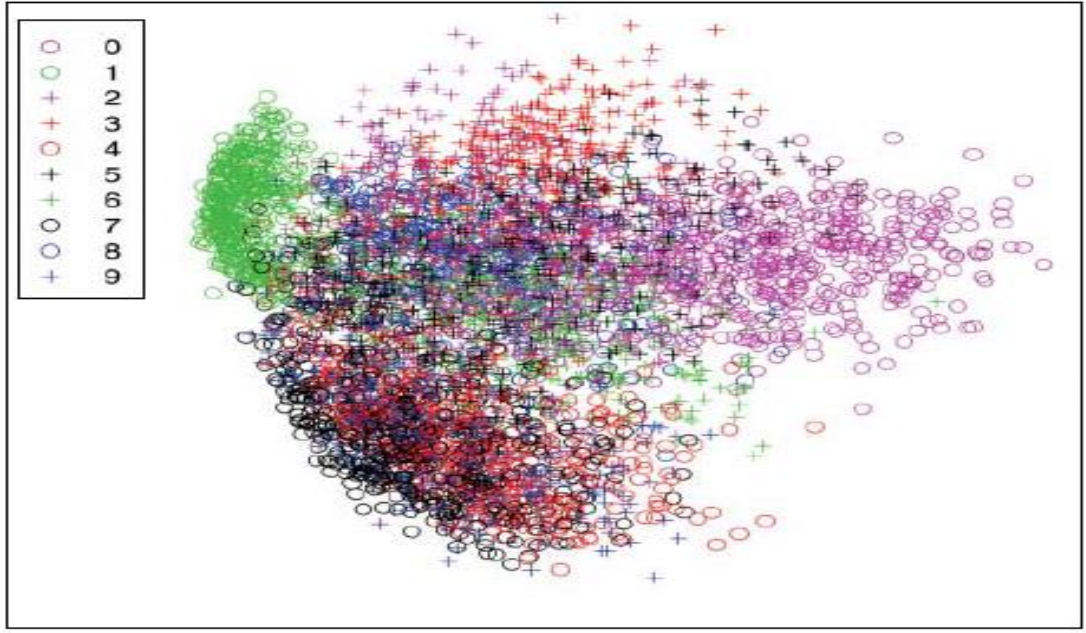
MNIST karakteri için orijinal piksel yoğunluğu 0-1 aralığında normalize edilmiştir. Deney sonuçları, ilk gizli katmanda 1.000 nörondan oluşan bir AE kullanarak elde edilmiştir. Bütün görüntüler öğrenme aşamasına sokulmadan önce rastsal piksel permütasyonu işleminden geçirilmiştir. Bu işlem, öğrenme programının daha önce kullanılan imgelerdeki benzeşik dönüşümleri önlemek adına yapılmaktadır [27].

Bütün 60.000 eğitim durumuna için 784-1.000-500-250-2 yapısında bir ağ ön-eğitimi (pre-training) uygulanmıştır. Ön-eğitim, sınıf etiketleri ile ilgili hiçbir veri kullanmamaktadır [27].

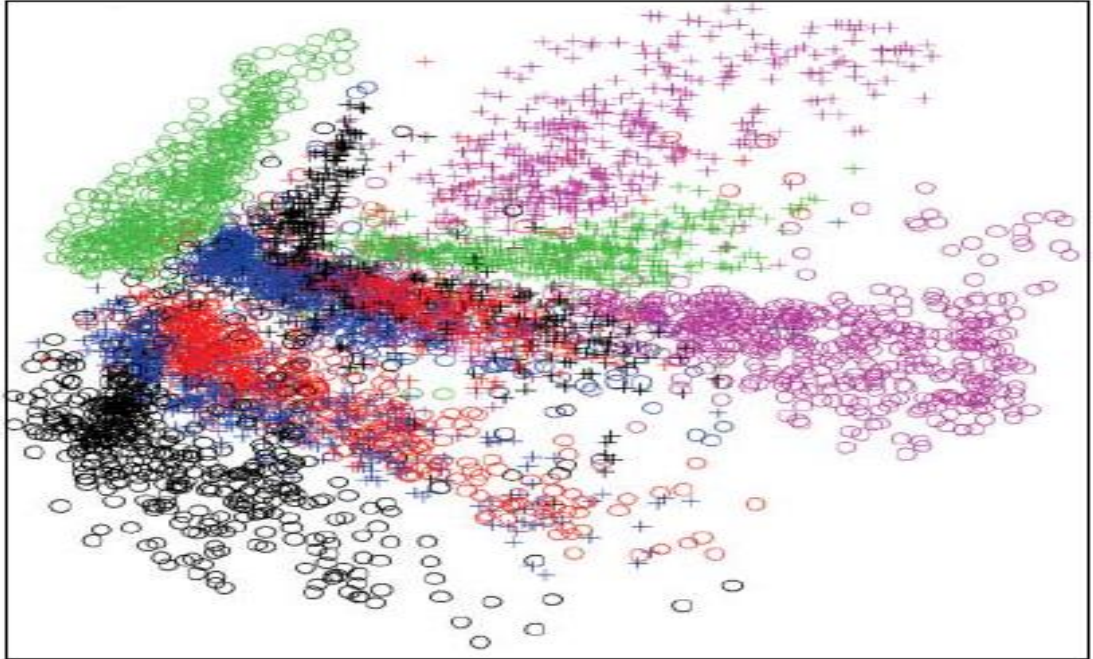


Şekil 4.1: Eğitim setinde test imgesi başına düşen ortalama kareli oluşturma (reconstruction) hatası [27].

Veri kümesinde tanıma işlemi tamamlandığında elde edilen sonuçlar Şekil 4.1’de gösterilmiştir. Burada karşılaştırma yapmak amacıyla, MNIST veri kümesinin Temel Bileşen Analizi (PCA-Principal Component Analysis) ile sınıflandırılması sonucu elde edilen grafiğe de yer verilmiştir. Görüldüğü üzere, derin öğrenme içeren bir sistemde karakter tanıma çok daha iyi sınıflandırılabilir.



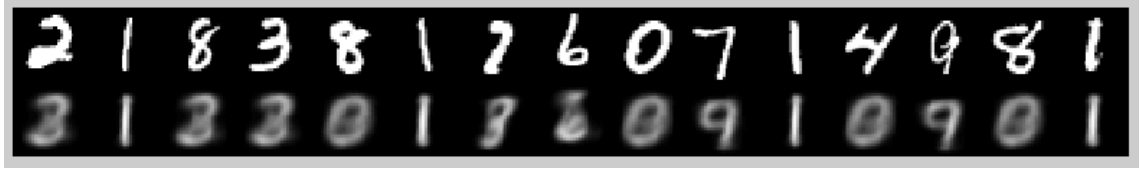
Şekil 4.2: MNIST veri kümesinin Temel Bileşen Analizi (PCA-Principal Component Analysis) ile sınıflandırılması sonuçları [27].



Şekil 4.3: MNIST veri kümesinin 784-1.000-500-250-2 yapısındaki AE içeren derin öğrenme yaklaşımıyla sınıflandırılması sonuçları [27].

4.2. KENAR BELİRLEME

İkinci aşamada autoencoder (AE) çıkışının (reconstructed image) kenar belirleme performansı araştırılmıştır. Yöntem olarak literatürde sık kullanılan "*Prewitt*", "*Sobel*" ve "*Roberts*" yöntemleri denenmiştir. Hem orijinal görüntü üzerindeki hem de AE çıkışındaki görüntü üzerindeki kenarların belirlenmesi uygulamaları yapılmıştır.

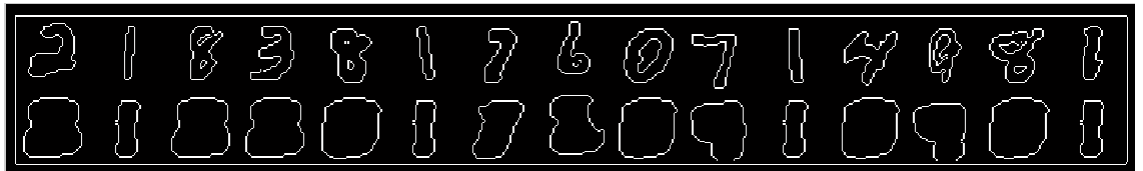


Şekil 4.4: MNIST karakter veri seti (AE çıkışı).

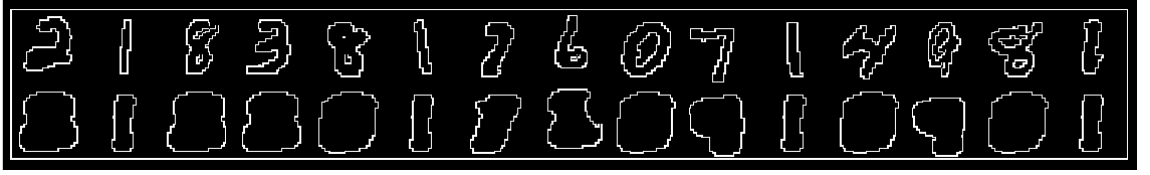
Orijinal görüntüden yeniden yapılandırılmış görüntüyü elde edebilmek için Derin inanç ağı oluşturuldu. Bu işlem için AE kullanıldı ve deneyi gerçekleştirmek için gerekli görsellerde MNIST karakter veri seti kullanıldı. Deneyimde [8]' de Derin inanç ağları kullanılarak yeniden yapılandırılan el yazısı karakterler kullanılmıştır bende deneyimde bu karakterleri kullandım. Bu görsellerin boyutu 14x14'tür. Her bir model yedi adet gizli katmana sahiptir ve ayrıca modelin eğitimi için 6000, testi içinde 1000 adet görsel kullanılmıştır. Test aşamasında kullanılan periyot sayısı 230'dur [8].



Şekil 4.5: MNIST karakter veri setine uygulanan Prewitt kenar belirleme yöntemi.



Şekil 4.6: MNIST karakter veri setine uygulanan Sobel kenar belirleme yöntemi.



Şekil 4.7: MNIST karakter veri setine uygulanan Roberts kenar belirleme yöntemi.

Şekil 4.5, 4.6 ve 4.7'de üç farklı derin öğrenme yapısının Prewitt, Sobel ve Roberts kenar belirleme kuralına uygulanmasının sonuçları görülmektedir.

5. TARTIŞMA VE SONUÇ

Bu tez çalışmasında derin öğrenme yöntemiyle MNIST veri kümesi kullanılarak karakter tanıma yapılmıştır. Elde edilen sonuçları karşılaştırmak amacıyla, MNIST'in Temel Bileşen Analizi (PCA-Principal Component Analysis) yöntemiyle sınıflandırılması da karşılaştırılmıştır. Yapılan karakter tanıma işleminde, derin öğrenme yönteminin PCA yöntemine göre daha üstün olduğu ve tanıma işlemini daha iyi gerçekleştirdiği sonucu elde edilmiştir.

Yapılan diğer uygulamada, kenar belirleme işlemi için AE çıkışı kullanılarak kenar belirleme uygulaması yapılmış ve bilgisayarlı görü (computer vision) alanında önemli bir yere sahip olan kenar tanıma simülasyonu gerçekleştirilmiştir. Uyguladığımız bu yöntem literatürde bu amaç için daha önce hiç uygulanmamıştır. Öncelikle sistemden geçip yeniden yapılandırılmış olarak çıkan görüntülerin çıktıları elde edilmiştir. En çok kullanılan kenar belirleme yöntemlerinden Sobel, Prewitt ve Roberts metodları kullanılmıştır.

İleride, farklı derin öğrenme yapıları kullanılarak yapılabilecek çalışmalarda, daha yüksek verim elde edilebilecek tasarımlar yapılabilir. Ayrıca, bilgisayarda görü alanında kenar belirleme haricinde değişik uygulamalar da gerçekleştirilebilir. Bunların haricinde, yapay zekâyı ilgilendiren diğer konularda da (robotik, biyoloji/tıp, finans, bilgisayarla görü, semantik web, ses vb.) derin öğrenme algoritmalarının iyi sonuçlar vereceği tahmin edilmektedir.

KAYNAKLAR

- [1]. Anil, K.J., Jianchang, M. and Mohuiddin, K.M., 1996, Artificial Neural Networks: A Tutorial, *Computer - Special issue: neural computing*, 29 (3) , 31-44.
- [2]. Salakhutdinov, R. and Hinton, G., 2012, An efficient learning procedure for deep Boltzmann machines, *Neural Computation*, 24 (8) , 1967-2006.
- [3]. Hinton, G., Osindero, S. and Teh, Y., 2006, A fast learning algorithm for deep belief nets, *Neural Computation*, 18 (7), 1527-1554.
- [4]. Deng, L., Platt, J. and Yu, D., 2012, Scalable stacking and learning for building deep architectures, *Int. Conf. on Acoustics, Speech and Signal Processing*, 25-30 Mar. 2012 Kyoto, Japan, IEEE, 2133-2136.
- [5]. Collobert, R., Mobahi, H., Rathe, F. and Weston, J., 2012, *Deep Learning via semi-supervised embedding*, Neural Networks: Tricks of the Trade, In: Grégoire, M. (ed.), Chapter 26, Springer Berlin Heidelberg, USA, ISBN: 978-3-642-35288-1, 639-655.
- [6]. Bengio, Y., Courville, A., Manzagol, P. A. and Vincent, P., 2010, Why does unsupervised pre-training help deep learning?, *The Journal of Machine Learning Research*, 11 (2), 625-660.
- [7]. Cheng, J., Eickholt, J. and Spencer, M., 2015, A deep learning network approach to ab initio protein secondary structure prediction, *IEEE Transactions on Computational Biology and Bioinformatics*, 12 (1), 103-112.
- [8]. Charalampous, K. and Gasteratos, A., 2014, A tensor-based deep learning framework, *Image and Vision Computing*, 32 (11), 916-929.
- [9]. Eswaran, C., Tan, C.C., 2008, Reconstruction of handwritten digit images using autoencoder neural networks, *Canadian Conference on Electrical and Computer Engineering*, 4-7 May 2008 Niagara Falls, Canada, IEEE, ISBN: 978-1-4244-1642-4, 465-470.
- [10]. Öztemel, E., 2013, *Yapay Sinir Ağları*, Papatya Yayıncılık, 3. Baskı, İstanbul, ISBN: 978-975-6797-39-6.
- [11]. Doğan, M., 2003, *Neurological diagnosis system based on neural networks*, Thesis (MSc), Anadolu University.
- [12]. Tolon, M. and Tosunoğlu, N.G., 2008, Tüketici tatmini verilerinin analizi: Yapay sinir ağları ve regresyon analizi karşılaştırması, *Gazi Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, 10 (2), 247-259.

- [13]. Çayıroğlu, İ., 2013, *İleri Algoritma Analizi*, <http://www.ibrahimcayiroglu.com/Dokumanlar/IleriAlgoritmaAnalizi/IleriAlgoritmaAnalizi-10.Hafta-ParcacikSuruAlgoritmasi.pdf>, [Ziyaret tarihi: 4 Mart 2015].
- [14]. Vanderplas, J., 2012, *Sklearn tutorial*, http://www.astroml.org/sklearn_tutorial/general_concepts.html, [Ziyaret tarihi: 9 Nisan 2015].
- [15]. Stelmo, M.B.N. and Vanessa, R.C.L., 2008, *Application on reinforcement learning for diagnosis based on medical image*, Reinforcement learning, In: Weber, C. and Elshaw, M. (ed.), Chapter 20, I-Tech education and publishing, ISBN: 978-3-902613-14-1, 400-410.
- [16]. Kölmek, M.A., 2012, *Modelling the day ahead price at electricity balancing and settlement market of Turkey by using artificial neural networks*, Thesis (MSc), Ankara University.
- [17]. Karakuzu, C., 2012, http://bm.bilecik.edu.tr/Dosya/Icerik/107/DosyaEki/nsg_ders_notu.pdf, [Ziyaret tarihi: 13 Nisan 2015].
- [18]. Özşahin, M., 2009, *Türkiye otomotiv sektörünün kendini örgütleyen haritalar ile finansal analizi*, Thesis (MSc), Çukurova Üniversitesi.
- [19]. Tosun, S., 2010, *Yapay sinir ağlar*, http://www.suleymantosun.com/wp-content/uploads/2010/05/Yapay_Sinir_Aglari.pdf, [Ziyaret tarihi: 15 Nisan 2015].
- [20]. Sezgin, N. and Tekin, R., 2012, Örüntü tanımda Hopfield ağıının kullanılması, *Yaşam bilimleri dergisi*, 1 (2), 257-266.
- [21]. Battle, A. and Raina, R., 2007, Self-taught learning: Transfer learning from unlabeled data, *24th Annual international conference on machine learning*, 20-24 June 2007 Corvallis, Oregon, 323-330.
- [22]. Faulkner, R., 2010, *Dyna learning with deep belief networks*, Thesis (MSc), McGill University.
- [23]. Hinton, G.E., 2007, *Boltzman machine*, http://www.scholarpedia.org/article/Boltzmann_machine, [Ziyaret tarihi: 16 Nisan 2015].
- [24]. Cho, K.H., 2011, *Improved Learning Algorithms for Restricted Boltzmann Machines*, Thesis (MSc), Aalto University.
- [25]. Dieterle, F., 2006, http://www.frank-dieterle.de/phd/2_7_1.html , [Ziyaret tarihi: 23 Nisan 2015].
- [26]. Eswaran, C. and Tan, C.C., 2008, Performance comparison of three types of autoencoder neural networks, *Second asia international conference on modelling & simulation*, 13-15 May 2008 Kuala Lumpur, IEEE Computer Society, ISBN: 978-0-7695-3136-6, 213-218.

- [27]. Hinton, G. E. and Salakhutdinov, R. R., 2006, Reducing the Dimensionality of Data with Neural Networks, *Science*, 313 (5786), 504-507.
- [28]. Hinton G., *Matlab for science*, <http://www.cs.toronto.edu/hinton/MatlabForSciencePaper.html>, [Ziyaret tarihi: 2 Haziran 2015].
- [29]. LeCun Y., *Mnist*, <http://yann.lecun.com/exdb/mnist/>, [Ziyaret tarihi: 4 Haziran 2015].

ÖZGEÇMİŞ



Kişisel Bilgiler

Adı Soyadı	Cihan TİKEN
Uyruğu	T.C
Doğum tarihi, Yeri	24/06/1985, Bismil
Telefon	05354790270
E-mail	cihantiken@gmail.com

Eğitim

Derece	Kurum/Anabilim Dalı/Programı	Yılı
Yüksek Lisans	İ.Ü. Fen Bilimleri Enstitüsü/ Bilgisayar Mühendisliği Anabilim / Bilgisayar Mühendisliği Programı	2015
Lisans	Yakın Doğu Üniversitesi (Burslu)	2009
Lise	Şanlıurfa Anadolu Lisesi	2003

Makaleler / Bildiriler

Bildiri: Tiken C., Ensari T., “Deep Learning for Edge Detection”, *International Symposium on Innovative Technologies In Engineering and Science (ISITES’2015)*, Universidad Politécnica de Valencia, Valencia, Spain, June 3-5, 2015.