YAPAY SİNİR AĞLARINDA MOMENTUMLU DİK İNİŞ VE EŞLENİK GRADYAN EĞİTİM ALGORİTMALARININ KARŞILAŞTIRILMASI

Engin TAŞ

Yüksek Lisans Tezi

Fen Bilimleri Enstitüsü
İstatistik Anabilim Dalı

Temmuz - 2005

JÜRİ VE ENSTİTÜ ONAYI

Engin Taş' ın "**Yapay Sinir Ağlarında Dik İniş ve Eşlenik Gradyan Eğitim Algoritmalarının Karşılaştırılması**" başlıklı **İstatistik Anabilim Dalındaki**, Yüksek Lisans tezi.....tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı-Soyadı	**İmza**
Üye (Tez Danışmanı)	**: Doç. Dr. Mammadagha MAMMADOV**	**……**
Üye	**: Prof. Dr. Ali Fuat YÜZER**	**……**
Üye	**: Doç. Dr. Vakıf CAFEROV**	**……**

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …… tarih ve …… sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ÖZET

Yüksek Lisans Tezi

**YAPAY SİNİR AĞLARINDA
MOMENTUMLU DİK İNİŞ VE EŞLENİK GRADYAN
EĞİTİM ALGORİTMALARININ KARŞILAŞTIRILMASI**

ENGİN TAŞ

**Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
İstatistik Anabilim Dalı**

Danışman: Doç. Dr. Mammadagha MAMMADOV
2005, 116 sayfa

Bu tezde, tahmin ve sınıflandırma problemlerinde istatistik yöntemlere alternatif olarak önerilen geriye yayılım yapay sinir ağı (y.s.a.) eğitim algoritmaları incelenmiştir. Kuadratik hata fonksiyonunun minimizasyonu probleminde öğrenme oranı ve momentum faktörünün süreç boyunca sabit olduğu momentumlu gradyan azalan metodunda, algoritmanın yakınsak olduğu parametre aralıkları türetilmiştir. Optimum öğrenme oranı ve momentum faktörlerinin dağılımı rassal olarak üretilmiş problemler üzerinde gözlemlenmiştir. Öğrenme oranı ve momentum faktörünün süreç boyunca değişken olduğu durumda ise sistemin öz bileşenlerinin tavrı gözlemlenmiş ve belirli bir fiziksel sistemle benzerliği incelenmiştir.

İncelenen eğitim algoritmaları güncel bir sınıflandırma problemi üzerinde bir simülasyon çalışmasıyla karşılaştırılmıştır. Çalışmada, nonobstrüktif azoospermi (ejakulatta sperm bulunmaması) ' ye sahip erkeklerde testis biopsisine dayanan spermatozoa (sperm hücresi) tahmini için bir y.s.a. geliştirilmiştir. Eğitim algoritmaların performansları değerlendirilirken yakınsama süreleri, devir sayıları ve doğru sınıflandırma oranları dikkate alınmıştır.

Anahtar Kelimeler: Geriye Yayılım, Momentum, Kararlılık, Eşlenik Gradyan

ABSTRACT

Master of Science Thesis

**COMPARISON OF
STEEPEST DESCENT WITH MOMENTUM AND CONJUGATE
GRADIENT TRAINING ALGORITHMS
IN NEURAL NETWORKS**

ENGİN TAŞ

**Anadolu University
Graduate School of Sciences
Statistics Program**

**Supervisor: Assoc. Prof. Dr. Mammadagha MAMMADOV
2005, 116 pages**

In this thesis, backpropagation training algorithm, proposed as an alternative to statistical methods in forecasting and classifying problems is analysed. Bounds for convergence on learning rate and momentum coefficients are derived in the case of quadratic error function minimization problem with constant learning rate and momentum coefficients. In experiments, distribution of optimum learning rate and momentum coeffiecients are observed on randomly generated problems. In particular, for a quadratic error function, behaviour of the dynamic system where the choice of learning and momentum parameters that has been referred to as 'optimally tuned' is observed. An analogy between the dynamic method and a prescribed physical system is examined.

Training methods that has been studied are compared on an up-to-date classification problem in a simulation. A neural network is developed for predicting spermatozoa prior to testicular biopsy in men with nonobstructive azoospermia. Convergence times, epoch numbers and correct classification rates are considered for the evaluation of training algorithm performances

Keywords: Backpropagation, Momentum, Stability, Conjugate Gradient

TEŞEKKÜR

Hazırlamış olduğum yüksek lisans tezinde bana sürekli yol gösteren, sorularımı bıkmadan cevaplayan, hazırlamış olduğum en ufak çalışmayı bile en ince ayrıntısına kadar değerlendiren, onca çalışmasının arasında bana zamanını ayıran, matematiği sevdiren ve benim bu meslekte heyecanlandıran, kanımı kaynatan hocalarımın başında gelen, kendisinden her konuda çok şey öğrendiğim çok değerli ve sevgili danışman hocam sayın Doç. Dr. Mammadagha MAMMADOV' a çok teşekkür ederim.

Çalışmam süresince beni hiç yalnız bırakmayan, bir problemim olduğunda her zaman yardımcı olan çok sevgili İstatistik Bölümü Araştırma Görevlisi arkadaşlarıma teşekkür ederim.

Hayatım boyunca her zaman yanımda olan, desteklerini esirgemeyen sevgili ağabeylerime, ablama ve beni yetiştiren canım anneme ve babama derin sevgi ve saygılarımla teşekkür ederim.

Engin TAŞ

Temmuz 2005

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER ve KISALTMALAR DİZİNİ

η	Öğrenme oranı
μ	Momentum faktörü
κ	Özdeğerler
∇	Gradyan operatörü
w	Ağırlık vektörü
α	Adım uzunluğu
ε	Hata fonksiyonu
p	Yön vektörü
r	Artık
A	Hessian matrisi
m	Kütle
v	Sürtünme Katsayısı
BP	Backpropagation (Geriye Yayılım)
BPM	Backpropagation with Momentum (Momentumlu Geriye Yayılım)
CG	Conjugate Gradient (Eşlenik Gradyan)
CGM	Conjugate Gradient Method (Eşlenik Gradyan Metodu)
SCG	Scaled Conjugate Gradient (Ölçekli Eşlenik Gradyan)
min	Minimum
max	Maksimum

1. GİRİŞ

Günümüzde bilgisayarlar çok karmaşık sayısal işlemleri ve matematiksel problemleri bir saniyeden çok daha kısa bir sürede çözümleyebilmekte ve bu noktada insan beyni karşısında önemli bir üstünlük sağlamaktadır. Sayısal işlemlerde bu derece üstün olan bilgisayarlar; öğrenme, tanıma ve deneyimlerle kazanılmış bilgileri kullanabilme noktasında ise çok yetersizdir. Örnek olarak, günlük yaşantımızda çevremizde gördüğümüz nesneleri tanımamız için gereken süre nanosaniyeler mertebesindedir. Yolda yürürken karşılaştığımız bir kişi ile ilgili pek çok özelliği (fiziksel durumu, giydiği giysinin rengi ve modeli, yüz ifadesi, taşıdığı çantanın şekli vs.) bir saniyeden daha az bir sürede tanımlayabilmekteyiz. Oysa ki bir bilgisayar bu işlemleri çok daha uzun bir sürede yapabilmekte ve elde ettiği sonuç, bir insan beyninin ürettiği sonuç kadar kesin ve yeterli olamamaktadır.

Bilgisayarların insan beyni karşısındaki bu yetersizliği, daha "zeki" bilgisayarlar ortaya çıkarmaya çalışan bilim adamlarını bu konuda araştırmalar yapmaya yöneltmiştir. Eğer insan beyninin öğrenme ve tanımayı ne şekilde gerçekleştirdiği matematiksel olarak modellenebilseydi ve bu model bilgisayar programlarına uyarlanabilseydi, "düşünen" bilgisayarlar yapma yolunda çok önemli bir engel aşılabilecekti. İşte bu konudaki ilk çalışmalar 1943 yılında Amerikalı iki bilim adamı, nörofizyolog Warren McCulloch ve matematikçi Walter Pitts tarafından yapılmıştır (McCulloch ve Pitts, 1943). McCulloch ve Pitts, insan beyninin hesaplama yeteneğinden esinlenmiş ve elektrik devreleriyle basit bir sinir ağını modellemişlerdir.

Bir sinir ağı için birincil öneme sahip özellik, ağın bulunduğu çevreden öğrenme yeteneği ve öğrenme süresince de performansını geliştirmesidir. Performansdaki gelişme önceden belirlenmiş bir ölçüye uygun olarak zaman içerisinde gerçekleşir. Bir sinir ağı çevresi hakkındaki bilgiyi sinaptik ağırlıklarına ve sapma düzeylerine uygulanan (interaktif) düzeltmeler süreciyle öğrenir. İdeal olarak ağ, öğrenme sürecinin her iterasyonundan sonra çevresi hakkında daha bilgili hale gelir.

Bu nedenle yapay sinir ağları üzerindeki en yoğun çalışmalar bu ağların eğitilmesi konusunda olmaktadır. Zira etkin bir algoritma ile eğitilebilen, yani

öğrenebilen ağlar, yeni şekilleri tanıyabilmekte (şekil tanıma) veya verilen bir girdinin hangi sınıfa ait olduğuna karar verebilmektedir (sınıflandırma).

İleri beslemeli yapay sinir ağlarının eğitimi için *geriye yayılım BP (backpropagation)* algoritması zor problemlerde bile güçlü olduğunu göstermiştir. Ama, BP' nin yüksek performansına, ağ parametrelerini düzenlemek için gerekli olan uzun eğitim zamanı maliyetinde erişilir ki bu gerçek uygulamalarda caydırıcı olabilir. Oldukça basit problemlerde bile, standart BP sıkça eğitim örneklerinin tamamının yüzlerce veya binlerce kez işlendiği uzun eğitim süreci gerektirir. Bu süreci hızlandırmak için literatürde birçok çalışma yapılmıştır.

Yu ve Chen (1997)' de, dinamik seçimli optimal öğrenme oranı ve momentum faktörü kullanan, etkin geriye yayılım eğitim algoritması incelenmiş, öğrenme oranı ve momentum faktörüne göre türevleri geliştiren bir yaklaşımlar ailesi sunulmuştur.

Kamarthi ve Pittner (1999)' da, geriye yayılım algoritması için her biri birbirine bağlı ağırlığın extrapolasyonuna dayanan genel bir hızlandırma tekniği sunulmuştur. Yeni metodun performansı, *eşlenik gradyan* algoritmasıyla karşılaştırılmıştır.

Qian (1999)' da, sürekli zaman limitinde momentum parametresinin, tutucu bir kuvvet alanı içinde yapışkan bir ortamda hareket eden Newton partikülleri kümesine benzer olduğu gösterilmiştir. Lokal minimum civarında sistemin davranışının, bir *eşli ve sönümlendirilmiş harmonik dalgalanıcılara* denk olduğu söylenmiştir.

Torii ve Hagan (2002)' de, kuadratik performans fonksiyonları için dik iniş eğitiminde momentum faktörünün etkisi analiz edilmiştir. Momentum katsayısının değerinin algoritmanın yakınsama özelliklerinin nasıl değiştirdiği gösterilmiştir.

Bhaya ve Kaszkurewicz (2004)' de, kuadratik hata fonksiyonu için, optimal olarak düzeltilmiş öğrenme oranı ve momentum parametrelerinin seçiminin, kesin olarak eşlenik gradyan algoritmasının kullanımına denk olduğu gösterilmiştir.

Buradan hareketle, bu tezde temel olarak yapay sinir ağlarının eğitimi ele alınmıştır. Bir eğitim algoritmasını diğerlerinden ayıran en önemli özelliklerin başında *kararlılığı ve yakınsama oranı* gelir. Bir yapay sinir ağı eğitilirken öğrenme oranı ve momentum faktörü gibi önemli parametre seçimleri yapılır. Bu

parametreler ise algoritmanın kararlılığını ve yakınsama hızını önemli derecede etkiler ve algoritmadan algoritmaya değişiklik gösterir. Eğer dikkatli seçim yapılmazsa algoritma kararsız hale gelebilir. Genelde bu seçimler pratikte kullanıcıya bırakılır. Aslında bu parametrelerin seçimi problemi de kendi başına bir optimizasyon problemidir ve değişik yaklaşımlar literatürde mevcuttur.

Bu tezde yapılan çalışmalar beş bölümde ele alınmıştır:

Birinci bölümde, konuya genel bir giriş yapılmış ve daha önce yapılan çalışmalara kısaca değinilmiştir. Yapay sinir ağı fikrinin temelde nereden geldiğine ve ilk çalışmaların nasıl başladığına yer verilmiştir. Bir yapay sinir ağının çalışma prensibini belirleyen en önemli unsurun ağın eğitim algoritması olduğu ve bu konuda karşılaşılan problemler kısaca özetlenmiştir.

İkinci bölümde, yapay sinir ağları, avantajları anlatılmış ve insan beyninin çalışma prensibinin bilgisayara uyarlanmasının ilk örnekleri olan nöron modellerinden bahsedilmiştir. İlk perceptron ve çok katmanlı perceptronlardan sonra ileri beslemeli temel ağ mimarileri kısaca anlatılmıştır. Ağın eğitim süreci yani öğrenme sürecine değinilmiş, hata düzeltmeli öğrenme kısaca anlatılmış ve son olarak ağ eğitim algoritmaları içerisinde en popüler olan ve sık kullanılan geriye yayılım algoritması detaylı olarak anlatılmıştır.

Üçüncü bölümde, öğrenme problemi matematiksel olarak ortaya konulmuş ve geriye yayılım algoritmasının temelini oluşturan gradyan yöntemleri anlatılmıştır. Öncelikle gradyan düşümü yöntemi genel olarak verilmiş ve bu yönteme momentum teriminin eklenmesiyle oluşan momentumlu gradyan düşümü metodu anlatılmıştır. Öğrenme oranının seçiminde kullanılan *doğru arama* genel olarak verilmiştir. Daha sonra gradyan yöntemleri ailesinin önemli bir sınıfını oluşturan eşlenik gradyan metodu ayrıntılı olarak ele alınmıştır. Doğrusal eşlenik gradyan ve doğrusal olmayan eşlenik gradyan anlatılmış ve son zamanlarda sıkça kullanılan *ölçekli eşlenik gradyan* algoritması incelenmiştir. Bu bölümde algoritmaların karşılaştırılmasını yapabilmek için güncel bir sınıflandırma problemi üzerinde çalışılmıştır. Bu çalışmada *nonobstrüktif azoospermi'* ye sahip erkeklerde *testicular biopsy'* ye dayanarak *spermatozoa* (sperm hücresi) bulunup bulunmadığı tahmin edilmiştir. Veriler 1997 ile 2000 yılları arasında Afyon Kocatepe Üniversitesi Araştırma Hastanesinde *testicular sperm extraction* (testiküler sperm ekstraksiyonu; **TESE**) uygulanmış 303 erkek hastanın medikal

kayıtlarından alınmıştır. Ağ mimarisi oluşturulduktan sonra eğitim aşamasında incelenen algoritmalar denenmiş ve yakınsama ve sınıflandırma performansları bir simülasyon çalışması ile değerlendirilmiştir.

Dördüncü bölümde, yapay sinir ağının ağırlıklarının fonksiyonu olan hata fonksiyonunun kuadratik olduğu durumda momentumlu gradyan düşümü algoritmasının kararlılığı ele alınmıştır. Öncelikle öğrenme oranının ve momentum faktörünün süreç boyunca sabit olduğu durumda kararlılık problemiyle ilişkili teorik incelemeler yapılarak 4.1, 4.2, 4.3 teoremlerinde verilen sonuçlar ispatlanmıştır. Daha sonra ise öğrenme oranı ve momentum faktörünün dinamik seçimli olduğu BPM algoritmasının kararlılığı ve yakınsama hızı incelenmiş ve eşlenik gradyan metodu ile karşılaştırılmıştır. Teorik olarak yapılan incelemeler sonrasında elde edilen sonuçlar, bir simülasyon çalışmasında rassal problemler üretilerek denenmiş ve karşılaştırılmıştır. Ayrıca yine deneylerden elde edilen sonuçlar teoride daha önce yapılan çalışmalarla da karşılaştırılmış, benzerlikler gözlemlenmiştir.

2. İLERİ SÜRÜMLÜ YAPAY SİNİR AĞ (YSA) MODELLERİ: GERİYE YAYILIM ALGORİTMASI

2.1. Ana Çizgileriyle Yapay Sinir Ağları

Yapay sinir ağlarındaki çalışmalar, başlangıcını insan beyninin günümüz dijital bilgisayarlarından tamamen farklı bir çalışma şekli olduğunun anlaşılması gerçeğinden almıştır. İnsan beyni çok karmaşık, doğrusal olmayan ve paralel bir bilgisayardır (bilgi-işleme sistemi). İnsan beyni ciddi hesaplamaları (örn: örnek tanıma, algılama ve motor kontrol gibi) günümüzün en hızlı bilgisayarlarından daha hızlı yapacak şekilde, nöronlar olarak bilinen yapısal bileşenlerini organize etme yeteneğine sahiptir. Örneğin, insanın görme işlevi önemli bir bilgi-işleme sürecidir (Marr 1982; Levine 1985; Churchland ve Sejnowski 1992). Görsel sistemin bizim etrafımızdaki ortamın bir temsilini vermesi ve daha da önemlisi, ortamla nasıl etkileşimde bulunacağımızla ilgili bilgiyi bize sağlaması fonksiyonudur. Daha açık olarak, beynimiz rutin olarak algısal tanıma görevlerini (örn: tanıdık olunmayan bir görüntü içinde gömülü olan tanıdık bir yüzü tanımamız gibi) yaklaşık 100-200 ms içinde gerçekleştirir, buna rağmen günümüz bilgisayarlarında daha az karmaşıklık içeren işlemler günlerce sürebilir.

Başka bir örnek vermek gerekirse, bir yarasanın sonarını düşünelim. Sonar bir yankı-yerleştirme sistemidir. Yarasanın sonarı, hedefin (örn: uçan bir sinek) ne kadar uzakta olduğu bilgisini verir ve ek olarak hedefin hızı, hedefin büyüklüğü, hedefin çeşitli niteliklerinin büyüklüğü, yatay sapma açısı ve irtifası hakkında bilgi toplar (Suga 1990a, Suga 1990b). Hedeften tüm bu bilgiyi çıkarsamak için gerekli olan karmaşık sinirsel hesaplamaların tümü erik büyüklüğündeki bir beyinde gerçekleşir. Gerçekten de, bir yankı-yerleştirmeli yarasa, hedefini bir radarın veya bir sonar mühendisinin gıpta edeceği kolaylıkla ve başarı oranıyla izler ve yakalar.

O zaman, bir insan beyni veya bir yarasa beyni bunu nasıl yapar? Doğumda, beyin çok büyük bir yapıya ve çoğunlukla “tecrübe” olarak nitelendirdiğimiz kendi kurallarını oluşturma yeteneğine sahiptir. Aslında, tecrübe zamanla oluşur ve insan beyninin en verimli gelişimi doğumdan itibaren ilk iki yılda gerçekleşir ama gelişim bu aşamadan sonrada devam eder.

Gelişen bir nöron esneyebilen plastik bir beyne benzetilebilir: Esneklik gelişen sinir sisteminin onu çevreleyen ortama adaptasyonuna izin verir. Esneklik, insan beyninde nöronların bilgi-işleme birimleri gibi çalışmaları için vazgeçilmez göründüğü için sinir ağları da yapay nöronlardan oluşturulur. En genel biçimiyle, bir sinir ağı beynin belirli bir görev veya ilgili fonksiyonunu gerçekleştirme yolunu modellemek için tasarlanan bir makinedir; ağ çoğu kez elektronik bileşenler kullanılarak veya dijital bir bilgisayardaki yazılımla benzetilir. Uyarlanabilir bir makine olarak görülen sinir ağının bir tanımını verecek olursak:

Bir yapay sinir ağı, tecrübesel bilgiyi saklamak için doğal bir eğilimi olan ve bunu kullanım için elverişli hale getiren, basit işleme birimlerinden oluşmuş geniş ölçekli paralel dağıtılmış bir işlemcidir. Beyni iki açıdan andırır:

1. Bilgi, ağ tarafından bir öğrenme süreciyle çevreden elde edilir.
2. İç nöron bağlantı güçleri, sinaptik ağırlıklar olarak bilinir, elde edilen bilgiyi kaydetmek için kullanılır.

Öğrenme sürecini gerçekleştirmek için kullanılan yordama öğrenme algoritması denir, arzu edilen tasarım amacına ulaşmak ağın sinaptik ağırlıklarını belirli bir sırada düzenleme fonksiyonudur.

2.1.1. Sinir Ağlarının Avantajları

Sinir ağı hesaplama gücünü şu özelliklerden aldığı açıktır, birincisi, geniş ölçekli paralel dağıtılmış yapısından, ve ikincisi, öğrenme kabiliyetinden ve böylelikle genelleştirebilmesinden. Genelleştirme, eğitim (öğrenme) sırasında karşılaşılmayan girdiler için mantıklı çıktılar üretmesini ifade eder. Sinir ağlarının bu iki bilgi-işleme kabiliyeti şu an çözülmesi zor kompleks (geniş-ölçekli) problemleri çözmesine olanak sağlar. Pratikte ise sinir ağları tek başına çalışarak çözüm sağlayamaz. Bunun yerine, tutarlı bir sistem mühendisliği yaklaşımına entegre edilmesi gerekir. Özellikle, ilgilenilen kompleks problem daha basit bölümlere ayrılır ve sinir ağları kendi doğal yeteneklerine uygun bir bölümler altkümesine atanır.

Sinir ağlarının kullanımının aşağıda belirtilen faydalı özellikleri ve yetenekleri vardır:

1. Doğrusal Olmama. Bir yapay nöron doğrusal olabilir veya olmayabilir. Birbirine bağlı doğrusal olmayan nöronlardan oluşmuş bir sinir ağının kendisi doğrusal değildir. Dahası, doğrusal olmama ağ boyunca dağıtıldığı için özel bir türdür. Doğrusal olmama çok önemli bir özelliktir, özellikle girdi sinyalini üreten fiziksel mekanizma (konuşma sinyali) doğuştan doğrusal olmayandır.

2. Girdi-Çıktı Eşlemesi. Popüler bir öğrenme şekli öğreticili öğrenme olarak bilinir. Belirlenmiş bir eğitim örnekleri veya süreç örnekleri kümesi uygulayarak bir sinir ağının sinaptik ağırlıkların düzenlenmesini içerir. Her örnek bir girdi sinyalinden ve karşılık gelen, arzu edilen hedeften oluşur. Ağa kümeden rasgele seçilmiş bir örnek verilir, ve sinaptik ağırlıklar (özgür parametreler), arzu edilen hedefle ağın girdi sinyalinden ürettiği çıktının arasındaki farkı, uygun istatistiksel kriterle beraber minimize edecek şekilde düzenlenmesidir. Ağın eğitimi, sinaptik ağırlıklarda daha fazla dikkate değer bir değişim olmadığı bir durağan duruma ulaşana kadar kümedeki birçok örnek için tekrarlanır. Daha önce uygulanmış eğitim örnekleri eğitim esnasında tekrar uygulanabilir ama farklı sırada. Böylece ağ, ele alınan problem için bir girdi-çıktı eşlemesi inşa ederek örneklerden öğrenir. Böyle bir yaklaşım akla model-bağımsız tahminle uğraşan istatistiğin bir branşı olan parametrik olmayan istatistiksel çıkarsama çalışmasını getirir; burada parametrik olmayan terimi girdi verisi ile ilgili istatistiksel model için herhangi bir ön varsayımın yapılmadığı gerçeğini belirtmek için kullanılmıştır. Örneğin,fiziksel bir objeyi veya olayı ifade eden girdi sinyalinin önceden belirlenmiş çeşitli kategorilere (sınıflara) atayan bir patern sınıflandırma sürecini düşünelim. Bu probleme parametrik olmayan yaklaşımda, gerekli olan bir örnekler kümesi kullanarak patern-sınıflandırma süreci için girdi sinyali uzayında keyfi karar sınırlarını tahmin etmektir ve bunu bir olasılıksal dağılım modeli ortaya koymadan yapmaktır. Benzer bir bakış açısı, girdi-çıktı eşlemesi gerçekleştiren bir sinir ağıyla parametrik olmayan istatistiksel çıkarsama arasında yakın bir benzerlik sunan öğreticili öğrenme algoritmasında gizlidir.

3. Uyarlanabilirlik. Sinir ağları kendilerini çevreleyen ortamdaki değişikliklere, sinaptik ağırlıklarını adapte edebilme yeteneğine sahiptir. Özellikle, belirli bir ortamda çalışacak şekilde eğitilmiş bir sinir ağı, çalışılan ortam koşullarındaki küçük değişikliklerle baş edebilecek şekilde tekrar kolayca eğitilebilir. Dahası, durağan olmayan bir ortamda çalışıyorsa (istatistiklerin zamanla değiştiği), sinir

ağı sinaptik ağırlıklarını zaman içinde değiştirecek şekilde tasarlanabilir. Örnek sınıflandırma, sinyal işleme ve kontrol uygulamalarında sinir ağının doğal mimarisi ağın adapte olma yeteneğiyle birleştirildiğinde, sinir ağını uyarlanabilir örnek sınıflandırma, uyarlanabilir sinyal işlemede ve uyarlanabilir kontrolde faydalı bir araç yapar. Genel bir kural olarak bir sistemi daha uyarlanabilir yaptığımızda, sistemin her zaman kararlı kalmasını sağlayarak sistemin kararlı olmayan ortamlarda çalışması gerektiğinde de performansının daha güçlü olacağı söylenir. Vurgulanması gerekir ki yinede uyarlanabilirlik her zaman güçlülükle sonuçlanmaz hatta tam tersini yapabilir. Örneğin kısa dönem sabitli bir uyarlanabilir sistem hızlı değişebilir ve bu nedenle gerçek olmayan dalgalanmalara cevap verme eğiliminde olabilir, bu da sistemin performansında oldukça kötü bir düşüşe neden olur. Uyarlanabilirliğin tüm faydalarını gerçekleştirmek için sistemin temel zaman sabitleri sistemin gerçek olmayan dalgalanmaları göz ardı etmesi için yeterince uzun olmalıdır ve ortamdaki anlamlı değişimlere cevap verecek kadar kısa olmalıdır; burada anlatılan problem kararlılık-esneklik dilemması olarak bilinir (Grossberg 1988).

4. Sebepsel Sonuç: Örnek sınıflandırma kapsamında, bir sinir ağı sadece belirli bir örneği seçmek için değil aynı zamanda verilen karardaki güvenilirlik hakkında da bilgi sağlayacak şekilde tasarlanabilir. Bu sonraki bilgi eğer şüpheli örnekler olursa bunları reddetmede kullanılabilir ve böylece ağın sınıflandırma performansını geliştirir.

5. Çevresel Bilgi: Bilgi tam olarak sinir ağının yapısıyla ve aktivasyon durumuyla ifade edilir. Ağdaki her nöron, ağdaki tüm diğer nöronların global aktivitesinden potansiyel olarak etkilenir. Sonuç olarak, çevresel bilgi bir sinir ağıyla doğal olarak idare edilir.

6. Hata Toleransı: Donanım şeklinde uygulanan bir sinir ağı doğal olarak hata toleransı olma potansiyeline veya kuvvetli hesaplama yeteneğine sahiptir yani kötü çalışma koşulları altında ağın performansı yumuşak bir şekilde azalır. Örneğin, eğer bir nöron veya onu bağlayan linkler zarar görürse, kaydedilmiş bir örneğin hatırlanması kalite olarak bozulur. Ama ağda kaydedilmiş bilginin dağıtılmış yapısı gereği ağın genel davranışının ciddi olarak düşmesi için hasarın yoğun olması gerekir. Böylece prensipte bir sinir ağı katastrofik bir hatadan çok performansında yumuşak bir düşüş gösterir. Güçlü hesaplama için bazı deneysel

kanıtlar vardır ama genellikle bu kontrol edilemez. Sinir ağının gerçekte de hata toleranslı olmasını garanti etmek için, ağı eğitmede kullanılan algoritmanın tasarımında düzeltici ölçümler almak gerekli olabilir (Kerlirzin ve Vallet 1993).

7. ÇBÖB (Çok Büyük Ölçekli Bütünleşik) Uygulanabilirliği: Sinir ağının kütlesel paralel doğası, ciddi görevlerin hesaplanması için potansiyel olarak hızlı yapar. Bu aynı özellik bir sinir ağını çok büyük ölçekli bütünleşik teknolojilerin kullanılarak uygulanmasına olanak sağlar. ÇBÖB' nin bir yararlı özelliği de gerçekten karmaşık davranışı, yüksek düzeyli hiyerarşik bir düzende yakalanmasını sağlar (Mead 1989).

8. Analiz ve Tasarım Tekdüzeliği: Esasen, sinir ağları bilgi işleyiciler olarak genel olma avantajına sahiptir. Bunu şu anlamda söyleriz; sinir ağları uygulamasını içeren tüm alanlarda aynı notasyon kullanılır. Bu özellik, değişik şekillerde kendini gösterir.

- Nöronlar, bir şekilde veya başka bir şekilde tüm sinir ağlarında ortak bir bileşendir.
- Bu ortaklık, sinir ağlarının farklı uygulamalarındaki teorilerin ve öğrenme algoritmalarının paylaşılmasına olanak sağlar.
- Modüler ağlar dikişsiz bir modüler entegrasyonuyla inşa edilebilir.

9. Nörobiyolojik Benzerlik: Sinir ağının tasarımı beyinle kurulan benzerlikten esinlenerek yapılmıştır, ki hata toleranslı paralel işlemenin sadece fiziksel olarak olası değil aynı zamanda hızlı ve güçlü olduğunun yaşayan bir kanıtıdır. Nörobiyologlar (yapay) sinir ağlarını nörobiyolojik fenomenlerin yorumlanması için bir araştırma aracı olarak görürler. Diğer taraftan, mühendislerde nörobiyolojiyi karmaşık problemleri çözmek için yeni fikirler üretmede kullanılan bir araç olarak görürler.

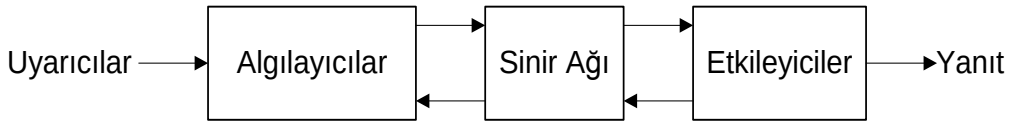
2.1.2. İnsan Beyni

Bir insanın sinir sistemi şekil 2.1 deki blok diagramda gösterildiği gibi üç aşamalı bir sistem olarak görülebilir (Arbib 1987). Sistemin merkezi sinir ağından oluşan, sürekli bilgi alan, algılayan ve uygun kararlar veren beyindir. Şekilde iki ok kümesi gösterilmiştir. Soldan sağa doğru olanlar sistemde bilgi taşıyıcı sinyallerin ileri aktarımını gösterir. Sağdan sola olanlar ise sistemdeki geri

bildirimin varlığını işaret eder. Algılayıcılar insan vücudundan veya dış ortamdan gelen uyarıları, bilgiyi sinir ağına (beyine) taşıyan elektrik sinyallerine çevirirler. Etkileyiciler sinir ağı tarafından oluşturulan elektrik sinyallerini sistem çıktısı olarak anlaşılabilir eylemlere dönüştürürler.

Beyni anlamak için verilen uğraş, beynin yapısal bileşenleri olan nöron fikrini bularak keşfedici bir iş yapan Roman y Cajál (1911) sayesinde hafiflemiştir. Tipik olarak, nöronlar silikon mantık kapılarına göre 5 veya 6 düzey daha yavaştır; silikon bir çipte olaylar nanosaniye düzeyinde gerçekleşir (10^{-9} s), sinirsel olaylar ise milisaniye düzeyinde gerçekleşir (10^{-3} s).

Buna rağmen nöronun çipe göre daha yavaş çalışmasının sebebi beynin birbirine büyük bağlantılarla bağlı şok edici nöron sayısına sahip olmasıdır. İnsan korteksinde yaklaşık olarak 10 milyar nöron ve 60 trilyon sinir ucu veya bağlantısı olduğu tahmin edilmektedir (Shepherd ve Koch 1990). Açık sonuç beynin inanılmaz verimli bir yapı olduğudur. Spesifik olarak, beynin enerjisel verimliliği yaklaşık olarak 10^{-16} joule'dur (J), saniye başına düşen operasyon sayısı olarak. Günümüzde kullanılan en iyi bilgisayarlarda ise karşılık gelen değer 10^{-6} joule' dur (J) (Faggin 1991).



Şekil 2.1. Sinir siteminin blok diyagram olarak gösterimi

2.1.3. Nöron Modelleri

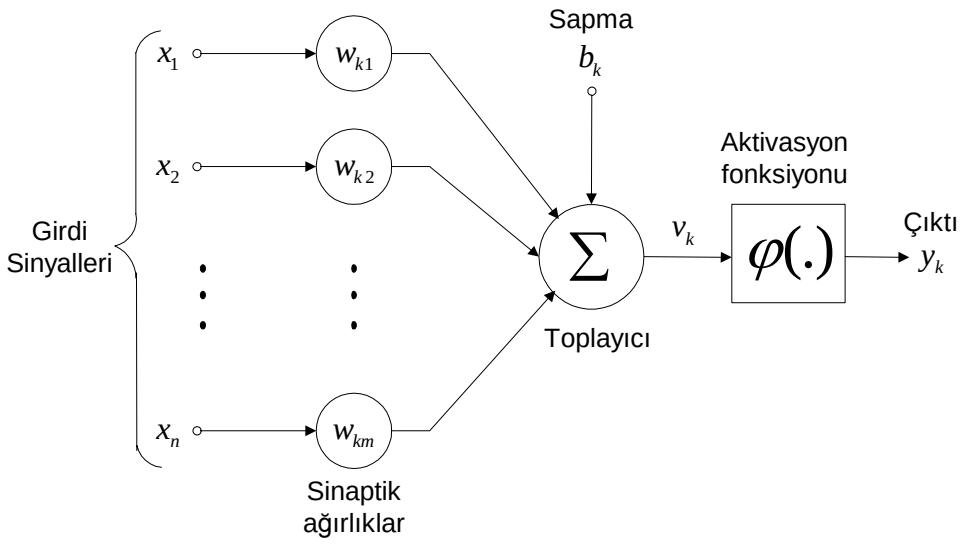
Bir nöron sinir ağının çalışma temeli olan bilgi-işleyici bir birimdir. Şekil 2.2 sinir ağlarını (yapay) tasarlamanın temelini oluşturan bir nöron modelini gösterir. Burada, sinirsel modelin üç temel elemanını belirleyeceğiz.

1. Her biri kendisine ait bir ağırlık veya güçle karakterize edilmiş bir sinapslar veya bağlantılı linkler kümesi. Özel olarak, k nöronuna bağlı j sinapsinin girişindeki bir x_j sinyali w_{kj} sinaptik ağırlığıyla çarpılır. Sinaptik ağırlık w_{kj}'nin alt indislerinin yazıldığına dikkat etmek gerekir. İlk alt indis sorgulanan nöronu

gösterir ve ikinci alt indis ağırlığın ilişkili olduğu sinapsın girdi ucunu gösterir. Beyindeki bir sinapsin tersine, yapay bir nöronun sinaptik ağırlığı pozitif olduğu kadar negatif değerler de içeren bir aralıkta olabilir.

2. Nöronun bağlı olduğu sinapsları tarafından ağırlıklandırılan girdi sinyallerini toplamak için bir toplayıcı; burada anlatılan işlemler bir doğrusal birleştirici oluşturur.

3. Nöronun çıktı büyüklüğünü kısıtlamak için bir aktivasyon fonksiyonu. Aktivasyon fonksiyonu aynı zamanda kabul edilebilir çıktı sinyali büyüklüğü aralığını belirli sonlu değere sıkıştıran (limitleyen) bir sıkıştırıcı fonksiyon olarak tanımlanır.



Şekil 2.2. Doğrusal olmayan nöron modeli.

Şekil 2.2' de gösterilen nöron modeli aynı zamanda harici olarak uygulanan bir sapma da içerir ve b_k ile gösterilir. Sapma b_k aktivasyon fonksiyonu net girdisini pozitif veya negatif olmasına bağlı olarak artırma veya azaltma etkisine sahiptir.

Matematiksel terimlerle, bir k nöronunu aşağıdaki denklemler çiftini yazarak ifade edebiliriz.

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

ve

$$y_k = \varphi(u_k + b_k) \tag{2.2}$$

burada $x_1, x_2, ..., x_m$ girdi sinyalleri; $w_{k1}, w_{k2}, ..., w_{km}$ k nöronunun sinaptik ağırlıkları; u_k girdi sinyallerine bağlı doğrusal birleştirici çıktısı; b_k sapma; $\varphi(.)$ aktivasyon fonksiyonu; ve y_k nöronun çıktı sinyalidir. b_k sapmasının kullanımı , Şekil 2.2' de modeldeki doğrusal birleştiricinin çıktısı u_k'ya afin dönüşümü uygulama etkisine sahiptir ve aşağıdaki şekilde gösterilebilir.

$$v_k = u_k + b_k \tag{2.3}$$

Sapma b_k yapay nöron k 'nın harici bir parametresidir. Denk olarak, 2.1-2.3 denklemlerinin kombinasyonunu aşağıdaki gibi formüle edebiliriz.

$$v_k = \sum_{i=1}^{m} w_{kj} x_j \tag{2.4}$$

ve

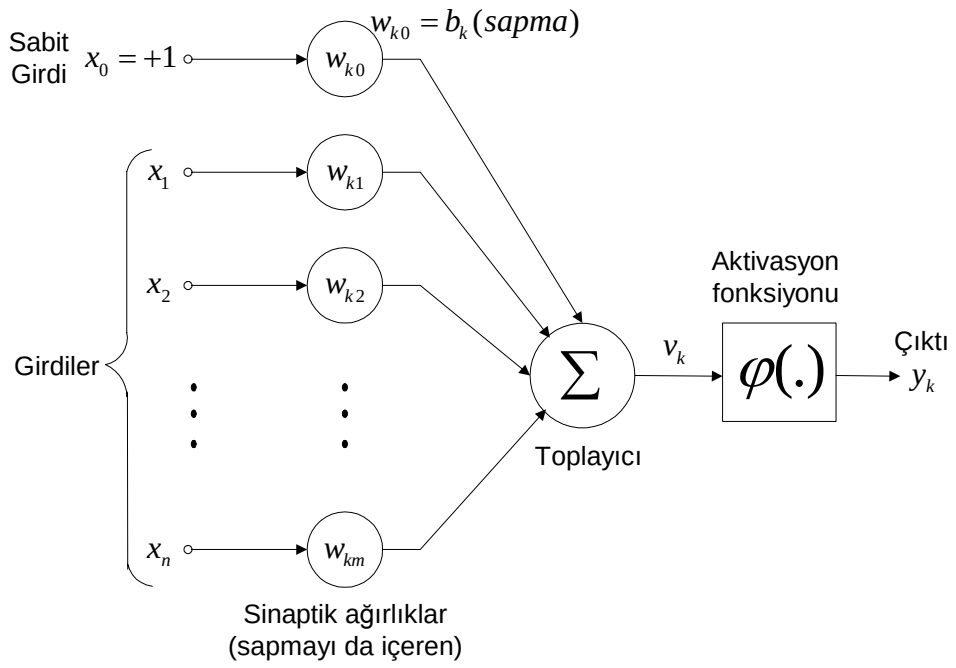
$$y_k = \varphi(v_k) \tag{2.5}$$

Eşitlik 2.4' e, yeni bir sinaps ekledik. Girdisi,

$$x_0 = +1 \tag{2.6}$$

ve ağırlığı ise

$$w_{k0} = b_k \tag{2.7}$$



Şekil 2.3. Başka bir doğrusal olmayan nöron modeli.

Böylece k nöronunun modelini şekil 2.3' deki gibi yeniden formüle edebiliriz. Bu şekilde, sapmanın etkisi iki şey yapılarak gerçekleştirilmiş oldu. (1) $+1$' de sabitlenen yeni bir girdi sinyali ve (2) b_k sapmaya eşit yeni bir sinaptik ağırlık. Şekil 2.2 ve .2.3' deki modeller farklı görünmelerine rağmen matematiksel olarak denktirler.

2.1.4. Aktivasyon Fonksiyonu Çeşitleri

Aktivasyon fonksiyonu, $\varphi(v)$ ile gösterilir ve nöronun v' ye göre çıktısını tanımlar. Burada temel aktivasyon fonksiyon çeşitlerini belirteceğiz.

1. Eşik Fonksiyon: Bu tip aktivasyon fonksiyonu genel olarak şu şekildedir.

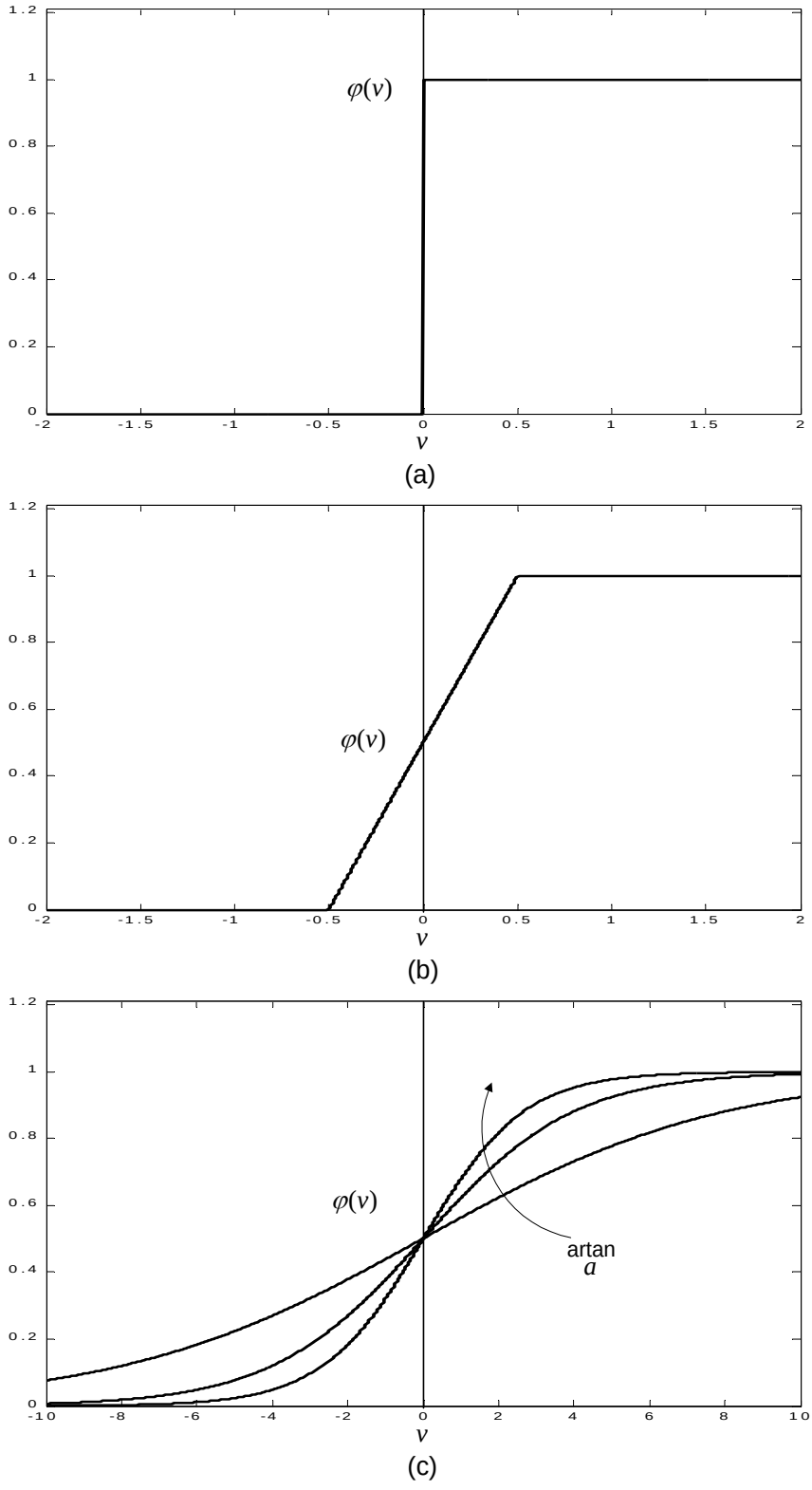
$$\varphi(v) = \begin{cases} 1, & eğer\ v \geq 0 \\ 0, & eğer\ v < 0 \end{cases} \tag{2.8}$$

Paralel olarak böyle bir eşik fonksiyon kullanan k nöronunun çıktısı aşağıdaki şekilde ifade edilebilir.

$$y_k = \begin{cases} 1, & eğer\ v_k \geq 0 \\ 0, & eğer\ v_k < 0 \end{cases} \tag{2.9}$$

burada v_k nöronunun net çıktısıdır yani

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k \tag{2.10}$$



Şekil 2.4. (a) Eşik fonksiyonu. (b) Parçalı-doğrusal fonksiyon. (c) Değişken a eğim parametreli sigmoid fonksiyonu.

Böyle bir nöron literatürde Mc Culloch ve Pitts (1943) tarafından yapılan öncü çalışmanın hatırına Mc Culloch-Pitts modeli olarak adlandırılır. Bu modelde, eğer nöronun girdisi negatif değilse nöronun çıktısı 1 değerini alır, diğer durumda ise 0 değerini alır. Bu ifade Mc Culloch-Pitts modelinin ya hep ya hiç özelliğini açıklar.

1. Parçalı-Doğrusal Fonksiyon: Şekil 2.4b' de anlatılan parçalı doğrusal fonksiyon için,

$$\varphi(v) = \begin{cases} 1, & v \geq \frac{1}{2} \\ v, & \frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \tag{2.11}$$

2. Sigmoid Fonksiyon: Sigmoid fonksiyonun grafiği s şeklindedir ve yapay sinir ağlarının oluşturulmasında kullanılan aktivasyon fonksiyonlarının içinde en yaygın olanıdır. Doğrusal ve doğrusal olmayan tavırlar arasında zarif bir denge gösteren ciddi artan bir fonksiyon olarak tanımlanır. Sigmoid fonksiyonun bir örneği aşağıdaki şekilde tanımlanan lojistik fonksiyondur.

$$\varphi(v) = \frac{1}{1+\exp(-av)} \tag{2.12}$$

burada a sigmoid fonksiyonun eğim parametresidir.

2.2. Ağ Mimarileri

Bir sinir ağının nöronlarının yapılandırılmasında izlenen yol sıkı bir şekilde ağın eğitimi için kullanılan eğitim algoritmasına bağlıdır. Bu yüzden yapılandırılacak sinir ağlarının tasarımında kullanılan eğitim algoritmalarından (kurallardan) bahsedeceğiz. Öğrenme algoritmalarını sonraki bölümde sınıflandıracağız. Bu bölümde ağ mimarilerine (yapılarına) odaklanacağız.

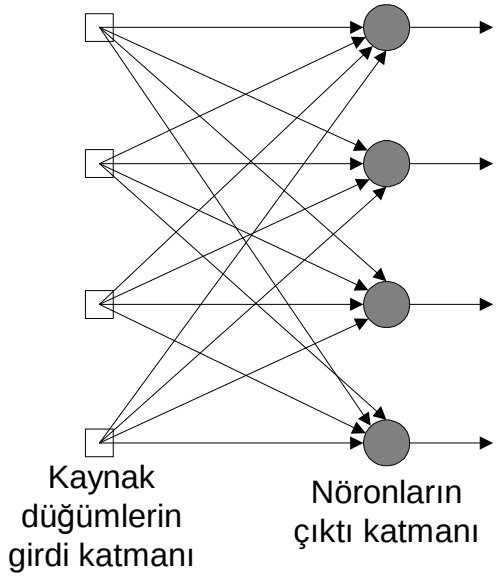
Genel olarak ağ mimarilerini üç ana sınıfta toplayabiliriz:

Tek Katmanlı İleri Beslemeli Ağlar

Katmanlı bir sinir ağında nöronlar katmanlar halinde organize olmuştur. Katmanlı bir ağın en basit şeklinde, nöronların çıktı katmanına (hesaplama düğümleri) yansıyan bir kaynak düğümlerin girdi katmanına sahibiz ama tersi geçerli değildir. Başka bir deyişle, bu ağ ciddi bir ileri beslemeli veya çevrimsel olmayan bir tiptir. Şekil 2.5' de girdi ve çıktı katmanlarında dört düğüm bulunan durum gösterilmiştir. Böyle bir ağ tek katmanlı ağ olarak tanımlanır, buradaki tek katman hesaplama düğümlerinin (nöronlarının) çıktı katmanını ifade eder. Kaynak düğümlerin girdi katmanı sayılmaz çünkü burada herhangi bir hesaplama gerçekleşmez.

Çok Katmanlı İleri Beslemeli Ağlar

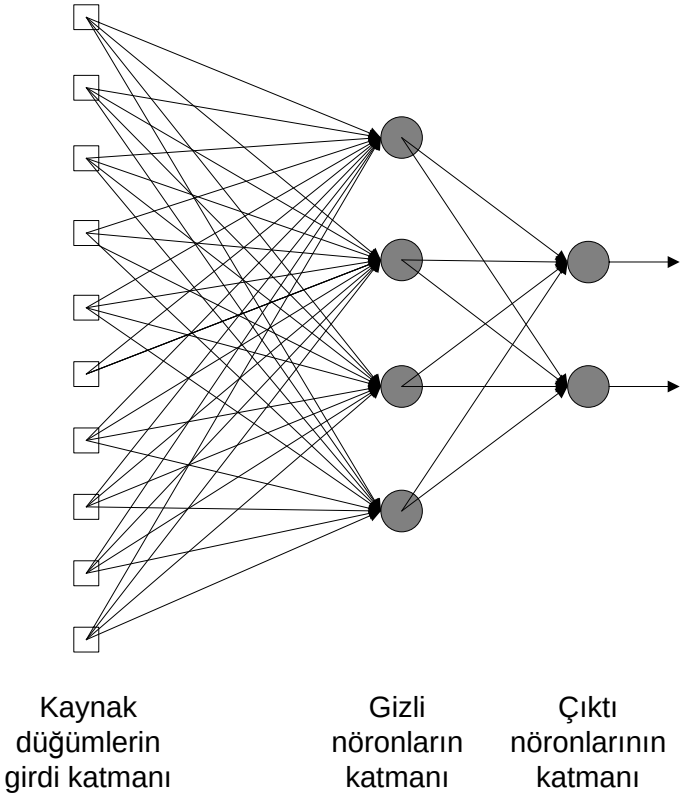
Bir ileri bildirimli sinir ağının ikinci sınıfı hesaplama düğümleri karşılıklı olarak gizli nöronlar veya gizli birimler olarak adlandırılan bir veya daha fazla gizli katmanın varlığıyla diğerlerinden ayrılır. Gizli nöronların fonksiyonu dış girdi ile ağın çıktısı arasında faydalı bir aracılık yapmaktır.



Şekil 2.5. Tek katmanlı ileri beslemeli veya çevrimsel olmayan ağ.

Bir veya daha fazla gizli katman ekleyerek, ağ üst düzey istatistikler çıkarsama yeteneğine sahip olur. Daha açık bir tanımla ağ, lokal bağlılığına rağmen ekstra sinaptik bağlantılar kümesi ve sinirsel interaksiyonların ekstra boyutu sayesinde global bir perspektif kazanır (Churcland ve Sejnowski 1992). Gizli nöronların üst düzey istatistikler çıkarsama yeteneği özellikle girdi katmanının boyutu büyük olduğunda değerlidir.

Ağın girdi katmanındaki kaynak düğümler, ikinci katmandaki (ilk gizli katmandaki) nöronlara uygulanan girdi sinyallerini oluşturan aktivasyon örneklerini (girdi vektörünü) sağlar. İkinci katmanın çıktı sinyalleri üçüncü katmanın girdi sinyalleri olarak kullanılır ve ağın geri kalan katmanları için bu şekilde devam eder. Tipik olarak, ağın her bir katmanındaki nöronlar girdi olarak sadece kendilerinden önce gelen katmanın çıktı sinyallerine sahiptirler. Ağın çıktı (son) katmanındaki nöronların çıktı sinyalleri kümesi, girdi (ilk) katmanındaki kaynak düğümler tarafından sağlanan aktivasyon örneklerine verilen ağın toplam (net) yanıtını oluşturur. Şekil 2.6' daki mimari grafik tek gizli katmanlı durum için çok katmanlı ileri beslemeli sinir ağının görünümünü gösterir. Sadelik için Şekil 2.6' daki ağ 10-4-2 bir ağ olarak tanımlanır çünkü 10 adet kaynak düğüme, 4 gizli nörona ve 2 çıktı nöronuna sahiptir. Başka bir örnek vermek gerekirse m kaynak düğümlü, ilk gizli katmanında h_1 nöron, ikinci gizli katmanında h_2 nöron ve çıktı katmanında q nöron bulunan ileri beslemeli bir ağ $m-h_1-h_2-q$ ağ olarak tanımlanır.



Şekil 2.6. Bir gizli katmanlı ve bir çıktı katmanlı tam bağlantılı ileri beslemeli ağ.

2.3. Öğrenme Süreçleri

Bir sinir ağı için birincil öneme sahip özellik, ağın bulunduğu çevreden öğrenme yeteneği ve öğrenme süresince de performansını geliştirmesidir. Performansdaki gelişme önceden belirlenmiş bir ölçüye uygun olarak zaman içerisinde gerçekleşir. Bir sinir ağı çevresi hakkındaki bilgiyi sinaptik ağırlıklarına ve sapma düzeylerine uygulanan (interaktif) düzeltmeler süreciyle öğrenir. İdeal olarak ağ, öğrenme sürecinin her iterasyonundan sonra çevresi hakkında daha bilgili hale gelir.

Öğrenme kavramını tam anlamıyla tanımlamak için bu konuyla ilgili birçok çalışma vardır. Hatta öğrenme süreci, kesin bir tanımın üzerinde anlaşılması zor olan bir konudur. Örneğin, bir psikolog için öğrenme bir sınıfta gerçekleşen öğrenmeden farklı görülür. Bizim ilgimizin özellikle sinir ağlarındaki öğrenme olduğunu hatırlarsak Mendel ve McClaren (1970)' den alınan bir tanımı kullanacağız. Sinir ağları kapsamında öğrenme aşağıdaki şekilde tanımlanır:

Öğrenme bir sinir ağının serbest parametrelerinin, ağın gömülü olduğu çevre tarafından uyarılması süreci boyunca düzenlendiği bir süreçtir. Öğrenme sürecinin bu tanımı aşağıdaki olaylar dizisini ima eder.

1. Sinir ağı bir çevre tarafından uyarılır.
2. Sinir ağının serbest parametreleri bu uyarılma sonucu değişikliğe uğrar.
3. Sinir ağı çevreye iç yapısında oluşan değişikliklerden dolayı yeni bir şekilde cevap verir.

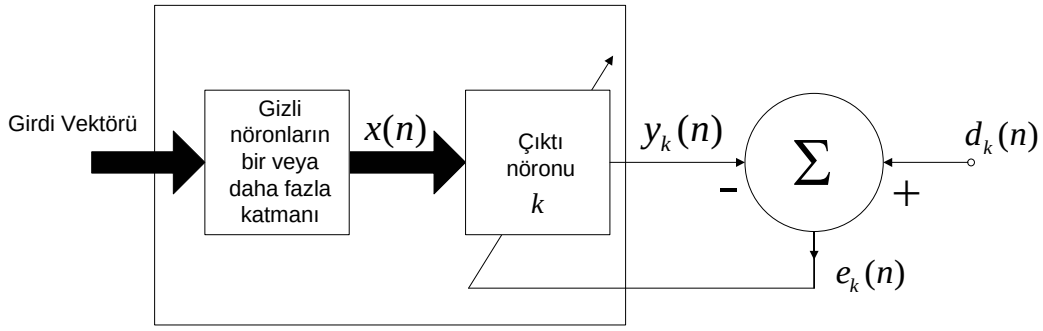
Öğrenme probleminin çözümü için iyi tanımlanmış bir belirli kurallar kümesine öğrenme algoritması denir. Beklenildiği üzere, sinir ağlarının tasarımı için tek bir öğrenme algoritması yoktur. Bunu yerine her biri kendi avantajlarını öneren değişik öğrenme algoritma tipleri olarak sunulan bir araçlar kitine sahibiz. Temel olarak, öğrenme algoritmaları bir nöronun sinaptik bir ağırlığına yapılan düzenlemenin formüle edilişi bakımından birbirinden ayrılır.

2.3.1. Hata Düzeltmeli Öğrenme

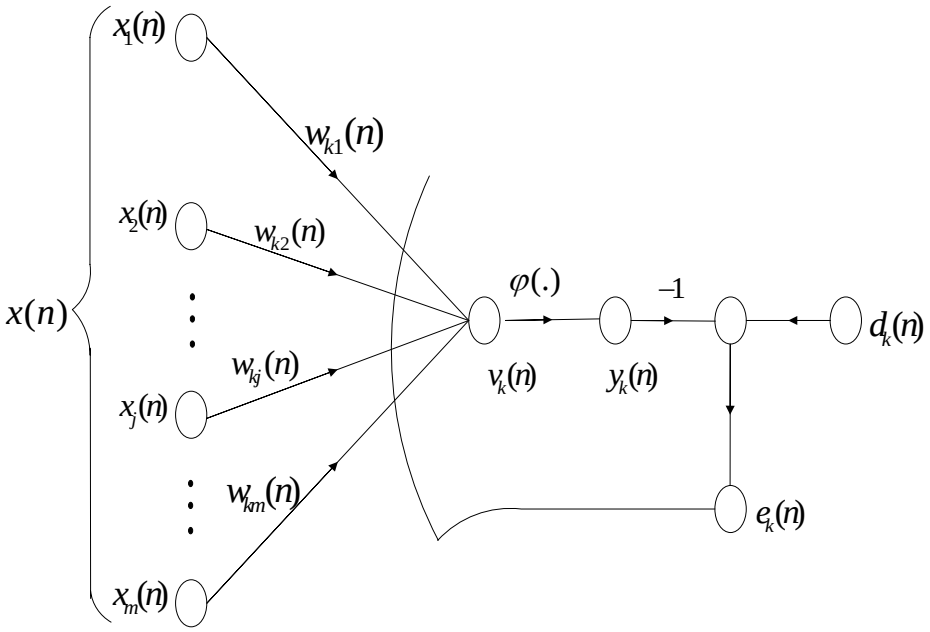
Şekil 2.7a' daki gibi bir k nöronunun ileri beslemeli bir sinir ağının çıktı katmanındaki tek hesaplama düğümünü oluşturduğu basit durumu düşünelim. k nöronu her biri sinir ağının kaynak düğümlerine (örneğin girdi katmanı) uygulanan bir girdi vektörü (uyarıcı) tarafından uyarılan bir veya daha fazla gizli katmanın nöronları tarafından üretilen bir $x(n)$ sinyal vektörüyle uyarılsın. n ifadesi kesikli zamanı gösterir veya daha açık olarak, k nöronunun sinaptik ağırlıklarının düzenlenmesi bir iteratif sürecin zaman adımıyla ilişkilendirilmiştir. k nöronunun çıktı sinyali $y_k(n)$ ile gösterilir. Bu çıktı sinyali sadece sinir ağının çıktısını gösterir ve $d_k(n)$ ile gösterilen arzu edilen bir cevap veya hedef çıktı ile karşılaştırılır. Devamında ise $e_k(n)$ ile gösterilen bir hata sinyali üretilir. Tanımdan aşağıdaki ifade yazılabilir.

$$e_k(n) = d_k(n) - y_k(n) \tag{2.13}$$

Hata sinyali $e_k(n)$, amacı k nöronunun sinaptik ağırlıklarına düzeltici düzenlemeler dizisi uygulamak olan bir kontrol mekanizmasını ortaya çıkarır. Düzeltici düzenlemeler, çıktı sinyali $y_k(n)$' i adım adım arzu edilen cevap $d_k(n)$' e yakın yapmak için tasarlanmıştır. Bu amaç hata sinyali $e_k(n)$ terimleri ile aşağıdaki gibi tanımlanmış bir maliyet fonksiyonunu veya performans indeksini $\varepsilon(n)$' i minimize ederek gerçekleştirilir.



a) Çıktı katmanındaki tek bir nöronu inceleyen bir sinir ağının blok diyagramı



b) Çıktı nöronunun sinyal düzeyindeki grafiği

Şekil 2.7. Hata-düzeltmeli öğrenmenin gösterimi.

$$\varepsilon(n) = \frac{1}{2} e_k^2(n) \tag{2.14}$$

Öyle ki $\varepsilon(n)$, hata enerjisinin anlık değeridir. k nöronunun sinaptik ağırlıklarına yapılan adım adım düzenlemeler, sistem kararlı (durağan) bir duruma ulaşana kadar devam eder (yani sinaptik ağırlıklar temel olarak kararlı hale getirilir). Bu noktada öğrenme süreci durdurulur.

Burada anlatılan öğrenme süreci açıkça hata-düzeltmeli öğrenme olarak tanımlanır. Özellikle $\varepsilon(n)$ maliyet fonksiyonunun minimizasyonu, geliştiricilerinin hatırına genellikle Widrow-Hoff veya Delta kuralı olarak adlandırılan bir öğrenme kuralıyla sonuçlanır (Widrow ve Hoff 1960). $w_{kj}(n)$'in n. zaman adımında girdi vektörü $x(n)$'in $x_j(n)$'ci elemanı tarafından uyarılan k nöronunun w_{kj}' inci sinaptik ağırlığının değerini göstersin. Delta kuralına göre w_{kj} sinaptik ağırlığına n. zaman adımında uygulanan $\Delta w_{kj}(n)$ düzenlemesi aşağıdaki şekilde tanımlanır.

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n) \tag{2.15}$$

burada η öğrenme sürecinde bir adımdan diğer bir adıma geçtiğimizdeki öğrenme oranını belirleyen pozitif bir sabittir. Bu yüzden η' yı öğrenme oranı parametresi olarak adlandırırız. Diğer bir deyişle, Delta kuralı şu şekilde ifade edilebilir:

Bir nöronun sinaptik bir ağırlığına yapılan düzenleme hata sinyaliyle, işlenen sinapsin girdi sinyalinin çarpımıyla orantılıdır. $\Delta w_{kj}(n)$ sinaptik düzenlemesi hesaplandıktan sonra w_{kj} sinaptik ağırlığının güncellenmiş değeri aşağıdaki şekilde belirlenir.

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n) \tag{2.16}$$

Aslında $w_{kj}(n)$ ve $w_{kj}(n+1)$ sinaptik ağırlık w_{kj}' nin sırasıyla eski ve yeni değerleri olarak görülebilir.

Şekil 2.7b k nöronu çevresindeki aktivite üzerine odaklanarak hata-düzeltmeli öğrenme sürecinin bir sinyal-akış grafiğinin gösterimidir. Girdi sinyali x_j ve k nöronunun net çıktısı v_k sırasıyla k nöronunun j. sinapsinin pre-sinaptik ve post-sinaptik sinyalleri olarak bilinir. Şekil 2.7b' den hata-düzeltmeli öğrenmenin bir kapalı döngülü geri beslemeli sistemin bir örneği olduğunu görürüz. Kontrol teorisinden biliyoruz ki böyle bir sistemin kararlılığı sistemin geri besleme döngülerini oluşturan parametreler tarafından belirlenir. Bizim durumumuzda tek bir geri besleme döngüsüne sahibiz ve bu parametrelerden özellikle bizi ilgilendireni öğrenme oranı parametresi η' dır. Bu yüzden gerçekleştirilen iteratif öğrenme sürecinin kararlılığının veya yakınsamasının sağlanması için η' nın dikkatli bir şekilde seçilmesi önemlidir. η' nın seçimi aynı zamanda öğrenme sürecinin doğruluğu ve diğer özellikleri üzerinde de çok büyük bir etkiye sahiptir. Kısaca, pratikte öğrenme oranı parametresi η hata düzeltmeli öğrenmenin performansını belirlemede anahtar bir rol oynar.

2.4. Tek Katmanlı Perceptronlar

Sinir ağlarının geliştiği yıllarda (1943-1958) çeşitli araştırmacılar öncü olan katkıları ile bilinirler.

- McCulloch ve Pitts (1943) hesaplama makineleri olarak sinir ağları fikrini ortaya attıkları için.
- Hebb (1949) kendi kendine organize öğrenme için ilk kuralı önerdiği için.
- Rosenblatt (1958) öğreticili öğrenme için perceptron' u ilk model olarak önerdiği için.

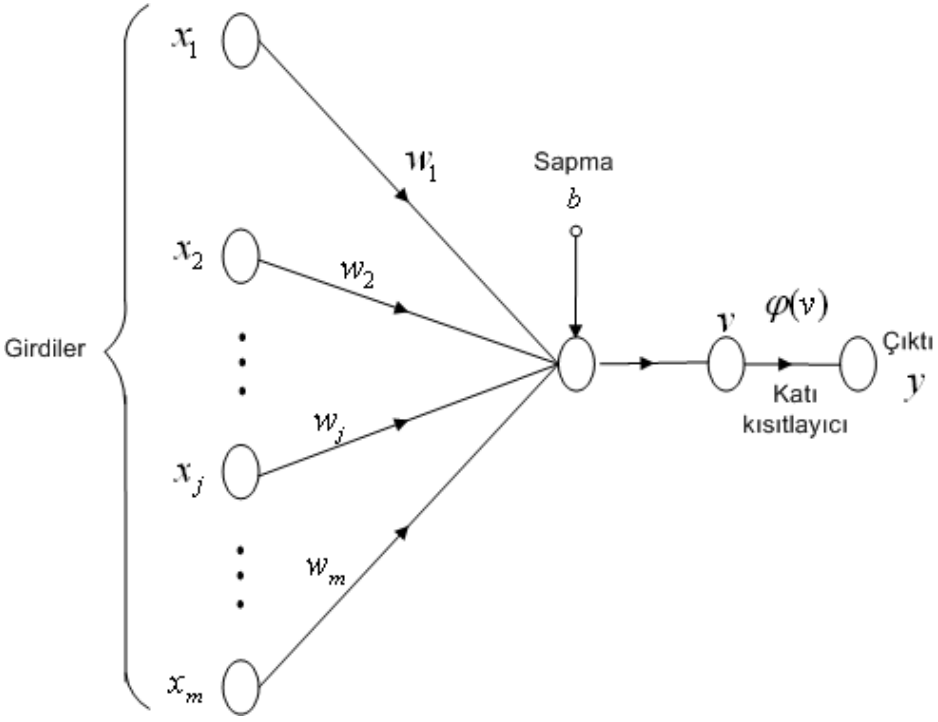
Bu bölümde bizim esas olarak ilgileneceğimiz Rosenblatt' ın perceptronudur. Perceptron, doğrusal olarak ayrılabilen örneklerin sınıflandırılması için kullanılan bir sinir ağının en basit şeklidir (örneğin bir hiperdüzlemin zıt taraflarında bulunan örnekler). Temel olarak düzenlenebilir sinaptik ağırlıkları ve sapması olan tek bir nöron içerir. Bu sinir ağının serbest parametrelerini düzenlemek için kullanılan algoritma ilk defa Rosenblatt' ın (1958, 1962) perceptron beyin modeli için geliştirdiği bir öğrenme yordamı içinde

görünür. Gerçekten Rosenblatt, eğer perceptronu eğitmek için kullanılan örnekler (vektörler) doğrusal olarak ayrılabilen sınıflardan seçilirse perceptron algoritmasının yakınsayacağını ve karar yüzeyini iki sınıf arasındaki bir hiperdüzlem şeklinde konumlandıracağını ispatlamıştır.

Algoritmanın yakınsamasının ispatı perceptron yakınsama teoremi olarak bilinir. Bir nöron etrafında inşa edilmiş perceptron sadece iki sınıflı (hipotezli) örnek sınıflandırma gerçekleştirebilme ile sınırlıdır. Çıktı (hesaplama) katmanını bir nörondan fazla nöron içerecek şekilde genişleterek iki sınıftan daha fazla sınıfın sınıflandırılması gerçekleştirilebilir. Yinede, perceptronun doğru çalışması için sınıfların doğrusal olarak ayrılabilir olması gerekir. Burada önemli nokta bir örnek sınıflandırıcı olarak düşünüldüğünde perceptronun temel teorisinde sadece tek nöron olduğu durum düşünülür. Teorini bir nörondan daha fazla nöronun bulunduğu durum için genişletilmesi sıradan bir konudur.

2.4.1. Perceptron

Önceki bölümlerde anlatıldığı üzere öğrenme sürecinde bahsedilen algoritma bir doğrusal nöron etrafında oluşturulmuştu, perceptron ise doğrusal olmayan bir nöron etrafında oluşturulur ve McCulloch-Pitts modeli olarak adlandırılır. Böyle bir sinirsel model Şekil 2.8’ de gösterildiği gibi bir katı kısıtlayıcı (signum fonksiyonu gerçekleştirilerek) tarafından takip edilen bir doğrusal birleştirici içerir. Sinirsel modelin toplayıcı düğümü, modelin sinapslarına uygulanan girdilerin doğrusal bir kombinasyonunu hesaplar ve aynı zamanda harici uygulanan bir sapmayı birleştirir. Oluşan toplam katı bir kısıtlayıcıya uygulanır. Uygun olarak nöron eğer katı kısıtlayıcının girdisi pozitif ise $+1$, negatif ise -1’ e eşit olan bir çıktı üretir.



Şekil 2.8. Perceptron sinyal akışı grafiği

Şekil 2.8' in sinyal-akış grafik modelinde, perceptronun ağırlıkları $w_1, w_2, ..., w_m$ ile gösterilir. Buna karşılık perceptrona uygulanan girdiler $x_1, x_2, ..., x_m$ ile gösterilir. Harici olarak uygulanan sapma b ile gösterilir. Modelden nöronun katı kısıtlayıcısının girdisini veya net çıktısını aşağıdaki şekilde buluruz.

$$v = \sum_{i=1}^{m} w_i x_i + b \tag{2.17}$$

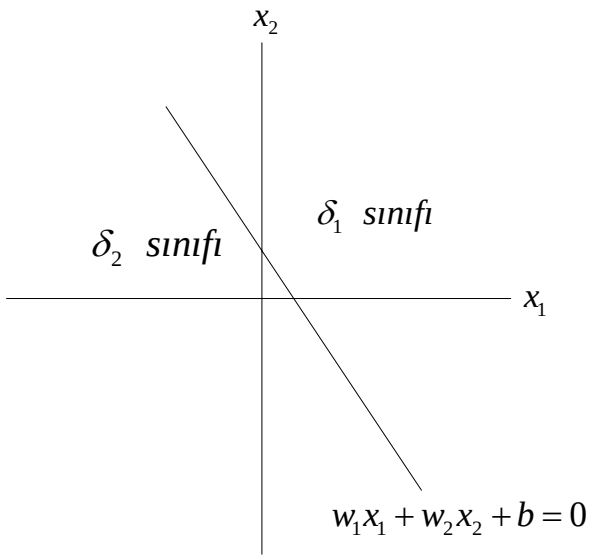
Perceptron' un hedefi harici olarak uygulanan uyarıcı $x_1, x_2, ..., x_m$ kümesini doğru olarak δ_1 ve δ_2 sınıflarından birine uygun olarak sınıflamaktır. Sınıflandırma için karar kuralı $x_1, x_2, ..., x_m$ girdileriyle ifade edilen noktayı eğer perceptronun çıktısı y $+1$ ise δ_1 sınıfına ve -1 ise δ_2 sınıfına atamaktır.

Bir örnek sınıflandırıcının tavrına bir bakış geliştirmek için, m girdi değişkeni $x_1, x_2, ..., x_m$ tarafından gerilen m boyutlu sinyal uzayında karar bölgelerinin bir grafiğini çizmek geleneksel bir durumdur. Perceptronun en basit

halinde aşağıdaki gibi tanımlanan bir hiperdüzlem tarafından ayrılmış iki karar bölgesi vardır.

$$\sum_{i=1}^{m} w_i x_i + b = 0 \qquad (2.18)$$

Bu, x_1 ve x_2 gibi iki girdi değişkeninin olduğu ve karar sınırının düz bir doğru şeklini aldığı durum için Şekil 2.9 da gösterilmiştir. Sınır doğrusunun üstünde yer alan bir (x_1, x_2) noktası δ_1 sınıfına atanır ve sınır doğrusunun altında yer alan bir (x_1, x_2) noktası da δ_2 sınıfına atanır. b sapmasının etkisi ise karar sınırını orijinden uzağa kaydırmaktır.



Şekil 2.9. Bir iki boyutlu, iki-sınıflı patern-sınıflandırma problemi için hiperdüzlemin gösterimi (bu örnekte düz bir çizgi).

2.5. Çok Katmanlı Perceptronlar

Bu bölümde sinir ağlarının önemli bir sınıfı olan çok katmanlı ileri beslemeli ağları inceleyeceğiz. Tipik olarak ağ girdi katmanını oluşturan bir sensör birimleri (kaynak düğümler) kümesini, bir veya daha fazla hesaplama düğümleri gizli katmanları ve hesaplama düğümlerinin bir çıktı katmanını içerir.

Girdi sinyalleri ağ boyunca bir katmandan diğer katmana ileri yönde yayılırlar. Bu sinir ağları önceki bölümdeki tek katmanlı perceptoronun bir genelleştirilmesi olarak sunulan çok katmanlı perceptronlar (ÇKP) olarak tanımlanır.

Çok katmanlı perceptronlar, hata geri-yayılım algoritması olarak bilinen oldukça popüler bir algoritmayla öğreticili bir şekilde eğitilerek bazı zor ve çeşitli problemleri çözmek için başarılı bir şekilde uygulanmaktadır. Bu algoritma hata-düzeltmeli öğrenme kuralına dayanır. Öyle ki, bu algoritma yine aynı derecede popüler olan ve çok sık rastlanan en küçük kareler algoritmasının bir genelleştirilmesi olarak görülebilir. Temel olarak hata geriye yayılımlı öğrenme ağın farklı katmanları boyunca iki geçişi içerir: Bir ileri ve bir geri geçiş. İleri geçişte, bir aktivite örneği (girdi vektörü) ağın sensör düğümlerine uygulanır ve bu girdinin etkisi ağda katmandan katmana yayılır. İleri geçiş sırasında ağın sinaptik ağırlıkları belirlidir. Diğer taraftan geri geçiş sırasında sinaptik ağırlıklarım tümü bir hata-düzeltme kuralı ile uyumlu olarak düzenlenir. Spesifik olarak, bir hata sinyali üretmek için ağın gerçek çıktısı arzu edilen (hedef) bir çıktıdan çıkartılır. Bu hata sinyali sinaptik bağlantıların tersi yönünde ağda geriye doğru yayılır- bu yüzden ismi “hata geriye-yayılım” dır.

Sinaptik ağırlıklar ağın gerçek çıktısını istatistiksel anlamda arzu edilen çıktıya yakın hale getirmek için düzenlenir. Hata geriye-yayılım algoritması literatürde aynı zamanda geriye-yayılım algoritması olarak bilinir. Bundan sonra geriye-yayılım olarak adlandıracağız. Bu algoritmayla gerçekleştirilen öğrenme süreci geriye-yayılımlı öğrenme olarak adlandırılır. Bir çok katmanlı perceptron üç tane ayırt edici özelliğe sahiptir.

1. Ağdaki her nöron modeli bir doğrusal olmayan aktivasyon fonksiyonu içerir. Burada vurgulanması gereken önemli nokta doğrusal olmama’ nın Rosenblatt’ ın perceptron’ undaki katı kısıtlayıcının aksine yumuşak (pürüzsüz) (yani her yerde türevlenebilen) olmasıdır. Bu gerekliliği sağlayan ve sıkça kullanılan bir doğrusal olmama formu lojistik fonksiyonla tanımlanan bir sigmoidsel doğrusal olmama’ dır.

$$y_j = \frac{1}{1+\exp(-v_j)} \tag{2.19}$$

burada v_j, j. nöronun net çıktısı (yani, tüm sinaptik girdilerin ağırlıklandırılmış toplamı artı sapma) ve y_j nöronunun çıktısıdır.

Doğrusal olmayan ilişkilerin varlığı önemlidir çünkü aksi halde ağın girdi-çıktı ilişkisi tek katmanlı bir perceptron durumuna indirgenir. Hatta, lojistik fonksiyonun kullanılması biyolojiksel olarak motive edilmiştir, çünkü lojistik fonksiyon gerçek nöronların kontrol edilmesi veya başa çıkılması zor safhasını gerçekleştirmeye çalışır.

1. Ağ, ağın girdi veya çıktı bölümlerine ait olamayan bir veya daha fazla gizli nöron katmanlarını içerir. Bu gizli nöronlar, girdi örneklerinden (vektörlerinden) gittikçe daha fazla anlamlı özellikler çıkarsayarak ağın karmaşık görevleri öğrenmesine olanak sağlar.

2. Ağ, ağın sinapsları tarafından belirlenen üst düzeyde bir ilişkisellik gösterir. Ağın ilişkiselliğindeki bir değişiklik sinaptik bağlantılar veya onların ağırlıklar popülasyonunda bir değişiklik gerektirir.

İşte bu özelliklerle beraber eğitim süresince tecrübeyle öğrenme yeteneğinin kombinasyonu çok katmanlı perceptron' un hesaplama gücünü türetir. Bu benzer özellikler, buna rağmen aynı zamanda ağın tutumu hakkındaki şu anki bilgi durumumuzdaki yetersizliklerden de sorumludur. Öncelikle, doğrusal olmamanın dağıtılmış bir şekildeki varlığı ve ağın üst düzeydeki ilişkiselliği çok katmanlı bir perceptron' un teorik analizini üstesinden gelinmesi zor bir durum yapmaktadır. İkinci olarak, gizli nöronların kullanımı öğrenme sürecini canlandırılması zor bir durum yapar. Daha açık olarak, öğrenme sürecinin girdi örneğinin hangi özelliklerinin gizli nöronlar tarafından ifade edileceğine karar vermesi gerekir. Bu yüzden öğrenme süreci daha zor bir şekle gelir çünkü olası fonksiyonların aranması daha geniş bir uzayda gerçekleştirilmelidir ve girdi örneklerinin alternatif görüntüleri arasından bir seçim yapılmaktadır (Hinton 1989).

Geriye-yayılım teriminin kullanımı, 1985 yılında yeni ufuklar açan Paralel Dağıtılmış İşleme Rumelhart ve McClelland (1986) adlı kitabın yayımlanmasından sonra popüler olmuş ve gelişmiştir.

Geriye-yayılım algoritmasının gelişimi yapay sinir ağlarında bir dönüm noktasıdır, öyle ki çok katmanlı perceptronların eğitimi için oldukça verimli bir

metot sağlar. Buna rağmen geriye-yayılım algoritmasının tüm çözülebilir problemler için optimal bir çözüm sağladığını iddia edemeyiz ama geriye yayılım algoritması Minsky ve Papert (1969) kitabında gözlenen çok katmanlı makinelerde öğrenme hakkındaki karamsarlıktan bizleri kurtarır.

2.6. Geriye Yayılım Algoritması

n. iterasyondaki (n. eğitim örneğinin ağa sunumu) çıktı nöronunun hata sinyali aşağıdaki şekilde tanımlanır.

$e_j(n) = d_j(n) - y_j(n)$, j nöronu bir çıktı düğümüdür. (2.20)

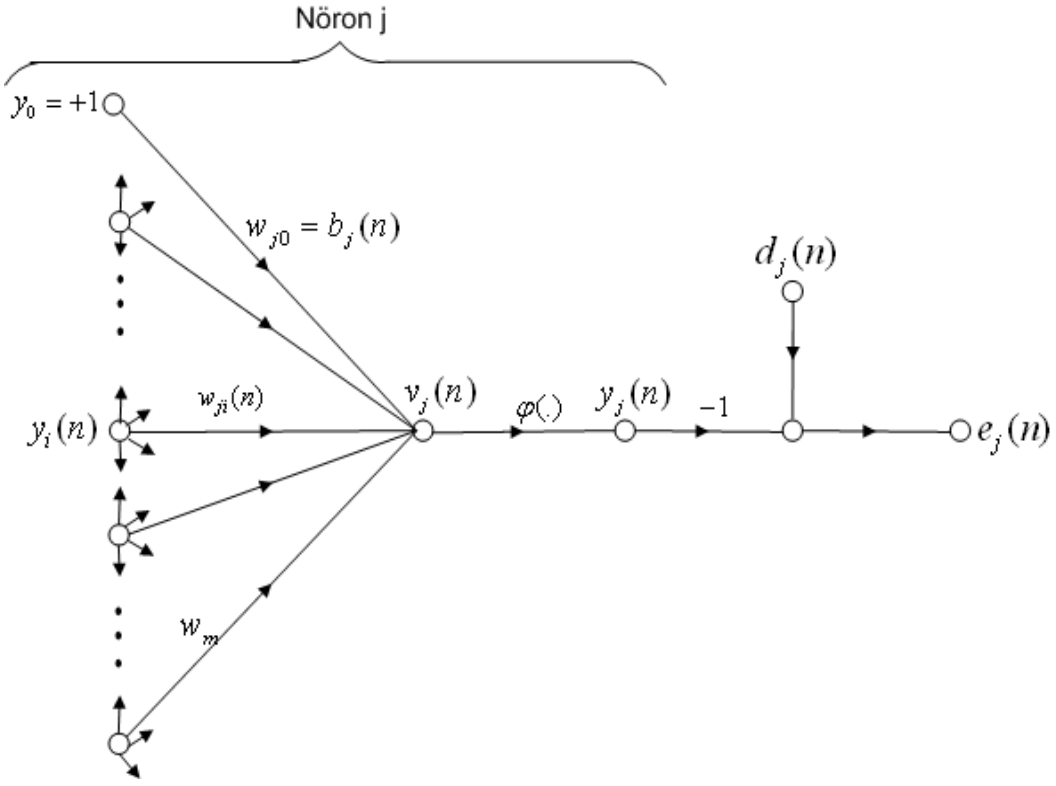
(2.20) j nöronu için hata enerjisinin anlık değeri $\frac{1}{2}e_j^2(n)$ olarak tanımlanır. Karşılık olarak, toplam hata enerjisinin anlık değeri, çıktı katmanındaki tüm nöronlar üzerinden $\frac{1}{2}e_j^2$'lerin toplamı alınarak elde edilir; ki sadece bunlar hata sinyallerinin doğrudan hesaplanabileceği "görünür" nöronlardır. Böylece aşağıdaki ifade yazılabilir.

$$\varepsilon(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \tag{2.21}$$

burada C kümesi ağın çıktı katmanındaki tüm nöronları içerir. Varsayalım ki N eğitim kümesindeki örneklerin toplam sayısını göstersin. Hata kareler ortalaması $\varepsilon(n)$'i tüm n'ler için hesaplayarak ve sonra küme genişliği N'ye göre normalize ederek aşağıda gösterildiği gibi elde edilir.

$$\varepsilon_{ort} = \frac{1}{N}\sum_{n=1}^{N} \varepsilon(n) \tag{2.22}$$

Anlık hata enerjisi $\varepsilon(n)$ ve böylece ortalama hata enerjisi ε_{ort} ağın tüm serbest parametrelerinin (yani sinaptik ağırlıkların ve sapma düzeylerinin) bir fonksiyonudur. Verilen bir eğitim kümesi için, ε_{ort} öğrenme performansının bir ölçümü olarak maliyet fonksiyonunu ifade eder. Öğrenme sürecinin amacı ε_{ort}' yı minimize edecek şekilde ağın serbest parametrelerini düzenlemektir. Bu minimizasyonu gerçekleştirmek için, HKO algoritmasının türetiminde kullanılan mantığa benzer bir tahmin kullanılır. Özel olarak, ağırlıkların bir tur tamamlanana kadar örnekten örneğe güncellendiği basit bir eğitim metodu düşünelim, öyle ki, dikkate alınacak olan tüm eğitim kümesinin bir kez tam olarak ağa sunumudur. Ağırlıklara yapılan düzenlemeler ağa sunulan her örnek için hesaplanan hatayla uyumlu olarak yapılır.



Şekil 2.10. j. çıktı nöronunun ayrıntılarını belirten sinyal-akış grafiği.

Bu yüzden eğitim kümesi üzerinden gerçekleştirilen ayrı ayrı ağırlık değişimlerinin aritmetik ortalaması, eğitim kümesi üzerinde gerçekleştirilen hata fonksiyonunun minimizasyonu üzerine dayanan ağırlık düzenlenmesinden

kaynaklanan gerçek değişimin bir tahminidir. Sol tarafındaki nöronlar katmanı tarafından üretilen fonksiyon sinyalleri kümesiyle beslenen j nöronunu gösteren Şekil 2.10 düşünülürse, nöron j ile ilişkilendirilmiş aktivasyon fonksiyonunun girdisinde üretilen net çıktı $v_j(n)$ aşağıdaki şekilde hesaplanır.

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \tag{2.23}$$

burada m nöron j'ye uygulanan girdilerin toplam sayısıdır (sapma hariç). Sinaptik ağırlık w_{j0} ($y_0 = +1$ sabit girdisine karşılık gelen) j nöronuna uygulanan b_j sapmasına eşittir. Bu sebepten n. iterasyonda j nöronunun çıktısında görünen fonksiyon sinyali $y_j(n)$ aşağıdaki şekilde hesaplanır.

$$y_j(n) = \varphi_j(v_j(n)) \tag{2.24}$$

HKO algoritmasına benzer olarak, geri-besleme algoritması, sinaptik ağırlık $w_{ji}(n)$'e kısmi türev $\partial\varepsilon(n)/\partial w_{ji}(n)$'le orantılı olan bir $\Delta w_{ji}(n)$ düzeltmesi uygular. Analizdeki zincir kuralına göre bu gradyanı aşağıdaki şekilde ifade edebiliriz:

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{2.25}$$

$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}$ kısmi türevi, sinaptik ağırlık w_{ji} için ağırlık uzayındaki aramanın yönünü belirleyen bir duyarlılık faktörünü gösterir.

(2.21) eşitliğinin her iki tarafının $e_j(n)$'e göre türevini alırsak,

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} = e_j(n) \tag{2.26}$$

(2.20) eşitliğinin her iki tarafının $y_j(n)$ ' e göre türevini alırsak,

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{2.27}$$

Sonra, (2.24) eşitliğinin $v_j(n)$ ' e göre türevini alırsak,

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi'(v_j(n)) \tag{2.28}$$

Son olarak (2.23) eşitliğinin $w_{ji}(n)$ ' e göre türevi alınırsa aşağıdaki eşitlik elde edilir.

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{2.29}$$

(2.25)' de (2.26) ve (2.29) eşitlikleri kullanılırsa,

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi'(v_j(n))y_i(n) \tag{2.30}$$

$w_{ji}(n)$ ' e uygulanacak $\Delta w_{ji}(n)$ düzeltmesi delta kuralı ile tanımlanır:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} \tag{2.31}$$

burada η geri besleme algoritmasının öğrenme oranı parametresidir. (2.31) eşitliğindeki eksi işaretinin kullanımı ağırlık uzayındaki gradyan azalanı ifade eder (yani, $\varepsilon(n)$ ' in değerini düşüren bir ağırlık değişimi için bir yön arayan). Karşılık olarak, (2.30) ve (2.31) eşitliklerinin kullanımı

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{2.32}$$

üretir. Burada $\delta_j(n)$ lokal gradyan aşağıdaki şekilde tanımlanır:

$$\begin{aligned} \delta_j(n) &= -\frac{\partial \varepsilon(n)}{\partial v_j(n)} \\ &= -\frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \\ &= e_j(n) \varphi_j'(v_j(n)) \end{aligned} \tag{2.33}$$

Lokal gradyan sinaptik ağırlıklardaki gerekli değişiklikleri işaret eder. (2.33) eşitliğine göre, j nöronunun çıktısı için lokal gradyan $\delta_j(n)$, o nöron için karşılık gelen hata sinyali $e_j(n)$ ve ilgili aktivasyon fonksiyonunun türevinin $\varphi_j'(v_j(n))$ çarpımına eşittir.

(2.32) ve (2.33) eşitliklerinden, ağırlık düzenlemesi $\Delta w_{ji}(n)$' nin hesaplanmasında kullanılan anahtar bir faktörün j nöronunun çıktısındaki hata sinyali $e_j(n)$ olduğuna dikkat edelim. Bu anlamda j nöronunun ağda nerede bulunduğuna bağlı olarak iki ayrı durum belirleyebiliriz.

1. durumda j nöronu bir çıktı düğümüdür. Bu durumun ele alınması kolaydır çünkü ağın her bir çıktı düğümü kendi arzu edilen cevabıyla beslenir, bu da ilgili hata sinyalinin hesaplanmasını kolay hale getirir. Durum 2' de ise, j nöronu gizli bir düğümdür. Gizli nöronlar doğrudan erişilebilir olmamasına rağmen ağın çıktısında yapılmış herhangi bir hatadaki sorumluluğu paylaşırlar. Buradaki esas soru, gizli nöronların sorumluluk paylaşımları için nasıl cezalandırılacağını veya ödüllendirileceğini bilmektir. Bu problem kredi-atama problemidir ve ağ boyunca hata sinyallerini geri yayarak zarif bir şekilde çözülür.

Durum 1 - j nöronu bir çıktı düğümüdür.

j nöronu ağın çıktı katmanında yer aldığında, kendi arzu edilen cevabıyla beslenir. Bu nöronla ilgili hata sinyali $e_j(n)$' i hesaplamak için (2.20) eşitliğini

kullanabiliriz (Şekil 2.10). $e_j(n)$ belirlendikten sonra lokal gradyanı $\delta_j(n)$' i (2.33) eşitliğini kullanarak hesaplamak çok açık bir yoldur.

Durum 2 - *j* nöronu bir gizli düğümüdür.

j nöronu ağın gizli katmanında yer aldığında, bu nöron için belirlenmiş arzu edilen bir cevap yoktur. Bu nedenle, gizli bir nöron için hata sinyali, o gizli nöronun doğrudan bağlı olduğu tüm nöronların hata sinyalleri anlamında yinelemeli olarak belirlenmek zorundadır; bu da geriye yayılım algoritmasının geliştirilmesinin karmaşıklaştığı yerdir. *j* nöronunun ağın gizli bir düğümü olduğu Şekil 2.11' de ifade edilen durumu düşünelim.

(2.33) eşitliğine göre lokal gradyan $\delta_j(n)$ gizli *j* nöronu için yeniden aşağıdaki şekilde tanımlayabiliriz.

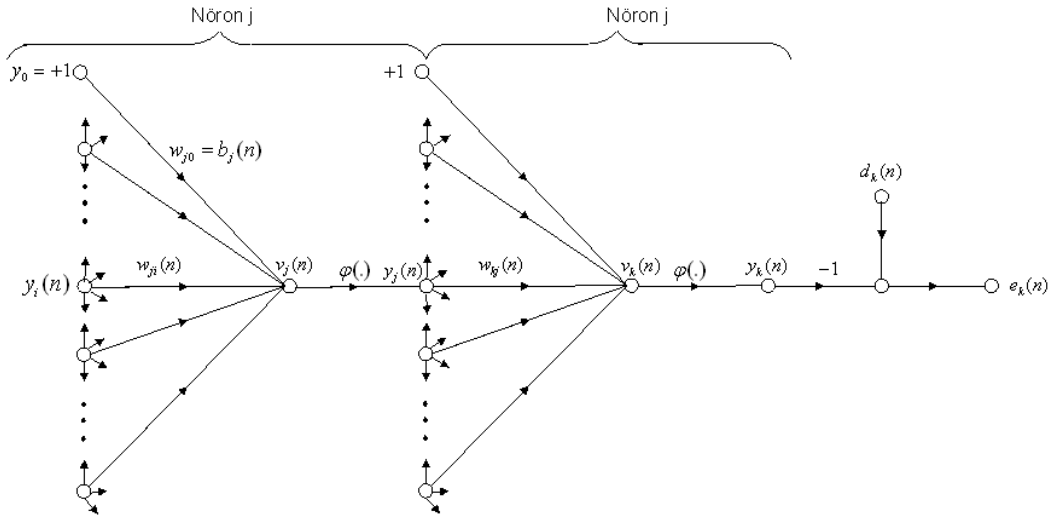
$$\begin{aligned}\delta_j(n) &= -\frac{\partial \varepsilon(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} \\ &= -\frac{\partial \varepsilon(n)}{\partial y_j(n)}\varphi_j'(v_j(n)), \quad j \text{ nöronu gizli}\end{aligned} \tag{2.34}$$

buradaki ikinci satırda (2.28) eşitliğini kullandık. $\frac{\partial \varepsilon(n)}{\partial y_j(n)}$ kısmi türevini hesaplamak için, aşağıdaki şekilde devam ederiz. Şekil 2.11' den şu görülür:

$$\varepsilon(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n), \quad k \text{ nöronu bir çıktı düğümüdür} \tag{2.35}$$

bu da *j* ' nin yerine *k* indeksinin kullanıldığı (2.21) eşitliğidir. Bunu yapmaktaki amacımız, durum 2' deki gizli bir nöronu belirten *j* indeksinin kullanımından doğacak karmaşıklıktan kurtulmaktır. (2.35) eşitliğinin fonksiyon sinyali $y_j(n)$' e göre türevini alırsak,

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \tag{2.36}$$



Şekil 2.11. Gizli nöron j' ye bağlı k çıktı nöronunun ayrıntılarını belirten sinyal akış grafiği.

Daha sonra, $\frac{\partial e_k(n)}{\partial y_j(n)}$ kısmi türevi için zincir kuralını kullanırız ve (2.36) eşitliğini denk olarak aşağıdaki gibi yeniden yazabiliriz.

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{2.37}$$

Bununla beraber, Şekil 2.11' den şunu belirtelim:

$$\begin{aligned} e_k(n) &= d_k(n) - y_k(n) \\ &= d_k(n) - \varphi(v_k(n)), \quad k \text{ nöronu bir çıktı düğümüdür} \end{aligned} \tag{2.38}$$

Böylece $$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)) \tag{2.39}$$

Aynı zamanda Şekil 2.11' ten k nöronunun net çıktısı,

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n) \tag{2.40}$$

burada m, k nöronuna uygulanan girdilerin toplam sayısıdır (sapma hariç). Tekrar burada, sinaptik ağırlık $w_{k0}(n)$, k nöronuna uygulanan sapma $b_k(n)$' e eşittir ve karşılık gelen girdi +1 değerinde sabitlenmiştir. (2.40) denkleminin $y_j(n)$' e göre türevini alırsak,

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{2.41}$$

(2.37)' de (2.39) ve (2.40) denklemlerini kullanırsak arzu ettiğimiz türevi elde edebiliriz:

$$\begin{aligned} \frac{\partial \varepsilon(n)}{\partial y_j(n)} &= -\sum_k e_k(n)\varphi_k'(v_k(n))w_{kj}(n) \\ &= \sum_k \delta_k(n) w_{kj}(n) \end{aligned} \tag{2.42}$$

burada ikinci satırda, (2.33) denkleminde j indeksi k indeksi ile değiştirilmiş $\delta_k(n)$ lokal gradyan tanımını kullandık.

Son olarak, (2.42) denklemini (2.34)' de kullanarak, lokal gradyan $\delta_j(n)$ için geriye yayılım formülünü elde ederiz:

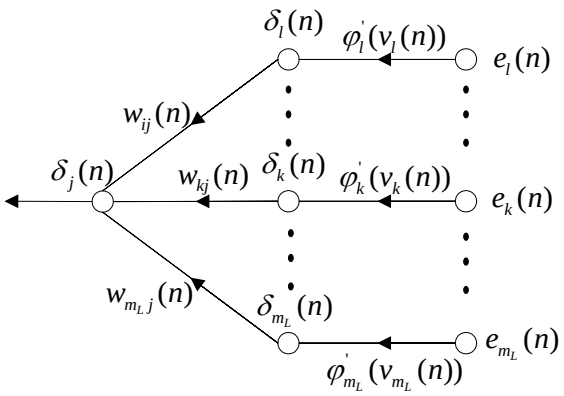
$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n) w_{kj}(n), \quad j \textit{ nöronu gizlidir}. \tag{2.43}$$

Şekil 2.12 (2.43) denkleminin sinyal akış grafiğini gösterir, çıkıt katmanının m_L nöron içerdiğini varsayarak. (2.43) denklemindeki $\delta_j(n)$ lokal gradyanın hesaplanmasında kullanılan $\varphi_j'(v_j(n))$ faktörü tamamen j gizli nöronunun ilgili aktivasyon fonksiyonuna bağlıdır. Bu hesaplamadaki geriye kalan faktör, kısaca k üzerinden toplam, iki terimler kümesine dayanır. İlk terimler kümesi, $\delta_k(n)$, j gizli nöronunun hemen sağındaki katmanda yer alan ve direkt j nöronuna bağlı

tüm nöronlar için $e_k(n)$ hata sinyallerinin bilgisini gerektirir (Şekil 2.11). İkinci terimler kümesi, $w_{kj}(n)$, bu ilişkilerle ilgili sinaptik ağırlıklardan oluşur.

Şimdi geriye yayılım algoritması için türettiğimiz ilişkileri özetleyelim. İlk olarak, i nöronunu j nöronuna bağlayan sinaptik ağırlığa uygulanan $\Delta w_{ji}(n)$ düzeltmesi delta kuralıyla tanımlanır:

$$\begin{pmatrix} Ağırlık \\ düzeltmesi \\ \Delta w_{ji(n)} \end{pmatrix} = \begin{pmatrix} öğrenme\text{-}oranı \\ parametresi \\ \eta \end{pmatrix} . \begin{pmatrix} lokal \\ gradyan \\ \delta_j(n) \end{pmatrix} . \begin{pmatrix} j\ nöronunun \\ girdi\ sinyali \\ y_i(n) \end{pmatrix} \qquad (2.44)$$



Şekil 2.12. Hata sinyallerinin geri yayılımıyla ilişkili olan eşlenik sistemin bir bölümünün sinyal akış grafiği.

İkinci olarak, $\delta_j(n)$ lokal gradyanı j nöronunun bir çıktı düğümü veya gizli bir düğüm olmasına bağlıdır.

1. Eğer j nöronu bir çıktı düğümü ise $\delta_j(n)$, her ikisi de j nöronuyla ilgili olan $\varphi_j'(v_j(n))$ türeviyle $e_j(n)$ hata sinyalinin çarpımına eşittir; bak denk. (2.33).

2. Eğer j nöronu bir gizli düğüm ise $\delta_j(n)$ $\varphi_j'(v_j(n))$ ilgili türevi ile sonraki gizli veya çıktı katmandaki j nöronuna bağlı nöronlar için hesaplanan δ'ların ağırlık toplamının çarpımına eşittir; denk. (2.43).

2.7. Hesaplamanın İki Geçişi

Geriye yayılım algoritmasının uygulanmasında, hesaplanan iki ayrı geçişi birbirinden ayrılır. İlk geçiş, ileri geçiş olarak bilinir ve ikinci ise geri geçiş olarak bilinir.

İleri geçişte sinaptik ağırlıklar ağ boyunca düzenlenmemiş olarak kalır ve ağın fonksiyon sinyalleri nörondan nörona temelinde hesaplanır. j nöronunun çıktısında ortaya çıkan fonksiyon sinyali aşağıdaki şekilde hesaplanır.

$$y_j(n) = \varphi(v_j(n)) \tag{2.45}$$

$v_j(n)$, j nöronunun net çıktısıdır ve şu şekilde tanımlanır:

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \tag{2.46}$$

m, j nöronuna uygulanan girdilerin toplam sayısıdır (sapma hariç) ve $y_i(n)$, j nöronunun girdi sinyalidir veya denk olarak, i nöronunu çıktısında ortaya çıkan fonksiyon sinyalidir.

Eğer j nöronu ağın ilk gizli katmanındaysa, $m = m_0$' dır ve (2.47) eşitliği için i indeksi ağın i. girdi terminalini gösterir.

$$y_i(n) = x_i(n) \tag{2.47}$$

Burada $x_i(n)$ girdi vektörünün i. elemanıdır. Diğer yandan, eğer j nöronu ağın çıktı katmanında ise, $m = m_L$' dir ve (2.48) eşitliğinde j indeksi ağın j. çıktı terminalini gösterir.

$$y_j(n) = o_j(n) \tag{2.48}$$

$o_j(n)$ çıktı vektörünün j. elemanıdır. Bu çıktı, j. çıktı nöronunun hata sinyali $e_j(n)$' i elde etmek için arzu edilen cevap $d_j(n)$ ile karşılaştırılır. Böylece hesaplamanın ileri aşaması ilk gizli katmanda girdi vektörü sunularak başlar ve çıktı katmanında her nöron için hata sinyali hesaplanarak sona erer.

Diğer taraftan geri geçiş çıktı katmanında hata sinyallerini ağ boyunca katmandan katmana sola doğru geçirerek ve her nöron için δ' yı (yani lokal gradyanı) özyinelemeli hesaplayarak başlar. Bu özyinelemeli süreç ağın sinaptik ağırlıklarının (2.44) denklemiyle uyumlu olarak değişiklikler geçirmesine izin verir. Çıktı katmanında yer alan bir nöron için δ basit olarak, nöronun hata sinyaliyle doğrusal olmayan transfer fonksiyonunun birinci derece türevinin çarpımına eşittir. Böylece (2.44) denklemini, çıktı katmanını besleyen tüm bağlantıların ağırlıklarındaki değişimleri hesaplamak için kullanırız. Çıktı katmanındaki nöronlar için δ ' lar verilirse, (2.43) denklemini sondan bir önceki katmandaki tüm nöronlar için δ' ları ve böylece onu besleyen tüm bağlantıların ağırlıklardaki değişimi hesaplamak için kullanırız. Özyinelemeli hesaplama, katmandan katmana ağdaki tüm sinaptik ağırlıklara değişiklikleri yayarak devam eder.

3. SAYISAL OPTİMİZASYONDA GRADYAN YÖNTEMLERİ

Bölüm 2' de sinir ağlarında öğrenme problemi, bir hata fonksiyonu ε' nun minimizasyonu anlamında formüle edilmişti. Bu hata, $w_1, w_2, ..., w_n$ bileşenleriyle tek bir n-boyutlu ağırlık vektöründe uygun olarak gruplayabileceğimiz ağdaki uyarlanabilir parametrelerin (ağırlıklar ve sapmalar) bir fonksiyonuydu.

İkinci bölümde çok katmanlı bir perceptron için hata fonksiyonunun ağ parametrelerine göre türevlerinin geriye yayılım algoritması kullanılarak verimli bir şekilde elde edilebileceği gösterildi. Geniş ölçekli uygulamalar için pratik kullanımda yeterince hızlı olan ağ eğitim algoritmalarının bulunmasında böyle bir gradyan bilgisinin kullanımının esas öneme sahip olduğu görülebilir.

Çok değişkenli sürekli türevlenebilen fonksiyonların minimizasyon problemi yaygın olarak çalışılmış problemlerdendir ve bu probleme olan geleneksel yaklaşımlar sinir ağlarının eğitimine direkt uygulanabilirdir. Bu bölümde pratik algoritmaların en önemlilerinden bazılarını ele alacağız. İlk olarak bunların en temeli ve basit formu olan gradyan düşümü ile başlayacağız ve sonra eşlenik gradyanlar konseptine dayanan ve bu konseptin oldukça yakın bir zamandaki varyantı olan ölçeklendirilmiş eşlenik gradyanı (scaled conjugate gradient) da içeren geleneksel optimizasyon algoritmalarının önemli bir sınıfını inceleyeceğiz.

Burada anlatacağımız algoritmalar geniş bir uygulama aralığında iyi performans gösterdiği bilinen algoritmalardır. Buna rağmen, farklı algoritmalar farklı problemlerde iyi performans gösterebilir ve bu yüzden bir tek genel optimizasyon algoritması önermek olası değildir. Bunun yerine, algoritmaların karşılaştırmalı avantajlarını ve kısıtlamalarını belirteceğiz.

Bazı bilinmeyen ağırlık (parametre) vektörü w' nin sürekli türevlenebilen fonksiyonu olan bir maliyet fonksiyonu $\varepsilon(w)$' yi düşünelim. $\varepsilon(w)$ fonksiyonu, w' ye karşı gelen bir gerçel sayı eşleştirir. $\varepsilon(w)$, algoritmanın optimum şekilde çalışması için ağırlık (parametre) vektörü w' nin nasıl seçileceğinin bir ölçüsüdür. Aşağıdaki koşulu sağlayan bir optimal çözüm w^* bulmak istiyoruz.

$$\varepsilon(w^*) \leq \varepsilon(w)$$

Yani, aşağıdaki şekilde ifade edilen bir sınırsız optimizasyon problemini çözmek istiyoruz:

ε -pürüzsüz, sürekli türevlenebilen fonksiyon olduğunda, $\varepsilon(w)$ *maliyet fonksiyonunu, ağırlık vektörü* w *' ye göre minimize et;*

$$\min_{w \in R^n} \varepsilon(w).$$

Optimallik için yeterli koşul:

$$\nabla \varepsilon(w^*) = 0$$

olmasıdır ve burada ∇ ile gradyan operatörü gösterilir:

$$\nabla = \left[\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, ..., \frac{\partial}{\partial w_n} \right]^T$$

ve $\nabla \varepsilon(w)$ ise maliyet fonksiyonunun gradyan vektörüdür.

$$\nabla \varepsilon(w) = \left[\frac{\partial \varepsilon}{\partial w_1}, \frac{\partial \varepsilon}{\partial w_2}, ..., \frac{\partial \varepsilon}{\partial w_n} \right]$$

3.1. Gradyan Düşümü

Ağ eğitim algoritmalarının en basitlerinden biri bazen dik iniş (steepest descent) olarak bilinen gradyan düşümüdür. Dik iniş metodunda, ağırlık vektörü w' ye uygulanan ardışık düzeltmeler dik-iniş yönündedir, yani gradyan vektörü $\nabla \varepsilon(w)$' ye zıt yöndedir. Anlatımdaki uyumluluk için aşağıdaki eşitlik yazılır.

$$g = \nabla \varepsilon(w) \tag{3.1}$$

Uygun olarak dik-iniş algoritması aşağıdaki şekilde ifade edilir.

$$w_{k+1} = w_k - \eta g_k \tag{3.2}$$

burada η adım büyüklüğü veya öğrenme-oranı parametresi olarak bilinen pozitif bir sabittir ve g_k ise hata fonksiyonunun w_k noktasında hesaplanan gradyan vektörüdür. k. adımdan $k+1$' inci adıma geçildiğinde algoritma aşağıdaki düzeltmeyi uygular:

$$\Delta w_k = w_{k+1} - w_k = -\eta g_k \tag{3.3}$$

(3.3) eşitliği aslında bölüm 2' de anlatılan hata-düzeltme kuralının düzgün bir ifadesidir.

Dik iniş algoritmasının iteratif olarak azalma sağladığını göstermek için, birinci düzey Taylor serisini kullanılarak, $\varepsilon(w_{k+1})$, w_k etrafında tahmin edilir:

küçük η için $\varepsilon(w_{k+1}) \simeq \varepsilon(w_k) + g_k^T \Delta w_k$ sağlanır.

Bu tahmin ilişkisinde (3.3) eşitliğini kullanırsak:

$$\varepsilon(w_{k+1}) \simeq \varepsilon(w_k) - \eta g_k^T g_k \simeq \varepsilon(w_k) - \eta \left\| g_k \right\|^2 \tag{3.4}$$

alırız. (3.4) pozitif bir öğrenme oranı parametresi için algoritmanın bir iterasyondan sonrakine ilerlediğinde maliyet fonksiyonunun azalacağını gösterir. Buradaki düşünce tahminidir, öyle ki bu son sonuç sadece yeterince küçük öğrenme oranları için doğrudur.

Dik-iniş metodu optimal çözüm w^*' a yavaş yakınsar. Öğrenme oranı parametresi η yakınsama tavrı üzerinde önemli bir etkiye sahiptir:

- η küçük olduğunda algoritmanın geçici yanıtı aşırı sönümlenmiştir, öyle ki w_k' nın izlediği yol w düzleminde pürüzsüz bir yoldur.

- η büyük olduğunda algoritmanın geçici yanıtı az sönümlenmiştir, öyle ki w_k' nın izlediği yol bir zig zag (dalgalı) bir yoldur.
- η belirli bir kritik değeri aştığında, algoritma kararsız hale gelir (yani ıraksar).

3.2. Momentum

Oldukça farklı özdeğerlere sahip problemlerin üstesinden gelebilmek için basit bir teknik, gradyan düşümü formülüne bir momentum terimi eklemektir (Plaut ve ark. 1986). Bu ağırlık uzayı boyunca gerçekleşen harekete eylemsizlik katar ve dalgalanmayı hafifletir. Düzenlenmiş gradyan düşümü formülü aşağıdaki şekilde verilir.

$$\Delta w_k = -\eta \nabla \varepsilon(w_k) + \mu \Delta w_{k-1} \tag{3.5}$$

burada μ momentum parametresi olarak tanımlanır.

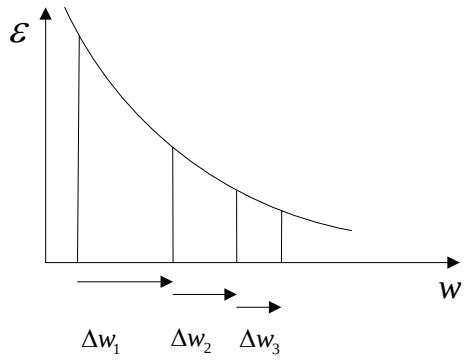
Momentum teriminin etkisini anlamak için, öncelikle Şekil 3.1' de gösterildiği gibi hata yüzeyinin oldukça düşük bir eğime sahip olduğu bir ağırlık uzayı bölgesi boyunca hareketi düşünelim. Eğer gradyanın değişmediğini tahminini yaparsak, o zaman (3.5)' i uzun bir ağırlık güncellemeleri serisine iteratif olarak uygulayabiliriz ve sonra oluşan geometrik serileri

$$\Delta w = -\eta \nabla \varepsilon \{1 + \mu + \mu^2 + ...\} = -\frac{\eta}{1-\mu} \nabla \varepsilon \tag{3.6}$$

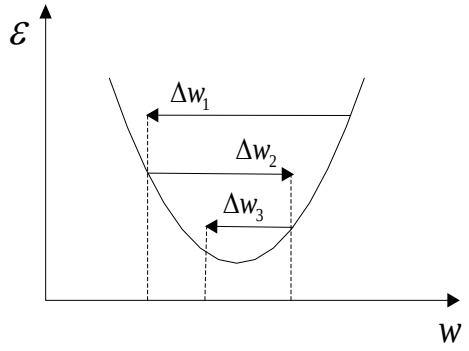
şeklinde toplayabiliriz ve momentum teriminin verimli öğrenme oranını η' dan $\eta/(1-\mu)$' ye yükselttiği sonucunu görürüz.

Tersi durumda, Şekil 3.2' de gösterildiği gibi gradyan düşümünün dalgalı olduğu bir yüksek eğrisel alanda, momentum teriminden gelen başarılı katkılar yok olma eğilimi içindedir ve verimli öğrenme oranı η' ya yakın olacaktır. Böylece, momentum terimi ıraksak dalgalanmalara yol açmadan minimuma doğru

hızlı yakınsama sağlayabilir. 3.6' dan μ' nün $0 \le \mu \le 1$ aralığında olması gerektiği görülür.



Şekil 3.1. Sabit bir öğrenme oranı parametresiyle, düşük eğimli bir yüzeyde gradyan düşümü ardışık küçük adımlara yol açar (doğrusal yakınsama). Böyle bir durumda, momentum teriminin etkisi, verimli öğrenme oranı parametresindeki bir artışa benzerdir.



Şekil 3.2. Gradyan düşümünün ardışık adımlarının dalgalı olduğu bir durum için, momentum terimi verimli öğrenme oranı parametresinin değeri üzerinde çok az etkiye sahiptir.

Momentum teriminin eklenmesi gradyan düşümünün performansında önemli bir gelişmeye yol açar. Yine de, algoritma hala verimsizdir. Momentum teriminin eklenmesi, öğrenme oranı parametresi η' ya ek olarak değerinin seçilmesi gereken ikinci bir parametre ortaya çıkarır.

3.3. Doğru Arama

Bu bölümde anlatılan algoritmalar ağırlık uzayında alınan bir adımlar sürecini içerir. Bu adımların her birini iki aşamada düşünmek uygun olur. Öncelikle hareket edeceğimiz yöne karar vermemiz gerekir ve ikinci olarak bu yönde ne kadar ilerleyeceğimize karar vermemiz gerekir. Basit gradyan düşümünde her adımın yönü hata fonksiyonunun lokal negatif gradyanıyla verilir ve adım büyüklüğü keyfi bir öğrenme oranı parametresiyle belirlenir. Negatif gradyan yönü boyunca hareket ederek hatanın minimum olduğu noktayı bulmanın daha iyi bir yordam olacağı beklenebilir. Daha genel olarak ağırlık uzayında belirli bir arama yönü düşünebiliriz ve sonra bu yön boyunca hata fonksiyonunun minimumunu buluruz. Bu yordam doğru arama olarak bilinir ve gradyan düşümünden oldukça daha güçlü olan çeşitli algoritmalar için temel oluşturur. Doğru aramalar pratikte aşağıdaki şekilde gerçekleşir.

Belirli bir algoritmanın k. adımındaki ağırlık vektörünün w_k olduğunu düşünelim ve ağırlık uzayı boyunca özel bir arama yönü p_k' yı düşünmek istiyoruz. O zaman arama yönündeki minimum ağırlık vektörünün yeni değerini verir:

$$w_{k+1} = w_k + \alpha_k p_k \tag{3.7}$$

burada α_k parametresi aşağıda (3.8) eşitliği ile verilen tek değişkenli $\varepsilon(\alpha)$ fonksiyonunu minimizasyonu probleminin çözümü olarak seçilir.

$$\varepsilon(\alpha) = \varepsilon(w_k + \alpha p_k) \tag{3.8}$$

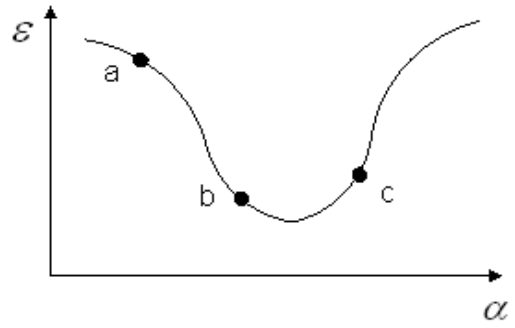
Bu bize, arama yönünü seçtiğimizde adım uzunluğunu belirlemek için otomatik bir yordam verir.

Doğru arama bir-boyutlu minimizasyon problemini ifade eder. Basit bir yaklaşım, hata fonksiyonunu her yeni pozisyonda hesaplayarak ve hata artmaya başladığında durarak arama yönü boyunca ufak adımlarla ilerlemektir (Hash ve Salas 1988). Buna rağmen daha verimli yaklaşımlar bulmak olasıdır (Press ve ark.

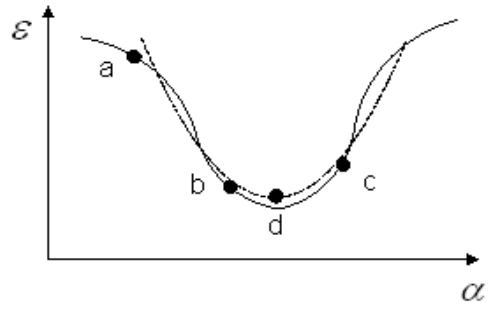
1992). Öncelikle bir doğru arama gerçekleştirirken gradyan bilgisini kullanma konusu düşünülsün. n-boyutlu ağırlık uzayında hata fonksiyonu ε' nun minimumunu arama genel problemi için gradyan bilgisini kullanmanın önemli bir avantaj olduğu genel olarak söylenebilir. Ama doğru aramanın alt problemi için durum biraz farklıdır. Bu bir boyutlu bir minimizasyon problemi olduğu için hem hata fonksiyonunu değeri hem de hata fonksiyonunun gradyanının değerinin her biri sadece tek parça bilgiyi ifade eder. Bir hata fonksiyonu hesaplaması bir ileri yayılım gerektirir ve $\sim 2Nn$ işlem gerektirir, burada N veri kümesindeki örnek sayısıdır. Bir hata fonksiyonu gradyanı hesaplaması bir ileri yayılım, bir geriye yayılım ve türevleri oluşturmak için bir çarpımlar kümesi gerektirir. Bu yüzden $\sim 5Nn$ işlem gerektirir. Bununla birlikte hata fonksiyonunun kendisinin de hesaplanmasına olanak verir. Sonuç olarak, doğru arama eğer sadece hata fonksiyonunun hesaplamalarını kullanırsa biraz daha verimli olur. Doğru arama belirli avantajlara sahip olmakla beraber, sonraki bölümlerde değinilecek bir takım problemleri de beraberinde getirir.

Her doğru arama iki aşamada gerçekleşir. İlk aşama Şekil 3.3' de gösterildiği gibi $\varepsilon(a) > \varepsilon(b)$ ve $\varepsilon(c) > \varepsilon(b)$ olacak şekilde arama yönü boyunca üç nokta $a < b < c$ bularak minimumu sıkıştırmaktır.

Hata fonksiyonu sürekli olduğu için (a, c) aralığında bir yerde bir minimum olduğu garantilenir (Press ve ark. 1992). İkinci aşama minimumun kendisini konumlandırmaktır. Hata fonksiyonu pürüzsüz ve sürekli olduğu için, bu parabolik interpolasyonla gerçekleştirilebilir. Bu üç ardışık noktada hesaplanan hata fonksiyonuna bir kuadratik polinom uydurmayı ve bu parabolün minimumuna hareket etmeyi içerir (Şekil 3.4). Bu süreç, hata fonksiyonunu yeni bir noktada hesaplayarak ve sonra bu nokta ve önceki iki noktaya yeni bir parabol uydurarak tekrarlanabilir. Pratikte çok güçlü Brent's algoritmasına ulaşan çeşitli iyileştirmeler de vardır (Brent 1973). Doğru arama algoritmaları ve durdurma kriterleri Luenberger (1984)' de ayrıntılı şekilde incelenmiştir.



Şekil 3.3. $\varepsilon(a) > \varepsilon(b)$ ve $\varepsilon(c) > \varepsilon(b)$ olacak şekilde üç nokta $a < b < c$.



Şekil 3.4. Doğru-arama minimizasyonunu gerçekleştirmek için kullanılan parabolik interpolasyon sürecinin bir gösterimi.

3.4. Eşlenik Gradyan Metodu

Eşlenik gradyan (Conjugate Gradient Method – CGM) metodu karmaşık doğrusal denklem sistemlerinin çözümü için en uygun tekniklerden birisidir ve doğrusal olmayan optimizasyon problemlerini çözmek için uyarlanabilir. Bu temel yaklaşımın sırasıyla doğrusal ve doğrusal olmayan eşlenik gradyan metotlar olarak tanımladığımız iki varyantının, bu bölümde anlatılacak dikkate değer özellikleri vardır.

Doğrusal eşlenik gradyan metodu pozitif tanımlı katsayı matrisli doğrusal sistemleri çözmek için bir iteratif metot olarak 1950' lerde Hestenes ve Stiefel tarafından önerilmiştir. Karmaşık problemleri çözmek için çok uygun olan Gaussian eleme yöntemine alternatifdir. Doğrusal eşlenik gradyan metodunun performansı katsayı matrisinin özdeğerlerinin dağılımına bağlıdır. Doğrusal

sistem dönüştürülerek veya ön koşullandırılarak bu dağılım daha uygun yapılabilir ve metodun yakınsamasını belirgin olarak geliştirilebilir. Ön koşullandırma pratik eşlenik gradyan stratejilerinin tasarımında hayati bir rol oynar. Bizim doğrusal eşlenik gradyan metodunu ele alış biçimimiz, optimizasyonda önemli olan metodun bu özelliklerini belirtecektir.

İlk doğrusal olmayan CGM, Fletcher ve Reeves tarafından 1960' larda sunulmuştur. Geniş ölçekli doğrusal olmayan optimizasyon problemlerini çözmek için yakın zamanda bilinen tekniklerden birisidir. Yıllar boyunca bu orijinal tasarımın birçok varyantı önerilmiştir ve bazıları pratikte geniş olarak kullanılmıştır. Bu algoritmaların anahtar özellikleri matris kaydedilmesini gerektirmemesi ve dik iniş metodundan daha hızlı olmasıdır.

3.4.1. Doğrusal Eşlenik Gradyan Metodu

Bu bölümde doğrusal eşlenik gradyan metodunu türetecek ve onun gerekli yakınsama özelliklerini tartışacağız. Sadelik için doğrusal nitelendirmesini kullanmayacağız.
CGM,

$$Ax = b \tag{3.9}$$

doğrusal denklem sistemlerini çözmek için iteratif bir metottur. Burada A, $n \times n$ boyutlu simetrik ve pozitif tanımlı bir matristir. (3.9) problemi aşağıdaki minimizasyon problemine denk olacak şekilde ifade edilebilir:

$$\phi(x) = \frac{1}{2} x^T Ax - b^T x, \tag{3.10}$$

öyle ki (3.9) ve (3.10) aynı tek çözüme sahiptir. Bu eşitlik CGM' yi hem doğrusal sistemleri çözmek için bir algoritma hem de konveks kuadratik fonksiyonların minimizasyonu için bir teknik olarak yorumlanmasına izin verecektir. İleri referans için ϕ' nin gradyanının doğrusal sistemin artığına eşit olduğunu not edelim,

$$\nabla\phi(x) = Ax - b = -r(x) \qquad (3.11).$$

Eşlenik Yön Metotları (Conjugate Direction Methods)

CGM' nin dikkate değer özelliklerinden birisi onun ekonomik anlamda eşleniklik (conjugacy) olarak bilinen özelliğe sahip bir vektörler kümesi üretme yeteneğidir.

Tanım 3.1. Bir sıfır-olmayan n boyutlu $\{p_0, p_1, ..., p_l\}$ vektörler kümesine, $n \times n$ boyutlu simetrik pozitif tanımlı A matrisine göre eşleniktir denir eğer,

$$tüm\ i \neq j\ için,\ \ p_i^T A p_j = 0\ \ ise \qquad (3.12).$$

Bu özelliği sağlayan vektörler kümesinin aynı zamanda doğrusal bağımsız sistem oluşturduğu görülebilir. Eşlenikliğin önemi, $\phi(.)$' yı bir eşlenik küme içerisindeki bağımsız yönler boyunca ardışık azaltarak n adımda minimize edebilmesinde yatar. Bu iddiayı onamak için aşağıdaki eşlenik yön metodu düşünülür. Verilen bir $x_0 \in R^n$ başlangıç noktasında ve eşlenik $\{p_0, p_1, ..., p_{n-1}\}$ yönler kümesinde aşağıdaki iterasyon süreciyle bir $\{x_k\}$ dizisi üretilsin.

$$x_{k+1} = x_k + \alpha_k p_k \qquad (3.13)$$

α_k katsayısı, kuadratik $\phi(.)$ fonksiyonunun, $x_k + \alpha_k p_k$ boyunca α_k' ya göre bir boyutlu minimizasyonu probleminden bulunabilir:

$$\alpha_k = \frac{r_k^T p_k}{p_k^T A p_k} \qquad (3.14)$$

Aşağıdaki sonuç elde edilebilir: (Nocedal ve Wright 1999; Greenbaum 1997).

Teorem 3.1. Herhangi $x_0 \in R^n$ başlangıç noktası için (3.13), (3.14) eşlenik yön algoritması tarafından üretilen $\{x_k\}$ dizisi, (3.9) doğrusal sisteminin x^* çözümüne en fazla n adımda yakınsar.

İspat. $\{p_i\}$ yönleri doğrusal olarak bağımsız olduğu için, tüm $\mathbb{R}^n$ uzayını oluşturmalıdır. Böylece x_0 ve x^* arasındaki fark, σ_k skalerlerinin belirli bir seçimi için aşağıdaki şekilde yazılabilir:

$$x^* - x_0 = \sigma_0 p_0 + \sigma_1 p_1 + ... + \sigma_{n-1} p_{n-1}, \tag{3.15}$$

Bu ifadeyi $p_k^T A$ ile soldan çarparsak ve (3.12) eşleniklik özelliğini kullanırsak aşağıdaki eşitlik elde edilir:

$$\sigma_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k} \tag{3.16}$$

Şimdi, bu σ_k katsayılarının, (3.14) formülüyle üretilen α_k adım uzunluklarıyla çakıştığını göstererek sonuç elde edilir.

Eğer x_k, (3.13) ve (3.14) algoritması tarafından üretilmişse, o zaman

$x_k = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + ... + \alpha_{k-1} p_{k-1}$ ' dır.

Bu ifadeyi $p_k^T A$ ile soldan çarpar ve eşleniklik özelliğini kullanırsak, şunu elde ederiz:

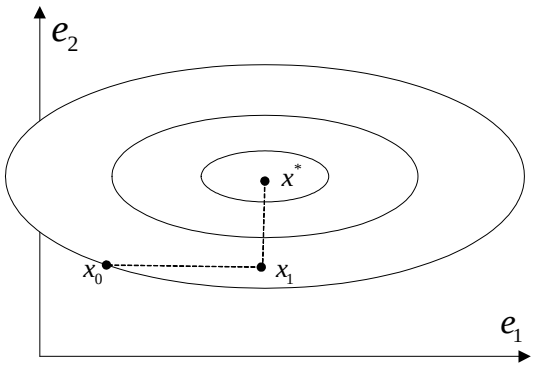
$$p_k^T A(x_k - x_0) = 0, \tag{3.17}$$

ve bu nedenle

$$p_k^T A(x^* - x_0) = p_k^T A(x^* - x_k) = p_k^T (b - Ax_k) = p_k^T r_k. \tag{3.18}$$

Bu ilişkiyi (3.14) ve (3.16) ile karşılaştırırsak, sonucu veren $\sigma_k = \alpha_k$ eşitliği bulunur. ■

Eşlenik yönlerin özellikleri şu şekilde yorumlanabilir; Eğer (3.10)' daki A matrisi köşegen matris ise, $\phi(.)$ fonksiyonunun seviye eğrileri, eksenleri koordinat yönlerine göre ayarlanmış paralel elipslerdir (Şekil 3.5). Bu fonksiyonun minimum noktası, koordinat yönleri boyunca bir boyutlu minimizasyonlar gerçekleştirilerek bulunabilir.

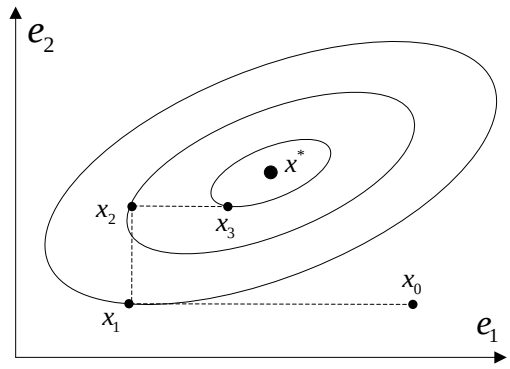


Şekil 3.5. Koordinat yönleri boyunca ardışık minimizasyonlar gerçekleştirilerek, bir köşegen Hessian (A) matrisli bir kuadratik fonksiyonun minimumu n iterasyonda bulunur.

A köşegen olmadığında, kontürleri hâla eliptiktir ama artık koordinat yönlerine göre ayarlanamazlar. Bu yönler boyunca sırasıyla yapılan ardışık minimizasyonlar stratejisi artık n iterasyonda çözüme ulaşmaz (hatta sınırlı sayıda iterasyonda). Bu fenomen Şekil 3.6' da iki boyutlu olarak gösterilmiştir.

Eğer problem A' yı diagonal yapmak üzere dönüştürülürse, koordinat yönleri boyunca minimize edilerek Şekil 3.5' deki durum geri getirilebilir. Bu durumda problemi yeni değişkenler tanımlayarak, doğrusal olarak aşağıdaki şekilde dönüştürmek gerekir:

$$\hat{x} = S^{-1}x \tag{3.19}$$



Şekil 3.6. Koordinat eksenleri boyunca ardışık minimizasyonlar gerçekleştirilerek, genel konveks kuadratik bir fonksiyon için minimum n iterasyonda bulunamaz.

(8.15) ifadesinde S, $n \times n$ matris şu şekilde tanımlanır:

$$S = [p_0 \; p_1 \ldots p_{n-1}]. \tag{3.20}$$

Burada $\{p_0, p_1, ..., p_{n-1}\}$, A' ya bağlı eşlenik yönler kümesidir. (3.10)' la tanımlanan ϕ fonksiyonu şimdi şu şekle gelir:

$$\hat{\phi}(\hat{x}) = \phi(S\hat{x}) = \frac{1}{2}\hat{x}^T (S^T AS)\hat{x} - (S^T b)^T \hat{x} \tag{3.21}$$

(3.12) eşleniklik özelliği sayesinde, $S^T AS$ matrisi köşegendir. Böylece $\hat{x}$' nın koordinat yönleri boyunca bir boyutlu minimizasyonlar gerçekleştirerek $\hat{\phi}$' yı minimize eden değeri bulabiliriz. Buna rağmen (3.19) ilişkisi nedeniyle $\hat{x}$-uzayındaki her koordinat yönü x-uzayındaki p_i yönüne karşılık gelir. Böylece ϕ' ye uygulanan koordinat arama stratejisi eşlenik yön algoritması (3.13), (3.14)' e eşdeğerdir. Sonuç olarak Teorem 3.1' deki gibi eşlenik yön algoritması en fazla n adımda sona erer.

(3.9)' a geri dönersek, bir başka ilginç özelliği not ederiz: Hessian matrisi köşegen olduğunda her koordinat minimizasyonu x^* çözümünün bir bileşenini doğru olarak belirler. Diğer bir deyişle, k kadar bir boyutlu minimizasyondan sonra, kuadratik ϕ fonksiyonunun, $e_1, e_2, ..., e_k$ ile oluşturulan alt uzayda minimize edilir.

Aşağıdaki teorem, kuadratik ϕ fonksiyonunun Hessian' ının köşegen olması gerekmediği genel durum için bu önemli sonucu ispatlar Burada ve sonra, $taban\{p_0, p_1, ..., p_k\}$ notasyonu, $p_0, p_1, ..., p_k$' nın tüm doğrusal kombinasyonlarını belirtmek için kullanılır:

$$taban\{p_0, p_1, ..., p_k\} = \left\{\alpha_0 p_0 + \alpha_1 p_1 + ... + \alpha_k p_k \ \ , \alpha_i \in R \ \ , i = \overline{1, n}\right\} \tag{3.22}$$

Sonucu ispat ederken, (3.9) ve (3.11) ilişkilerinden kolayca onanan aşağıdaki ifadeyi kullanılır:

$$r_{k+1} = r_k - \alpha_k A p_k \tag{3.23}$$

Teorem 3.2. (Altuzay minimizasyonunun genişletilmesi) (Nocedal ve Wright 1999).

$x_0 \in R^n$ herhangi bir başlangıç noktası olsun ve $\{x_k\}$ dizisi eşlenik yön algoritması (3.13), (3.14) tarafından üretilmiş olsun. O zaman

$$i = 0, ..., k-1 \ \ için, \ \ r_k^T p_i = 0 \tag{3.24}$$

ve x_k, $\phi(x) = \frac{1}{2} x^T A x - b^T x$ kuadratik fonksiyonunun

$$\{x \mid x = x_0 + taban\{p_0, p_1, ..., p_{k-1}\}\} \tag{3.25}$$

alt uzayında minimum noktasıdır.

İspat. Bir $\tilde{x}$ noktasının (3.25) kümesinde yalnız ve yalnız, her $i = 0,1,...,k-1$ için $r(\tilde{x})^T p_i = 0$ olur ise ϕ' yi minimize edeceğini göstererek başlayalım. $h(\sigma) = \phi(x_0 + \sigma_0 p_0 + ... + \sigma_{k-1} p_{k-1})$ tanımlayalım, burada $\sigma = (\sigma_0, \sigma_1, ..., \sigma_{k-1})^T$. $h(\sigma)$ ciddi konveks kuadratik olduğu için, aşağıdaki eşitliği sağlayan bir tek minimum noktası σ^*' a sahiptir. Optimum için gereklilik koşuluna göre

$$\frac{\partial h(\sigma^*)}{\partial \sigma_i} = 0, \quad i = 0,1,...,k-1. \tag{3.26}$$

Zincir kuralından (3.26) aşağıdaki eşitliği ifade eder.

$$\nabla \phi(x_0 + \sigma_0^* p_0 + ... + \sigma_{k-1}^* p_{k-1})^T \, p_i = 0, \; i = 0,1,...,k-1. \tag{3.27}$$

(3.11) ' i hatırlarsak, arzu edilen sonuç elde edilir. x_k ' nın (3.24)' yi sağladığını göstermek için tümevarım kullanılır. α_k her zaman bir boyutlu minimum olduğu için, $r_1^T p_0 = 0$ hemen elde edilir. Şimdi tümevarım hipotezini kuralım; şöyle ki, $r_{k-1}^T p_i = 0, \; i = 0,...,k-2.$ (5.9)' dan,

$r_k = r_{k-1} - \alpha_{k-1} A p_{k-1}$, buradan α_{k-1} ' in (3.14) tanımından.

$p_{k-1}^T r_k = p_{k-1}^T r_{k-1} - \alpha_{k-1} p_{k-1}^T A p_{k-1} = 0$ elde edilir.

Aynı zamanda diğer vektörler için $p_i, \; i = 0,1,...,k-2$ aşağıdaki ifade sağlanır,

$$p_i^T r_k = p_i^T r_{k-1} - \alpha_{k-1} p_i^T A p_{k-1} = 0 \tag{3.28}$$

tümevarım hipotezinden ve p_i ' nin eşlenikliğinden $i = 0,1,...,k-1 \; için, \; r_k^T p_i = 0$ sonucuna ulaşırız, böylece ispat tamamlanır. ■

Teorem 3.2' ye göre cari artık r_k önceki arama yönlerinin tümüne ortogonaldir (diktir).

Buraya kadar anlatılanlar geneldi, çünkü herhangi bir eşlenik yön $\{p_0, p_1, ..., p_{n-1}\}$ kümesinin seçimine dayanan eşlenik yön metodunu ele almıştı. Eşlenik yönler kümesi seçmenin birçok yolu vardır. Örneğin, A ' nın özvektörleri $v_1, v_2, ..., v_n$ A ' ya göre eşlenik olduğu gibi aynı zamanda karşılıklı ortogonaldir,

böylece bu özvektörler $\{p_0, p_1, ..., p_{n-1}\}$ vektörleri olarak kullanılabilir. Ama buna rağmen, geniş-ölçekli uygulamalar için özvektörler kümesinin tamamını hesaplamak pratik olmaz, bunun için büyük ölçüde hesaplama gerekir. Bir alternatifte Gram-Schmidt ortogonalleştirme sürecini ortogonal yönler kümesi yerine eşlenik yönler kümesi üretmesi için düzenlemektir (Bu düzenlemeyi oluşturmak kolaydır, zaten eşleniklik ve ortogonallik birbirleriyle yakın ilişkilidir). Bu yaklaşım da oldukça maliyetlidir çünkü tüm yön kümesini kaydetmemizi gerektirir.

CGM' nin Temel Özellikleri

CGM çok özel bir niteliğe sahip bir eşlenik yön metodudur: CGM eşlenik vektörler kümesi oluştururken sadece önceki p_{k-1} vektörünü kullanarak yeni vektör p_k' yı hesaplayabilir. Bu metodun, eşlenik kümenin k. elemanı p_k' yı hesaplamak için daha önceki elemanlarını, $p_0, p_1, ..., p_{k-2}$ bilmesine gerek yoktur; p_{k-1} vektörüyle belirlenen p_k otomatik olarak bu vektörlere eşlenik olur. Bu dikkate değer özellik metodun az saklama alanı ve hesaplama gerektirdiğini açıklar.

Şimdi CGM' nin detaylarını inceleyelim. Bu metotta her p_k yönü dik iniş yönünün $-\nabla\phi(x_k)$' nın (artık r_k ile aynı olan; (3.11)' den) ve bir önceki yönün p_{k-1}' in doğrusal kombinasyonu olarak seçilir ve aşağıdaki şekilde yazılır:

$$p_k = r_k + \beta_k p_{k-1} \tag{3.29}$$

β_k katsayısı, p_{k-1} ve p_k' nın A' ya göre eşlenik olması koşulundan belirlenir. (3.29)' u p_{k-1}^T ile çarparsak ve $p_{k-1}^T A p_k = 0$ koşuluna dikkat edersek aşağıdaki ifade bulunur:

$$\beta_k = \frac{-r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}} \tag{3.30}$$

Başlangıç noktası x_0' da ilk arama yönü p_0' ı dik iniş yönü olarak seçmek sezgisel olarak bir anlam ifade eder. Genel CGM' deki gibi her bir arama yönünde ardışık bir-boyutlu minimizasyonlar gerçekleştirilir. Böylece tam bir algoritma belirlenmiş olur ve aşağıdaki şekilde gösterilir.

Algoritma 3.1 (CG).

x_0 *verilsin.*

$r_0 \leftarrow Ax_0 - b, \;\; p_0 \leftarrow -r_0, \;\; k \leftarrow 0;$

$r_k \neq 0$ *iken*

$$\alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k}; \qquad (3.31a)$$

$$x_{k+1} \leftarrow x_k + \alpha_k p_k; \qquad (3.31b)$$

$$r_{k+1} \leftarrow r_k + \alpha_k A p_k; \qquad (3.31c)$$

$$\beta_{k+1} \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}; \qquad (3.31d)$$

$$p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k; \qquad (3.31e)$$

$$k \leftarrow k+1; \qquad (3.31f)$$

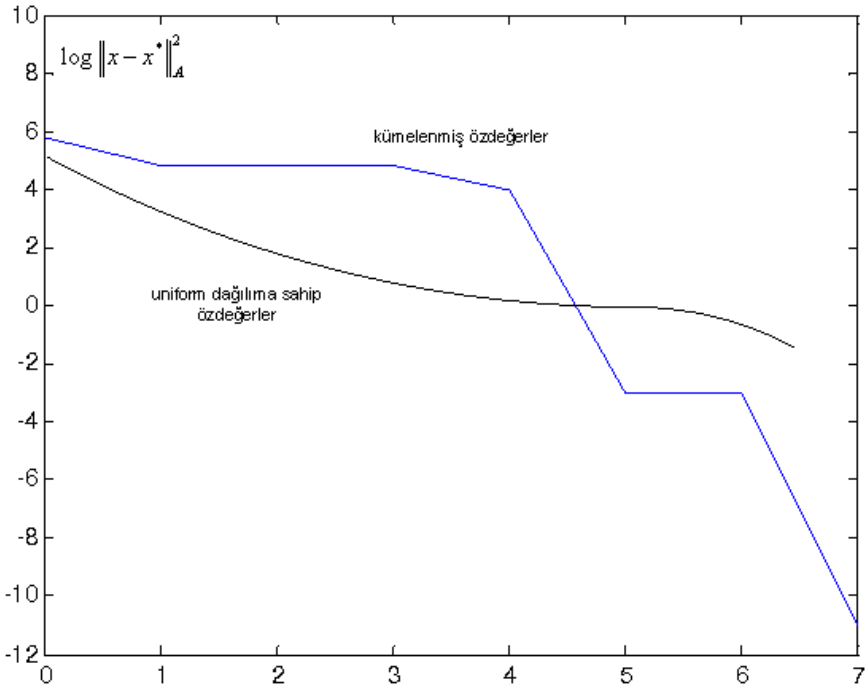
son (*iken*)

Verilen herhangi bir noktada x, r ve p vektörlerini son iki iterasyondan daha fazla bilmemiz gerekmez. Böylelikle algoritmanın uygulamalarında saklama alanından kazanmak için vektörlerin yeni değerleri eski değerleri üzerine yazılır. Esas hesaplama işleri her adımda yapılan matris-vektör çarpımlarıdır. Ap_k, iç çarpımların $p_k^T(Ap_k)$ ve $r_{k+1}^T r_{k+1}$ hesaplaması ve üç vektör toplamının hesabıdır. İç çarpım ve vektör toplamları küçük çoklu n-kayan nokta işlemi ile gerçekleştirilir. Matris-vektör çarpımı ise tabii ki probleme bağlıdır. CG metodu büyük problemler için tavsiye edilir, diğer hallerde ise Gauss elemesi veya tekil değer dekompozisyonları gibi diğer faktörleştirme algoritmaları tercih edilir ki onlar yuvarlama hatalarına daha az duyarlıdır. Büyük problemler için CG metodunun avantajı katsayı matrisini değiştirmemesidir ve faktörizasyon teknikleri gibi matrisi sağlayan dizilerde doluluk üretmez. Bir diğer anahtar özellik ise CGM çözüme bazen çok hızlı ulaşır. Kesin aritmetik CGM' nin çözüme en fazla n

adımda ulaşacağını söylemiştik. Daha dikkat çeken ise A' nın özdeğerlerinin dağılımı ciddi faydalı özelliklere sahip olduğunda, algoritma n adımdan çok daha önce çözüme ulaşır. Aşağıdaki teorem ispatlanabilir (Nocedal veWright 1999).

Teorem 3.3. Eğer A sadece r tane ayrık özdeğere sahipse o zaman CG iterasyonu en fazla r iterasyonda çözüme ulaşır.

Şekil 3.7. CG' nin beş tane büyük özdeğerle beraber tüm küçük özdeğerleri 0.95 ile 1.05 arasında kümelenmiş bu tip bir problemdeki tutumunu gösterir ve bu tutumu özdeğerlerinin dağılımı belirli bir rassal dağılımı sağlayan bir problemdeki tutumuyla karşılaştırır. Her iki durumda da her iterasyondan sonra ϕ' nin logaritmasının grafiği çizilir.



Şekil 3.7. Eşlenik gradyan metodunun performansı (a) beş özdeğeri büyük olan ve kalan özdeğerlerin 1 etrafında kümelendiği bir problem ve (b) uniform dağılmış özdeğerlere sahip bir matris

Kümelenmiş özdeğerler problemi için yakınsama oranının daha hızlıdır, tersine rasgele dağılmış özdeğerli problemin yakınsama oranı daha yavaş ve uniform'

dur.CG için yakınsama oranına bir tahmin de Euclidean koşul sayısı üzerine kurulur ve bu koşul aşağıdaki şekilde tanımlanır.

$$\kappa(A) = \|A\|_2 \left\|A^{-1}\right\|_2 = \lambda_1 / \lambda_n \tag{3.32}$$

Burada gösterilir ki (Nocedal ve Wright 1999),

$$\left\|x_k - x^*\right\|_A \leq \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\right)^{2k} \left\|x_0 - x^*\right\|_A \tag{3.33}$$

Bu sınır genellikle hatanın büyük aşırı tahminini verir, ama A hakkında sahip olduğumuz tek bilginin uç özdeğerler λ_1 ve λ_n' nin tahminleri olduğunda faydalı olabilir.

3.4.2. Doğrusal Olmayan Eşlenik Gradyan Metodu

CG metodunun, Algoritma 3.1 (3.10)' la tanımlanmış konveks kuadratik ϕ fonksiyonu için minimizasyon algoritması olarak görülebilir. Bu yaklaşımı genel konveks fonksiyonlara veya genel doğrusal olmayan fonksiyonlara uyarlanabilir mi diye sormak doğaldır.

Fletcher-Reeves Metodu

Algoritma 3.1' de iki basit değişiklik yaparak bu tipin bir genişletilmesinin mümkün olduğunu Fletcher ve Reeves göstermiştir. Öncelikle (3.31a) seçimi için, adım uzunluğu α_k için (p_k arama yönü boyunca ϕ' yi minimize eden), p_k boyunca doğrusal olmayan fonksiyonunun yaklaşık minimumunu bulan bir doğru arama gerçekleştirmemiz gerekir. İkincisi, Algoritma 3.1' de basitçe ϕ' nin gradyanı olan artık r, doğrusal olmayan amaç fonksiyonu f' nin gradyanıyla değiştirilmesi gerekir. Bu değişiklikler doğrusal olmayan optimizasyon için aşağıdaki algoritmayı açığa çıkarır.

Algoritma 3.2 (FR-CG)

x_0 *verilsin.*

$f_0 = f(x_0)$' *hesapla* , $\nabla f_0 = \nabla f(x_0)$;

$p_0 = -\nabla f_0$, $k \leftarrow 0$;

$\nabla f_k \neq 0$ *iken*

α_k '*yı hesapla ve* $x_{k+1} = x_k + \alpha_k p_k$;

∇f_{k+1} '*i hesapla*;

$$\beta_{k+1}^{FR} \leftarrow \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}; \qquad (3.34a)$$

$$p_{k+1} \leftarrow -\nabla f_{k+1} + \beta_{k+1}^{FR} p_k \qquad (3.34b)$$

$$k \leftarrow k+1; \qquad (3.34c)$$

son (*iken*)

Eğer f' yi kesin konveks kuadratik ve α_k' yı da kesin minimizer olarak seçersek, bu algoritma doğrusal CGM' ye indirgenir; Algoritma 3.1. Algoritma 3.2 büyük doğrusal olmayan optimizasyon problemleri için uyarlanabilirdir çünkü her iterasyon sadece amaç fonksiyonu ve gradyanın hesaplanmasını gerektirir. Herhangi bir matris operasyonu gerçekleştirilmez ve sadece birkaç vektör kaydını gerektirir.

Polak-Ribière metodu

Fletcher-Reeves metodunun temelde β_k parametresinin seçiminde farklılaşan birçok çeşidi vardır. Bunların en önemlisi Polak ve Ribière tarafından önerilmiştir. Bu parametreyi aşağıdaki şekilde tanımlar:

$$\beta_{k+1}^{PR} = \nabla f_{k+1}^T \frac{(\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2} \qquad (3.35)$$

(3.35)' in (3.34a)' nın yerini aldığı algoritmayı PR-CG olarak adlandıracağız ve algoritma 3.2' yi Algoritma FR-CG olarak tanımlayacağız. f ciddi konveks kuadratik fonksiyon ve doğru arama kesin olduğunda bunlar aynıdır. Genel

doğrusal olmayan fonksiyonlara kesin olmayan doğru arama uygulandığında iki algoritma belirgin olarak farklı davranır. Sayısal tecrübe gösterir ki PR-CG algoritması daha güçlü ve verimlidir.

Amaç fonksiyonun kuadratik ve doğru aramanın kesin olduğu durum için Fletcher-Reeves β_{k+1}^{FR} formülüyle çakışan, birçok başka β_{k+1} seçimi vardır. Hestenes-Stiefel formülü

$$\beta_{k+1}^{HS} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{(\nabla f_{k+1} - \nabla f_k)} \tag{3.36}$$

PR-CG algoritmasına benzer bir algoritmanın oluşmasını sağlar; hem teorik yakınsama özellikleri anlamında hem de pratikteki performans anlamında. Diğer önerilen β_k tanımlarından hiçbiri Polak-Ribière seçiminden daha verimli olduğunu ispatlayamamıştır.

3.5. Ölçekli Eşlenik Gradyan (Scaled Conjugate Gradients)

Eşlenik gradyan algoritmasındaki adım büyüklüğü, doğru aramanın yardımıyla Hessian matrisinin hesaplanmasına gerek kalmadan seçilebilir. Buna rağmen, doğru aramanın kendisi bazı problemleri beraberinde getirir. Özellikle, her doğru minimizasyonu her biri hesaplama olarak masraflı olan çeşitli hata fonksiyonu hesaplamalarını içerir. Aynı zamanda doğru arama yordamının kendisi, her doğru arama için durdurma kriterini belirleyen bazı parametreler içerir. Algoritmanın genel performansı bu parametre değerine duyarlı olabilir çünkü yeterli derecede kesin olmayan bir doğru arama α_k değerinin doğru olarak belirlenmediğini ifade ederken aşırı kesin bir doğru arama boşa harcanmış bir hesaplama anlamına gelebilir.

Møller (1993) geleneksel eşlenik gradyanların doğru arama yordamında kurtulmanın bir yolu olarak ölçekli eşlenik gradyan algoritmasını sunmuştur. Öncelikle Hessian matrisinin, (3.31a) α_k formülüne Hessian ile bir p_k vektörünün çarpımı şeklinde girdiğine dikkat edelim.

Çeşitli hata fonksiyonu hesaplamalarını içeren doğru minimizasyonu kullanmanın yerine yukarıda bahsettiğimiz yöntemleri kullanarak basitçe Hp_k hesaplanır. Ama bu basit yaklaşım başarısız olur çünkü kuadratik olmayan hata fonksiyonu durumunda, Hessian matrisi pozitif tanımlı olmayabilir. Bu durumda ise (3.31a)' daki payda negatif olabilir ve ağırlık güncellemesi hata fonksiyonunun değerinde bir artışa yol açabilir. Hessian matrisini, pozitif tanımlı olmasını sağlayacak şekilde düzenleyerek bu problemin üstesinden gelinebilir. Bu Hessian' a birim matrisin belirli bir çarpımı eklenerek gerçekleştirilir ve böylece Hessian

$$H + \lambda I \tag{3.37}$$

şekline gelir. Burada I ; birim matris ve $\lambda \geq 0$ bir ölçeklendirme katsayısıdır. λ yeterince büyük olursa, bu düzenlenmiş Hessian' ın pozitif tanımlı olması garantilenmiş olur. Adım uzunluğu için formül aşağıdaki şekilde verilir:

$$\alpha_k = -\frac{p_k^T r_k}{p_k^T H p_k + \lambda_k \left\| p_k \right\|^2} \tag{3.38}$$

λ_k parametresinin seçimi ve algoritmanın adım adım ayrıntılı olarak anlatımı Møller (1993b) ve Williams (1991)' da bulunabilir.

3.6. Uygulama

İntrasitoplazmik sperm enjeksiyonu (intracytoplasmic sperm injection-ICSI), erkeklerde nonobstrüktif azoospermi' yi de (nonobstructive azoospermia – hastada sperm bulunmaması) içeren ciddi kısırlık problemleri için bir çözüm önermektedir. Nonobstrüktif azoospermi' ye sahip erkeklerin değerlendirilmesinin bir bölümü, testislerdeki spermatozoa' yı yani hastada sperm hücrelerinin var olup olmadığını tahmin etmek için kullanılan bir algoritmadır. Algoritma fiziksel incelemeyi ve laboratuar verisini birleştirir ve özellikle serum hormon düzeyleri üzerine odaklanır (Samlı ve Dogan, 2004).

Bu bölümde (nonobstructive azoospermia' ya sahip erkeklerde testicular biopsy' ye dayanan spermatozoa tahmini için bir yapay sinir ağı oluşturuldu. Oluşturulan yapay sinir ağı önceki bölümlerde anlatılan ve teorik açıdan incelenen değişik eğitim algoritmalarıyla eğitildi. Ağın performansı, eğitim algoritmalarının yakınsama hızları karşılaştırılmalı olarak değerlendirildi. Çalışmada kullanılan veriler ,1997 ve 2000 yılları arasında fiziksel nonobstructive azoospermia' ya dayanan kısırlık sorunu yaşayan, Afyon Kocatepe Üniversitesi Araştırma hastanesindeki 303 hastanın medikal kayıtlarından ve fiziksel incelemelerinden elde edilmiştir.

Girdi değişkenleri hastanın yaşı, kısırlık süresi, serum follicle-stimulating hormone (serum follikül uyarıcı hormon), luteinizing hormone, total testosterone, proluctin ve sol ve sağ testis hacmi olarak alınmıştır. Yapay sinir ağı testiküler sperm ekstraksiyonu (testicular sperm extraction-TESE)' de spermatozoa' nın görüldüğü ve görülmediği durumlardaki veri temel alınarak eğitilmiştir.

Veri seti rassal olarak aşağıdaki şekilde üç gruba ayrılmıştır:

- Eğitim kümesi: 152 hasta
- Geçerlilik kümesi: 75 hasta
- Test kümesi: 76 hasta

Eğitim kümesi ağı eğitmek için kullanılan girdi örneklerinin bulunduğu kümedir. İkinci küme olan geçerlilik kümesine ağın verdiği cevaplar eğitim süresince izlenir. Bu kümede bulunan örnekler eğitimde kullanılmaz. Eğitim sırasında eğitim kümesinin hatası azalırken geçerlilik kümesinin hatası da azalıyorsa eğitime devam edilir. Eğitim kümesi için oluşan hata azalmaya devam ederken geçerlilik kümesi için oluşan hata belirli bir adım süresince artarsa eğitim sona erer. Bu işleme çapraz-geçerlilik (cross-validation) denilmektedir. Test kümesi ağın performansını ölçmek için kullanılır.

Ağ mimarisi, farklı denemeler sonucunda aşağıdaki şekilde oluşturulmuştur.

Girdi Katmanı: 8 yapay sinir hücresi

1. Gizli Katman: 13 yapay sinir hücresi

Çıktı Katmanı: 1 yapay sinir hücresi

Gizli katmanlarda ve çıktı katmanında kullanılan aktivasyon fonksiyonları Tangent Sigmoid olarak seçilmiştir.

Çizelge 3.1. Geriye yayılım eğitim algoritmaları

GD	Gradyan azalan.
GDM	Momentumlu gradyan azalan.
GDA	Değişken öğrenme oranlı gradyan azalan.
GDX	Değişken öğrenme oranlı ve momentumlu gradyan azalan.
CGF	Eşlenik gradyan (Fletcher-Reeves metodu).
CGP	Eşlenik gradyan (Polak-Ribière metodu).
SCG	Ölçekli eşlenik gradyan.

Çizelge 3.2. Geriye yayılım eğitim algoritmalarının yakınsama ve sınıflandırma performansları.

	Epoch (Döngü Sayısı)	***Geçen Süre (saniye)***	***Doğru Sınıflandırma Oranı (Eğitim)***	***Doğru Sınıflandırma Oranı (Test)***
GD	1349.4	10.225	0.67105	0.76316
GDM	424.42	3.3276	0.65789	0.82895
GDA	54.774	0.47365	0.63816	0.78947
GDX	47.387	0.42213	0.64474	0.77632
CGF	17.613	0.29648	0.67105	0.76316
CGP	11.968	0.23319	0.63816	0.76316
SCG	14.161	0.19726	0.67105	0.80263

Çizelge 3.2’ ye bakıldığında algoritmaların doğru sınıflandırma oranlarına bakıldığında test grubu için elde edilen değerlerin eğitim grubuna göre daha iyi olduğu açıkça görülür. Bu çalışmada bizim dikkate aldığımız doğru sınıflandırma oranı esas olarak test grubununkidir. Bunun sebebi yapay sinir ağı eğitim süreci ağın genelleştirme performansı iyi olacak şekilde tasarlanmıştır. Çünkü, gerek sınıflandırma problemlerinde gerekse tahmin problemlerinde bir tercih yapmak zorunda kalırız. Arzu edilen ağın genelleştirme performansının iyi olması ise, ağın eğitimi bir geçerlilik kümesine dayanarak sonlandırılır (bizim uygulamamızda olduğu gibi) ve böylece ağın yeni veriye tepkisi daha iyileştirilir. Diğer taraftan ağın eğitim verisini daha iyi tahmin etmesi isteniyorsa (interpolasyon) eğitim

süreci istenilen hata değerine ulaşıncaya kadar sürdürülebilir ama bu gerçek hayatta istenmeyen bir durumdur. Bizim istediğimiz ağın yeni güncel veriyi iyi tahmin edebilmesidir.

Algoritmaların döngü sayılarına ve yakınsama sürelerine bakıldığında eşlenik gradyana dayanan yöntemlerin diğer gradyan azalan yöntemlere göre daha üstün olduğu rahatlıkla söylenebilir. Ama momentumlu algoritmalardan GDA, ve GDX' in diğer gradyan azalan yöntemlerden GD ve GDM' den daha üstün olduğu görülmektedir. Hatta, yakınsama süreleri ve doğru sınıflandırma oranlarının da neredeyse eşlenik gradyan yöntemlere yakın olduğu söylenebilir.

4. KUADRATİK HATA FONKSİYONU İÇİN MOMENTUMLU GRADYAN DÜŞÜMÜ ALGORİTMASININ KARARLILIĞI

2. ve 3. bölümlerde geriye yayılım y.s.a. eğitim metodu ve onunla bağlı olan gradyan terimli optimizasyon algoritmaları hakkında bilgi verildi. Momentum teriminin gradyan düşümü algoritmasının kararlılığı ve hızlı yakınsaması problemlerinde etkili olduğu görülmektedir. Bu bölümde hata fonksiyonunun kuadratik olması durumunda momentumlu BP (BPM) algoritmasının kararlılığı ve yakınsama hızı incelenmiştir. Öğrenme oranı ve momentum katsayısının dinamik seçimi halinde BPM algoritmasının CG algoritması gibi çalışabileceği gösterilmiştir. Bir tutucu kuvvet alanı içinde yapışkan bir ortamda hareket eden Newton parçacığının hareket denkleminin BPM sürecine benzer olduğu açıklanmıştır. Elde olunan teorik sonuçlar deneysel çalışmalarda gösterilmiştir.
BPM probleminde x ağırlık vektörünün kuadratik fonksiyonu olan ve minimize edilen hata fonksiyonunun aşağıda verilen şekilde olduğunu varsayalım:

$$F(x) = \frac{1}{2} x^T A x - b^T x + c \,. \tag{4.1}$$

Burada A $n \times n$ boyutlu simetrik pozitif tanımlı matris, B n boyutlu vektör, c verilmiş bir sabittir. F fonksiyonunun x noktasındaki gradyanı $\nabla F(x) = Ax - b$ olur. $Ax = b$ doğrusal denkleminin çözümü açısından $b - Ax = r$ artık olarak düşünülebilir ve $\nabla F(x) = -r$ olur. (4.1) ile verilen fonksiyonun minimum değerini veren x^* ağırlık vektörü,

$$Ax = b \tag{4.2}$$

doğrusal denkleminin çözüm vektörüdür. Böylece, (4.1) ile verilen fonksiyonunun minimumunu bulan bir iteratif süreç, (4.2)' de verilen denklemin çözümü için de bir iteratif süreç olarak kabul edilebilir. Standart BP algoritması,

$$x_{k+1} - x_k = -\eta \nabla F(x_k) \equiv \eta r_k \tag{4.3}$$

şeklindedir. Bu algoritma η öğrenme oranı ile A matrisinin en büyük özdeğerinin çarpımı, 2' den küçük olduğunda kararlıdır (Hagan ve ark. 1996).

(4.3) ile verilen algoritmaya momentum terimi eklendiğinde algoritma aşağıdaki gibi yazılabilir;

$$x_{k+1} - x_k = \mu(x_k - x_{k-1}) - (1-\mu)\eta\nabla F(x_k) \tag{4.4}$$

burada μ momentum katsayısı olarak tanımlanır. μ negatif olduğunda algoritmanın yakınsama hızı çok yavaştır (Phansalkar ve ark. 1994), μ>1 olduğunda ise algoritma kararlı değildir (Torii ve ark. 1996). Bu nedenle incelemelerimizde μ parametresinin değerleri 0< μ <1 aralığında göz önüne alınacaktır. (4.4) ile verilen BPM algoritmasında μ ve η katsayıları zamana göre değişmezdir. Bu parametrelerin her iterasyon adımında değişmesi durumunda ise BPM aşağıdaki şekilde yazılır (Amit ve Kaszkurewicz, 2004).

$$x_{k+1} - x_k = \mu_k(x_k - x_{k-1}) - (1-\mu_k)\eta_k\nabla F(x_k) \tag{4.5}$$

Sabit katsayılı BPM algoritmalarının analiziyle ilgili literatürde çok sayıda çalışma yapılmıştır (Haykin, 1999). Katsayıların zamana bağlı olarak değişmesine yönelik çalışmalarda az değildir (Qian, 1999).

4.1. Sabit Katsayılı BPM Algoritmasının Kararlılığı

Bu bölümde (4.4) ile verilen sabit katsayılı BPM algoritması ele alınacaktır. $\nabla F(x) = Ax - b$ ifadesini göz önüne alındığında, (4.4) ile verilen algoritma aşağıdaki şekilde yazabilir;

$$x_{k+1} = [(1+\mu)I - (1-\mu)\eta A]x_k - \mu x_{k-1} + (1-\mu)\eta b\,. \tag{4.6}$$

(4.6) ile verilen eşitlik $\tilde{x}_k = \begin{bmatrix} x_{k-1} \\ x_k \end{bmatrix}$, $2n$ boyutlu bir vektör gibi düşünülerek aşağıdaki şekilde gösterilebilir;

$$\tilde{x}_{k+1} = Q\tilde{x}_k + q, \tag{4.7}$$

burada $Q = \begin{bmatrix} 0 & I \\ -\mu I & (1+\mu)I - (1-\mu)\eta A \end{bmatrix}$ $2n \times 2n$ boyutlu bir matris, $q = \begin{bmatrix} 0 \\ (1-\mu)\eta b \end{bmatrix}$ $2n$ boyutlu bir vektördür. (4.7) ile verilen doğrusal dinamik denklem, (4.6) denklemi ile $x_k = x_k$ denkliğinin birleştirilmesidir. Böylece, (4.7) ile verilen doğrusal dinamik denklem, BPM algoritmasının değişik bir şekilde yazılmasıdır.

(4.7) ile verilen doğrusal dinamik sistemi, Q matrisinin özdeğerlerinin büyüklüğü (modu) bir' den küçük olduğunda kararlı olur (Brogan, 1991). Böylece (4.4) BPM algoritmasının kararlılığı problemi ile Q matrisinin özdeğerlerinin modu arasında bir ilişki kurulabilir. Özdeğer ve özvektör tanımına göre,

$Qz = \lambda z$, $z \neq 0$ veya

$$\begin{bmatrix} 0 & I \\ -\mu I & (1+\mu)I - (1-\mu)\eta A \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \lambda \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \tag{4.8}$$

yazılabilir. Burada z_1, z_2 n boyutlu vektörler, $z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ $2 \times n$ boyutlu vektördür ve $z \neq 0$ 'dır. λ sayısı özdeğer, z ise ona uygun özvektördür. (4.8) ifadesi aşağıdaki biçimde yazılabilir;

$$z_2 = \lambda z_1, \; -\mu z_1 + \left[(1+\mu)I - (1-\mu)\eta A\right] z_2 = \lambda z_2. \tag{4.9}$$

$\lambda \neq 0$ olmalıdır, aksi halde $z \neq 0$ koşulu bozulur. Buna göre de eşitlik (10)' dan $z_1 = \frac{1}{\lambda} z_2$ ve

$$\left[(1+\mu)I - (1-\mu)\eta A\right] z_2 = (\lambda + \frac{\mu}{\lambda}) z_2 \tag{4.10}$$

elde edilir. Özdeğer – özvektör tanımına göre ,

$$Az_2 = \kappa z_2, \; z_2 \neq 0 \tag{4.11}$$

yazılabilir. Burada κ sayısı z_2 özvektörüne uygun bir özdeğerdir. (4.10) ve (4.11) denklemlerini sağlayan λ ' nın bulunması için;

$$(1+\mu) - (1-\mu)\eta\kappa = \lambda + \frac{\mu}{\lambda}$$

veya

$$\lambda^2 - [(1+\mu) - (1-\mu)\eta\kappa]\lambda + \mu = 0 \tag{4.12}$$

kare denklemini incelemek gerekir. $A, n \times n$ pozitif tanımlı bir matris olduğundan, n sayıda pozitif $\kappa_1, \kappa_2, ..., \kappa_n$ özdeğere sahiptir. Her bir κ_i.'ye $(i = 1, 2, ..., n)$ karşılık, (4.12) ile verilen denklemin iki kökü (gerçel veya kompleks) vardır. Böylece $Q_{2n \times 2n}$ matrisinin $2n$ sayıda özdeğeri olur.

(4.7) ile verilen doğrusal iteratif sürecinin kararlılığı için (4.12) ile verilen denklemin her bir kökünün modu 1' den küçük olmalıdır. Bu nedenle (4.4) momentumlu gradyan düşümü algoritmasının kararlılığı problemi (4.12) denkleminin incelenmesi problemi haline gelir. (4.12) denkleminin kökleri aşağıdaki gibi hesaplanabilir:

$$\lambda = \frac{[(1+\mu) - (1-\mu)\eta\kappa] \pm \sqrt{[(1+\mu) - (1-\mu)\eta\kappa]^2 - 4\mu}}{2}. \tag{4.13}$$

Şimdi (4.12) denkleminin diskriminantını yazalım.

$$D = [(1+\mu) - (1-\mu)\eta\kappa]^2 - 4\mu\,. \tag{4.14}$$

D<0 olduğunda, her bir κ_i $(i = 1, 2, ..., n)$ değeri için (4.12) denkleminin kökleri eşlenik kompleks sayılardır.

Lemma 4.1. $0 < \mu < 1$ koşulu sağlandığında, (4.12) denkleminin herhangi bir kompleks kökünün modu 1' den küçüktür.

İspat: λ' nın eşitlik (4.13) ile verilen denklemin kompleks kökü olduğunu varsayalım. Bu durumda D<0' dır. (4.13) ve (4.14) eşitliklerine göre ve D<0 olduğu için

$$\lambda = \frac{(1+\mu) - (1-\mu)\eta\kappa}{2} \pm \frac{\sqrt{-D}}{2}\, i \qquad\qquad i = \sqrt{-1} \tag{4.15}$$

olur. Buradan, kompleks sayının modunun tanımına göre;

$$|\lambda|^2 = \frac{[(1+\mu) - (1-\mu)\eta\kappa]^2}{4} + \frac{-D}{4} = \mu \text{ veya } |\lambda| = \sqrt{\mu} \tag{4.16}$$

olur. $0 < \mu < 1$ olduğu için $|\lambda| < 1$ olduğu görülür. ■

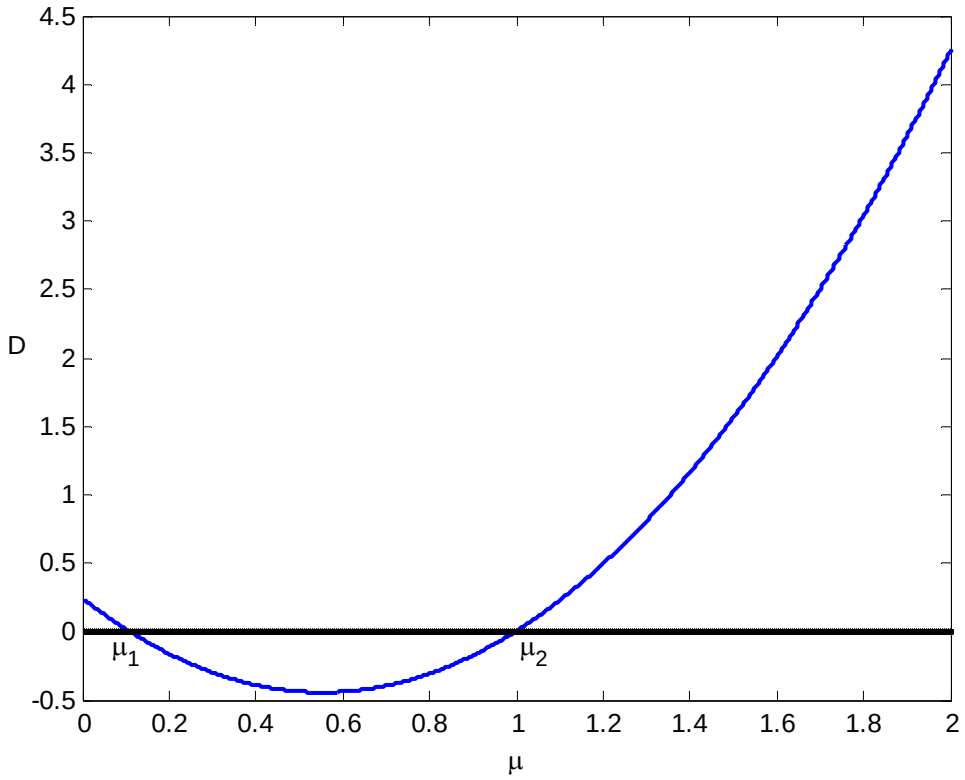
Şimdi eşitlik (4.14) ifadesi ele alınırsa, diskriminantın negatif veya pozitif olması, μ ve η katsayılarına bağlı olarak belirlenebilir. (4.14) ile verilen ifadeyi μ' nün derecelerine göre yazalım;

$$D(\mu) = (1+\eta\kappa)^2\mu^2 - 2(1+\eta^2\kappa^2)\mu + (1-\eta\kappa)^2\,. \tag{4.17}$$

(4.17) μ değişkenine göre kare formdur. μ^2' nin katsayısı $(1+\eta\kappa)^2 > 0$ dır ve bu formun diskriminantı,

$$d = 4[(1+\eta^2\kappa^2)^2 - (1-\eta^2\kappa^2)^2] = 16\eta^2\kappa^2 \geq 0. \tag{4.18}$$

Bunun içinde, (4.17) ile verilen kare formun grafiği şematik olarak aşağıdaki gibi çizilebilir (Şekil 4.1).



Şekil 4.1. (4.17) ile verilen kare formun şematik grafiği.

Şekil 4.1' den görüldüğü gibi,

$$D(\mu)\begin{cases} < 0 & , \quad \mu_1 < \mu < \mu_2 \\ = 0 & , \quad \mu = \mu_1 \ veya \ \mu = \mu_2 \\ > 0 & , \quad \mu > \mu_2 \ veya \ \mu < \mu_1 \end{cases}. \tag{4.19}$$

Şimdi (4.17) kare formunun köklerini bulalım;

$$\mu_{1,2} = \frac{2(1+\eta^2\kappa^2) \pm \sqrt{d}}{2(1+\eta\kappa)^2} = \frac{(1+\eta^2\kappa^2) \pm 2\eta\kappa}{(1+\eta\kappa)^2} \tag{4.20}$$

Buradan $\mu_1 = \frac{(1-\eta\kappa)^2}{(1+\eta\kappa)^2}$, $\mu_2 = 1$ köklerini buluruz. Bunun için de (18) ifadesini şu şekilde yazabiliriz;

$$D(\mu)\begin{cases} <0 & , \frac{(1-\eta\kappa)^2}{(1+\eta\kappa)^2} < \mu < 1 \\ =0 & , \mu = 1 \; veya \; \mu = \frac{(1-\eta\kappa)^2}{(1+\eta\kappa)^2} \\ >0 & , \mu > 1 \; veya \; \mu < \frac{(1-\eta\kappa)}{(1+\eta\kappa)} \end{cases}. \tag{4.21}$$

(4.21) ifadesinde kullanılan $\frac{(1-\eta\kappa)^2}{(1+\eta\kappa)^2}$ ifadesinde η öğrenme oranı, κ A matrisinin özdeğeridir $(\kappa_i > 0, i = 1,2,...,n)$. κ_i özdeğerleri A matrisinin verilmesiyle kesin olarak belirlenebilir. η öğrenme oranı ise değiştirilebilen bir parametredir. Böylece, verilen A matrisi için

$$S(\eta\kappa) = \frac{(1-\eta\kappa)^2}{(1+\eta\kappa)^2}, \tag{4.22}$$

η değişkeninin bir fonksiyonudur. Şimdi bu fonksiyonu inceleyelim. (4.22) denkleminin $(\eta\kappa)$ değişkenine göre 1. ve 2. dereceden türevlerini bulalım.

$$S'(\eta\kappa) = \left[\frac{(1-\eta\kappa)^2}{(1+\eta\kappa)^2}\right]' = -8\frac{(1-\eta\kappa)}{(1+\eta\kappa)^3} \tag{4.23}$$

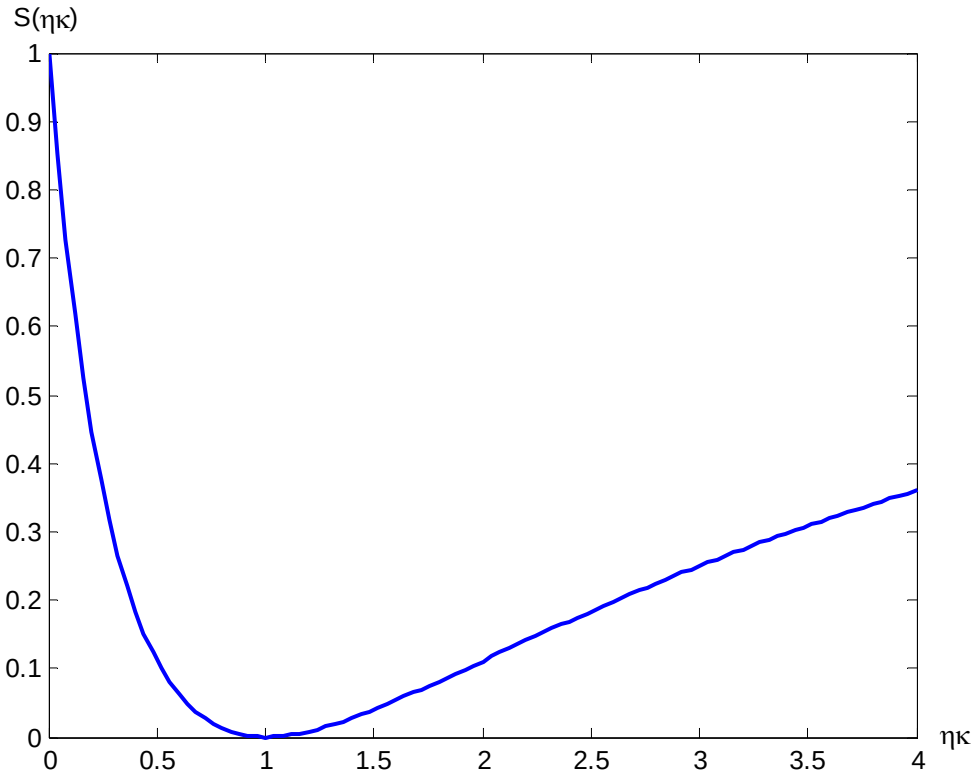
$$S''(\eta\kappa) = 16\frac{(2-\eta\kappa)}{(1+\eta\kappa)^4} \tag{4.24}$$

(4.23) ve (4.24) eşitliklerine dayanarak aşağıdaki ifadeler yazılabilir;

$$S'(\eta\kappa)\begin{cases} > 0 & , \eta\kappa > 1 \\ = 0 & , \eta\kappa = 1 \\ < 0 & , 0 \le \eta\kappa < 1 \end{cases} \tag{4.25}$$

$$S''(\eta\kappa)\begin{cases} < 0 & , \eta\kappa > 2 \\ = 0 & , \eta\kappa = 2 \\ > 0 & , 0 \le \eta\kappa < 2 \end{cases} \tag{4.26}$$

(4.25) ve (4.26) ifadeleri esasında, $S(\eta\kappa)$ fonksiyonu ve onun grafiği hakkında kesin fikir söyleyebiliriz. $S(\eta\kappa)$ fonksiyonu, $0 \le \eta\kappa \le 1$ aralığında 1 değerinden 0 değerine kadar azalır ve $\eta\kappa = 1$' de minimum '0' değerini alır. $\eta\kappa > 1$ aralığında ise bu fonksiyon artandır. $0 \le \eta\kappa < 2$ aralığında $S(\eta\kappa)$ fonksiyonu konveks, $(2, +\infty)$ aralığında ise konkavdır. $\eta\kappa = 2$, dönüm noktasıdır (Şekil 4.2).



Şekil 4.2. $S(\eta\kappa)$ fonksiyonunun grafiği.

(4.21) ifadesini şimdi şu şekilde de yazabiliriz;

$$D(\mu)\begin{cases} <0 & ,S(\eta\kappa)<\mu<1 \\ =0 & ,\mu=1 \; veya \; \mu=s(\eta\kappa) \\ >0 & ,\mu>1 \; veya \; \mu<S(\eta\kappa) \end{cases} \tag{4.27}$$

Teorem 4.1. η öğrenme oranı ve A matrisinin tüm κ_i , $(i=1,2,...,n)$ özdeğerleri için $0<\eta\kappa_i\le 2$ koşulu sağlandığında (4.4) eşitliği ile verilen momentumlu BP algoritması, $0<\mu<1$ aralığındaki herhangi μ momentum katsayısı için kararlıdır.

İspat. (4.4) veya (4.7) eşitliği ile verilen algoritmanın kararlılığını ispatlamak için, Q matrisinin özdeğerlerinin modunun 1' den küçük olduğunu göstermek gerekir. Lemma 3.1' e dayanarak, kompleks özdeğerler için bu koşulun sağlandığını kolaylıkla söyleyebiliriz. Geriye herhangi bir gerçel özdeğerin mutlak değerinin 1' den küçük olduğunu göstermek kalır.

(4.12) ile verilen kare denklemin sol kısmındaki kare fonksiyonu $\varphi(\lambda)$ ile belirtelim;

$$\varphi(\lambda)=\lambda^2-[(1+\mu)-(1-\mu)\eta\kappa]\lambda+\mu\,. \tag{4.28}$$

Burada κ sayısı A matrisinin herhangi bir özdeğeridir ($\kappa>0$). (4.28) ile verilen kare fonksiyonunu inceleyelim. Burada

$$\varphi(0)=\mu>0 \tag{4.29}$$

' dır ve $\varphi(\lambda)$ fonksiyonunun minimum noktası,

$$\lambda_{\min}=\frac{(1+\mu)-(1-\mu)\eta\kappa}{2} \tag{4.30}$$

olur. Minimum değeri ise

$$\varphi(\lambda_{\min}) = -\frac{D(\mu)}{4} \tag{4.31}$$

olur. (4.31) ile verilen eşitliğe dayanarak, $D(\mu) \geq 0$ durumunda $\varphi(\lambda_{\min}) \leq 0$ olduğu söylenebilir. $\eta\kappa$ sayısına göre λ özdeğerlerini inceleyelim;

a) $\eta\kappa = 1$ durumunda, $0 < \mu < 1$ için $D(\mu) < 0$' dır ve uygun λ özdeğerleri kompleks sayılardır. Lemma 4.1' e göre bu durumda $|\lambda| = \sqrt{\mu} < 1$.

b) $0 < \eta\kappa \leq 2$, $\eta\kappa \neq 1$ durumunda ise, $0 < \mu < 1$ katsayısının seçimine bağlı olarak, $D(\mu)$ sıfırdan küçük, sıfırdan büyük ve sıfır olabilir. Bu durumda (4.30) ifadesi ile tanımlanan $\lambda_{\min}$ sayısını değerlendirelim;

$$\frac{3\mu - 1}{2} < \lambda_{\min} = \frac{(1+\mu) - (1-\mu)\eta\kappa}{2} < \frac{1+\mu}{2} \tag{4.32}$$

olur. μ katsayısı (0,1) aralığında değiştiği için,

$$\frac{-1}{2} < \lambda_{\min} < 1 \tag{4.33}$$

olduğunu görürüz. Diğer yandan,

$$\varphi(1) = (1-\mu)\eta\kappa > 0 \text{ , (} 0 < \mu < 1 \text{ olduğu için)} \tag{4.34}$$

ve

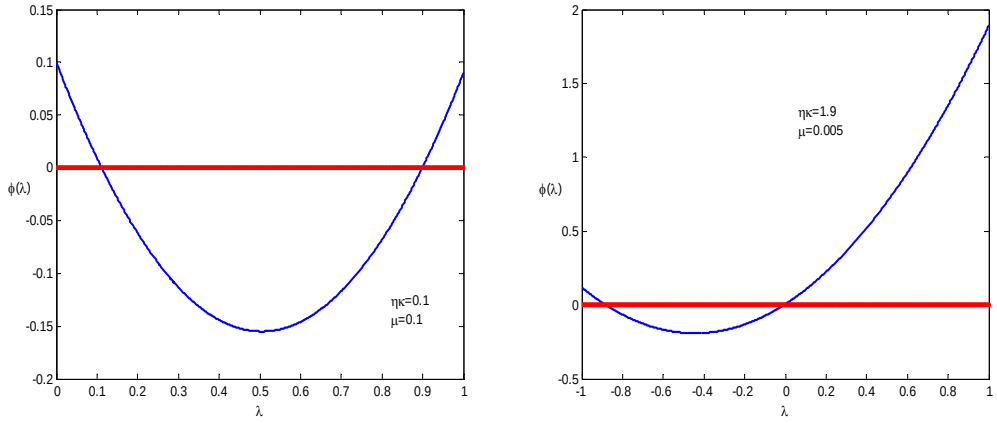
$$\varphi(-1) = 1 + (1+\mu) - (1-\mu)\eta\kappa + \mu = 2(1+\mu) - (1-\mu)\eta\kappa \tag{4.34}$$

(4.34)' ye göre $\varphi(-1)$, $0 < \eta\kappa < 2$ aralığında $(\eta\kappa)$' nın azalan fonksiyonudur ve $\eta\kappa = 2$ olduğunda,

$$\varphi(-1) = 2(1+\mu) - (1-\mu)2 = 4\mu > 0 \qquad (4.35)$$

olur. Buna göre de $0 < \eta\kappa \le 2$ koşulu sağlandığında herhangi $0 < \mu < 1$ katsayısı için $\varphi(-1) > 0$ olur.

$\varphi(0), \varphi(1), \varphi(-1) > 0$ ve $\frac{-1}{2} < \lambda_{\min} < 1$ değerlendirmelerine esasen, $D(\mu) > 0$ olduğu durumlarda $\varphi(\lambda)$ fonksiyonunun grafiği şematik olarak aşağıdaki iki şekilden birine benzer olur (Şekil 4.3).



Şekil 4.3. $\varphi(\lambda)$ fonksiyonunun şematik grafikleri.

Her iki durumda da $|\lambda| < 1$ olduğu görülebilir.■

Teorem 4.2. η öğrenme oranı ve A matrisinin $\kappa_i, i = 1,2,...,n$ özdeğerleri için $\eta\kappa_i > 2, i = 1,2,...,n$ koşulu sağlandığında (4.4) momentumlu BP algoritması $\frac{\eta\bar{\kappa} - 2}{\eta\bar{\kappa} + 2} < \mu < 1$ aralığındaki tüm μ momentum katsayıları için kararlıdır. Burada $\bar{\kappa} = \min_i \{\kappa_i\}$.

İspat Teoremde verilen koşullara dayanarak (4.12) denkleminin köklerinin modunun 1' den küçük olduğunu göstermeliyiz. Eğer A' nın bir κ özdeğeri için $D(\mu) < 0$ ise lemma 4.1' e göre bu özellik sağlanır. Buna göre de $D(\mu) > 0$ durumu incelenmelidir. Bu durumda (4.12) denkleminin uygun kökleri gerçel

sayılardır ve mutlak değerlerinin 1' den küçük olması için (4.30) formülü ile tanımlanan $\lambda_{\min} \in (-1,1)$ ve $\varphi(-1) > 0$ koşulları sağlanmalıdır. (4.34)' e göre

$$\varphi(-1) = (\eta\kappa + 2)\mu - (\eta\kappa - 2).$$

Buradan $\eta\kappa > 2$ olduğunda μ' nün

$$\mu > \frac{\eta\kappa - 2}{\eta\kappa + 2} \tag{4.36}$$

eşitsizliğini sağlayan değerlerinde $\varphi(-1) > 0$ olduğunu görürüz. Diğer taraftan, (4.21) formülü ile tanımlanan $D(\mu)$ için, $D(\mu) > 0$ koşulu sağlandığında $\mu < \frac{(\eta\kappa - 1)^2}{(\eta\kappa + 1)^2}$ olduğunu görürüz. Buradan $\frac{|\eta\kappa - 1|}{\eta\kappa + 1} < 1$ olduğu için

$$\mu < \frac{|\eta\kappa - 1|}{\eta\kappa + 1} \tag{4.37}$$

yazabiliriz. (4.30) ve (4.37) formüllerinden, $\eta\kappa > 1$ durumunda

$$\lambda_{\min} = \frac{1}{2}\left[(\eta\kappa + 1)\mu - (\eta\kappa - 1)\right] < 0 \tag{4.38}$$

sayısının negatif olduğunu bulunur. Diğer yandan (4.30) ve (4.36)'e göre

$0 < (\eta\kappa + 2)\mu - (\eta\kappa - 2) = [(\eta\kappa + 1)\mu - (\eta\kappa - 1)] + \mu + 1 = 2\lambda_{\min} + (\mu + 1)$. Buradan, μ "0" ve "1" arasında değerler alabileceğinden $\lambda_{\min} > -\frac{\mu + 1}{2} > -1$ olduğunu buluruz. Böylece $\lambda_{\min} \in (-1, 0)$ ve μ momentum katsayısı $\frac{\eta\kappa - 2}{\eta\kappa + 2} < \mu < 1$ aralığında değiştiğinde (4.4) iteratif süreci kararlı olur. ■

Not 4.1. 4.1 ve 4.2 teoremlerinin ispatlarına dikkat edilirse aşağıda belirtilen ifadelerin doğruluğu görülebilir:

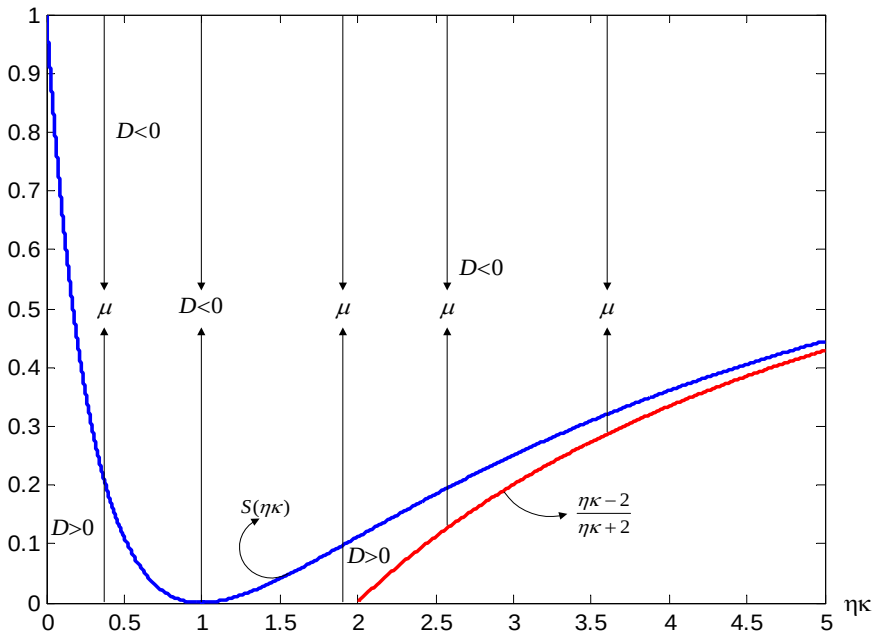
a) $0 < \eta\kappa < 1$ ve $D(\mu) > 0$ olduğunda (4.12) denklemini uygun kökleri $(0,1)$ aralığında yerleşir.

b) $1 < \eta\kappa \leq 2$ ve $D(\mu) > 0$ durumunda ise (4.12) denkleminin uygun kökleri $(-1,0)$ aralığında yerleşir.

4.1 ve 4.2 teoremleri birleştirilerek aşağıdaki şekilde ifade edilebilir:

Teorem 4.3. η öğrenme oranı ve simetrik pozitif tanımlı A matrisinin tüm $\kappa_i, i = 1,2,...,n$ özdeğerleri olduğunda (4.4) eşitliği ile verilen momentumlu BP algortiması $0 < \eta\kappa_i \leq 2$ koşulu sağlandığında (0,1) aralığındaki herhangi μ momentum faktörü, $\eta\kappa_i > 2$ koşulu sağlandığında ise $\max_i \frac{\eta\kappa_i - 2}{\eta\kappa_i + 2} < \mu < 1$ aralığında olan μ momentum faktörleri için kararlıdır.

Şekil 4.4' de $\eta\kappa$' nın değerine karşılık μ momentum faktörünün (4.4) sürecinin kararlılığını sağlayan değişim aralıkları geometrik olarak açık olarak gösterilmektedir.



Şekil 4.4. $S(\eta\kappa)$ ve $\frac{\eta\kappa - 2}{\eta\kappa + 2}$ fonksiyonlarının grafikleri ve geçerli μ aralıkları.

A matrisinin $\kappa_i, i = 1,2,...,n$ özdeğerlerinin büyükten küçüğe doğru sıralandığını varsayalım;

$$\kappa_1 \geq \kappa_2 \geq ... \geq \kappa_{n-1} \geq \kappa_n \tag{4.39}$$

Bu durumda $\eta\kappa_1 = \max_i \eta\kappa_i$ ve $\eta\kappa_n = \min_i \eta\kappa_i$ olur. Buna göre de $\eta\kappa_1 \leq 2$ ise $\mu \in (0,1)$ ve $\eta\kappa_1 > 2$ ise $\mu \in \left(\frac{\eta\kappa_{k-1} - 2}{\eta\kappa_{k-1} + 2}, 1 \right)$ aralığında değiştiğinde (4.4) algoritması kararlı olur.

4.2. Öğrenme Oranı ve Momentum Faktörünün Dinamik Seçimi ve Eşlenik Gradyan Metodu

Bir önceki alt bölümde (4.1) kuadratik hata fonksiyonu için (4.4) denklemiyle verilen momentumlu BP algoritmasında η-öğrenme oranı ve μ-momentum katsayısı iterasyon süreci boyunca sabit olarak ele alınmaktaydı. Şimdi Amit, (2004) makalesi göz önüne alarak aşağıdaki BPM algoritmasını ele alalım:

$$x_{k+1} - x_k = \mu_k (x_k - x_{k-1}) - (1 - \mu_k)\eta_k \nabla F(x_k)\,. \tag{4.41}$$

Burada η_k ve μ_k, sırasıyla k. adımdaki öğrenme ve momentum katsayıları, $\nabla F(x_k)$ kuadratik F fonksiyonunun x_k noktasındaki gradyanıdır; $\nabla F(x_k) = Ax_k - b = -r_k$. r_k k. adımdaki artıktır.

(4.41) denklemi (4.4)' e benzemektedir. Farkları ise (4.41)' de öğrenme ve momentum katsayılarının her adımda değişmesidir. Tabii ki, katsayıların dinamik olarak uygun bir şekilde seçimi algoritmanın yakınsama hızını çok yükseltebilir. Bu bakımdan eşlenik gradyan (CG) yöntemleri optimizasyon algoritmaları arasında belirli avantajlara sahiptir. (4.41) algoritmasının yanı sıra, eşlenik gradyan algoritmasını ele alalım. Bu amaçla önce

$$x_{k+1} = x_k + \alpha_k r_k , \qquad r_k = b - Ax_k \tag{4.42}$$

gradyan düşümü formülünü yazalım. Yeni bir algoritmada, r_k-düşüm vektörüne bir önceki $x_k - x_{k-1}$ değişiminin aşağıdaki şekilde verildiğini varsayalım:

$$x_{k+1} - x_k = \alpha_k \left[r_k + \gamma_k (x_k - x_{k-1}) \right] \tag{4.43}$$

(4.43) denkleminde $\Delta x_k = x_{k+1} - x_k$ vektörü r_k ve $\Delta x_{k-1} = x_k - x_{k-1}$ vektörlerinin bir doğrusal kombinasyonu gibi düşünülebilir. (4.43) formülü şöyle yazılabilir:

$$x_{k+1} = x_k + \alpha_k p_k \tag{4.44}$$

burada p_k aşağıdaki şekilde tanımlanır.

$$p_k = r_k + \gamma_k (x_k - x_{k-1}) = r_k + \gamma_k \alpha_{k-1} p_{k-1} \tag{4.45}$$

$$= r_k + \beta_{k-1} p_{k-1} , \tag{4.46}$$

burada $p_{k-1} := \gamma_k \alpha_{k-1}$. Bu formüller birleştirilerek aşağıdaki şekilde yazılabilir:

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k p_k \\ r_{k+1} &= r_k - \alpha_k A p_k \\ p_{k+1} &= r_{k+1} + \beta_k p_k \quad , \beta_k = \gamma_{k+1} \alpha_k \end{aligned} \tag{4.47}$$

(4.47)' deki $r_{k+1} = r_k - \alpha_k A p_k$ formülü aşağıdaki şekilde açıklanabilir:

$$r_{k+1} = b - Ax_{k+1} = b - A(x_k + \alpha_k p_k) = (b - Ax_k) - \alpha_k A p_k = r_k - \alpha_k A p_k \tag{4.48}$$

(4.47) formülleri standart CG formülleridir (Greenbaum 1997). Burada geçen α_k ve p_k katsayıları, CG algoritmasını bölüm 3' de açıklandığı gibi aşağıdaki şekilde hesaplanırlar (Greenbaum 1997; Nocedal ve Wright 1999).

$$\alpha_k = \frac{(r_k, r_k)}{(p_k, Ap_k)}, \quad \beta_k = \frac{(r_{k+1}, r_{k+1})}{(r_k, r_k)} \tag{4.49}$$

(Bhaya ve Kaszkurewicz, 2002) makalesinde (4.47)' de son iki satırın oluşturduğu

$$\begin{cases} r_{k+1} = r_k - \alpha_k A p_k \\ p_{k+1} = r_{k+1} + \beta_k p_k \end{cases} \tag{4.50}$$

bilineer sistemi için α_k ve β_k katsayıları, kontrol teorisine uygun olarak, r_k ve p_k' nın geri beslemeli bir fonksiyonu gibi ele alınmış ve sonuç olarak bu katsayılar aşağıdaki şekilde bulunmuştur:

$$\alpha_k = \frac{(r_k, p_k)}{(Ap_k, p_k)}, \quad \beta_k = -\frac{(p_k, Ar_{k+1})}{(p_k, Ap_k)} \tag{4.51}$$

(4.51) ve (4.49) formülleri denk formüllerdir (Greenbaum 1997). Bhaya ve Kasckurewicz (2002) makalesinde katsayıların seçimi için kontrol teorisi yaklaşımının uygulanabilirliği önerilmiştir.

Şimdi (4.41) BPM ve (4.43) CG formüllerini karşılaştırarak η_k, μ_k ve α_k, β_k parametreleri arasındaki denklik ilişkilerini yazalım. (4.41) ve (4.43) ifadeleri sırasıyla şu şekilde yazılabilir:

$$\begin{aligned} x_{k+1} - x_k &= \mu_k (x_k - x_{k-1}) + (1 - \mu_k)\eta_k r_k \\ x_{k+1} - x_k &= \alpha_k \gamma_k (x_k - x_{k-1}) + \alpha_k r_k \end{aligned} \tag{4.52}$$

Buradan

$$\alpha_k = (1 - \mu_k)\eta_k \qquad \text{ve} \qquad \alpha_k \gamma_k = \mu_k \tag{4.53}$$

olduğunu buluruz. (4.53) eşitliklerinden μ_k ve η_k katsayıları için

$$\mu_k = \frac{\alpha_k}{\alpha_{k-1}}\beta_{k-1} \qquad \text{ve} \qquad \eta_k = \frac{\alpha_k \alpha_{k-1}}{\alpha_{k-1} - \alpha_k \beta_{k-1}} \tag{4.54}$$

formülleri bulunur. (4.54) formülleri ile belirlenen μ_k momentum ve η_k öğrenme oranı katsayıları (4.41) algoritmasının CG algoritması gibi çalışmasını sağlıyor. Yukarıdaki açıklamalar ışığında Teorem 4.4 söylenebilir.

Teorem 4.4 Bhaya (2004). $f(x) = \frac{1}{2}x^T Ax - b^T x + c$ kuadratik hata fonksiyonunun minimalleştirilmesi için η_k öğrenme oranı ve μ_k momentum faktörlü

$$x_{k+1} - x_k = \mu_k (x_k - x_{k-1}) + (1 - \mu_k)\eta_k r_k \tag{4.55}$$

geriye yayılım BP algoritmasını ele alalım, burada A simetrik pozitif tanımlı matris ve $-r_k = \nabla f(x_k) = Ax_k - b$ gradyandır ($Ax_k = b$ denkleminin çözümü bakımından artıktır). Ayrıca, α_k ve β_k parametreli

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k p_k \\ r_{k+1} &= r_k - \alpha_k A p_k \\ p_{k+1} &= r_{k+1} + \beta_k p_k \quad , \beta_k = \gamma_{k+1}\alpha_k \end{aligned} \tag{4.56}$$

CG metodunu ele alalım, burada

$$\alpha_k = \frac{(r_k, r_k)}{(p_k, Ap_k)}, \quad \beta_k = \frac{(r_{k+1}, r_{k+1})}{(r_k, r_k)} \tag{4.57}$$

Bu durumda öğrenme oranı ve momentum faktörü dinamik olarak aşağıdaki şekilde seçilebilir.

$$\mu_k = \frac{\alpha_k}{\alpha_{k-1}}\beta_{k-1} \qquad \lambda_k = \frac{\alpha_k \alpha_{k-1}}{\alpha_{k-1} - \alpha_k \beta_{k-1}} \tag{4.58}$$

ve başlangıç koşulları uygun seçildiklerinde her iki metot aynı x_k ve r_k vektörlerini üretir. η_k ve μ_k parametrelerinin bu şekilde seçilmesiyle, her iterasyonda r_k ve p_k vektörlerinin 2-normunun azalması anlamında, BPM metodunun optimum düzeltildiği söylenir.

(4.53) ve (4.54) ifadeleri optimum öğrenme oranı ve momentum faktörünün (r, p) durum değişkenlerinin fonksiyonu gibi hesaplandığını gösterir. (4.54) formüllerinde daha çok iç çarpım işlemleri yapılmaktadır. Buna göre de "optimal düzeltilmiş" BPM algoritması yerine ona denk olan CG algoritmasının uygulanması önerilir (Amit 2004). Ayrıca CG algoritmasının farklı hızlandırılmış versiyonlarının da olduğu bilinmektedir. (bkz. bölüm 3)

(4.54) formülündeki μ_k' nın ifadesi ilk olarak Yu&Chen (1997) makalesinde optimum momentum faktörü olarak bulunmuştur. Uygun η_k ifadesi ise ilk olarak Amit (2004) de yazılmıştır.

4.3. Tutucu Kuvvet Alanı İçinde Newton Partiküllerinin Hareketi ve Momentum. Yakınsama Hızı.

Qian (1999) makalesinde gradyan düşümü algoritmasındaki momentum etkeniyle bir tutucu kuvvet alanı içinde hareket eden Newton partikülleri arasındaki benzerliği gösterilmiştir.

Bir m nokta-kütlenin $E(w)$ potansiyel enerjisine sahip bir tutucu kuvvet alanı etkisi altında v sürtünme katsayılı yapışkan ortamda hareketinin Newton denklemini yazalım:

$$m\frac{\partial^2 w}{\partial t} + v\frac{\partial w}{\partial t} = -\nabla_w E(w) \tag{4.59}$$

Burada m-kütle, w-parçacığın koordinat vektörü, $\nabla_w E(w)$ - E potansiyel enerji fonksiyonunun w-değişkenine göre gradyanıdır.

Eğer $m = 0$ olursa, (4.59) denklemi basit gradyan düşümü denklemine dönüşür:

$$\frac{\partial w}{\partial t} = -\eta \nabla_w E(w), \quad \eta = \frac{1}{v} \tag{4.60}$$

(4.59) denklemini kesikli durumda aşağıdaki şekilde yazılabilir:

$$m\frac{w(t+\Delta t) - w(t-\Delta t)}{\Delta t^2} + \frac{w(t+\Delta t) - w(t)}{\Delta t} = -\nabla_w E(w) \tag{4.61}$$

(4.61) denklemini $w_{t+\Delta t}$, w_t, $w_{t-\Delta t}$ terimlerine göre yazılırsa:

$$w_{t+\Delta t} - w_t = -\frac{(\Delta t)^2}{(m + v\Delta t)} \nabla_w E(w) + \frac{m}{m + v\Delta t}(w_t - w_{t-\Delta t}) \tag{4.62}$$

veya

$$w_{t+1} - w_t = -\frac{1}{m+v} \nabla E(w) + \frac{m}{m+v}(w_t - w_{t-1}) \tag{4.63}$$

(4.62) ifadesinde, $\eta = \frac{1}{v}$, $\mu = \frac{m}{m+v}$ eşitlikleri kullanılarak aşağıdaki şekilde yazılabilir:

$$w_{t+1} - w_t = -(1-\mu)\eta \nabla E(w) + \mu(w_t - w_{t-1}) \tag{4.63}$$

(4.63) momentumlu gradyan düşümü iteratif sürecinin aynısıdır. Buna göre de momentumlu gardyan düşümü tutucu kuvvet alanı etkisi altında yapışkan bir ortamdaki Newton partikülünün hareketine denktir.

Burada momentum etkeni kütle rolünü oynar. Çünkü $m = 0$ ise $\mu = 0$ olur, ve tersine (4.63) momentumlu gradyan düşümü ifadesinde $E(w)$ potansiyel enerjisi hata fonksiyonu olarak algılanabilir. Öyle ki, w-hareket vektörünün seçimi $E(w)$ fonksiyonunun minimalleştirilmesi problemiyle bağlanmış olur. Kararlılık durumunda w ağırlık vektörünün değişimi (veya denk olarak, partikülün hızı) sıfıra yaklaşmış olur.

w_0' ın $E(w)$ fonksiyonunun lokal minimumu olduğunu varsayalım ($\nabla_w E(w) = 0$). Bu durumda w_0' ın yakın komşuluğunda

$$\nabla E(w) \approx \nabla E(w_0) + H(w - w_0) = H(w - w_0) \tag{4.64}$$

yazabiliriz, burada H simetrik ve pozitif tanımlı Hessian matrisidir;

$$H = (h_{ij}) \ , \ h_{ij} = \left. \frac{\partial^2 E(w)}{\partial w_i \partial w_j} \right|_{w_0} \tag{4.65}$$

(4.64)' yı (4.63) ifadesinde göz önüne alarak, ($w_0 = 0$) lokal minimumu civarında (4.63) denklemi aşağıdaki şekilde yazılabilir:

$$w_{t+1} - w_t = -(1-\mu)\eta H w_t + \mu(w_t - w_{t-1}) \tag{4.66}$$

veya

$$w_{t+1} = [(1+\mu)I - (1-\mu)\eta H]w_t - \mu w_{t-1} \tag{4.67}$$

H simetrik ve pozitif tanımlı matris olduğundan belirli bir Q ortogonal matrisinin yardımıyla (Q matrisi H' ın ortonormal özvektörlerinden oluşturulan bir matristir) köşegenleştirilebilir:

$$H = QKQ^T \quad , \quad QQ^T = I \tag{4.68}$$

burada K matrisi H' ın özdeğerlerinde oluşan köşegen bir matristir;

$$K = \begin{pmatrix} \kappa_1 & & & 0 \\ & \kappa_2 & & \\ & & . & \\ & & & . \\ 0 & & & \kappa_n \end{pmatrix} \quad , \quad (\kappa_i > 0) \tag{4.69}$$

$w' = Q^T w$ dönüşümünü uygulamakla, (4.67) denklemini

$$w'_{t+1} = [(1+\mu)I - (1-\mu)\eta K]w'_t - \mu w'_{t-1} \tag{4.70}$$

veya koordinatlarla yazılırsa

$$w'_{i,t+1} = [1+\mu-(1-\mu)\eta\kappa_i]w'_{i,t} - \mu w'_{i,t-1} \quad , \quad i = \overline{1,n} \tag{4.71}$$

$w'_{i,t} = w'_{i,t}$ yapay denklemini dahil etmekle, (4.71) denklemini matris formunda yazabiliriz:

$$\tilde{w}_{i,t+1} = Q w'_{i,t} \quad , \quad \tilde{w}'_{i,t} = \begin{pmatrix} w'_{i,t-1} \\ w'_{i,t} \end{pmatrix} \tag{4.72}$$

burada $Q = \begin{pmatrix} 0 & 1 \\ -\mu & 1+\mu-(1-\mu)\eta\kappa_i \end{pmatrix}$ şeklinde 2×2 boyutlu bir matristir. Bu matrisin karakteristik denklemini yazıp, özdeğerlerinin bulunması için uygun denklemi bulalım:

$$|Q - \lambda I| = \begin{vmatrix} -\lambda & 1 \\ -\mu & 1+\mu-(1-\mu)\kappa_i - \lambda \end{vmatrix} = 0 \tag{4.73}$$

(4.73) denkleminin kökleri ise şu şekilde hesaplanır:

$$\lambda_{1,2} = \frac{(1+\mu-(1-\mu)\eta\kappa_i) \pm \sqrt{(1+\mu-(1-\mu)\eta\kappa_i) - 4p}}{2} \tag{4.74}$$

(4.73) denklemi (4.12) denkleminin aynısıdır ve bu denklemlerin köklerini hesaplayan (4.74) ve (4.13) formülleri aynı formüllerdir.

Şimdi Qian (1999) makalesinde bulunan sonuçları ve formülleri yorumlayalım.

Bu bölümde açıkladığımız gibi (4.63) momentumlu iterasyon sürecinin kararlılığı için (4.73) denkleminin (4.74) formülü ile verilen $\lambda_{i,1}, \lambda_{i,2}$ $i=\overline{1,n}$ köklerinin modları 1' den küçük olmalıdır; $\max(|\lambda_{i,1}|, |\lambda_{i,2}|) < 1$.

Sonuç 3. Qian (1999). (4.71) denklemi ile ifade edilen sistem ancak ve ancak

$$-1 < p < 1 \;\; ve \;\; 0 < \varepsilon\kappa_i < 2 + 2p \tag{4.75}$$

olduğunda yakınsaktır. Burada p bizim işaretlerde μ' ye, ε ise $(1-\mu)\eta$' ya uygun gelmektedir. μ momentum katsayısının (0,1) aralığında değerler almasının avantajı hakkında yukarıda tartışılmıştı. Buna göre de (4.75) ifadesinde μ, $0 < \mu < 1$ aralığından seçilebilir. Şimdi (4.75) ifadesinin bizim işaretlerle uygun dönüşümünü yapalım. (4.75)' de 2. eşitsizlik $0 < \mu < 1$ olduğunda şöyle yazılabilir:

$$0 < (1-\mu)\eta\kappa_i < 2 + 2\mu \tag{4.76}$$

Buradan $0 < \eta\kappa_i < \dfrac{2+2\mu}{1-\mu}$ veya

$$\mu > \frac{\eta\kappa_i - 2}{\eta\kappa_i + 2}, \quad i = \overline{1,n} \tag{4.77}$$

alırız. (4.77) koşulu Teorem 4.2' de (4.4) sisteminin kararlılığı için μ-momentum katsayısının sağladığı koşulun aynısıdır.

(4.77) ifadesinde $\eta\kappa_i \le 2$ ise μ parametresi için alt sınır sıfır olur ve $0 < \mu < 1$ koşulunu sağlamalıdır. (4.77)' da $\eta\kappa_i > 2$ durumunda μ parametresi için alt sınır

$\max_i \frac{\eta\kappa_i - 2}{\eta\kappa_i + 2}$ olur. Bu durumda (4.71) sisteminin "yakınsak" olması için $\frac{\eta\kappa_i - 2}{\eta\kappa_i + 2} < \mu < 1$ koşulunu sağlamalıdır.

Böylelikle, Qian (1999) makalesindeki Sonuç 3 aşağıdaki şekilde yazılabilir.

Sonuç 3. (4.71) denklemi ile ifade edilen sistem ancak ve ancak $\frac{\eta\kappa_i - 2}{\eta\kappa_i + 2} < \mu < 1$, $i = 1, 2, ..., n$ koşulu sağlandığında kararlıdır.

Qian (1999)' da (35) formülü ile en iyi yakınsamayı sağlayan durum açıklanmıştır. Bu $\lambda_{i,1} = \lambda_{i,2}$ gerçel köklerinin eşit olduğu durumdur, yani (4.73) kare denkleminin diskriminantının sıfıra eşit olduğu durumdur.

Qian (1999) makalesinde (35) formülü aşağıdaki şekildedir:

$$p = (1 - \sqrt{\varepsilon\kappa_i})^2 \tag{4.78}$$

Bizim işaretlerde $\mu = p$, $\varepsilon = (1-\mu)\eta$ olduğunu göz önüne alarak (4.78) ifadesinin dönüşümünü yapalım.

$$\mu = \left(1 - \sqrt{(1-\mu)\eta\kappa_i}\right)^2 \tag{4.79}$$

Bu ifadeden sonuç olarak μ' yü bulabiliriz:

$$\mu = \frac{(1-\eta\kappa_i)^2}{(1+\eta\kappa_i)^2} \tag{4.80}$$

(4.80) formülündeki $\frac{(1-\eta\kappa_i)^2}{(1+\eta\kappa_i)^2}$ ifadesi (4.22) ifadesinin aynısıdır, yani $S(\eta\kappa)$ fonksiyonudur. (4.21)' e göre (4.80) sağlandığında $D(\mu) = 0$, yani (4.12) veya (4.73) kare denkleminin diskriminantı sıfır olur ve $\mu = S(\eta\kappa)$ değerini alır.

Böylelikle aşağıdaki sonuç söylenebilir:

η öğrenme oranı ve κ_i özdeğeri için μ momentum katsayısı (4.80) eşitliğini sağlıyorsa (4.71) denkleminin i. Bileşenine uygun W_i' değişkenine göre en hızlı yakınsama gerçekleşmiş olur.

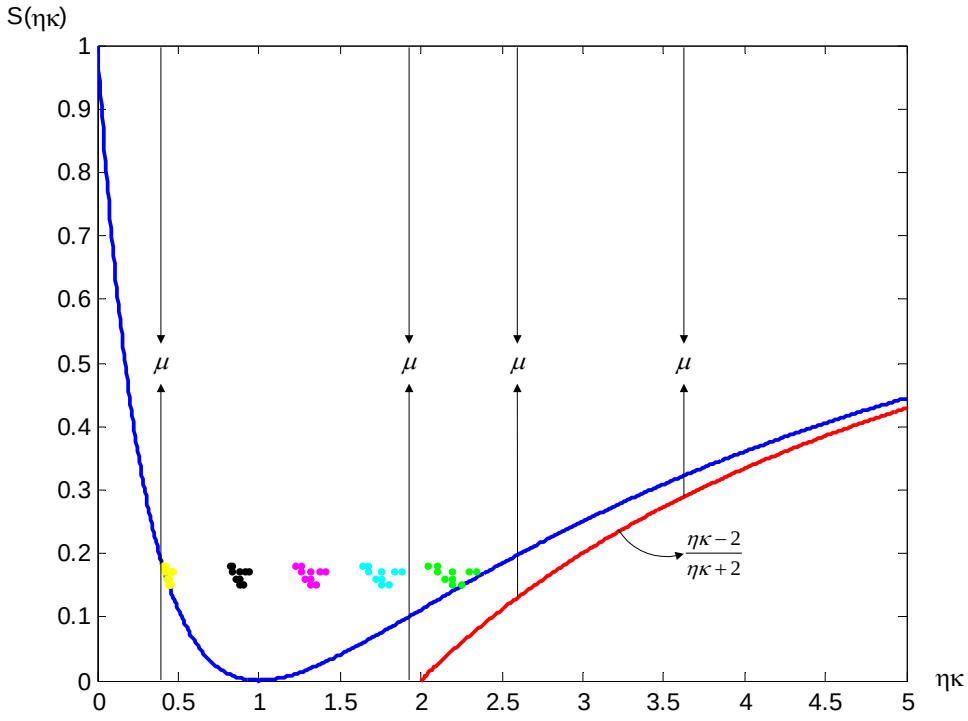
Bu sonuca dayanarak, ayrı ayrı w_i' , $i = 1,...,n$ bileşenlerine uygun adım adım μ_i' lerin seçimi fikri ortaya çıkabilir. Ancak bizim amacımız (4.67) sisteminin hızlı yakınsamasını gerçekleştirmektir. Bu durumda $w = Q^{T-1}w'$ dönüşümü w_i bileşeninin w_j'' in lineer kombinasyonu olduğunu gösterir. Bu nedenle (4.80) formülüne uygun μ seçimi mümkün değildir.

4.4. Deneyler

Sabit Parametreli Momentumlu Gradyan Düşümü. Teorem 4.1' de sunulan önerilerin uygulamadaki geçerliliğini göstermek için keyfi problemler oluşturmak gerekir. (4.4) algoritmasında keyfi simetrik pozitif tanımlı A matrisi ve b vektörü üretilerek çeşitli problemler oluşturulabilir. $A = K\Lambda K^T$, $KK^T = I$, eşitliği kullanılarak rassal A matrisleri elde edilebilir. Burada K ortogonal matris, Λ ise A matrisinin özdeğerlerini içeren köşegen matristir. Çalışmadaki deneylerde 5 boyutlu problemler ele alınmıştır. Uniform dağılımdan rasgele sayılar üretilerek 5 farklı A matrisi ve b vektörü oluşturulmuştur. $\eta = \{0.01, 0.02, ..., 0.99\}$ ve $\mu = \{0.01, 0.02, ..., 0.99\}$ olacak şekilde seçilmiş ve (4.4) iterasyonu tüm kombinasyonlar için çalıştırılmıştır. Her farklı A matrisi ve b vektörü için 9801 deneme yapılmıştır. Bu denemelerden elde edilen en iyi η ve μ seçimleri aşağıdaki tabloda özetlenmiştir.

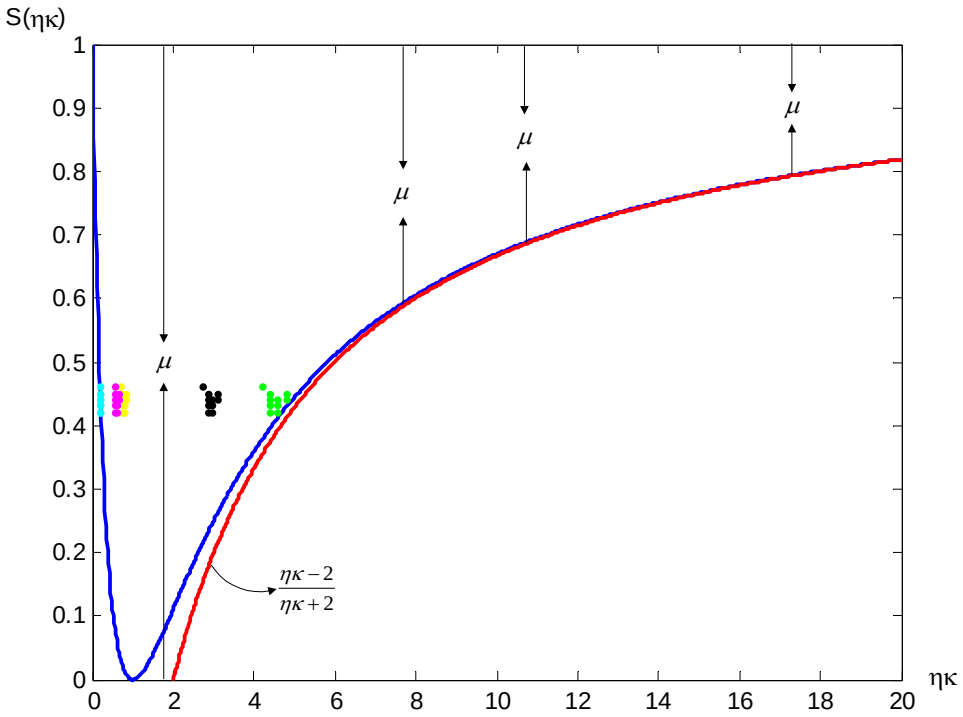
Çizelge 4.1. Farklı özdeğer dağılımına sahip Hessian matrisli problemler için optimum sabit parametre seçimleri.

No	A matrisinin özdeğerleri (κ_i)	Öğrenme oranı	Momentum katsayısı	İterasyon sayısı
1	{1, 2, 3, 4, 5}	0.44	0.16	19
2	{0.9, 2.7, 3.5, 13, 20}	0.24	0.45	40
3	{0.1, 0.2, 0.3, 0.6, 0.9}	0.99	0.68	77
4	{0.5, 12, 53, 105, 300}	0.08	0.85	246
5	{0.1130, 0.2610, 0.4453, 0.5162, 34}	0.51	0.80	137



Şekil 4.5. 1 nolu matris için en iyi η ve μ değerleri dağılımı.

Çizelge 4.1' deki 1 nolu matris için en iyi η ve μ seçimleri için $\eta\kappa_i$ değerleri ve bu değerlere karşılık gelen μ katsayılarının dağılımı Şekil 4.5' de gösterilmiştir. Şekildeki her bir renk farklı özdeğere karşılık gelen $\eta\kappa_i$ ve μ değerlerini göstermektedir. Şekilden de görüldüğü üzere $\eta\kappa_i$ değerlerinin ve bu değerlere karşılık gelen μ katsayısı seçimlerinin Teorem 4.3' de belirtilen sınırlar içerisinde olduğu görünmektedir.

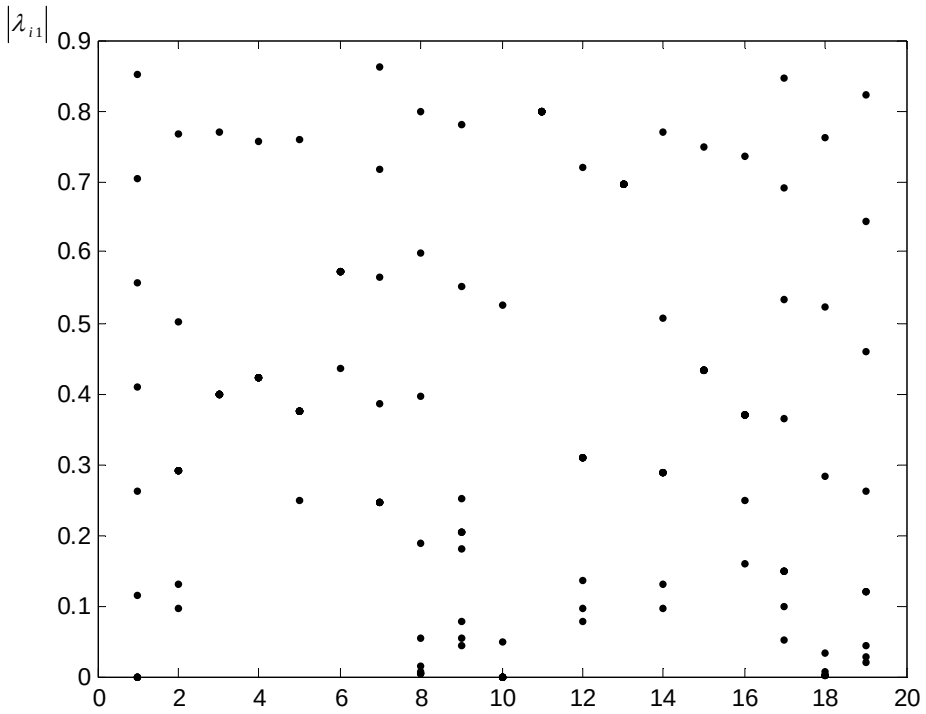


Şekil 4.6. 2 nolu matris için en iyi η ve μ değerleri dağılımı.

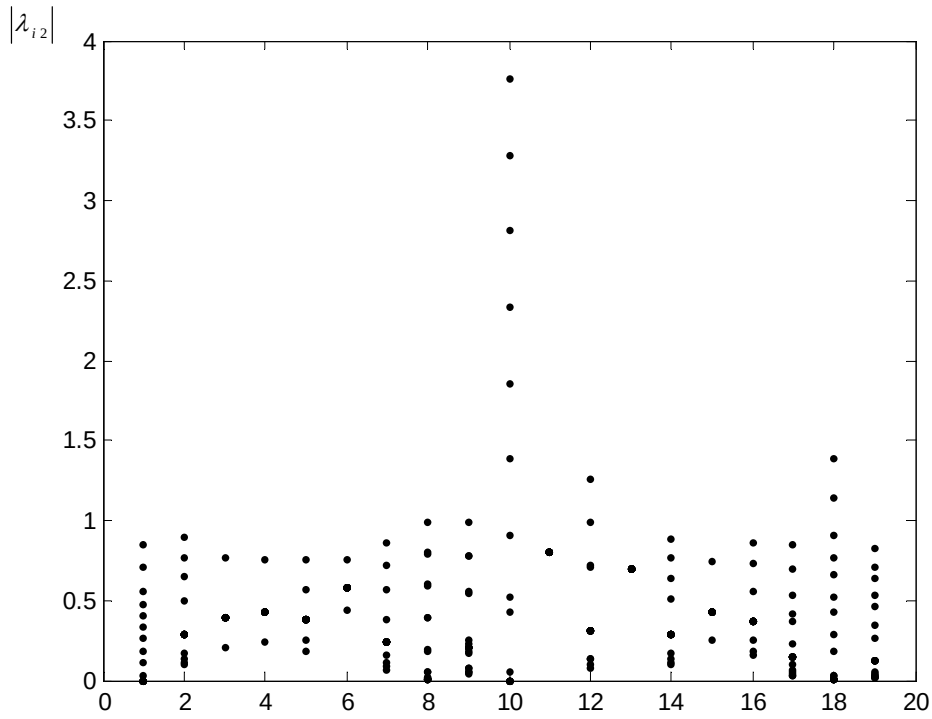
Çizelge 4.1' deki 2 nolu matris için en iyi η ve μ seçimlerinde uygun $\eta\kappa_i$ değerleri ve bu değerlere karşılık gelen μ katsayılarının dağılımı Şekil 4.6.' da gösterilmiştir. κ_i değerleri büyüdükçe algoritmanın kararlı olduğu μ katsayısının seçim aralığı daralmaktadır. Bu da μ momentum katsayısının seçimini daha duyarlı hale getirir. Denemelerde farklı K ortogonal matrislerinin kullanımı algoritmanın yakınsama özelliklerini etkilemediği görülmüştür. Fakat özdeğer dağılımını belirleyen Λ matrisi algoritmanın yakınsama özelliklerini ve katsayı seçimlerini önemli derecede etkilemektedir.

Öğrenme Oranı ve Momentum Katsayısının Dinamik Seçilmesi:
Denemelerde 10×10 boyutlu farklı özdeğer dağılımına sahip simetrik pozitif tanımlı A matrisleri oluşturulmuştur. (4.57) ve (4.58) formüllerinin yardımıyla k. Adımdaki η_k, μ_k parametreleri hesaplanmış ve her bir κ_i özdeğeri için uygun $|\lambda_{i1}^k|$, $|\lambda_{i2}^k|$ modları bulunmuştur. Şekil 4.7 ve 4.8' de modların iterasyon süreci

boyunca dağılımı gösterilmiştir. λ_{i1} ve λ_{i2} özdeğerlerinin modunun tüm κ_i 'ler için süreç ilerledikçe belirli bir değer etrafında toplandığı görünür. Hatta sürecin bazı adımlarında λ_{i1} ve λ_{i2} özdeğerlerinin modları eşit olur ve bu durumda kökler kompleks sayılardır ve modları $\sqrt{\mu}$ 'ye eşittir.Qian (1999) makalesinde gradyan düşüm eğitim algoritmasındaki momentum parametresi ile tutucu bir kuvvet alanı altında yapışkan bir ortamda hareket eden Newton partiküllerinin kütlesi arasında benzerlik gösterilmiştir. Bu çalışmadaki sonuçlar, Qian (1999)' da momentum teriminin mantıklı aralıktan seçildiğinde, sistemdeki çoğu öz bileşeni kritik sönümlenme noktasına yakın bir konuma getirerek yakınsamayı hızlandırdığı yorumuna uygundur.



Şekil 4.7. Dinamik seçimli iterasyon için örnek λ_{i1} modları dağılımı.



Şekil 4.8. Dinamik seçimli iterasyon için örnek λ_{i2} modları dağılımı.

5. SONUÇ VE TARTIŞMA

Geriye yayılım yapay sinir ağı eğitim algoritmaları tahmin ve sınıflandırma problemlerinde istatistik yöntemlere alternatif olarak önerilmektedirler.Geriye yayılım, sinir ağının ağırlıklarının fonksiyonu olan hata kareler toplamının minimumunun bulunmasına yönelik gradyan düşümü optimizasyon metodudur. Momentumlu geriye yayılım algoritmasının kararlılığı ve yakınsama hızı gibi önemli özellikleri, öğrenme ve momentum katsayılarının seçimine bağlı olarak değişim göstermektedir. Momentum katsayısının gradyan düşüm algoritmasına dahil edilmesi, algoritmanın bu özelliklerini geliştirmektedir.

Genelde gizli katmana sahip çok katmanlı yapay sinir ağlarında hata kareler fonksiyonu doğrusal olmayan bir fonksiyondur. Ancak her adımda uygun bir nokta komşuluğunda bu fonksiyon yaklaşık olarak kuadratik bir fonksiyondur. Bu durumda, hata fonksiyonunun uygun iterasyon noktasındaki Hessian matrisi kullanılır. Bu nedenle çalışmada kuadratik hata kareler fonksiyonunun minimizasyonu probleminde momentumlu gradyan azalan algoritmasının yakınsak olduğu momentum ve öğrenme katsayıları sınırları türetilmiştir (Teorem 4.1-4.3). Elde edilen sonuçlar algoritmaya uyarlanarak, optimum parametre seçimlerinin dağılımı gözlemlenmiştir.

Öncelikle yapay sinir ağlarında öğrenme problemi ifade edilmiş, öğrenme oranı ve momentum parametrelerinin sabit olduğu durumda algoritmaya momentum teriminin eklenmesi, diğer çalışmalardan farklı bir yaklaşımla teoride incelenmiş ve yakınsamanın sağlanması için gerekli koşullar verilmiştir. Simülasyon çalışmasında değişik karakterli rassal problemler oluşturulmuş ve algoritmanın yakınsama özellikleri gözlemlenmiştir. Bu gözlemler iki önemli noktada yoğunlaşır; algoritmanın bir çözüme yakınsaması ve yakınsama oranı.

Parametrelerin “optimal düzeltilmiş” dinamik seçimli olduğu durumda ise algoritmanın eşlenik gradyan algoritmasına denk olduğu gösterilmiştir. Eşlenik gradyan algoritması ayrıntılı olarak ele alınmış ve momentumlu geriye yayılım algoritmasıyla karşılaştırılmıştır. Güncel bir sınıflandırma problemi üzerinde de geriye yayılım eğitim algoritmasının, hem eşlenik gradyan metoduna dayanan hem de gradyan azalana dayanan farklı türlerinin yakınsama performansları, yine bir simülasyon çalışması ile değerlendirilmiştir.

Simülasyon çalışmasında nonobstrüktif azoospermi (ejakulatta sperm bulunmaması) ‘ ye sahip erkeklerde testis biopsisine dayanan spermatozoa (sperm hücresi) tahmini için bir y.s.a. geliştirilmiştir. Eğitim algoritmaların performansları değerlendirilirken yakınsama süreleri, devir sayıları ve doğru sınıflandırma oranları dikkate alınmıştır.

Parametrelerin dinamik olarak seçildiği durumda algoritmanın süreç boyunca önemli bileşenleri incelenmiş ve Qian (1999)’ da geçen “momentum teriminin mantıklı aralıktan seçildiğinde, sistemdeki çoğu öz bileşeni kritik sönümlenme noktasına yakın bir konuma getirerek yakınsamayı hızlandırdığı” yorumuna paralel sonuçlar elde edilmiştir.

Bu çalışmadaki denemelerde keyfi rassal problemler üretilerek simülasyonlar gerçekleştirilmiştir (örneğin simetrik pozitif tanımlı Hessian matrisinin rassal olarak oluşturulması). Pratikte Hessian matrisini hesaplanması mümkün olmayabilir veya pratik olmayabilir. Diğer yandan bu bilginin kullanımı algoritmayı önemli derecede hızlandırmaktadır. Bu yüzden Hessian matrisini kesin olarak hesaplamak yerine, değişik tahmin yaklaşımlarıyla bu önemli bilgiyi en azından özdeğer veya özvektör derecesinde algoritmaya katmak faydalı olabilir. Dinamik seçimli algoritmada ise süreç daha değişik açılardan incelenerek, örneğin fiziksel bir sistemle karşılaştırılarak, fiziksel sistemin yakınsama prensipleri, bu çalışmada tartıştığımız algoritmaların yakınsama prensipleriyle karşılaştırılabilir.

KAYNAKLAR

ARBIB, M. A. *Brains, Machines and Mathematics, 2nd edition*, New York:Springer-Verlag (1987).

BHAYA, A. ve KASZKUREWICZ E. *Steepest descent with momentum for quadratic functions is a version of the conjugate gradient method*. Neural Networks **17**, 65-71 (2004).

BISHOP, C. M. *Neural networks for pattern recognition*. Oxford Univ. Press (1995).

BRENT, R. P. *Algorithms for Minimization without Derivatives*. Englewood Cliffs, NJ: Prentice-Hall (1973).

BROGAN, W.L. *Modern Control Theory*. Englewood Cliffs, NJ: Prentice-Hall (1991).

CHURCLAND, P.S. ve SEJNOWSKI, T.J. *The Computational Brain*, Cambridge, MA:MIT Press (1992).

FAGGIN, F. *VLSI implementation of neural networks, Tutorial Notes*, International Joint Conference on Neural Networks, Seattle (1991).

FLETCHER, R. *Practical Methods of Optimization (Second ed.)* New York: John Wiley (1987).

GREENBAUM, A. *Iterative methods for solving linear systems*. Philadelphia: SIAM (1997).

GROSSBERG, S. *Neural Networks and Natural Intelligence*, Cambridge, MA:MIT Press (1988).

HAGAN, M.T., DEMUTH, H.B. ve BEALE, M.H. *Neural Network Design.* Boston, MA: PWS (1996).

HAYKIN, S. *Neural Networks (2nd ed.)*. Upper Saddle River. NJ:Prentice-Hall (1999).

HEBB, D.O. The *Organization of Behavior: A Neuropsychological Theory*, New York: Wiley (1949).

HESTENES, M. R. ve STIEFEL E. *Methods of conjugate gradients for solving linear systems*. Journal of Research of the National Bureau of Standards **49(6)**, 409-436 (1952).

HINTON, G.E. *Connectionist learning procedures*, Artificial Intelligence, **40**, 185-234 (1989).

KAMARTHI, S. V. ve PITTNER, S. *Accelerating neural network training using weight extrapolations*. Neural Networks, **12(9)**, 1285-1299 (1999).

KERLIRZIN, P. ve VALLET, F. *Robustness in multilayer perceptrons*, Neural Computation, **5**, 473-482 (1993).

LEVINE, M. *Man and Machine Vision*. New York: McGraw-Hill (1985).

LUENBERGER, D.G. *Linear and Nonlinear Programming. 2nd edition*, *Reading*, MA:Addison-Wesley (1984).

MARR, D. *Vision*, New York: W.H. Freeman and Company (1982).

MC CULLOCH W.S. ve PITTS W. *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics, **5**, 115-133 (1943).

MEAD, C.A. *Analog VLSI and Neural Systems*, *Reading*, MA: Addison-Wesley (1989).

MENDEL, J.M. ve MC LAREN. *Reinforcement-learning control and pattern recognition systems in Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications*, **66**, J.M. Mendel and K.S. Fu, eds. 287-318, New York: Academic Press (1970).

MINSKY, M.L. ve PAPERT, S.A. *Perceptrons*, Cambridge, MA:MIT Press (1969).

MØLLER, M. *A scaled conjugate gradient algorithm for fast supervised learning*. Neural Networks **6(4)**, 525-533 (1993).

NOCEDAL J. ve WRIGHT S. *Numerical Optimization*. Springer (1999).

PHANSALKAR, V.V. ve SASTRY, P.S. *Analysis of the backpropagation algorithm with momentum*. IEEE Trans. Neural Networks, **5**, May (1994).

PLAUT, D.; NOWLAN, S. ve HINTON, G.E. *Experiments on learning by backpropagation*. Technical Report CMU-CS-86-126, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA (1986).

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T. ve FLANNERY, B. P. *Numerical Recipes in C: The Art of Scientific Computing. (Second ed.)*. Cambridge University Press (1992).

QIAN, N. *On the momentum term in gradient descent learning algorithms*. N Netks, **12(1)**, 145-151 (1999).

ROSENBLATT, F. *The Perceptron: A probabilistic model for information storage and organization in the brain, Psychological Review*, **65**, 386-408 (1958).

ROSENBLATT, F. *Principles of Neurodynamics*, Washington, D.C.: Spartan Books (1962).

RUMELHART, D.E. ve MC CLELLAND, J.L. eds, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. **1**, Cambridge, MA:MIT Press (1986).

SAMLI, M.M. ve DOGAN, I. *An artificial neural network for predicting the presence of spermatozoa in the testes of men with nonobstructive azoospermia*. The Journal of Urology, **171**, 2354-2357 (2004).

SUGA, N. *Cortical computational maps for auditory imaging*, Neural Networks, **3**, 3-21 (1990a).

SUGA, N. *Computations of velocity and range in the bat auditory system for echo location*, in Computational Neuroscience, E.L. Schwartz, ed. 213-231, Cambridge, MA:MIT Press (1990b).

TORII, M. ve HAGAN, M. T. *Stability of steepest descent with momentum for quadratic functions*. IEEE Transactions on Neural Networks, **13(3)**, 752-756 (2002).

WIDROW, B. ve HOFF, M.E. *Adaptive switching circuits*, IRE WESCON Convention Record, 96-104 (1960).

WILLIAMS, P. M. *A Marquardt algorithm for choosing the step-size in backpropagation learning with conjugate gradients*. Technical Report CSRP **299**, University of Sussex, Brighton, UK (1991).

YU, X.-H. ve CHEN, G.-A. *Efficient backpropagation learning using optimal learning rate and momentum*. Neural Networks, **10(3)**, 517-527 (1997).

EKLER

EK 1. Uygulama için yazılan Matlab simülasyon kodu.

```
function [tr_results,oran,net,tt,testp]=uygulama3(p,t)

[pn,minp,maxp,tn,mint,maxt]=premnmx(p,t);
%[pn,meanp,stdp,tn,meant,stdt]=prestd(p,t);

[R,Q]=size(p);
[ptr,ttr,val.P,val.T,test.P,test.T] = sample2(p,t,152,75,76);
testp=test.P;
tt=postmnmx(test.T,mint,maxt);
tt1=postmnmx(ttr,mint,maxt);
tt2=postmnmx(val.T,mint,maxt);
%tt=poststd(test.T,meant,stdt);
algorithm=strvcat('traingd
','traingdm','traingda','traingdx','traincgf','traincgp','trainscg');

for alg=1:7

net = newff(minmax(ptr),[13 1],{'tansig','tansig'},'');
net.trainFcn=deblank(algorithm(alg,:));
net.trainparam.epochs=10000;
net.trainparam.goal=0.01;

for k=1:31
   net=init(net);

   [net,tr]=train(net,ptr,ttr,[],[],val,test);
%   [net,tr]=train(net,ptr,ttr);
   an=sim(net,test.P);
   a=postmnmx(an,mint,maxt);
```

```
    bn=sim(net,ptr);
    b=postmnmx(bn,mint,maxt);
    vn=sim(net,val.P);
    v=postmnmx(vn,mint,maxt);
    %a=poststd(an,meant,stdt);
    [R,Q]=size(tr.epoch);
    trr(k,:)=[tr.epoch(Q),tr.perf(Q),tr.vperf(Q),tr.tperf(Q),tr.ttime(Q)];
    for i=1:76
        if a(i)>0.5 c(k,i)=1; else c(k,i)=0; end
    end
    for i=1:152
        if b(i)>0.5 c1(k,i)=1; else c1(k,i)=0; end
    end
    for i=1:75
        if v(i)>0.5 c2(k,i)=1; else c2(k,i)=0; end
    end

end

for i=1:76
    t=0;
    f=0;
    for k=1:31
        if c(k,i)==1 t=t+1; else f=f+1; end
    end
    if t>=f d(i)=1; else d(i)=0; end
end

for i=1:76
    s(i)=d(i)-tt(i);
end

dogru=0;
```

```
for i=1:76
   if s(i)==0 dogru=dogru+1; end
end
oran(alg,1)=dogru/76
tr_results(alg,:)=mean(trr) ;

for i=1:152
   t=0;
   f=0;
   for k=1:31
      if c1(k,i)==1 t=t+1; else f=f+1; end
   end
   if t>=f d1(i)=1; else d1(i)=0; end
end

for i=1:152
   s1(i)=d1(i)-tt1(i);
end

dogru=0;
for i=1:152
   if s1(i)==0 dogru=dogru+1; end
end
oran(alg,2)=dogru/152

end

for i=1:75
   t=0;
   f=0;
   for k=1:31
      if c2(k,i)==1 t=t+1; else f=f+1; end
   end
```

```
    if t>=f d2(i)=1; else d2(i)=0; end
end

for i=1:75
    s2(i)=d2(i)-tt2(i);
end

dogru=0;
for i=1:75
    if s2(i)==0 dogru=dogru+1; end
end
oran(alg,3)=dogru/75
```

EK 2. Dinamik seçimli momentumlu gradyan azalan programı.

```
function [alpha,momentum,x,r,p] = momentum(x,Hessian,d,max_itr)
   r(:,1)=d-Hessian*x(:,1);
   p(:,1)=r(:,1);  % ilk yön gradyanın tersi olarak seçildi.
   a(:,1)=(r(:,1)'*r(:,1))/(p(:,1)'*Hessian*p(:,1)); % ilk adım uzunluğunun
belirlenmesi
   alpha(:,1)=a(:,1);
   x(:,2)=x(:,1)+a(:,1)*p(:,1);  % iterasyon yapılır.
   for k=2:max_itr
      r(:,k)=r(:,k-1)-a(:,k-1)*Hessian*p(:,k-1);
      if (abs(r(:,k))<1e-24), break, end;
      b(:,k-1)=(r(:,k)'*r(:,k))/(r(:,k-1)'*r(:,k-1));
      p(:,k)=r(:,k)+b(:,k-1)*p(:,k-1);
      a(:,k)=(r(:,k)'*r(:,k))/(p(:,k)'*Hessian*p(:,k));
      momentum(:,k)=(a(:,k)/a(:,k-1))*b(:,k-1);
      alpha(:,k)=(a(:,k)*a(:,k-1))/(a(:,k-1)-a(:,k)*b(:,k-1));
      x(:,k+1)=x(:,k)+momentum(k)*(x(:,k)-x(:,k-1))+(1-
momentum(:,k))*alpha(:,k)*r(:,k);
   end;
   itr=k;
```

EK 3. Eğitim, geçerlilik ve test veri gruplarının oluşturulması için Matlab kodu.

```
unction [tr_pn,tr_tn,val_pn,val_tn,test_pn,test_tn] =
sample2(p,t,tr_size,val_size,test_size)

[pn,minp,maxp,tn,mint,maxt]=premnmx(p,t);
%[pn,meanp,stdp,tn,meant,stdt]=prestd(p,t);

for i=1:tr_size
   index=ceil(rand*size(pn,2));
   tr_pn(:,i)=pn(:,index);
   tr_tn(:,i)=tn(:,index);
   pn(:,index)=[];
   tn(:,index)=[];
end

for j=1:val_size
   index=ceil(rand*size(pn,2));
   val_pn(:,j)=pn(:,index);
   val_tn(:,j)=tn(:,index);
   pn(:,index)=[];
   tn(:,index)=[];
end

for k=1:test_size
   index=ceil(rand*size(pn,2));
   test_pn(:,k)=pn(:,index);
   test_tn(:,k)=tn(:,index);
   pn(:,index)=[];
   tn(:,index)=[];
end
```

EK 4. Dinamik seçimli momentumlu gradyan azalan algoritmasında iterasyon süreci boyunca özdeğerlerin modlarını hesaplayan Matlab kodu.

```
function [l1,l2,u1,u2]=modsim(a,m,lis)
for i=1:1 %size(a,2)
   alpha=a(:,:,i);
   momentum=m(:,:,i);
   for j=1:size(alpha,2)
      [l1(:,i,j),l2(:,i,j)]=calcevs(alpha(j),momentum(j),lis);
      u1(:,i,j)=abs(l1(:,i,j));
      u2(:,i,j)=abs(l2(:,i,j));
      figure=plot(1:size(lis,1),u1(:,i,j));
      t=sprintf('%d nolu Hessian matrisi, %d. adım',i,j);
      title(t);
      hold on;
      plot(1:size(lis,1),u2(:,i,j),'r-');
      dosya=sprintf('H%d_Step%d',i,j);
      hold off;
      %pause;
      %saveas(figure,dosya);
   end;
end;
```