YAPAY SİNİR AĞLARI İLE DİFERANSİYEL DENKLEM ÇÖZÜMÜ ÜZERİNE YAPILAN BİR ARAŞTIRMA

Aslıhan YILDIRIM

Yüksek Lisans Tezi
Matematik Ana Bilim Dalı
Doç. Dr. Mesut KARABACAK
2024

T.C.
ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MATEMATİK ANA BİLİM DALI

YAPAY SİNİR AĞLARI İLE DİFERANSİYEL DENKLEM ÇÖZÜMÜ ÜZERİNE YAPILAN BİR ARAŞTIRMA

(Research On The Solution of Differential Equations With Artificial Neural Networks)

YÜKSEK LİSANS TEZİ

Aslıhan YILDIRIM

Danışman: Doç. Dr. Mesut KARABACAK

Erzurum

Ocak, 2024

KABUL VE ONAY TUTANAĞI

Aslıhan YILDIRIM tarafından hazırlanan "Yapay Sinir Ağları ile Diferansiyel Denklem Çözümü Üzerine Yapılan Bir Araştırma" başlıklı çalışması 22 / 01 / 2024 tarihinde yapılan tez savunma sınavı sonucunda başarılı bulunarak jürimiz tarafından Matematik Ana Bilim Dalı, Uygulamalı Matematik Bilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Jüri Başkanı:	Doç. Dr. Yeşim AKBULUT *Atatürk Üniversitesi*	Aslı ıslak imzalıdır
Danışman:	Doç. Dr. Mesut KARABACAK *Atatürk Üniversitesi*	Aslı ıslak imzalıdır
Jüri Üyesi:	Dr. Öğr. Üyesi Muhammed YİĞİDER *Erzurum Teknik Üniversitesi*	Aslı ıslak imzalıdır

Enstitü Yönetim Kurulunun …./…/…. tarih ve …… sayılı kararı.

Bu tezin Atatürk Üniversitesi Lisansüstü Eğitim ve Öğretim Yönetmeliği'nin ilgili maddelerinde belirtilen şartları yerine getirdiğini onaylarım.

Prof. Dr. Saltuk Buğrahan CEYHUN
Enstitü Müdürü

Aslı ıslak imzalıdır

Not: Bu tezde kullanılan özgün ve başka kaynaklardan yapılan bildiriş, çizelge, şekil ve fotoğrafların kaynak olarak kullanımı, 5846 sayılı Fikir ve Sanat Eserleri Kanunundaki hükümlere tabidir.

ETİK BİLDİRİM VE İNTİHAL BEYAN FORMU

Yüksek Lisans Tezi olarak *Doç. Dr. Mesut KARABACAK* danışmanlığında sunulan "Yapay Sinir Ağları ile Diferansiyel Denklem Çözümü Üzerine Yapılan Bir Araştırma" başlıklı çalışmanın tarafımızdan bilimsel etik ilkelere uyularak yazıldığını, yararlanılan eserlerin kaynakçada gösterildiğini, Fen Bilimleri Enstitüsü tarafından belirlenmiş olan Turnitin Programı benzerlik oranlarının aşılmadığını ve aşağıdaki oranlarda olduğunu beyan ederiz.

Tez Bölümleri	**Tezin Benzerlik Oranı (%)**	**Maksimum Oran (%)**
Giriş	*2*	**30**
Kuramsal Temeller	*19*	**30**
Materyal ve Yöntem	*9*	**35**
Araştırma Bulguları ve Tartışma	*10*	**20**
Sonuç ve Öneriler	*0*	**20**
Tezin Geneli	*17*	**25**

Not: *Yedi kelimeye kadar benzerlikler ile Başlık, Kaynakça, İçindekiler, Teşekkür, Dizin ve Ekler kısımları tarama dışı bırakılabilir. Yukarıdaki azami benzerlik oranları yanında tek bir kaynaktan olan benzerlik oranlarının* ***%5'den*** *büyük olmaması gerekir.*

Beyan edilen bilgilerin doğru olduğunu, aksi halde doğacak hukuki sorumlulukları kabul ve beyan ederiz.

Tez Yazarı (Öğrenci)	**Tez Danışmanı**
Aslıhan YILDIRIM	**Doç. Dr. Mesut KARABACAK**
1.2.2024	**1.2.2024**
İmza: Aslı ıslak imzalıdır	İmza: Aslı ıslak imzalıdır

***** Tez ile ilgili YÖKTEZ'de yayınlamasına ilişkin bir engelleme var ise aşağıdaki alanı doldurunuz.

☐ Tezle ilgili patent başvurusu yapılması / patent alma sürecinin devam etmesi sebebiyle Enstitü Yönetim Kurulunun …./…/…. tarih ve ……. sayılı kararı ile teze erişim 2 (iki) yıl süreyle engellenmiştir.

☐ Enstitü Yönetim Kurulunun …./…/…. tarih ve ……. sayılı kararı ile teze erişim 6 (altı) ay süreyle engellenmiştir.

TEŞEKKÜR

Tez çalışmam boyunca bana her türlü kolaylığı sağlayan ve gösterdiği yol ve yöntemlerle desteğini esirgemeyen sayın hocam Doç. Dr. Mesut KARABACAK'a ve maddi manevi desteklerini esirgemeyen aileme teşekkür ederim.

Yüksek Lisans eğitimim boyunca "Yurt İçi Yüksek Lisans Burs Programı" ile tarafıma vermiş olduğu destek için TÜBİTAK'a şükranlarımı sunmayı bir borç bilirim.

Aslıhan YILDIRIM

ÖZET

YÜKSEK LİSANS TEZİ

YAPAY SİNİR AĞLARI İLE DİFERANSİYEL DENKLEM ÇÖZÜMÜ ÜZERİNE YAPILAN BİR ARAŞTIRMA

Aslıhan YILDIRIM

Danışman: Doç. Dr. Mesut KARABACAK

Amaç: Bu yüksek lisans tezi, yapay sinir ağları kullanarak diferansiyel denklemlerin çözümü konusunda literatürdeki önemli çalışmaları derlemeyi ve analiz etmeyi amaçlamaktadır. Diferansiyel denklemler, birçok bilimsel ve mühendislik alanında karşımıza çıkan matematiksel modelleme zorluklarını içermekte olup, yapay sinir ağlarının bu alandaki rolü büyük bir öneme sahiptir. Geleneksel nümerik yöntemlerin ötesinde, yapay sinir ağlarının bu alandaki potansiyelini anlamak ve değerlendirmek amacıyla gerçekleştirilen bu çalışma, hem teorik hem de pratik bir perspektif sunmaktadır.

Yöntem: Yapay sinir ağları ile diferansiyel denklem çözümü için matematiksel yöntemler kullanılarak gösterilmiştir.

Bulgular: Bu tezde, ilk olarak yapay sinir ağı ile diferansiyel denklemlerin çözümü üzerine yapılan ilk çalışma olan Hopfield sinir ağı modelinden bahsedilmiştir. Ardından yapay sinir ağı ile lineer ve lineer olmayan adi diferansiyel denklemlerin çözümü üzerinde durulmuştur. Son olarak yapay sinir ağı ile adi ve kısmi diferansiyel denklemlerin çözümü üzerinde genel bir yol izlenmiştir.

Sonuç: Bu çalışma, yapay sinir ağlarının birçok farklı alanda kullanıldığını ortaya koymak için yapılmış bir içerik analizi örneğidir. Çalışma kapsamında, yapay sinir ağları birçok farklı sınıflandırma başlığı altında tek bir kaynakta toplanarak incelenmiştir. Bu sayede, yapay sinir ağlarının hangi alanlarda ve hangi problemleri çözmek için kullanıldığı daha açık bir şekilde ortaya konulmuştur. Elde edilen bulguların bu alandaki tezlerin niteliği hakkında araştırmacılara tek bir kaynaktan birçok sınıflandırma ile bilgi sunması açısından önemlidir.

Anahtar Kelimeler: Yapay sinir ağları, Aktivasyon fonksiyonları, Gradyan iniş algoritması, Temel fonksiyonlar, Lineer diferansiyel denklemler, Lineer olmayan diferansiyel denklemler, Adi diferansiyel denklemler, Kısmi diferansiyel denklemler.

Ocak 2024, 112 sayfa

ABSTRACT

MSC. THESİS

RESEARCH ON THE SOLUTION OF DIFFERENTIAL EQUATIONS WITH ARTIFICIAL NEURAL NETWORKS

Aslıhan YILDIRIM

Supervisor: Doç. Dr. Mesut KARABACAK

Purpose: This master's thesis aims to review and analyze important works in the literature on solving differential equations using artificial neural networks. Differential equations pose mathematical modeling challenges in many scientific and engineering fields, and the role of artificial neural networks in this field is of great importance. Beyond traditional numerical methods, this study provides both a theoretical and practical perspective to understand and evaluate the potential of artificial neural networks in this field.

Method: It is demonstrated using mathematical methods for solving differential equations with artificial neural networks.

Findings: In this thesis, firstly, the Hopfield neural network model, which is the first study on the solution of differential equations with artificial neural network, is mentioned. Then, the solution of linear and nonlinear ordinary differential equations with artificial neural network is discussed. Finally, a general path is followed on the solution of ordinary and partial differential equations with artificial neural network.

Result: This study is an example of content analysis to reveal that artificial neural networks are used in many different fields. Within the scope of the study, artificial neural networks are analysed under many different classification headings in a single source. In this way, it has been revealed more clearly in which areas and which problems artificial neural networks are used to solve. The findings obtained are important in terms of providing information to researchers about the quality of theses in this field with many classifications from a single source.

Keywords: Artificial neural networks, Activation functions, Gradient descent algorithm, Basis functions, Linear differential equations, Nonlinear differential equations, Ordinary differential equations, Partial differential equations.

January 2024, 112 pages

İÇİNDEKİLER

TABLOLAR DİZİNİ

ŞEKİLLER DİZİNİ

KISALTMALAR DİZİNİ

ÇKA	Çok katmanlı algılayıcı (Multılayer Perceptron)
FFANN	İleri beslemeli yapay sinir ağı (Feed Forward ANN)
MWR	Ağırlıklı kalanlar yöntemi (Method of Weighted Residuals)
ODE	Adi diferansiyel denklem (Ordinary Differential Equation)
PDE	Kısmi diferansiyel denklem (Partial Differential Equation)
YSA	Yapay sinir ağı (ANN: Artificial Neural Network)

GİRİŞ

Diferansiyel denklemler, farklı bilim ve mühendislik alanlarındaki problemlerin çözümü için temel araçlardır. Diferansiyel denklemlerin analitik çözümleri, genellikle basit ve idealize edilmiş durumlar için elde edilebilir. Ancak gerçek dünyadaki pek çok problem daha karmaşık denklemlerle modellendiği için analitik çözüm elde etmek zor olabilir hatta imkansız olabilir. Bu durumda çözüme iteratif yöntemlerle nümerik bir şekilde yaklaşılmaya çalışılır. İteratif yöntemler, denklemleri adım adım çözerek yaklaşır. Runge-Kutta, Sonlu Fark metodu ve Atış metodu gibi metodlar, diferansiyel denklemlerin çözümü için tercih edilen yöntemler arasında yer alır. Bunanla birlikte Lee ve Kang'ın araştırması, diferansiyel denklemlerin nümerik çözümlerini elde etmek için, klasik iterasyon yöntemlerine bir alternatif olarak Yapay Sinir Ağları'nın (YSA) önerdiği öneriyi içermektedir (Lee ve Kang 1990). Bilgisayar biliminin ilerlemesi ile birlikte yapay sinir ağları, diferansiyel denklemleri çözmek için daha yeni ve güçlü araçlar haline getirildi. Yapay sinir ağları, büyük miktarda veri ve karmaşık ayrıntıları içeren denklemleri çözmede etkili olabilirler. Bu yöntem özellikle veriye dayalı programlarda ve tahminlerde yaygın olarak kullanılmaktadır. Sonuç olarak, diferansiyel denklemlerin mümkün olmayan analitik çözümleri, dijital yöntemler ve yapay sinir ağları gibi modern yaklaşımlar ile farklı disiplinlerdeki araştırmacılara ve mühendislere geniş bir araç yelpazesi sunarak karmaşık problemleri çözme imkanı sağlar.

İlk olarak Lee ve Kang birinci mertebeden adi diferansiyel denklemleri çözmek için Hopfield yapay sinir ağı modelini önermiştir (Lee ve Kang 1990). Daha sonra, Meade ve Fernandez lineer ve lineer olmayan adi diferansiyel denklemleri çözmek için ileri beslemeli yapay sinir ağı ve $B1$-spline'lar kullanmıştır (Meade ve Fernandez 1994). Ardından Lagaris vd. yapay sinir ağı içeren bir deneme çözümü oluşturarak adi ve kısmi diferansiyel denklemleri çözmüştür (Lagaris vd 1998). Bu deneme çözümü iki bölümden oluşmaktadır. İlk bölüm diferansiyel denklemin başlangıç ve sınır koşullarını sağlar, ikinci bölümü ise koşullardan bağımsız olup ileri beslemeli bir yapay sinir ağı içermektedir. Yazdi vd. adi diferansiyel denklemleri çözmek için yapay sinir ağlarına dayanan Koçan en küçük ortalama kareler (Kernel Least Mean Squares) algoritmasını önermiştir (Yazdi vd 2011). Kumar ve Yadav, diferansiyel denklemlerin çözümünde kullanılan Çok katmanlı alyılayıcı (Multilayer Perceptron) ve Radyal tabanlı fonksiyon sinir ağı modellerini içeren kapsamlı bir çalışma önermiştir (Kumar ve Yadav 2011).

Yapay sinir ağlarının diferansiyel denklemlerin çözümündeki etkisi görüldükçe, farklı tip yapay sinir ağları da bu amaçla kullanılmaya başlanmıştır. Örneğin, Fonksiyonel Bağlantılı Yapay Sinir Ağları diferansiyel denklemleri çözmek amacıyla kullanılan ağ türlerinden biridir. Bu ağ modelinin hesaplama maliyeti az olduğu için, uygulamada kolaylık sağlamaktadır. Mall ve Chakraverty, başlangıç değer ve sınır değer problemlerini çözmek amacıyla Legendre yapay sinir ağı (LeNN) modelini geliştirmişlerdir (Mall ve Chakraverty 2016). Hadian-Rasanan vd., Lane-Emden diferansiyel denklemlerini çözmek için kesirli Legendre yapay sinir ağından faydalanmıştır (Hadian-Rasanan vd 2020). Mall ve Chakraverty, adi ve kısmi diferansiyel denklemleri çözmek için fonksiyonel bağlantılı yapay sinir ağında Chebyshev polinomlarını kullanmıştır (Mall ve Chakraverty 2014, 2015, 2020). Yang vd, Legendre yapay sinir ağı (LeNN) ile aşırı öğrenme makinesi (extreme learning machine) algoritmasını kullanarak adi diferansiyel denklem sistemlerini çözmüştür (Yang vd 2018). Chakraverty, Lane-Emden denklemini ve Duffing osilatör denkleminin bir uygulama problemini çözmek için Laguerre yapay sinir ağı (LgNN) modelini önermiştir (Chakraverty 2020). Chen vd., farklı bir Laguerre yapay sinir ağı modeli önermiş ve bu model ile Black-Scholes diferansiyel denklemini çözmüştür (Chen vd 2021).

Bu çalışma, yapay sinir ağlarının birçok farklı alanda kullanıldığını ortaya koymak için yapılmış bir içerik analizi örneğidir. Çalışma kapsamında, yapay sinir ağları birçok farklı sınıflandırma başlığı altında tek bir kaynakta toplanarak incelenmiştir. Bu sayede, yapay sinir ağlarının hangi alanlarda ve hangi problemleri çözmek için kullanıldığı daha açık bir şekilde ortaya konulacaktır. Elde edilen bulguların bu alandaki tezlerin niteliği hakkında araştırmacılara tek bir kaynaktan birçok sınıflandırma ile bilgi sunması açısından önemlidir.

KURAMSAL TEMELLER

Beyin çok karmaşık bir yapıya sahip olmakla birlikte; öğrenme, hatırlama, karar verme gibi öğeleri gerçekleştirebilen bir organdır. Canlıların beyinleri aracılığıyla öğrenme, saklama ve farklı birçok yönde kullanma gibi testler yıllar boyunca bilim insanlarının ilgisini çekmiştir. Bilim insanları çevrelerindeki gözlemlerinden ilham alma ve taklit etme istekleriyle birlikte; yapay zeka ve yapay zekanın alt dalı olan yapay sinir ağları ortaya çıkmıştır.

Yapay Zeka

Yapay zeka ilk duyulduğu günden günümüze kadar kişilerin ilgi alanı ne olursa olsun merak uyandırmıştır. Bazı insanlar tehlikeli bazıları ise bulunmaz bir fırsat olarak görmüştür. Geçmişte birçok çalışma yapılmış ve günümüzde de yapılmaya devam eden hızla gelişen bir çalışma alanıdır.

Yapay zeka, insan zekasına özgü yetenek veya otonominin taklit edilmesini amaçlayan bir bilim ve teknoloji alanıdır. Bu yetenekler arasında algılama, öğrenme, düşünme, fikir yürütme, sorun çözme, iletişim kurma, çıkarım yapma ve karar verme gibi yüksek düzeydeki yöntemler bulunur.

Yapay zeka, bilgisayarlar ve makineler aracılığıyla zeka ve gelişimi simüle etme veya gerçekleştirme amacı güder. Zorlu problemlerin çözümünü daha erişilebilir hale getiren bu alan, optimizasyon problemlerinden, ses ve yüz tanımaya, güvenlikten rota planlamaya, kanser hücresi tespitine, sürücüsüz araçlara, matematiksel işlemlere kadar birçok farklı konuda etkili çözümler sunuyor. Yapay zekanın farklı problemi çözmek için yararlandığı birçok tekniği vardır bu da onu modern teknolojik yönde önemli bir araç haline getirir.

Yapay Sinir Ağı (ANN: Arficial Neural Network)

Literatürde yapay sinir ağının farklı tanımları yer almaktadır. Bu tanımlardan biri, Simon Haykin'in (Haykin 2010) yapay sinir ağı ile beynin benzerliğine dikkat çektiği tanımdır: "Sinir ağı, basit işlem birimlerinden oluşan, deneyimsel bilgileri depolamaya yönelik doğal bir eğilimi olan ve bu bilgilerin kullanılmasını sağlayan büyük ölçekte paralel dağıtılmış bir işlemcidir. İki yönden beyin ile benzerlik göstermektedir:

1. Bilgi, ağ tarafından öğrenme süreciyle çevreden elde edilir.
2. Elde edilen bilgileri biriktirmek için sinaptik ağırlıklar olarak da bilinen nöronlar arası bağlantı güçleri kullanılır."

Yapay sinir ağı, insan beyninin ve sinir ağlarının çalışma prensiplerine dayanarak matematiksel olarak modellenmiş bir yapay zeka sistemidir, amacı ise bilgisayar sistemlerine insan öğrenme yeteneği kazandırmaktır. İnsan öğrenme sürecini göz önünde bulundurarak yapay sinir ağları bilgisayar için öğrenme yeteneğini sağlamayı hedefler. Bu yaklaşımla bilgisayarların elde ettiği bilgileri kullanarak deneyimlerini öğrenmelerini, bu bilgileri geliştirmelerini, yeni bilgi oluşturmalarını, bu bilgileri yorumlayıp çalıştırmalarını ve karar vermelerini mümkün kılmayı sağlar.

Yapay sinir ağları, insan beyninde milyarlarca olan sinir hücrelerinin yapısını taklit ederek yapılmıştır. Bu modeller, beyin gibi adaptif olmuştur. Yapay sinir ağları, biyolojik sinir sistemlerinden esinlenerek tasarlandığı için bu iki sistem birbirine benzer özellikler taşır. O halde biyolojik sinir ağları sinir hücrelerine sahip olduğundan yapay sinir ağlarının da yapay sinir hücrelerine sahip olduğunu diyebiliriz. Bu hücreler, beynin sinir hücreleri gibi çalışır ve sinyalleri birbirleriyle paylaşırlar. Bu sinyal paylaşımı, beynin nasıl çalıştığını anlamaya yardımcı olur böylelikle yapay sinir ağının nasıl çalıştığını anlamaya yardımcı olur. Yapay sinir ağları, biyolojik sinir ağlarının çalışma prensiplerine benzer şekilde yürüttüğü sinyal paylaşımı ve bilgi işleme yoluyla karmaşık problemler için kullanılır.

Yapay Sinir Ağlarının Kullanım Alanları

Yapay sinir ağlarının hata toleransı ve öğrenme özellikleri sayesinde gerçek hayat problemlerinde kullanılabilir olmaları mümkündür. Yapay sinir ağları, veri girdileri ve beklenen çıkışlar arasındaki ilişkileri öğrenerek ve bu öğrenme sürecindeki hataları düzelterek çalışırlar. Ayrıca, öğrenme yeteneği sayesinde yapay sinir ağları, verilerdeki düzenlilikleri veya kalıpları öğrenerek gelecekteki olayları veya sonuçları tahmin edebilirler. Bu özellikler, yapay sinir ağlarının birçok alanda kullanılmasını mümkün kılar.

Tanı: Yapay sinir ağları, hastalıkların teşhisinde ve tanısında kullanılabilir. Örneğin, kanser tespiti veya nörolojik bozuklukların teşhisi gibi durumlarda kullanılabilir.

Tahmin: Yapay sinir ağları, gelecekteki olayların veya sonuçların tahmin edilmesinde kullanılabilir. Örneğin, hisse senedi fiyatları veya hava durumu gibi alanlarda kullanılabilir.

Sınıflandırma: Yapay sinir ağları, verileri sınıflandırmak veya kategorize etmek için kullanılabilir. Örneğin, bir pazarlama kampanyası için hedef müşteri segmentlerinin belirlenmesi gibi durumlarda kullanılabilir.

Kontrol: Yapay sinir ağları, makine veya cihazların kontrolünde kullanılabilir. Örneğin, robotik kol gibi cihazların hareket kontrolü veya trafik akışının yönetimi gibi durumlarda kullanılabilir.

Doğal Dil İşleme: Yapay sinir ağları, doğal dil işleme gibi alanlarda da kullanılmaktadır. Örneğin, bir web sitesi kullanıcıların sorduğu soruları anlamak ve doğru cevapları vermek için kullanılabilir.

Optimizasyon: Yapay sinir ağları, karmaşık problemleri optimize etmek için kullanılabilir. Örneğin, bir işletmenin satışını artırmak için en iyi pazarlama stratejilerini belirlemek, enerji tüketimini azaltmak veya üretim sürecindeki verimliliği artırmak gibi konularda kullanılabilirler.

Verilen kullanım alanlarını genişletmek mümkündür. Yapay sinir ağları, sahip olduğu özellikler sayesinde gerçek hayat problemlerinde yaygın olarak kullanılmaktadır ve gelecekte daha da yaygınlaşması beklenmektedir.

Yapay Sinir Ağlarının Genel Özellikleri

Yapay sinir ağları, birçok farklı özelliğiyle öne çıkan matematiksel modellerdir. Yapay sinir ağlarının bazı genel özellikleri şunlardır:

Paralellik Özelliği: Yapay sinir ağları genellikle paralel olarak çalışırlar ve bu özellikleri, aynı anda birçok işlem yapabilme yetenekleri ile ilişkilidir. Yapay sinir ağların paralel özelliği, her katmanın ayrı ayrı işlem yapması ve sonuçların bir sonraki katmana aktarılmasıdır. Ayrıca bazı yapay sinir ağ modelleri özellikle; grafik işleme birimleri gibi özel donanım cihazları kullanılarak paralelleştirilebilirler. Bu sayede, Yapay sinir ağlarının hesaplama hızı ve verimliliği arttırılabilir. Sonuç olarak, Yapay sinir ağlarının paralel özelliği, büyük veri kümeleri üzerinde hızlı ve etkili işlem yapabilme yeteneğiyle birlikte, eğitim sürecini hızlandırarak ve özel donanımlar kullanılarak yapay zeka uygulamalarının geliştirilmesinde önemli bir rol oynamaktadır.

Doğrusal Olmama: Yapay sinir ağları, veriler içindeki hem doğrusal hem de doğrusal olmayan ilişkileri modelleyebilirler. Bu doğrusal olmama, yapay sinir ağları karmaşık problemleri çözmede güçlü kılan temel özelliklerden biridir.

Doğrusal modellerin verilerdeki karmaşık kalıpları yakalama yetenekleri sınırlıdır çünkü değişkenler arasında sabit bir değişim hızı varsayarlar. Buna karşılık, Yapay sinir ağları, özellikle de çok katmanlı yapay sinir ağları, doğrusal olmayan işlevlere yaklaşabilir. Bu onların görüntü ve konuşma tanıma, doğal dil işleme ve girdiler ile çıktılar arasındaki temel ilişkilerin genellikle doğrusal olmayan ve karmaşık olduğu diğer birçok gerçek dünya uygulaması gibi görevlerde başarılı olmalarını sağlar.

Yapay sinir ağların doğrusal olmama özelliği genellikle her bir nöronun çıktısına uygulanan aktivasyon fonksiyonları yoluyla ortaya çıkar. Sigmoid, tanh ve ReLU (düzeltilmiş doğrusal birim) gibi doğrusal olmayan aktivasyon fonksiyonlarının yapay nöronlara dahil edilerek girdiler ve çıktılar arasındaki karmaşık eşlemeleri modellemesine olanak tanıyan doğrusal olmayan dönüşümler sağlar. Özetle, bir yapay sinir ağının doğrusal veya doğrusal olmayan şekilde tasarlanıp tasarlanmayacağı, problemin doğasına ve karmaşıklığına bağlıdır. Yapay sinir ağlarının doğrusal olmama özelliği onları oldukça çok yönlü hale getirir ve doğrusal modellerin yetersiz kalacağı çok çeşitli problemlerin üstesinden gelmek için uygundur.

Hata Toleransı: Seri işlemciye sahip sistemlerde, bir hata genellikle tüm sistem birimlerine yayılabilir ve sistemlerin çalışması ciddi şekilde bozulurken yapay sinir ağları paralel yapısı sayesinde daha dayanıklıdır. Nöronlarda veya bağlantı değerlerinde oluşan hasarlar, yapay sinir ağlarında genellikle sadece küçük sapmalara yol açar. Bu durum yapay sinir ağlarını hatalara karşı daha toleranslı hale getirir. Ayrıca Yapay sinir ağları, verilerin eksik veya hatalı olması durumunda, veriler arasındaki ilişkiyi bularak güvenilir sonuçlar üretebilir. Sonuç olarak, yapay sinir ağları paralel yapısı ve veri işleme yeteneği sayesinde hatalara karşı daha dayanaklı ve sistem genelinde bir hatanın tüm sistemi etkileme olasılığı daha düşüktür. Bu özellikler, pek çok uygulama alanında yapay sinir ağlarını tercih edilen bir seçenek haline getirmiştir.

Öğrenme, Genelleme ve Uyum Sağlama: Yapay sinir ağlarında öğrenme, ağa sunulan örnek durumlar aracılığıyla gerçekleşir. Yapay sinir ağı bir problemin örnek verileri arasındaki ilişkiyi keşfederek sinaptik bağlantı değerlerini ayarlar. Uygun sinaptik ağırlık değerlerinin belirlenmesi ile yapay sinir ağlarında öğrenme gerçekleşir ve bu sürece yapay sinir ağının eğitilmesi adı verillir. Eğitim tamamlandığında yapay sinir ağı, eğitildiği problemlerde genelleme yeteneğine sahip olur. Yani aynı problem ile ilgili farklı veriler ile karşılaşıldığında uygun çıktılar üretebilir ve ayrıca verilerde eksik veya hatalı kısımlara rağmen uygun çıktılar üretilebilir. Yapay sinir ağları, eğitildiği bir problemde yapılan değişikliklere uyum sağlayabilirler. Yapılan değişikliğe göre, Yapay Sinir Ağları ağırlıklarını tekrar güncelleyebilir.

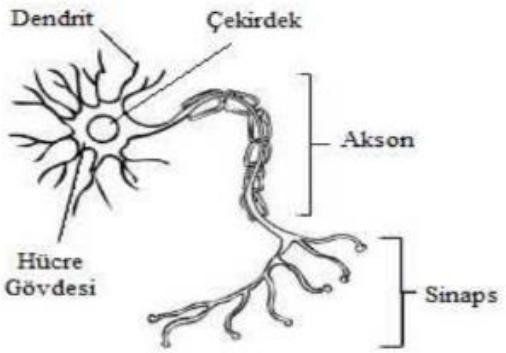
Yapay sinir ağları, bu özellikleri sayesinde birçok farklı uygulama alanında başarıyla kullanılmaktadır. Bu özellikler, ağların geniş bir problem yelpazesine uyarlanabilmesini sağlar ve bu da yapay zeka alanındaki hızlı ilerlemeyi destekler.

Biyolojik Sinir Hücresi

İnsan beyninin bilgiyi biyolojik olarak nasıl öğrendiğini inceleyerek yapay sinir ağlarının çalışması tasarlanmıştır. Sinir hücreleri, sinir sisteminin temel yapı taşlarıdır ve bu

hücreler nöron olarak da adlandırılırlar. İnsan merkezi sinir sisteminin yaklaşık 100 milyar sinir hücresinden oluştuğu tahmin edilmektedir (Da Silva vd 2017). Sinir hücreleri arasında yapısal değişkenlikler olmasına rağmen temel işleyiş tarzları genellikle benzerdir.

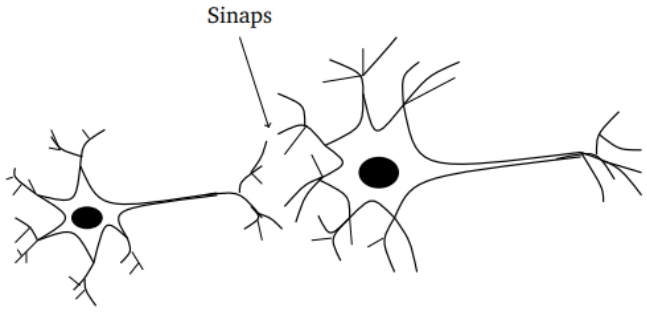
Temel bir biyolojik sinir hücresi dentrit, hücre gövdesi, akson ve sinapslerden oluşmaktadır ve hücre gövdesinde bir çekirdek bulunmaktadır (Şekil 1).



Şekil 1. Biyolojik sinir hücresi

Hücre gövdesinden çıkan ve bir ağacın dallarına benzeyen dendritler, diğer sinir hücrelerinden veya dış ortamdan gelen elektrokimyasal sinyalleri hücre gövdesine iletirler. Çekirdek ve plazma içeren hücre gövdesi, hücrenin yönetimini sağlamaktadır. Hücre gövdesi, dendritler aracılığıyla sürekli olarak sinyaller alır. Gelen uyarılar o nöronun eşik değerine ulaşırsa bir elektriksel çıktı oluşturulur ve aksona iletilir. Eşik değeri, çıktı oluşması için gereken minimum uyaran şiddetidir. Her bir nöronun spesifik bir eşik değeri vardır. Eşik değerine ulaşamayan uyarılara sinir hücresi yanıt vermez.

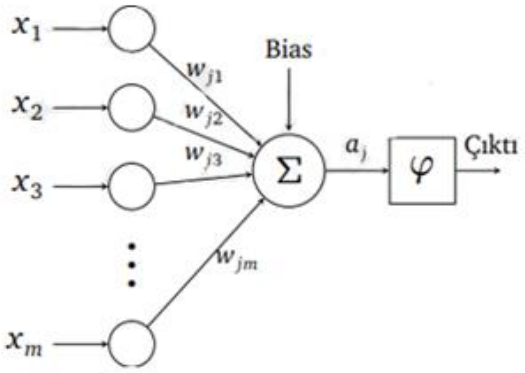
Hücre gövdesinden çıkan uzantı akson olarak adlandırılır. Aksonlar, nöronda oluşan çıktının diğer sinir hücrelerine iletilmesini sağlamaktadır. Aksonlar, son kısımlarında ki uzantılar aracılığıyla diğer hücrelerle bağlantı kurarlar. Aksonun uzantıları ile diğer bir sinir hücresinin dendritleri arasında sinaps adı verilen boşluklar vardır (Şekil 2). Bu sinaptik boşluklar da yeni üretilen sinyalleri diğer sinir hücrelerine iletirler.



Şekil 2. Sinir hücreleri arası sinaptik bağlantının gösterimi

Yapay Nöron

Biyolojik sinir hücrelerinde olduğu gibi yapay sinir hücrelerinde de aralarında bağ kurarak yapay sinir ağları oluşturulur. Yapay sinir ağlarının temel elemanı olan yapay sinir hücrelerine, işlem birimi de denilmektedir. Şekil 3'de yapay sinir hücresi yapısının basitleştirilmesiyle oluşturulmuş bir modelleme gösterilmiştir.



Şekil 3. Yapay nöron modeli

Yapay nöron aşağıdaki temel bölümlerden oluşmaktadır:

Ağırlıklar: Yapay sinir hücresine gelen bilginin hücre üzerindeki etkisini gösteren değere "ağırlık" denir. Ağırlıklar yapay sinir hücresine gelen girdilere verilen önemleri gösterir ve yapay sinir hücresinin çıktısını etkileyen önemli bir parametredir. Ağırlıklar yapay sinir ağının eğitim sırasında optimize edilir ve ağın problemi nasıl çözeceğine dair bilgi içerir. Ağırlıklar sabit değerler alabilir veya değişebilir. Pozitif ağırlık girdinin etkisini artırır iken, negatif ağırlık girdinin etkisini azaltır. Ağırlıkların değişimi, yapay sinir ağının çıktısını etkileyecektir. Şekil 3'de gösterilen w_{jm} ağırlık değeri, j. yapay sinir hücresinin x_m girdisinin bağlantı değerini sembolize etmektedir.

Toplama Fonksiyonu: Yapay sinir hücresine çevreden gelen net girdi toplama fonksiyonu ile bulunur. Bunun için farklı fonksiyonlar kullanılmaktadır. Genellikle ağırlıklı toplam fonksiyonu kullanılır. Bu fonksiyon, her bir girdi değeri ile o girdi değerine ait ağırlık değerini çarparak toplanır ve ağa gelen net girdi bulunmuş olur. Şekil 3 esas alınarak, j. sinir hücresinin a_j net girdisi;

$$a_j = \sum_{i=1}^{m} x_i w_{ji} \tag{1}$$

eşitliği ile ifade edilir. Bu eşitlikte verilen; girdi değerleri $(x_1, x_2, \dots, x_m)$, ağırlık değerleri $(w_{j1}, w_{j2}, \dots, w_{jm})$ ve hücreye gelen toplam girdi sayısı ise m ile ifade edilir. Net girdiyi hesaplamak için ağırlıklı toplam fonksiyonu kullanılmıştır.

Aktivasyon Fonksiyonu: Hücreye gelen net girdiyi kullanarak hücrenin bu girdiye üreteceği çıktıyı belirler. Aktivasyon fonksiyonu geniş değer aralığına sahip girdilerin daha sınırlı bir değer aralığına aktarılmasını sağlar. Çıktıyı hesaplamak için farklı aktivasyon fonksiyonları kullanılmaktadır. Şekil 3 esas alınarak, j. sinir hücresinin y_j çıktısı:

$$y_j = \varphi(a_j) \tag{2}$$

Burada, j. sinir hücresinin y_j çıktısını hesaplamak için net girdinin φ aktivasyon fonksiyon değeri hesaplanmaktadır.

Tablo 1'de Literatürde sık kullanılan bazı aktivasyon fonksiyonları ve bu fonksiyonların türevleri gösterilmiştir.

Tablo 1. Aktivasyon Fonksiyonları ve Türevleri

Aktivasyon Fonksiyonu	Türev	Açıklama
Lineer Fonksiyon $\varphi(x) = x$	$\varphi'(x) = 1$	Yapay nöronun çıktısı net girdiye eşit olur.
Eşik Fonksiyonu $\varphi(x) = \begin{cases} 1, \text{ if } x \geq 0 \\ 0, \text{ if } x < 0 \end{cases}$	x=0 noktasında türevi yoktur.	Net girdinin büyüklüğüne göre yapay nöron 0 veya 1 değerini alır. Sınıflandırma problemlerinde kullanılır.
Sigmoid Fonksiyonu $\varphi(x) = \frac{1}{1+\exp(-x)}$	$\varphi'(x) = \varphi(x)[1-\varphi(x)]$	Çok sık kullanılan bir aktivasyon fonksiyonudur. $[0, 1]$ aralığında değerler alır. Sürekli ve lineer olmayan bir fonksiyondur.
Hiperbolik tanjant Fonksiyonu $\varphi(x) = tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$	$\varphi'(x) = 1-\varphi^2(x)$	$[-1, 1]$ aralığında değer alır.
ReLU $\varphi(x) = max(0, x)$	$\varphi'(x) = \begin{cases} 1, \text{ if } x \geq 0 \\ 0, \text{ if } x < 0 \end{cases}$	$[0, \infty)$ aralığında değerler alır. Hesaplama kolaylığı nedeniyle sık kullanılan bir aktivasyon fonksiyonudur.
Gauss Radyal Temel Fonksiyonu $\varphi(x) = \exp(\frac{\|x-c\|^2}{2\delta^2})$	$\varphi'(x) = \frac{-2(x-c)\varphi(x)}{\delta^2}$	Radyal tabanlı fonksiyon yapay sinir ağları (RBFNN) için sık kullanılan temel fonksiyonlardan biridir. c ve δ parametreleri probleme göre belirlenmektedir.

Bias: Net girdi hesaplanırken bias değeri toplama fonksiyonu ile elde edilen sonuca eklenebilir. Net girdiye eklenmesi, aktivasyon fonksiyonun sonucunu arttırır veya azaltır. Ağırlık değerleri gibi bias değerinin de ağ için en uygun değeri bulunmaya çalışılır. Her yapay sinir hücresine bias değeri eklenmek zorunda değildir.

$$a_j = \sum_{i=1}^{m} x_i w_{ji} + b_j \tag{3}$$

Burada, j. sinir hücresine ait bias değeri b_j ile gösterilmektedir.

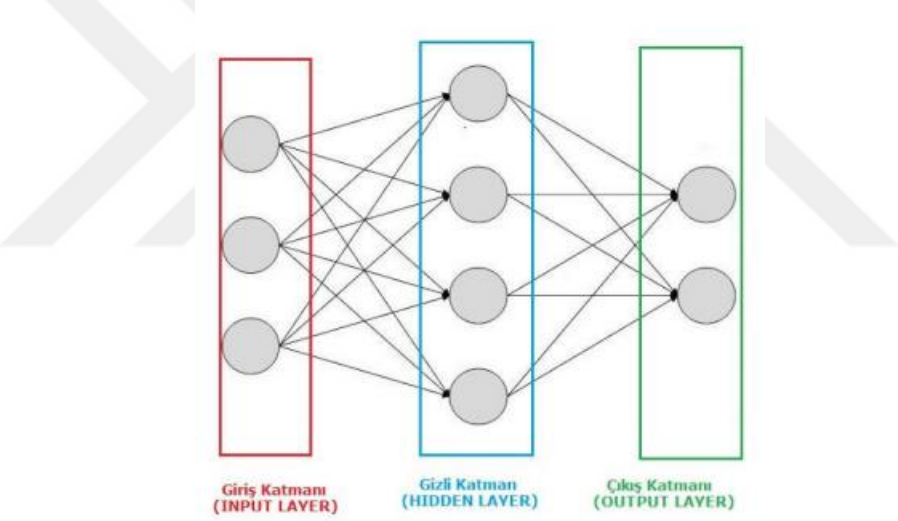
Denklem (3)'de görüldüğü gibi bias değeri toplam fonksiyonunun sonucuna eklenmektedir. Bunun yerine bias değerini denklemde ağın bir ağırlık değeri gibi yazabiliriz:

$$a_j = \sum_{i=0}^{m} x_i w_{ji} \tag{4}$$

Burada $b_j = w_{j0}$ olarak alınmıştır. Bias değerine ait girdi değeri ise $x_0 = +1$ olarak alınabilir.

Yapay Sinir Ağının Yapısı

Yapay sinir ağları, birbirlerine ağırlık değerleri ile bağlanan yapay sinir hücrelerinden oluşmaktadır. Yapay sinir hücreleri katmanlardan oluşur ve bu katmanlar arasında bağlantılar vardır. Genel olarak bakacak olursak; yapay sinir ağı üç ana katmanda incelenir: Giriş Katmanı, Ara (Gizli) Katman/Katmanlar ve Çıktı Katmanı. Şekil 4' de yapay sinir ağlarının katmanları gösterilmiştir.



Şekil 4. Yapay sinir ağı katmanları

Girdi katmanı: Yapay sinir hücresine çevreden gelen bilgilere "girdi" denir. Bu girdiler yapay sinir hücresine hesaplama yapması için gereken verileri sağlar. Bu katmanda yer alan sinir hücreleri, çevreden aldıkları verileri diğer katmanlara iletirler. Girdi katmanında bulunan sinir hücreleri gelen veriler üzerinde herhangi bir işlem yapmaz. . Bu nedenle bazı araştırmacılar ağın katman sayısını hesaplarken girdi katmanını bu hesaplamaya dahil etmezler. Girdi sayıları ise tamamen probleme özgüdür.

Gizli katmanlar: Girdi katmanı ile çıktı katmanı arasında bulunan tüm katmanlar gizli (ara) katman denir. Bu katmanlar, girdi katmanından gelen verileri alır. Gizli katmanlar, bu verileri belirli bir aktivasyon fonksiyonu kullanarak burada işlenip çıktılarını bir sonraki

katmana iletilirler. Bu katmandaki sinir hücreleri, gelen bilgiler arasındaki ilişkileri ortaya çıkarmak için önemli bir rol oynarlar. Ara katman sayısı ve ara katman nöron sayılarının belirlenmesi için genel bir kural olmadığı gibi bu durum eşlemenin karmaşıklığına göre değişebilir.

Çıktı katmanı: Herhangi bir hücreden aktivasyon fonksiyonu ile çıkan değer o hücrenin çıktı değeri olarak adlandırılır. Bu çıktı değeri ister yapay sinir ağının çıktısı olarak direkt çevreye aktarılabileceği gibi tekrardan ağın içinde bir başka hücreye girdi olarak da aktarılabilir. Her hücrenin birden fazla girdisi olmasına rağmen bir tek çıktısı olmaktadır. Bu çıktı istenilen sayıda hücreye bağlanabilir.

Yapay Sinir Ağlarında Öğrenme

Yapay sinir ağlarının en önemli avantajlarından biri, ağın öğrenme kapasitesidir. Yapay sinir ağlarında, öğrenme süreci ağın eğitimiyle başlar. Sinir ağının eğitimi, ağırlık değerlerinin belirlenmesiyle ilerler. Başlangıçta, bu ağırlık değerleri rastgele atanır. Optimal ağırlık değerleri bulunana kadar, ağa örnekler gönderilir ve ağırlık değerleri güncellenir. Ağın doğru ağırlık değerlerini seçmesi, temsil ettiği olay hakkında doğru yorum yapabilme yeteneğine ulaşması anlamına gelir. Sinir hücresi, bu doğru yorum yapabilme özelliğine ulaştığında öğrenme sürecini başarıyla tamamlamış olur. Ağın eğitilmesi sırasında yapılan işlemlerin tamamına öğrenme algoritması denmektedir. Ağın optimal ağırlık değerlerini elde etmesine de ağın öğrenmesi denmektedir.

Öğrenme algoritmalarına göre yapay sinir ağları

Yapay sinir ağları öğrenme algoritmalarına göre üç başlık altında toplanmaktadır. Bunlar;

- Denetimli Öğrenme
- Denetimsiz Öğrenme
- Destekli Öğrenme

Denetimli öğrenme

Bu öğrenme türüne eğiticili öğrenme, öğretmen ile öğrenme adı da verilmektedir. Eğitici, sisteme öğrenilmesi istenen olay ile ilgili örnekleri girdi ve çıktı birimi olarak öğretir. Girdi ve istenen çıktı değerleri bilindiği için bu veriler, YSA üzerinde bir nevi öğretmen görevi görürler. Ağ istenen çıktı değerlerine yaklaşmak için elde ettiği çıktı değerleri ile istenen çıktı değerleri arasındaki hatayı minimize etmeye çalışır. Hata değerini minimize etmek için ağırlık ve eşik değerlerini sürekli olarak günceller. Bu güncellemeler, optimal ağırlık değerleri elde

edilene kadar tekrarlanır ve öğrenme işlemi gerçekleşmiş olur. Denetimli öğrenme yöntemine örnek olarak çok katmanlı algılayıcılar verilebilir.

Denetimsiz öğrenme

Denetimsiz öğrenme sisteminde, denetimli öğrenmenin aksine girdilere karşılık gelen çıktı değerleri bilinmemektedir. Ağ girdi değerleri arasındaki ilişkiyi veya modelleri bularak çıktıları oluşturur. Model oluşturulup öğrenildikten sonra çıktı değerlerinin yorumlanması, kullanıcı tarafından belirlenir. Bu yöntem genellikle sınıflandırma problemlerinde kullanılan bir yaklaşımdır.

Destekli öğrenme

Denetimsiz öğrenmeye benzer şekilde, bu öğrenme türünde de ağ istenen çıktı değerlerini bilmemektedir. Sistem, öğretici olarak önceden belirlenmiş bir giriş ayarı ve buna karşılık gelen çıktı seti yerine, kendisine sunulan girilenlere karşılık çıktıları üretmeyi öğrenir. Destekli öğrenmede, genellikle ağın eğitilmesine yardımcı olan bilgi, ağın hata değeridir. Çıkan hata değerine göre ağın ağırlıklarında değişiklikler yapılır.

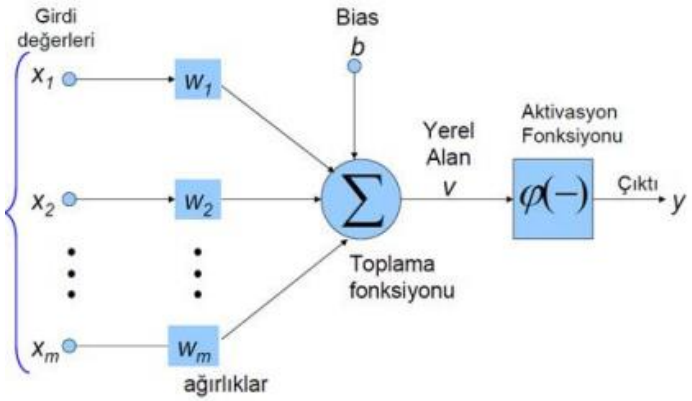
Katman Sayısına Göre Yapay Sinir Ağları

Yapay sinir ağları, katman sayısına göre iki başlık altında bahsedilecektir. Bunlar;

- Tek katmanlı Yapay Sinir Ağları
- Çok katmanlı Yapay Sinir Ağları

Tek katmanlı yapay sinir ağları

Tek katmanlı yapay sinir ağları sadece girdi ve çıktı katmanlarından oluşan ve doğrusal problemlerin çözümünde kullanılan bir modeldir. Tek katmanlı yapay sinir ağı modellerinde çıktı fonksiyonu doğrusal olup bu çıktı değeri 1 veya -1 değerlerini almaktadır. Eğer çıktı 1 ise birinci sınıfa, -1 ise ikinci sınıfa kabul edilmektedir (Öztemel, 2003). Şekil 5' te tek katmanlı yapay sinir ağı modeli gösterilmiştir.

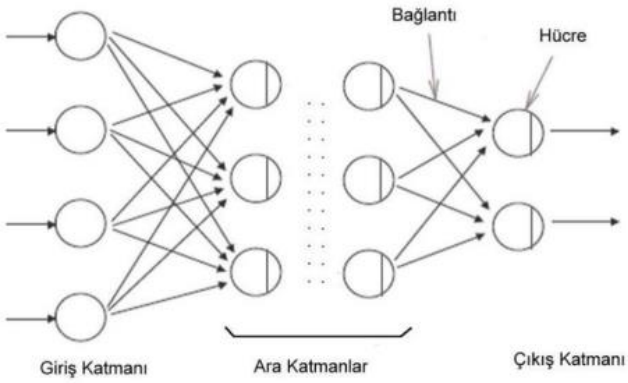


Şekil 5. Tek katmanlı yapay sinir ağının yapısı

Çok katmanlı yapay sinir ağları

Çok katmanlı modeller yapay sinir ağlarının en yaygın ve en etkili olan modeller arasında yer almaktadır. Bunun nedeni, tek katmanlı yapay sinir ağlarının doğrusal olmayan problemleri çözmek için yetersiz kalabilen basit bir ağ olmalarıdır. Dolayısıyla çok katmanlı yapay sinir ağları, daha karmaşık ve doğrusal olmayan ilişkileri öğrenmek ve modellenmesinde oldukça etkili olan yapay sinir ağı modelleridir.

Çok katmanlı yapay sinir ağları, giriş katmanı, bir veya daha fazla ara katman ve çıkış katmanından oluşur. Şekil 6'da çok katmanlı yapay sinir ağının yapısı verilmektedir.



Şekil 6. Çok katmanlı yapay sinir ağının yapısı

Doğrusal olmayan aktivasyon fonksiyonu içeren birçok dağıtımın birbirine bağlandığı yapı çok katmanlı yapay sinir ağı olarak adlandırılır. Bu dönüşümün doğrusal olmamasının sebebi, gizli katmanlarda kullanılan doğrusal olmayan aktivasyon fonksiyonlarıdır. Doğrusal olmayan aktivasyon fonksiyonu, karmaşık özellikleri öğrenmeye olanak tanır. Örneğin sigmoid, ReLU, tanh gibi fonksiyonlar bu aktivasyon fonksiyonlarına örnek olarak verilebilir. Bu fonksiyonlar, doğrusal olmayan davranışlar sergiler ve bu yapılar sayesinde çok katlı yapay sinir ağları daha karmaşık problemleri çözebilir hale gelir.

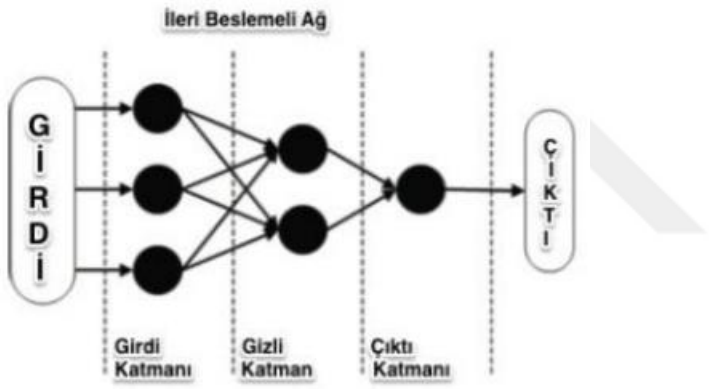
Yapılarına Göre Yapay Sinir Ağları

Yapay sinir ağları yapılarına göre iki başlık altında bahsedilecektir. Bunlar;

- İleri Beslemeli Yapay Sinir Ağları
- Geri Beslemeli Yapay Sinir Ağları

İleri beslemeli yapay sinir ağları (Feed Forward ANNs)

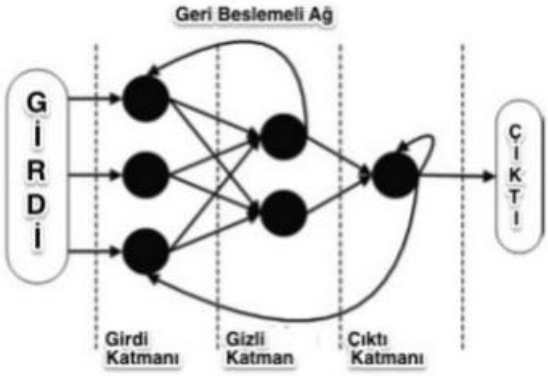
İleri beslemeli yapay sinir ağlarında her nöron bitişik katmandaki tüm nöronlarla bağlantılıdır ancak aynı katman içindeki diğer nöronlarla bağlantısı yoktur. Bu yapı, katman içindeki bağlantıların yasaklanmasını sağlar ve böylece ağ daha basit ve daha hızlı çalışır. Bu tip ağlarda veriler giriş katmanından başlayarak ara katmanlara ve son olarak çıkış katmanına ileri beslemeli olarak aktarılır. Bu yapıda her bir katmanın sadece bir önceki katmandan veri alması ve bir sonraki katmana veri göndermesi sağlanır. Şekil 7'de ileri beslemeli yapay sinir ağının yapısı verilmiştir.



Şekil 7. İleri beslemeli yapay sinir ağının yapısı

Geri beslemeli yapay sinir ağları (Recurrent ANNs)

Geri beslemeli yapay sinir ağlarında, en az bir hücrenin çıkışı kendi girişine veya başka bir hücreye giriş olarak atanır. Bu yapı, geri beslemeli yapay sinir ağlarını doğrusal olmayan ve dinamik bir davranış sergileyen bir form haline getirir. Geri beslemeli yapay sinir ağı, çıkış katmanı ve ara katmanlardaki giriş birimlerine veya önceki ara katmanlara geri beslendiği bir ağ yapısını temsil eder ve bu, hem ileri yönlere hem de geri yönlere bir iletişimin olduğu anlamına gelir. Bu nedenle, geri besleme yöntemine bağlı olarak farklı yapı ve davranışlarda geri beslemeli yapay sinir ağı modelleri elde edilebilir. Şekil 8, geri beslemeli yapay sinir ağının bu yapısal özelliklerini göstermektedir.



Şekil 8. Geri beslemeli yapay sinir ağının yapısı

MATERYAL ve YÖNTEM

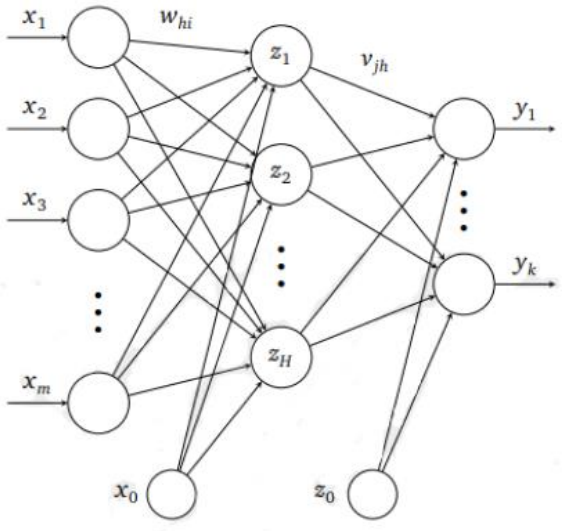
Çok Katmanlı Algılayıcı (Multılayer Perceptron)

ÇKA, ileri beslemeli yapay sinir ağları içinde önemli bir türüdür. ÇKA'lar, en az bir giriş katmanı, bir veya daha fazla gizli katman ve çıkış katmanı içeren bir yapıya sahiptir.

Çok Katmanlı Algılayıcının çalışmasını kısaca özetlemek gerekirse:

İlk olarak girdi katmanında bulunan girdiler birinci gizli katmana iletilir. Gizli katmandaki her bir gizli nöron, girdi değeri ile her bir girdiye ait ağırlık ve bias değerini hesaba katarak gizli nöronun çıktısını hesaplar ve daha sonra bu değerler ikinci gizli katmandaki nöronlara iletilir. Böylelikle bilgi, girdi katmanından çıktı katmanına doğru ilerlemiş olur. Bu sebepden dolayı çok katmanlı algılayıcılara ileri beslemeli sinir ağı da denilmektedir. Öğrenme süreci çok katmanlı algılayıcıda ilerledikçe gizli nöronlar eğitim verilerini karakterize eden belirgin özellikleri aşamalı olarak "keşfetmeye" başlarlar. Bunu, girdi verilerinden nitelik uzayı adı verilen yeni bir alana doğrusal olmayan bir dönüşüm gerçekleştirerek yaparlar (Haykin 2010). Bu dönüşümün doğrusal olmamasının sebebi, gizli katmanlarda kullanılan doğrusal olmayan aktivasyon fonksiyonlarıdır.

ÇKA'nın çalışmasının daha kolay anlaşılması için tek bir gizli katmana sahip çok katmanlı algılayıcıyı ele alalım.



Şekil 9. Tek gizli katmana sahip çok katmanlı algılayıcı modeli

Şekil 9'da ÇKA modelinde verilen $x_1,x_2,\ldots,x_m$ ağın girdi değerleri, w_{hi} girdi katmanını gizli katmana bağlayan ağırlık değerleri, $z_1,z_2,\ldots,z_H$ gizli katmanın nöronları, v_{jh} gizli katmanı çıktı katmanına bağlayan ağırlık değerlerini ve $y_1,y_2,\ldots,y_k$ ise ağın çıktı değerlerini belirtmektedir. Burada verilen x_0 girdi katmanının bias nöronu, z_0 ise gizli katmanın bias nöronudur. Ağın elde ettiği çıktıyı şu ifadeyle formülize edebiliriz:

$$y_j = \varphi\left(\sum_{h=1}^{H} v_{jh}\sigma\left(\sum_{i=1}^{m} w_{hi}x_i + w_{h0}\right) + v_{j0}\right) \quad j = 1,2,\ldots,k \tag{5}$$

Burada verilen σ, gizli katmanda kullanılan aktivasyon fonksiyonunu, φ ise çıktı katmanında kullanılan aktivasyon fonksiyonunu belirtmektedir. Denklem (4)'de gösterildiği gibi $x_0 = 1$ ve $z_0 = 1$ alınarak bias değerlerini ağın ağırlık değerleri kümesinin içine ekleyebiliriz. O halde denklem (5),

$$y_j = \varphi\left(\sum_{h=0}^{H} v_{jh}\sigma\left(\sum_{i=0}^{m} w_{hi}x_i\right)\right) \quad j = 1,2,\ldots,k \tag{6}$$

şekilde tekrar yazılabilir.

Çok katmanlı algılayıcının öğrenme kuralı

Geri yayılım algoritması, çok katmanlı algılayıcıları eğitmek için yaygın kullanılan yöntemlerden biridir. Geri yayılım algoritması genelleştirilmiş delta kuralı olarak da adlandırılır. Bu algoritma bir denetimli öğrenme türüdür, yani istenen çıktı değerleri verilir ve ağın çıktı değerlerinin bu istenilen değerlere yakınsaması amaçlanır. Geri yayılım algoritması; ileri yayılım ve geri yayılım olarak iki ana aşamadan oluşur.

İleri yayılım

İleri yayılım, girdinin algoritmaya girmesinden çıktının elde edilmesine kadar yapılan işlemlere denir. İlk olarak çok katmanlı algılayıcıya girdi değerleri sunulur ve ardından ağın çıktı değerleri hesaplanır. Bu hesaplama sürecinde ağın ağırlık ve bias değerleri değiştirilmez. Girdi katmanına iletilen bilgiler, herhangi bir değişiklik yapılmadan ilk gizli katmana iletilir. Girdi değerleri, ilk gizli katmandaki nöronlarda işlenir ve bir sonraki katmana iletilir. Bilgi, ağın girdi katmanından çıktı katmanına doğru ilerlemiş olur.

Gizli katmanlarda ve çıktı katmanında yer alan her bir nöron, bir önceki katmandan gelen değerler ile ağırlıkları hesaba katarak net girdiyi denklem (7) ile hesaplar:

$$v_j(n) = \sum_i w_{ji}(n)y_i(n) \tag{7}$$

Burada, $y_i(n)$'ler, j. nöronun girdilerini belirtmektedir. Eğer j. nöron ilk gizli katmanda yer alıyorsa, $y_i(n)$'ler ağın girdi değerleri olduğunu söyleyebiliriz. $w_{ji}(n)$ ise, i. nöronu j. nörona bağlayan ağırlık değeridir. Daha öncede ifade edildiği gibi bias değeri ekstra bir nöron kullanılarak net girdiye eklenebilmektedir. Dolayısıyla, bias değerini ayrı bir şekilde incelemeye gerek yoktur. j. nöronun çıktısı ise, bir aktivasyon fonksiyonunda net girdinin değeri hesaplanarak bulunur:

$$y_j(n) = \varphi_j\left(v_j(n)\right) \tag{8}$$

Ağın çıktı katmanında yer alan nöronların da çıktı değerleri hesaplandıktan sonra geri yayılım algoritmasının ilk aşaması tamamlanmış olur.

İleri yayılım ile tahmin yapıldıktan sonra maliyet fonksiyonu ile hata hesaplanır ve meyilli azalım ile maliyet değeri minimize edilir. Ancak meyilli azalım algoritmasının kullanılması için maliyet fonksiyonunun parametrelere göre türevleri bulunmalıdır. Bu işlemi geri yayılım algoritması yapmaktadır.

Geri yayılım

İlk olarak toplam hata hesaplanır (Haykin 2010). Toplam hata, ağın çıktı değerleri ile istenen çıktı değerleri arasındaki farktır. Çıktı katmanındaki herhangi bir yapay nöronda oluşan hata,

$$e_k(n) = t_k(n) - y_k(n) \tag{9}$$

şeklinde hesaplanabilir. $e_k(n)$, k. yapay nöronun hatasını belirtmektedir. Burada, k. nörondan istenen çıktı değeri $t_k(n)$ ile, bu nöronun çıktı değeri ise $y_k(n)$ ile gösterilmektedir. Çıktı katmanındaki bütün işlem birimlerinde oluşan toplam hata,

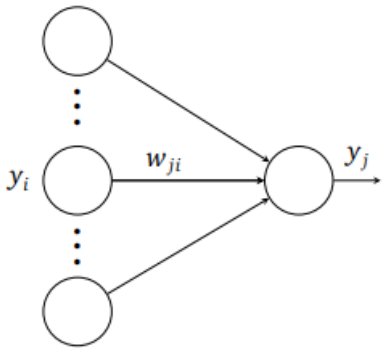
$$E(n) = \frac{1}{2}\sum_k e_k^2(n) \tag{10}$$

şekilde ifade edilebilir. Denklem (10)'a maliyet fonksiyonu da denilmektedir. Maliyet fonksiyonunu hesaplamak için, denklem (10)'da belirtilen fonksiyondan farklı bir fonksiyon da kullanılabilir. Toplam hatayı en aza indirmek için, hata ağın ağırlık değerlerine dağıtılır ve ağırlık değerleri değiştirilir. Ağırlıklar değiştirilirken, ağırlıkların yapay sinir ağında bulunduğu yere göre, iki farklı durum söz konusu olur:

- Gizli katman ile çıktı katmanı arasındaki ağırlıkların değiştirilmesi
- Gizli katmanlar arasındaki ağırlıkların veya girdi katmanı ile gizli katman arasındaki ağırlıkların değiştirilmesi

Gizli katman ile çıktı katmanı arasındaki ağırlıkların değiştirilmesi

Çıktı katmanından bir önceki katmanda yer alan i. nöron ile çıktı katmanındaki j. nöronu birbirine bağlayan $w_{ji}(n)$ ağırlığını ele alalım (Şekil 10).



Şekil 10. Çıktı katmanındaki j. nöronun sinaptik ağırlığını ayarlamak için nöral bağlantıların gösterimi

Öncelikle, ağırlıkta yapılacak değişimden hata fonksiyonunun ne kadar etkilendiğini bulmak için, $\frac{\partial E(n)}{\partial w_{ji}(n)}$ kısmi türevi hesaplanmalıdır:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{11}$$

Çıktı katmanındaki nöronu j notasyonu ile ifade ettiğimiz için, (9) ve (10) denklemlerini şu şekilde tekrar yazabiliriz:

$$E(n) = \frac{1}{2}\sum_j e_j^2(n) \tag{12}$$

$$e_j(n) = t_j(n) - y_j(n)$$

O halde, $\frac{\partial E(n)}{\partial e_j(n)}$ ve $\frac{\partial e_j(n)}{\partial y_j(n)}$ kısmi türevleri,

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \tag{13}$$

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

şeklindedir.

(7) ve (8) denklemleri kullanılarak, $\frac{\partial y_j(n)}{\partial v_j(n)}$ ve $\frac{\partial v_j(n)}{\partial w_{ji}(n)}$ kısmi türevleride,

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'\big(v_j(n)\big) \tag{14}$$

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

şeklinde hesaplanabilir.

(13) ve (14) denklemlerini, (11) denkleminde yerine yazdığımızda;

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'\left(v_j(n)\right)y_i(n) \tag{15}$$

elde ederiz.

O halde (15) denklemini şu şekilde yazabiliriz:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\delta_j(n)y_i(n); \quad \delta_j(n) = \frac{\partial E(n)}{\partial v_j(n)} \tag{16}$$

Burada, $\delta_j(n)$ sembolü, j. çıktı nöronunun hata değerini belirtmektedir. Ayrıca (16) denkleminden görüleceği üzere; toplam hatanın, ara katman ile çıktı katmanı arasındaki ağırlığa göre türevi, çıktı katmanındaki nöronun $\delta_j(n)$ hatası ile o nörona gelen girdinin çarpımına eşittir.

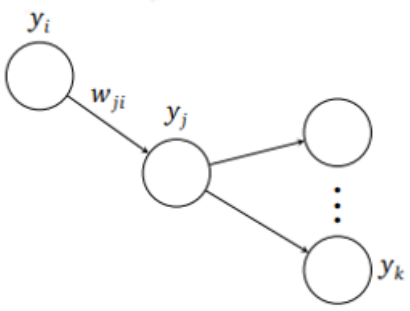
Hata fonksiyonunun ağırlık değişiminden etkilenme miktarı $\frac{\partial E(n)}{\partial w_{ji}(n)}$ hesaplandıktan sonra; $\Delta w_{ji}(n)$ ağırlık değerinde yapılacak değişim, gradyan inişi algoritması kullanılarak hesaplanabilir:

$$\Delta w_{ji}(n) = -\eta\frac{\partial E(n)}{\partial w_{ji}(n)} \tag{17}$$

Burada η öğrenme katsayısıdır. Ağırlıkta yapılacak olan değişimin miktarını belirler. Genellikle 0 ile 1 arasında değerler alır.

Gizli katmanlar arasındaki ağırlıkların veya girdi katmanı ile gizli katman arasındaki ağırlıkların değiştirilmesi

Gizli katmanda yer alan bir nöron ele alındığında, bu nörondan istenen çıktı değeri ve dolayısıyla bu nöronun hata değeri bilinmemektedir. Geri yayılım algoritmasında; gizli katmanlardaki nöronların hataları, kendinden sonraki katmanlardaki nöronların hatalarına bağlı olarak hesaplanır. Gizli katmanda yer alan bir j. nöronu ele aldığımızda, bu nörona gelen ağırlıklardan birini $w_{ji}(n)$ sembolü ile gösterebiliriz, bu bağlantı, i. nöronu j. nörona bağlayan ağırlık değerini belirtmektedir (Şekil 11).



Şekil 11. Gizli katmanda bulunan j. nöronun sinaptik ağırlığını ayarlamak için nöral bağlantıların gösterimi

Toplam hatanın $w_{ji}(n)$ ağırlığına göre türevini, zincir kuralını kullanarak şu şekilde ifade edebiliriz:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{18}$$

Burada, j. nöronun çıktı ve net girdi değerleri sırasıyla, $y_j(n)$ ve $v_j(n)$ ile gösterilmektedir. Toplam hata, daha önce denklem (10) ile şu şekilde ifade edilmişti:

$$E(n) = \frac{1}{2} \sum_k e_k^2(n) \tag{19}$$

O halde $\frac{\partial E(n)}{\partial y_j(n)}$ kısmi türevini şu şekilde yazabiliriz:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)} \tag{20}$$

Burada, k notasyonu çıktı katmanındaki nöronları belirtmek için kullanılmaktadır. Denklem (20)'de yer alan $\frac{\partial e_k(n)}{\partial y_j(n)}$ kısmi türevine zincir kuralı uygulanırsa, (20) denklemini tekrar şu şekilde yazabiliriz:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{21}$$

Burada $v_k(n)$, k. çıktı nöronunun net girdisini belirtmektedir. Denklem (9)'da belirtildiği gibi, çıktı katmanındaki k. nöronunun hata değeri $e_k(n)$ ise,

$$e_k(n) = t_k(n) - y_k(n) \tag{22}$$

şeklindedir.

Ayrıca, (7) ve (8) denklemleri kullanılarak, k. nöronun çıktısı şu şekilde yazılabilir:

$$y_k(n) = \varphi\big(v_k(n)\big) \tag{23}$$

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

Burada m, k. nörona gelen toplam girdi sayısını belirtmektedir. O halde (22) ve (23) denklemleri kullanılarak, $\frac{\partial e_k(n)}{\partial v_k(n)}$ ve $\frac{\partial v_k(n)}{\partial y_j(n)}$ türevleri;

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)) \tag{24}$$

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

şeklinde yazılabilir.

Denklem (24)'ü, (21) denkleminde yerine yazarsak;

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n)\, \varphi_k'\big(v_k(n)\big) w_{kj}(n) \tag{25}$$

elde edilir.

Daha önce (15) ve (16) eşitliklerinde belirtildiği gibi, j. çıktı nöronunun hatası $\delta_j(n)$,

$$\delta_j(n) = e_j(n)\varphi_j'\left(v_j(n)\right) \tag{26}$$

şeklinde gösterilmektedir. Burada, çıktı katmanındaki nöron için k notasyonunu kullandığımız için (25) denklemi şu şekilde yeniden yazılabilir:

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k \delta_k(n) w_{kj}(n) \tag{27}$$

(18) denklemindeki, $\frac{\partial y_j(n)}{\partial v_j(n)}$ ve $\frac{\partial v_j(n)}{\partial w_{ji}(n)}$ türevleri ise,

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'\big(v_j(n)\big); \quad y_j(n) = \varphi_j\big(v_j(n)\big) \tag{28}$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n); \qquad v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n)$$

şeklinde bulunur. Burada $y_i(n)$, j. gizli nörona gelen girdiyi belirtmektedir. O halde (27) ve (28) denklemlerini, (18) denkleminde yerine koyarsak;

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\sum_k \delta_k(n) w_{kj}(n) \varphi_j'\left(v_j(n)\right) y_i(n) \tag{29}$$

elde ederiz. j. gizli nöronun hatasını belirten $\delta_j(n)$ ise şu şekildedir:

$$\delta_j(n) = \sum_k \delta_k(n) w_{kj}(n)\, \varphi_j'\left(v_j(n)\right) \tag{30}$$

Denklem (30)'u, (29) denkleminde yerine koyarsak;

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\delta_j(n) y_i(n) \tag{31}$$

denklemini elde ederiz.

$\frac{\partial E(n)}{\partial w_{ji}(n)}$ hesaplandıktan sonra, denklem (17)'deki gibi gradyan inişi algoritması kullanılarak ağırlıkta yapılacak değişimler hesaplanabilir.

(16) ve (31) denklemlerinde görüldüğü üzere; $\frac{\partial E(n)}{\partial w_{ji}(n)}$ toplam hatanın gizli veya çıktı katmanda yer alan bir nöronun ağırlığına göre türevi, o nöronun hatası $\delta_j(n)$ ile o nörona gelen girdinin çarpımının eksi işaretlisine eşittir. Ayrıca, bir nöronun hatası $\delta_j(n)$, o nöronun gizli veya çıktı katmanında yer almasına göre şu şekilde bulunmaktadır:

• Eğer j. nöron çıktı katmanında yer alıyorsa; $\delta_j(n)$, nöronun çıktısının türevi $\varphi_j'\left(v_j(n)\right)$ ile o nöronun hata değerinin $e_j(n)$ çarpımına eşittir.

• j. nöron gizli katmanda yer alan bir nöron ise; $\delta_j(n)$, j. nöronun bulunduğu katmandan sonraki katmanda yer alan nöronların hatalarının ağırlıklı toplamı ile $\varphi_j'\left(v_j(n)\right)$'nin çarpımıdır.

Denklem (30)'da görüldüğü gibi, gizli katmandaki bir nöronun hatasını hesaplayabilmek için, o nöronun bulunduğu katmandan bir sonraki katmanda yer alan nöronların hatalarının bilinmesi gerekmektedir. Dolayısıyla çıktı katmanındaki nöronların hataları hesaplandıktan sonra, hatalar ağ üzerinde geriye doğru yayılır ve girdi katmanına kadar olan nöronların hataları sırasıyla hesaplanır. Bu nedenle, algoritmaya geri yayılım algoritması denmektedir.

(10) ile gösterilen hatayı minimize etmek için, gradyan inişi dışında farklı iteratif yöntemler de kullanılabilir. Bu yöntemlerde, iterasyona başlamak için, ilk olarak başlangıç ağırlık değerleri atanır. Ardından hata minimize olana kadar ağırlık değerleri değiştirilir. Aşağıda bazı gradyan inişi optimizasyon algoritmalarından bahsedilecektir.

Gradyan inişi (Gradient descent)

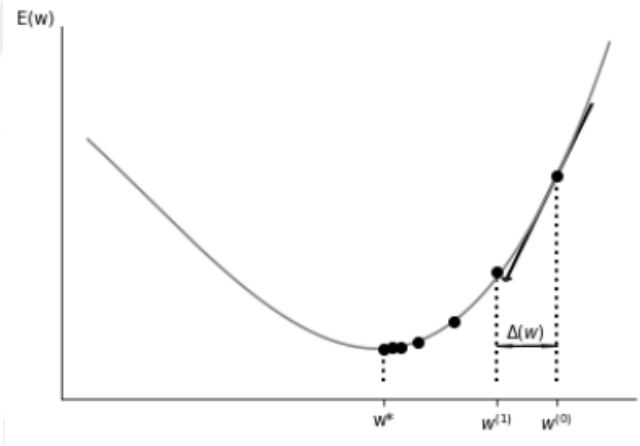
Gradyan inişi algoritması, uygulama kolaylığı nedeniyle, en sık kullanılan tekniklerden biridir. Diferansiyellenebilir $E(w)$ fonksiyonunu minimize etmek için, bu yöntem fonksiyonun w vektörüne göre gradyanını kullanmaktadır:

$$\nabla E(w) = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_l}\right]^T$$

Algoritma $w^{(0)}$ başlangıç ağırlık değerinden başlayarak, gradyanın tersi yönde ağırlık değerlerini günceller (Şekil 12):

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E\left(w^{(\tau)}\right) \tag{32}$$

Burada, τ iterasyonun adım sayısını göstermektedir. Yukarıdaki denklemde kullanılan η öğrenme katsayısının seçimi önemlidir. Öğrenme katsayısı büyük bir değer seçilirse, algoritma hata fonksiyonunun minimum değerini aşabilir ve bu minimum değere yakınsamayabilir. Eğer öğrenme katsayısı küçük bir değer seçilirse, minimum değere yakınsamak çok uzun vakit alabilir.



Şekil 12. Gradyan inişinin geometrik gösterimi

Bir fonksiyonu minimize etmek için, gradyan inişi algoritmasını hesaplarken, ne kadar veri kullandığımıza bağlı olarak üç farklı gradyan inişi çeşidinden bahsedebiliriz: Toplu gradyan inişi (batch gradient descent), stokastik gradyan inişi (stochastic gradient descent) ve mini-toplu gradyan inişi (mini-batch gradient descent).

Toplu gradyan inişi (Batch gradient descent)

Toplu gradyan inişi hesaplanırken eğitim kümesindeki tüm veriler kullanılır. Bütün veri kümesi için gradyan vektörü hesaplandıktan sonra, ağırlıklar güncellenir:

$$w_{\tau+1} = w_\tau - \eta \nabla_w E(w_\tau) \tag{33}$$

Burada, τ iterasyonun adım sayısını göstermektedir. $\nabla_w E(w_\tau)$ ise E fonksiyonunun w parametresine bağlı gradyanını belirtmektedir.

Her seferinde tüm veri için gradyan inişinin hesaplanması, bu algoritmanın çok yavaş ilerlemesine sebep olur. Ayrıca, belleğe sığamayacak büyük veri kümeleri için uygulanması zordur. Fakat her bir güncellemede tüm veri seti incelendiği için uygun şartlar altında, algoritma hata fonksiyonunun minimum değerine yakınsamayı garanti eder.

Stokastik gradyan inişi (Stochastic gradient descent)

Stokastik gradyan inişinde, ağa rastgele bir eğitim örneği x_i ve t_i sunulur. Ardından sunulan örnek için gradyan inişi hesaplanarak parametreler (ağırlık ve bias değerleri) güncellenir. Bu işlem tüm eğitim örnekleri için gerçekleştirilir ve örnekler rastgele seçilir.

$$w_{\tau+1} = w_\tau - \eta \nabla_w E(w_\tau; x_i; t_i) \tag{34}$$

Parametreler güncellenirken her yinelemede bir örnek ele alındığı için stokastik gradyan inişi algoritmasının hesaplanması hızlıdır ve uygulanması kolaydır. Dolayısıyla büyük veri kümeleri söz konusu olduğunda; maliyet fonksiyonunun minimum değerine, stokastik gradyan inişi toplu gradyan inişine göre daha hızlı yakınsar. Fakat, tek bir örnek üzerinden parametrelerde değişiklik yapıldığı için algoritma parametrelerde yüksek varyanslı güncellemelere neden olur.

Mini-toplu gradyan inişi (Mini-batch gradient descent)

Mini-toplu gradyan inişi, diğer iki gradyan inişi yönteminin de iyi özelliklerini almaktadır. Her bir parametre güncellemesini, n tane eğitim örneğinden oluşan eğitim setinin küçük bölümleri için gerçekleştirir:

$$w_{\tau+1} = w_\tau - \eta \nabla_w E\left(w_\tau; x_{(i:i+n)}; t_{(i:i+n)}\right) \tag{35}$$

Mini-toplu gradyan inişi, toplu gradyan inişine göre daha hızlı, stokastik gradyan inişine göre ise daha kararlıdır.

Gradyan inişi optimizasyon algoritmaları

Gradyan inişi algoritmasının dezavantajlarını gidermek için zaman içinde alternatif algoritmalar geliştirilmiştir. Bu bölümde bir fonksiyonu minimize etmek için birinci dereceden türevini içeren bazı algoritmalardan bahsedilmiştir. Verilen bu metodlar, hesaplama ve uygulama açısından ikinci dereceden türev içeren metodlara göre daha kolay olduğu gözlemlenmiştir.

Momentum

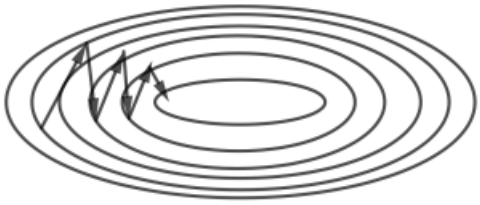
Gradyan inişi algoritması, maliyet fonksiyonunu en aza indirmek için oldukça kullanışlıdır ancak bazı problemlerle karşılaşılabilir. Momentumlu gradyan inişi, karşılaşılan bu problemlerin çözümü için gradyan inişine önceki gradyan değerlerini de eklemektedir. Bu gradyan inişinin daha hızlı ve daha düzgün bir şekilde yakınsamasına yardımcı olabilir. Momentunmlu gradyan başlangıcı aşağıdaki gibi bir güncelleme kuralıyla ifade edilir:

$$v_{t+1} = \gamma v_t + \eta \nabla_w E(w) \tag{36}$$

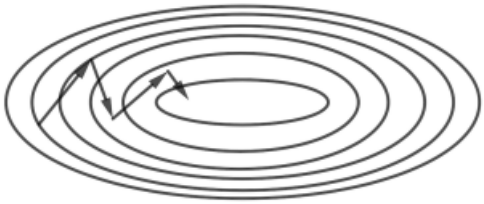
$$w_{t+1} = w_t - v_{t+1}$$

Burada γ sembolüne momentum sabiti denir. Momentum sabiti, önceki gradyan değerlerinin ne kadarının güncelleme işlemine dahil edileceğini belirler. Bu sabit, öğrenme sonuçlarını kontrol etmek ve gradyan güncellemelerinin daha düzgün ve hızlı bir şekilde ilerlemesine yardımcı olmak için kullanılır ve genellikle 0 ile 1 arasında değerler alır. Yüksek momentum değeri, gradyan inişlerini daha fazla düzleştirebilir, ancak aynı zamanda aşırı atlamalara neden olabilir. Bu nedenle genellikle 0.9 gibi orta bir değer tercih edilir ancak bu değer belirli bir probleme ve veri setine bağlı olarak değişebilir. Ayrıca burada iterasyon basamağı sayısı t ile gösterilmektedir. v_t ise momentum değerini belirtir ve başlangıçta genellikle 0 olarak alınır.

"Yapay sinir ağlarının farklı katmanlarında yer alan ağırlıkların kısmi türevleri genellikle farklı büyüklüktedir" (Aggarwal 2020). Gradyan inişinde kısmi türevi büyük olan ağırlıklar, gradyanın büyük bir değere sahip olduğu bölgeleri temsil eder ve bu ağırlıklar daha hızlı güncellenir. Aynı şekilde kısmi türevi küçük olan ağırlıklar, gradyanın küçük bir değere sahip olduğu bölgeleri temsil eder ve bu ağırlıklar daha yavaş güncellenir. Gradyan inişi, bir hata fonksiyonunun minimum noktasının bulunması için kullanılan bir teknik olup kısmi türevler, onun bir ağırlık parametresinin hata fonksiyonuna olan katkısı ölçer. Dolayısıyla gradyan inişi algoritması hata fonksiyonunun minimum noktasını tespit etmeye çalışırken aşırı salınım sorunlarıyla karşılaşılabilmektedir (Şekil 13).



Şekil 13. Momentum kullanılmadan gradyan inişi



Şekil 14. Momentum ile gradyan inişi

Momentum sabiti, işlemcilerin ardışık iterasyonlardaki kısmi türevlerinin işaretine bağlı olarak güncellenir. Bu, momentumlu grandyan inişinin davranışını şekillendiren önemli bir faktördür.

Eğer yapay sinir ağındaki parametrelerin belirli bir kısmi türevleri ardışık iterasyonlarda aynı işarete sahipse momentum değeri artar. Bu parametrelerin güncellenmesinin daha büyük olmasına ve hızlı bir şekilde yakınsamaya katkıda bulunmasına neden olur.

Eğer yapay sinir ağındaki parametrelerin belirli bir kısmi türevleri ardışık iterasyonlarda farklı işaretlere sahipse momentum değeri azalır, bu parametrenin güncellenmesinin daha küçük olması ve eğimin azalmasının daha yavaş yakınsamasına neden olur. Böylece, momentum algoritması gradyan inişine göre daha hızlı yakınsar ve daha az salınım yapar (Şekil 14).

Momentum sabiti (γ) seçimi dikkatli olunmalıdır. Yüksek bir momentum sabiti büyük hızlanmaya neden olabilir, bu da yakınsama miktarını artırabilir ve aşırı sallanmalara yol açabilir. Düşük bir momentum sabiti ise daha yavaş bir yakınsamaya ve daha düşük frekansta salınım yapmaya yol açabilir. Bu nedenle momentum sabiti problemi ve veri setine bağlı olarak uygun bir şekilde ayarlanması gerekir.

Nesterov hızlandırılmış gradyan

Momentum algoritması geçmiş gradyan vektörlerini ekleyerek gradyan inişine göre daha büyük adımlar atar ancak bu özelliği nedeniyle global minimum noktasını geçme riski taşır ve salınım yapabilir. Momentum algoritmasının bu eksikliklerini gidermek için Nesterov hızlandırılmış gradyan algoritması kullanılır (Nesterov 1983).

Nesterov Hızlandırılmış Gradyan Algoritması, momentum algoritmasının iyileştirilmiş bir versiyonudur. Temel fikir gradyan vektörünün gelecekteki tahmini konumunu hesaplayarak daha iyi bir yaklaşım yapmaktır. Bu yaklaşım momentumlu gradyan inişindeki yukarı ivmeyi düzeltmeye yardımcı olur ve global minimumu geçme olasılığını azaltır.

Nesterov Hızlandırılmış Gradyan Algoritması'nın güncelleme kuralı aşağıdaki gibidir:

$$v_{t+1} = \gamma v_t + \eta \nabla_w E(w - \gamma v_t) \tag{37}$$

$$w_{t+1} = w_t - v_{t+1}$$

Nesterov Hızlandırılmış Gradyan Algoritması, gradyan hesaplarken mevcut konumun gradyanı yerine bir sonraki konumun gradyanını dikkate alır. Bu algoritmanın daha iyi bir tahmin yapmasına ve daha iyi yakınsamasına yardımcı olur. Dolayısıyla denklem (37)'de v_{t+1}

ile verilen terimi hesaplarken, mevcut konumun gradyanı yerine bir sonraki konumun gradyanı dikkate alınmaktadır. Verilen γ, momentum sabiti 0.9 olarak alınmaktadır.

Momentum algoritması ve Nesterov Hızlandırılmış Gradyan Algoritması arasındaki temel fark, gradyan güncellemelerini hesaplarken hangi sırayı takip ettiğidir. Momentum algoritmasında, mevcut konumun gradyanı hesaplandıktan sonra önceden hesaplanmış olan tüm gradyanların toplamı yönünde parametrelerde güncelleme yapılır. Nesterov hızlandırılmıış gradyan algoritmasında ise, önce geçmiş gradyanların toplamı yönünde bir ilerleme sağlıyor, ardından bu yeni konumdaki gradyan hesaplanarak parametreler güncellenir. Algoritmanın gelecekteki konumunun gradyanını dikkate alması, algoritmanın aşırı büyük adımlardan kaçınmasına yardımcı olur. Dolayısıyla performans daha dikkatli ve kontrollü bir şekilde devam eder.

Adagrad

Gradyan inişi, momentum algoritmalarında öğrenme katsayısı sabit bir değer olarak belirlenir ve tüm iterasyon boyunca aynı kalır. Ancak Adagrad algoritması farklı bir yaklaşım benimser. Adagrad (Duchi vd 2011) algoritmasında öğrenme katsayısı, her bir parametre ve her bir iterasyon adımı için değiştirilmektedir. Adagrad algoritması bu özelliği, her bir parametre için farklı öğrenme katsayılarına izin vererek daha hızlı yakınsama sağlar. Adagrad algoritması, öğrenme oranını parametrelere uyarlar, seyrek parametreler için daha büyük güncellemeler ve sık parametreler için daha küçük güncellemeler gerçekleştirir. Bu nedenle seyrek verilerle başa çıkmak için çok uygundur.

Ağırlık vektöründeki her bir ağırlık değerini w_i ile gösterirsek, t. İterasyon adımında ki hata fonksiyonunun gradyanı $g_{t,i}$ şu şekilde ifade edilebilir:

$$g_{t,i} = \nabla_{w_t} E\left(w_{t,i} \right) \tag{38}$$

Adagrad algoritması, her bir parametre için öğrenme katsayısını belirlerken geçmiş gradyanları da içermektedir. Adagrad algoritmasının parametreleri güncelleme kuralı şu şekildedir:

$$w_{t+1,i} = w_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} g_{t,i} \tag{39}$$

Burada $G_t \in \mathbb{R}^{d \times d}$ köşegen matris olup $G_{t,ii}$ bu matrisin (i, i) köşegen elemanıdır ve w_i parametresine göre t. adıma kadar olan geçmiş gradyanların karelerinin toplamına eşittir. ϵ ise sıfıra bölünmeyi önlemek için atanan bir sabittir ve genellikle 10^{-8} gibi küçük bir değer alınır.

Adagrad algoritması öğrenme katsayısını kendi belirlediği için, önceki algoritmalara göre daha avantajlıdır. Ancak Adagrad'ın dezavantajı, geçmiş gradyanların karelerinin paydası tarafından toplandığı ve bu toplamının arttığıdır. Bu nedenle sürekli paydanın değeri büyür ve böylece öğrenme katsayısının değeri giderek azalır.

Sonuç olarak, öğrenme çok küçük olduğunda parametre güncellemeleri de çok küçük olur. Bu programın yavaşlamasına ve hatta durmasına neden olabilir. Bu sorunun aşılması için Adadelta, RMSprop ve Adam gibi yeni yöntemler geliştirildi. Bu değişiklikler, payda değerinin daha etkili bir şekilde kontrol edilmesine yardımcı olur ve öğrenmenin hızlı bir şekilde küçülmesini önler.

Adadelta

Adagrad'ın sahip olduğu öğrenme katsayısı problemini çözmek için, Adadelta algoritması (Zeiler 2012) geliştirilmiştir. "Önceki gradyanların karelerini verimsiz bir şekilde depolamak yerine; gradyanların toplamı, tüm geçmiş gradyanların karelerinin azalan bir ortalaması olarak, özyinelemeli bir şekilde tanımlanır" (Ruder 2016).

Adadelta algoritmasının parametreleri güncelleme kuralı şu şekildedir:

$$\Delta w_t = -\frac{RMS[\Delta w]_{t-1}}{RMS[g]_t} g_t \tag{40}$$

$$w_{t+1} = w_t + \Delta w_t$$

Burada g_t, t. iterasyon basamağındaki gradyan değerini, $RMS[g]_t$ ise t. adıma kadar olan gradyanların karelerinin ortalama karekök hatasını hesaplamaktadır:

$$RMS[g]_t = \sqrt{E[g^2]_t + \epsilon} \tag{41}$$

Burada verilen ϵ değeri, paydanın sıfıra eşit olmasını engelleyen sabittir. $E[g^2]_t$ ise mevcut konumdaki gradyanın karesinin ve geçmiş gradyanların karelerinin üssel olarak azalan bir ortalamasıdır ve şu şekilde tanımlanır:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \tag{42}$$

Burada γ sabiti, momentum algoritmasındaki momentum sabitine benzerdir ve genellikle 0.9 alınır.

Benzer şekilde, $RMS[\Delta w]_t$ ve $E[\Delta w^2]_t$ ifadeleri sırasıyla, parametre değişimlerinin karelerinin ortalama karekök hatasını ve parametre değişimlerinin karelerinin üssel olarak azalan ortalamasını belirtir:

$$RMS[\Delta w]_t = \sqrt{E[\Delta w^2]_t + \epsilon} \tag{43}$$

$$E[\Delta w^2]_t = \gamma E[\Delta w^2]_{t-1} + (1-\gamma)\Delta w_t^2 \quad (44)$$

RMSprop

RMSprop algoritması, Adadelta gibi, Adagrad algoritmasının sürekli azalan öğrenme katsayısı sorununu çözmek için geliştirilmiştir. Bu algoritma, Adadelta algoritması ile aynı zaman aralığında, Geoff Hinton tarafından ortaya konulmuştur (Hinton vd 2012):

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2 \quad (45)$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

γ sabitinin 0.9, η öğrenme katsayısının da 0.001 olarak alınması Hinton tarafından önerilmiştir (Hinton vd 2012).

Adaptif moment tahmin (Adaptive moment estimation - adam)

Adaptif moment tahmin algoritması (Kingma vd 2014), RMSprop ve momentum algoritmalarının özelliklerini taşır. Adam algoritması, geçmiş gradyanların üssel olarak azalan m_t ortalamasını ve geçmiş gradyanların karelerinin üssel olarak azalan v_t ortalamasını kullanır:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t \quad (46)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$$

Burada, m_t gradyanların birinci momentini, v_t ise gradyanların ikinci momentini belirtmektedir. Başlangıçta m_t ve v_t momentleri sıfır vektörleri olarak alınırlar. Momentlere başlangıçta 0 değeri atandığı için, iterasyonun başlarında momentler sıfıra yakın değerler alır. Aynı zamanda β_1 ile β_2, 1'e yakın değerler aldığında da, momentler sıfıra meyilli olurlar. Bu nedenlerden dolayı, momentler normalize edilir:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad (47)$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$\hat{m}_t$ ve $\hat{v}_t$ değişkenlerini hesapladıktan sonra, parametreler denklem (48) kullanılarak güncellenir:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \quad (48)$$

Yazar (Kingma vd 2014), β_1 ve β_2 değerlerinin sırasıyla, 0.9 ve 0.999 olarak alınmasını önermiştir. Ayrıca ϵ, diğer algoritmalarda olduğu gibi 10^{-8} olarak alınır.

Radyal Tabanlı Fonksiyon Sinir Ağı

Radyal tabanlı fonksiyon ağları, çok katmanlı algılayıcılı sinir ağı yapılarına benzer şekilde girdi katmanı, gizli katman ve çıktı katmanından oluşur. Girdi katmanı girdi vektör uzayı, çıktı katmanı da örüntü sınıfları ile ilişkilidir. Böylelikle tüm yapı, gizli katmanın yapısı ve gizli katman ile çıktı katmanı arasındaki ağırlıkların belirlenmesine indirgenir. Gizli katmandaki nöronların aktivasyon fonksiyonları bir c_i merkezi ve σ_i bant genişliği ile belirlenir. Aktivasyon fonksiyonu;

$$\varphi_i(x) = exp\left[-\frac{\|x - c_i\|^2}{2\sigma_i^2}\right] \tag{49}$$

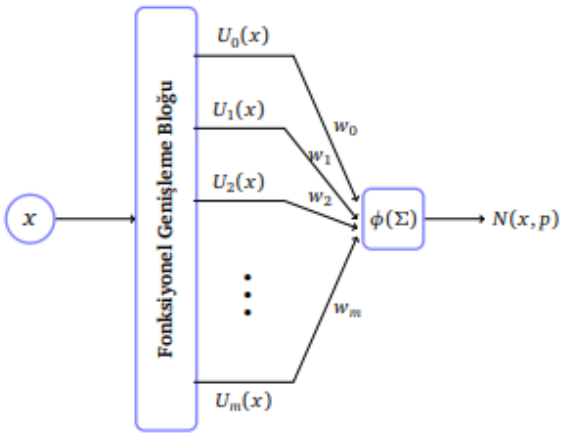
eşitliği ile tanımlanan bir Gauss eğrisidir. Çıktı katmanındaki j. Nöronun çıktısı için genel eşitlik ise şu şekildedir:

$$y_j(x) = \sum_{i=1}^{N} w_{ij}\varphi_i(x) + b_j \tag{50}$$

Burada w_{ij}, i. gizli nöron ve j. çıktı nöronu arasındaki ağırlık değeridir (Verleysen vd 1994; Park vd 2002).

Tek Katmanlı Fonksiyonel Bağlantılı Yapay Sinir Ağı (Sıngle-Layer Functıonal Lınk ANN)

Tek katmanlı fonksiyonel bağlantılı yapay sinir ağı (FLANN), gizli katmanı olmayan bir sinir ağı çeşididir. FLANN modeli, Pao ve Philips tarafından tanıtılmıştır (Pao vd 1995). Fonksiyonel bağlantılı yapay sinir ağı bir girdi nöronu, bir çıktı nöronu ve gizli katman yerine bir fonksiyonel genişleme bloğundan oluşmaktadır (Şekil 15). Ağa sunulan girdiler ortogonal, trigonometrik, üstel gibi polinomlar kullanılarak genişletilir. Ardından, fonksiyonel genişleme bloğu tarafından üretilen genişletilmiş girdiler, ağırlık değerleri ile çarpılır. Elde edilen çarpımlar toplanır ve bu toplamın aktivasyon fonksiyonu değeri hesaplanarak ağın çıktı değeri elde edilir.



Şekil 15. Tek katmanlı fonksiyonel bağlantılı yapay sinir ağı modeli

$x = (x1, x2, \dots, xh)^T$ girdi vektörü, $(U_0, U_1, U_2, \dots, U_m)$ genişletilmiş girdi verileri ve $(w_0, w_1, w_2, \dots, w_m)$ ağırlık değerleri olmak üzere, yapay sinir ağı çıktısını şu şekilde yazabiliriz:

$$z = \sum_{j=0}^{m} w_j U_j(x) \tag{51}$$

$$N(x,p) = \phi(z)$$

Burada, ϕ aktivasyon fonksiyonunu belirtmektedir. Yapay sinir ağının ağırlık değerleri ise $p = [w_0, w_1, w_2, \dots, w_m]$ vektörü ile gösterilmiştir.

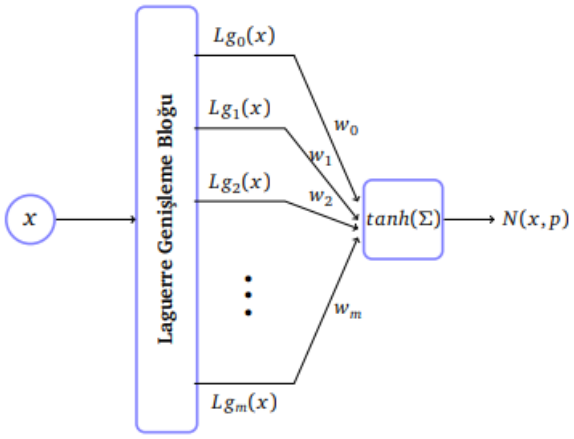
Tek katmanlı fonksiyonel bağlantılı sinir ağının parametre sayısı, çok katmanlı algılayıcının parametre sayısına göre daha azdır. Bu nedenle, FLANN daha hızlı eğitilir ve işlem kolaylığı sağlar. Dolayısıyla, uygulanması çok katmanlı algılayıcıya göre daha kolaydır.

Tek katmanlı fonksiyonel bağlantılı yapay sinir ağı modelleri

Bu kısımda Laguerre, Chebyshev ve Legendre yapay sinir ağlarından bahsedilecektir.

Laguerre yapay sinir ağı (Laguerre neural network)

Tek katmanlı Laguerre yapay sinir ağı modeli, geleneksel yapay sinir ağlarından farklı bir yapı kullanır ve Laguerre ortogonal polinomlarını kullanan fonksiyonel genişleme bloğu içerir (Şekil 16). Burada Laguerre polinomları, girilen verileri daha yüksek boyutlu bir uzaya dönüştürür.



Şekil 16. Tek katmanlı Laguerre yapay sinir ağı modeli

Laguerre polinomları, denklem (52) ile verilen Laguerre diferansiyel denklemini sağlayan ortogonal polinomlardır:

$$xy''(x) + (1-x)y'(x) + ny(x) = 0 \tag{52}$$

Aşağıda verilen rekürans bağıntısı ile yüksek mertebeden Laguerre polinomları elde edilebilir:

$$\begin{aligned} &Lg_0(x) = 1 \\ &Lg_1(x) = -x + 1 \\ &Lg_{n+1}(x) = (2n + 1 - x)Lg_n(x) - n^2 Lg_{n-1}(x), \quad (n \geq 1) \end{aligned} \tag{53}$$

Şekil 16'da, ilk m Laguerre polinomunu kullanan bir Laguerre yapay sinir ağı gösterilmiştir. Burada, girdi olarak h boyutlu $x = (x1, x2, \dots, xh)^T$ girdi vektörü ele alınırsa, Laguerre polinomları kullanılarak elde edilen genişletilmiş girdi değerleri şu şekilde yazılabilir:

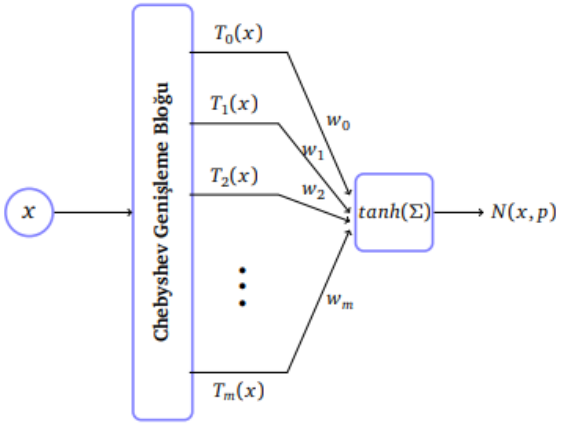
$$\begin{aligned} [Lg_0(x_1), Lg_1(x_1), Lg_2(x_1), \dots, Lg_m(x_1); \quad Lg_0(x_2), Lg_1(x_2), Lg_2(x_2), \dots, Lg_m(x_2); \\ \dots; Lg_0(x_h), Lg_1(x_h), Lg_2(x_h), \dots, Lg_m(x_h)] \end{aligned} \tag{54}$$

Laguerre yapay sinir ağı modelinde, $tanh$ aktivasyon fonksiyonu kullanılmıştır. x girdi vektörü ve $p = [w_0, w_1, w_2, \dots, w_m]$ ağırlık vektörüne göre yapay sinir ağının çıktısı şu şekilde hesaplanabilir:

$$N(x,p) = tanh\left(\sum_{j=0}^{m} w_j Lg_j(x)\right) \tag{55}$$

Chebyshev yapay sinir ağı (Chebyshev neural network)

Tek katmanlı Chebyshev yapay sinir ağı modeli, Şekil 17 ile gösterilmektedir. Bir girdi nöronu, bir çıktı nöronu ve gizli katman yerine fonksiyonel genişleme bloğundan oluşmaktadır. Bu genişleme bloğunda Chebyshev polinomları kullanılmaktadır.



Şekil 17. Tek katmanlı Chebyshev yapay sinir ağı modeli

Bu yapay sinir ağı modelinde, Chebyshev polinomları kullanılarak genişletilmiş girdi değerleri elde edilir. Yapay sinir ağına girdi vektörü olarak da h boyutlu $x = (x1, x2, \dots, xh)^T$ vektörü verilmiştir.

İlk iki Chebyshev polinomu şu şekildedir:

$$T_0(x) = 1 \tag{56}$$

$$T_1(x) = x$$

Diğer Chebyshev polinomları ise aşağıda verilen rekürans formülü yardımıyla elde edilebilir:

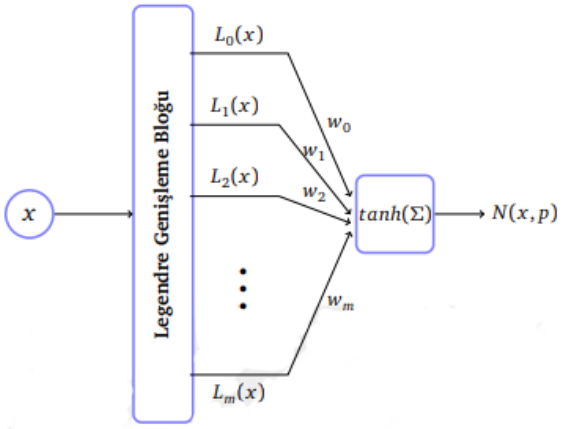
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{57}$$

İlk m Chebyshev polinomu ile m boyutlu $p = [w_0, w_1, w_2, \dots, w_m]$ ağırlık vektörü kullanılarak elde edilen, Chebyshev yapay sinir ağı modelinin çıktısını şu şekilde yazabiliriz:

$$N(x,p) = tanh\left(\sum_{j=0}^{m} w_j T_j(x)\right) \tag{58}$$

Legendre yapay sinir ağı (Legendre neural network)

Tek katmanlı Legendre yapay sinir ağı modeli, Legendre polinomlarını kullanarak girdi hücrelerini daha yüksek boyutlu bir uzaya dönüştürür (Şekil 18). Böylece, gizli katmana ve girdi katmanını gizli katmana bağlayan ağırlık değerlerine ihtiyaç kalmaz.



Şekil 18. Tek katmanlı Legendre yapay sinir ağı modeli

Legendre diferansiyel denkleminin bir çözümü olan ortogonal Legendre polinomları, aşağıda verilen rekürans bağıntısı ile elde edilir:

$$L_0(x) = 1$$
$$L_1(x) = x \tag{59}$$
$$L_{n+1}(x) = \frac{1}{n+1}[(2n+1)xL_n(x) - nL_{n-1}(x)]$$

h boyutlu $x = (x1, x2, \dots, xh)^T$ girdi vektörü yapay sinir ağına sunulursa, Legendre polinomları kullanılarak oluşturulan genişletilmiş girdi verileri şu şekilde yazılabilir:

$$[L_0(x_1), L_1(x_1), L_2(x_1), \dots, L_m(x_1);\ L_0(x_2), L_1(x_2), L_2(x_2), \dots, L_m(x_2);$$
$$\dots; L_0(x_h), L_1(x_h), L_2(x_h), \dots, L_m(x_h)] \tag{60}$$

x girdi değerine ve parametrelere bağlı Legendre yapay sinir ağı modelinin çıktısı, $tanh$ aktivasyon fonksiyonu kullanılarak şu şekilde hesaplanır:

$$N(x,p) = tanh\left(\sum_{j=0}^{m} w_j L_j(x)\right) \tag{61}$$

ARAŞTIRMA BULGULARI

Hopfield Sinir Ağı

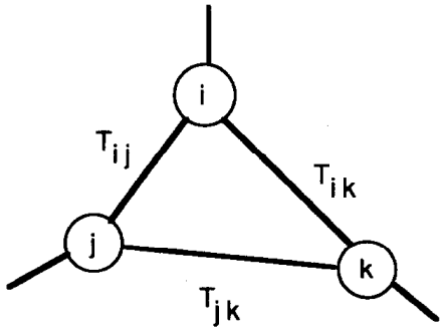
Literatürde yapay sinir ağları ile diferansiyel denklemlerin nümerik çözümlerinin elde edilmesi üzerine görülen ilk çalışmada, Hopfield Sinir Ağı modeli ile sonlu fark denklemlerinin çözümü gerçekleştirilmiştir (Lee vd 1990). Böylelikle yapay sinir ağları ile fonksiyonlara nümerik olarak yaklaşılabileceği gösterilmiştir.

Sinir minimizasyon algoritmaları

i. Sinir Ağlarının Toplu Hesaplanması

Sinir ağları bireysel işlemcilerden ve işlemciler arasındaki ara (gizli) bağlantılardan oluşur. Örnek olarak basit bir sinir ağının şematik gösterimi Şekil 19'da gösterilmiştir. Her işlemciye, biyolojik sinir sistemlerinin de bir benzeri olan nöron denir. Her nöron, 1 ve 0 (veya 1 ve -1) ikili sayıları ile temsil edilen açık ve kapalı olmak üzere iki farklı duruma sahip olabilir. Tüm nöronların çalışması aynıdır. Tipik olarak, her nöron ağırlıklı ara bağlantılar aracılığıyla diğer tüm nöronlardan gelen tüm sinyalleri toplar, toplanan sinyali 0 veya 1 (veya -1 veya 1) olarak eşikler ve eşikli çıktıya göre durumunu değiştirir. Bu işlem ağdaki her nöronda aynı anda veya rastgele gerçekleşir. Bu nedenle, ağdaki toplam nöron sayısı büyükse bireysel nöron durumundan oluşan tüm ağın durumu dinamik olarak değişir ve ağ işbirlikçi etkiler sergiler.

Yukarıda açıklanan sinir ağı hesaplama için kullanılır. Nöronların ikili durumları, bazı değişkenlerin ikili gösterimi olarak tanımlanabilir. Nöronlar arasındaki ara bağlantı güçleri belirli bir problemin bilgisini temsil edebilir.



Şekil 19. Basit bir sinir ağının şematik diyagramı. T_{ij},T_{ik}, ve T_{jk} (i,j),(i,k), ve (j,k) nöron çiftleri arasındaki bağlantı güçleridir

İlk deneme çözümünü temsil eden ağ için bir başlangıç deneme durumu, sinir dinamiğine göre ağın son durumuna yakınlaşabilir ve sorunun çözümünü verebilir. Bu hesaplama yöntemi, nöronlar arasındaki kolektif etkileşime dayanır ve yüksek derecede paralellik gösterir. Sinir ağının bir başka özelliği de her bir nöronun işlenmesinin son derece basit olmasıdır, bununla birlikte çok sayıda nöron ve ara bağlantı muazzam bir hesaplama gücü sağlar.

ii. Sürekli Sinir Minimizasyon Algoritması

Hopfield modeli (Hopfield 1982), birbirine bağlı nöronlar arasındaki doğrusal olmayan dinamik etkileşeme dayanmaktadır. Her bir nöronun durumu, 0 ile 1 arasında değişen V_i çıktıları ile belirlenir. Hopfield modelinde nöronların dinamikleri, yani nöron durumlarının güncelleme kuralları aşağıdaki gibi özetlenebilir. Sürekli modelde, nöronlar durumlarını aşağıdaki dinamik denklemlerine göre değiştirirler:

$$\frac{dU_i}{dt} = \sum_j T_{ij} V_j \tag{62}$$

$$V_i = g(U_i) \tag{63}$$

Burada t, güncelleme parametresine karşılık gelen sürekli zamandır. T_{ij} ara bağlantı gücüdür ve $g(x)$ lineer olmayan bir fonksiyondur:

$$g(x) = \frac{1}{2}\left[1 + tanh\left(\frac{x}{x_0}\right)\right] \tag{64}$$

0 ile 1 arasında sınırlanan monoton olarak artan bir fonksiyondur. x_0 bir sabittir. Hoplield, $T_{ij} = T_{ji}$ ise, sürekli modelde nöronlar tarafından tanımlanan bir enerji fonksiyonunu minimize edecek şekilde durumlarını değiştirirler;

$$E = -\left(\frac{1}{2}\right)\sum_i \sum_j T_{ij} V_i V_j \tag{65}$$

ve bu fonksiyonun minimum değerinde durur. Enerji fonksiyonu (65)'i minimize etmek için denklem (62), (63) ve (64) ile temsil edilen güncelleme kuralı ile oldukça paraleldir (Psaltıs vd 1985).

Yukarıda açıklanan Hopfield modeli yalnızca simetrik T matrislere sahip ikinci dereceden enerji fonksiyonlarının minimizasyonuna uygulanabilir. Bununla birlikte keyfi enerji fonksiyonları için Hopfield modeli de araştırılmıştır (Koch 1986).

Aşağıdaki şekilde yazılabilen bir minimizasyon problemi için enerji fonksiyonu verilsin:

$$E = F(V_i) \tag{66}$$

Burada F, V_i değişkenlerinin tekil olmayan ve sınırlı bir fonksiyonudur ve V_i' ye göre kısmi türevlerin iyi tanımlı olduğu varsayılır. E'nin her zaman pozitif veya sıfır olduğu varsayılır, bu da E sınırlı olduğu için geçerlidir. Eşitlik (66) ile tanımlanan enerji fonksiyonunun zaman içindeki değişimi şu şekilde verilmektedir:

$$\frac{dE}{dt} = \sum_i \frac{\partial F}{\partial V_i} \frac{dV_i}{dt} \tag{67}$$

Güncelleme kuralı şu şekilde tanımlanır:

$$\frac{dU_i}{dt} = -\frac{\partial F}{\partial V_i} \tag{68}$$

$$V_i = h(U_i) \tag{69}$$

Burada $h(x)$, 0 ile 1 arasında sınırlandırılmış monoton olarak artan bir fonksiyon ve U_i ara değişken olmak üzere, eşitlik (68) ve (69)'un enerji fonksiyonunu aşağıdaki gibi minimize ettiği kanıtlanabilir. Eşitlik (69)'dan şunu elde ederiz:

$$\frac{dV_i}{dt} = h'(U_i)\left(\frac{dU_i}{dt}\right) \tag{70}$$

Burada h', x'e göre türevdir. $h'(x)$ her zaman pozitif veya sıfırdır çünkü $h(x)$ monoton olarak artan bir fonksiyondur. Eşitlik (68) ve (69) ve eşitlik (70) ile temsil edilen güncelleme kuralı eşitlik (67)'de kullanılırsa;

$$\frac{dE}{dt} = -\sum_i h'(U_i)\left(\frac{dU_i}{dt}\right)^2 \tag{71}$$

elde edilir. Eşitlik (71) her zaman negatif veya sıfırdır. Bu nedenle enerji fonksiyonunun eşitlik (68) ve (69)'daki güncelleme kuralına göre zaman içindeki değişimi, enerji fonksiyonunun minimize edilmesini garanti eder.

iii. Ayrık Sinir Minimizasyon Algoritması

Çeşitli enerji fonksiyonlarını minimize etmek için ayrık sinir algoritması geliştirilmiştir (Lee 1989). Enerji fonksiyonunun, durum değişkenlerinin keyfi bir polinom fonksiyonu olduğu varsayılır. 1 ve -1 değerlerine sahip ikili değişkenler durum değişkenleri olarak kabul edilir. Enerji fonksiyonu şu şekilde tanımlanabilir:

$$E = F(\{B_1, B_2, \ldots, B_N\}). \tag{72}$$

Burada B'ler durum değişkenleridir ve toplam durum değişken sayısı N'dir.

En genel durum için kısmen eş zamanlı minimizasyon dikkate alınır. Tamamen eş zamanlı veya tamamen eşzamansız minimizasyonlar genel durumun spesifik örnekleridir. Her adımda M durum değişkeninin rastgele seçildiğini ve minimizasyon işleminin M durum değişkenlerinin eşzamanlı olarak güncellenmesi ve diğer tüm durum değişkenlerinin değişmeden bırakılmasıyla gerçekleştirildiğini varsayalım. M, 1'den N'ye kadar herhangi bir tam sayı olabilir ve M, 1'e veya N'ye eşitse minimizasyon algoritması tamamen eşzamansız veya tamamen eşzamanlı hale gelir. Seçilen durum değişkenlerinin indislerinden oluşacak şekilde bir P kümesi tanımlanır. Değişmeyen durum değişkenlerinin indislerini temsil etmek üzere başka bir $P' = \{1,2, \dots, N\} - P$ kümesi tanımlanmıştır. Güncellenmiş B_i' durum değişkenleri ve ΔB_i durum değişkenlerinin değişimi şu ilişkiyi sağlar:

$$B_i' = B_i + \Delta B_i \tag{73}$$

Burada $i \in P$. Güncellenen durum değişkenleri de 1 ve -1 değerlerine sahip ikili değişkenlerdir. Bu nedenle ΔB_i'nin olası değerleri şunlardır:

$$\Delta B_i = -2,0,2 \tag{74}$$

Eşitlik (73) ile verilen güncellenmiş durum değişkenleri nedeniyle ΔE enerjisindeki artan değişim, eşitlik (72) ile tanımlanan enerji fonksiyonunu en aza indiren bir algoritma geliştirmek için dikkate alınır. ΔE şu şekilde tanımlanır:

$$\Delta E = E\left[\{B_i', B_j\}\right] - E\left[\{B_i, B_j\}\right] \tag{75}$$

Burada $i \in P$ ve $j \in P'$ ve eşitlik (73) kullanılarak şu hale gelir:

$$\Delta E = E\left[\{B_i + \Delta B_i, B_j\}\right] - E\left[\{B_i, B_j\}\right] \tag{76}$$

Eşitlik (76)'nın sağ tarafındaki ilk terim, çeşitli değişkenlerde Taylor serisi olarak genişletilebilir çünkü E, durum değişkenlerinin bir polinomudur. Bu nedenle artan enerji değişimleri ΔE şu şekilde yazılabilir:

$$\Delta E = \sum_{m=1} \sum_{i1 \in P} \dots \sum_{im \in P} \left(\frac{1}{m!}\right) [\Delta B_{i1} \dots \Delta B_{im}] D[B_{i1} \dots B_{im}] E \tag{77}$$

Burada $D[B_{i1} \dots B_{im}]E$, tüm $i \in P$ için $\Delta B_i = 0$'daki $B_{i1} \dots B_{im}$ durum değişkenlerine göre kısmi türevdir. Eşitlik (77)'nin toplamındaki toplam terim sayısı sonludur çünkü E bir polinomdur.

Eşitlik (77)'deki değişim çarpımlarını lineer formlara indirgemek için aşağıdaki bağıntılar türetilmiştir. Rastgele bir B durum değişkeni ve pozitif bir n tamsayısı için $(\Delta B)^n$ şu koşulları sağlar:

$$(\Delta B)^n = (-2B)^{n-1}\Delta B \tag{78}$$

Bu da şu şekilde kanıtlanabilir:

ΔB sıfır ise eşitlik (78) otomatik olarak sağlanır. ΔB 2 ise B ve B' eşitlik (73)'e göre -1 ve 1 olmalıdır. Bu durumda eşitlik (78)'in sağ tarafi sol tarafla aynı olur:

$$[(-2)(-1)]^{n-1}[2] = 2^n = (\Delta B)^n \tag{79}$$

Diğer taraftan $\Delta B\ -2$ ise B ve B' eşitlik (73)'e göre 1 ve -1 olmalıdır. Dolayısıyla eşitlik (78)'in sağ tarafı şu hale gelir:

$$[(-2)(1)]^{n-1}[-2] = [-2]^n \tag{80}$$

Bu da $\Delta B = -2$ durumunda eşitlik (78)'in sol tarafı ile aynıdır.

Şimdi $A\Delta B_{i1} \dots \Delta B_{im}$ tarafından verilen ve keyfi bir A katsayısına sahip bir değişim çarpımını ele alalım; varsayalım ki $m > 1$, şu şekilde yazılabilir:

$$A\Delta B_{i1} \dots \Delta B_{im} = |A| \left(\frac{1}{2}\right) [-\{S(A)\Delta B_{i1} - B_{i2} \dots B_{im}\}^2 + (\Delta B_{i1})^2 + (\Delta B_{i2} \dots B_{im})^2] \tag{81}$$

$$\leq |A| \left(\tfrac{1}{2}\right) [(\Delta B_{i1})^2 + (\Delta B_{i2} \dots B_{im})^2] \tag{82}$$

Burada $S(A)$, A'nın işaretidir. Eşitlik (81)'deki ilk terim her zaman negatif veya sıfırdır ve bunu eşitlik (82) takip eder. Eşitlik (82)'deki ikinci terim, eşitlik (81)'deki sol terimden daha az sayıda değişikliğin bir sonucudur. Bu nedenle bu teknik, eşitlik (81)'deki değişikliklerin çarpımını bireysel değişikliklerin toplamına indirgemek için iteratif olarak uygulanabilir ve şu şekilde verilir:

$$A\Delta B_{i1} \dots \Delta B_{im} \leq |A| \left[\sum_{j=1}^{m-1} 2^{-j} \left(\Delta B_{ij}\right)^{2^j} + 2^{-(m-1)} (\Delta B_{im})^{2^{(m-1)}}\right] \tag{83}$$

Denklem (83), $\{i1, \dots, im\}$ indislerinin m döngüsel sıralaması dikkate alınarak simetrik hale getirilebilir. Denklem (83) şu şekilde verilir:

$$A\Delta B_{i1} \dots \Delta B_{im} \leq \left(\frac{1}{m}\right) |A| \sum_{k=1}^{m} \left[\sum_{j=1}^{m-1} 2^{-j} (\Delta B_{ik})^{2^j} + 2^{-(m-1)} (\Delta B_{ik})^{2^{(m-1)}}\right] \tag{84}$$

Eşitlik (78) kullanılırsa, eşitlik (84) şu şeklide yazılabilir:

$$A\Delta B_{i1} \dots \Delta B_{im} \leq |A| I(m) \sum_{k=1}^{m} B_{ik} \Delta B_{ik} \tag{85}$$

Burada,

$$I(m) = -\left(\frac{1}{m}\right)\left[\sum_{j=1}^{m-1} 2^{\left(2^j - j - 1\right)} + 2^{\left(2^{(m-1)} - m\right)}\right]. \tag{86}$$

Eşitlik (85) ve (86), eşitlik (77)'de kullanılırsa, artan enerji değişimi şu şekilde verilir:

$$\Delta E \leq -\sum_i G_i \Delta B_i \tag{87}$$

Burada

$$G_i = -\left[B_i\left\{\sum_{m=1}\sum_{i1\in P}\dots\sum_{im\in P}\left(\frac{1}{m!}\right) I(m+1) \times |D[B_i B_{i1} \dots B_{im}]E|\right\} + D[B_i]E\right]. \tag{88}$$

Eşitlik (87)'den açıkça görülmektedir ki eğer,

$$G_i \Delta B_i \geq 0 \qquad \forall i \in P \tag{89}$$

artan enerji değişimi her zaman negatif veya sıfırdır. Bu nedenle eşitlik (89) durum değişkenlerini birçok iterasyon sınırında enerji fonksiyonunu minimize edecek şekilde günceller. Pozitif G_i için ΔB_i pozitif veya sıfır ise eşitlik (89) sağlanır. $B_i = 1$ ise $B_i' = 1$ olmalıdır çünkü $\Delta B_i = 0$ tek çözümdür. Ancak $B_i = -1$ ise $B_i' = 1$ olmalıdır çünkü bu artan enerji değişimini $\Delta B_i = 0$ koşulunu kullanmaktan daha negatif hale getirir. Bu nedenle G_i pozitifse güncellenen değer 1 olur ve bu da G_i değerinin işaretiyle aynıdır. G_i negatifse B_i' için güncellenmiş değerin -1 olduğunu göstermek için yukarıdaki argüman uygulanabilir. Bu G_i'nin işareti ile aynıdır. G_i sıfır ise B_i' herhangi bir değere sahip olabilir. Yukarıdaki sonucu özetleyecek olursak güncelleme kuralı şu şekilde yazılabilir:

$$B_i' = \mathrm{T}(G_i) \quad \forall i \in P \tag{90}$$

Burada T, $x > 0$ ise $T(x) = 1$ ve $x \leq 0$ ise $T(x) = -1$ ile tanımlanan bir birim adım fonksiyonudur. Denklem (90), durum değişkenlerinin keyfi polinom türlerinden oluşan enerji fonksiyonlarını kısmen senkronize bir şekilde minimize eden genel ayrık sinir algoritmasıdır.

Örnek 1: $u' = f(u)$ için Sürekli Algoritma

Sinir minimizasyon algoritmalarının diferansiyel denklemleri sayısal olarak çözmek ve nasıl kullanılabileceğini açıklamak için basit bir örnek ile ele alınmıştır (Lee vd 1990).

Diferansiyel denklem şu şekilde seçilir:

$$\frac{du}{dx} = f(u) \tag{91}$$

Burada f, u'nun bir polinom fonksiyonudur. Eşitlik (91) için sonlu fark denklemi şu şekilde verilir:

$$\frac{(U_{a+1} - U_{a-1})}{2h} = f(U_a) \tag{92}$$

Burada h ayrıklaştırma için en büyük boyuttur ve U_a, ağ indeksi a'ya karşılık gelen ayrık değişkendir. Sürekli sinir algoritması için U_a değişkeninin ikili gösterimi şu şekilde verilebilir:

$$U_a = \sum_{s=-K}^{K} 2^{s-1} V_{as} - \left\{\left(\frac{1}{2}\right)[2^K - 1]\right\} \tag{93}$$

Burada K ve K' pozitif tamsayılar ve V_{as} ikili değişkenlerdir.

Sürekli model için enerji fonksiyonu şu şekilde tanımlanır:

$$E = \sum_{a} A\left[\frac{(U_{a+1} - U_{a-1})}{2h} - f(U_a)\right]^2 \tag{94}$$

Burada A pozitif bir sabittir ve güncelleme kuralı zaman türevinden türetilmiştir:

$$\frac{dE}{dt} = \sum_{abs} 2A\left[\frac{(U_{a+1} - U_{a-1})}{2h} - f(U_a)\right]\left\{\frac{\frac{\partial U_{a+1}}{\partial V_{bs}}}{2h} - \frac{\frac{\partial U_{a-1}}{\partial V_{bs}}}{2h} - \frac{\partial f(U_a)}{\partial V_{bs}}\right\}\left(\frac{dV_{bs}}{dt}\right) \tag{95}$$

Eşitlik (68) ile verilen güncelleme kuralı için ara değişkenlerin ifadesi şu şekilde olur:

$$\frac{dW_{bs}}{dt} = -\sum_{a} 2A\left[\frac{(U_{a+1} - U_{a-1})}{2h} - f(U_a)\right]\left\{\frac{\frac{\partial U_{a+1}}{\partial V_{bs}}}{2h} - \frac{\frac{\partial U_{a-1}}{\partial V_{bs}}}{2h} - \frac{\partial f(U_a)}{\partial V_{bs}}\right\} \tag{96}$$

$$= -2A\left\{\frac{\left[\frac{(U_b - U_{b-2})}{2h} - f(U_{b-1})\right]}{2h} - \frac{\left[\frac{(U_{b+2} - U_b)}{2h} - f(U_{b+1})\right]}{2h}\right.$$

$$\left. - \left[\frac{(U_{b+1} - U_{b-1})}{2h} - f(U_b)\right]\left(\frac{df(U_b)}{dU_b}\right)\right\}\left(\frac{\partial U_b}{\partial V_{bs}}\right) \tag{97}$$

Bu nedenle eşitlik (91)'in çözümü için sürekli sinir argoritması aşağıdakilerden oluşur:

$$\frac{dW_{bs}}{dt} = -2A\left\{\left(\frac{1}{2h}\right)^2(-U_{b+2} + 2U_b - U_{b-2}) + \left(\frac{1}{2h}\right)\left[f(U_{b+1}) - f(U_{b-1})(U_{b+1} - U_{b-1})\left(\frac{df(U_b)}{dU_b}\right) + f(U_b)\left(\frac{df(U_b)}{dU_b}\right)\right]\right\}\left(\frac{\partial U_b}{\partial V_{bs}}\right) \quad (98)$$

$$V_{bs} = g(W_{bs}) \quad (99)$$

Burada g; eşitlik (64) ile verilen eşik fonksiyonudur ve çözüm $V_{bs}(t = \infty)$ ile verilir.

Diferansiyel denklemlerin çözümü için genel formül

Bu kısımda ele alınacak diferansiyel denklemlerin genel formu şu şekilde gösterilir:

$$F_i\left(x_l, u_j, d_m u_k,\ Yüksek\ Mertebeden\ Türevler\right) = 0 \quad (100)$$

Burada $l,\ m = 1$'den M'ye, x_l bağımsız reel değişkenler ve $i, j, k = 1$'den N'ye, u_j bağımlı reel değişkenler ve $d_m u_k$ birinci dereceden kısmi türevlerdir ve i ile indekslenen N vardır. F_i fonksiyonlarının tekil olmadığı ve x_i,u_j,$d_m u_k$'ya göre kısmi türevlerin ve yüksek mertebeden türevlerin iyi tanımlı olduğu varsayılır. Diferansiyel denklem (100)'ü çözmek için sonlu farklar yöntemi kullanılmıştır. Bağımsız değişkenler için M boyutlu uzaydaki sonlu alan, birim hacim h^M ile ayrıklaştırılır. Böyle bir uzayı tanımlamak için $a(l),\ l = 1$'den M'ye kadar bileşenlere sahip ayrık indeks vektörü a kullanılır. Her bir $a(1)$ bileşeni, x_i bağımsız değişkeni için ayrık koordinatı temsil eder. Daha sonra sürekli büyüklüklere karşılık gelen ayrıklaştırılmış büyüklükler şu şekilde yazılabilir:

$$x_{ia} = x_{l0} + a(l)h \quad (101)$$

$$U_{ja} = u_j\left(X_l(a)\right) \quad (102)$$

$$D_m U_{ka} = \left(\frac{1}{2h}\right)\left(U_{k,a+m'} - U_{k,a-m'}\right) = \left(\frac{1}{2h}\right)\sum_b\left(\delta_{a+m',b} - \delta_{a-m',b}\right)U_{kb} \quad (103)$$

$$D_{lm}U_{ka} = \left(D_1(D_m U_k)\right)_a = D_1\left(\frac{\left(U_{k,a+m'} - U_{k,a-m'}\right)}{2h}\right) = \left(\frac{1}{2h}\right)^2\sum_b\{\delta_{a+m'+l',b} - \delta_{a+m'-l',b}$$

$$-\delta_{a-m'+l',b} + \delta_{a-m'-l',b}\}U_{kb} \quad (104)$$

Burada δ birim matris, x_{l0}, x_l koordinatının başlangıç değeri, m' ve l' aksi takdirde $m'(m) = 1$ ve $m' = 0$ ve aksi takdirde $l'(l) = 1$ ve $l' = 0$ koşullarını sağlayan M boyutlu vektörlerdir. Diferansiyel denklem (100) için sonlu fark denklemi şu şekilde verilir:

$$F_{ia}\left(X_{la}, U_{ja}, D_m U_{ka}, \text{Yüksek mertebeden ayrık türevler}\right)$$

$$= F'_{ia}\left(U_{jb}\right) = 0 \quad (105)$$

İlk olarak sürekli algoritma ele alınır. U_{jb} değişkenleri sürekli reel değişkenlerdir. Bununla birlikte reel değişkenin farklı sayı gösterim türlerini kullanabiliriz. Reel sayının olası gösterimlerinden biri şu şekilde verilsin:

$$U_{jb} = R\left(B_{jbr}\right) \quad (106)$$

Burada R temsil fonksiyonunu, r bitleri, B_{jbr} ise 0 ve 1 olmak üzere ikili değere sahip bit değişkenleridir. Eşitlik (106) kullanılırsa sonlu fark denklemi (105) şu hale gelir:

$$F_{ia}\left(X_{la}, U_{ja}, D_m U_{ka}, \text{Yüksek mertebeden ayrık türevler}\right)$$

$$= F'_{ia}\left(U_{jb}\right) = F''_{ia}\left(B_{jbr}\right) = 0 \quad (107)$$

Sürekli bir algoritmadan yararlanmak için 0 ve 1 arasında sınırlandırılmış sürekli değişkenler V_{jbr} , eşitlik (107)'de B_{jbr} ile değiştirilir. Daha sonra eşitlik (106) ve (107) şu hale gelir:

$$U_{jb} = R\left(V_{jbr}\right) \quad (108)$$

$$F_{ia}\left(X_{la}, U_{ja}, D_m U_{ka}, \text{Yüksek mertebeden ayrık türevler}\right)$$

$$= F'_{ia}\left(U_{jb}\right) = F''_{ia}\left(V_{jbr}\right) = 0 \quad (109)$$

Denklem (109), bir minimizasyon algoritması kullanılarak çözülebilir. Eşitlik (109) için bir enerji fonksiyonu şu şekilde verilir:

$$E\left(V_{jbr}\right) = A\sum_{ia} F''^{2}_{ia} \quad (110)$$

Burada A pozitif bir sabittir. Eşitlik (110) ile verilen enerji fonksiyonu V_{jbr}'ye göre minimize edilirse, diferansiyel denklemin çözümü eşitlik (110)'u minimize eden $U_{jb}(\infty)$ değerleri ile verilir. Eşitlik (110)'un zaman türevi şu şekilde verilir:

$$\frac{dE\left(V_{jbr}(t)\right)}{dt} = \sum_{ias}\left(\frac{\partial E}{\partial V_{ias}}\right)\left(\frac{dV_{ias}}{dt}\right) \tag{111}$$

Eşitlik (68) ve (69)'dan minimizasyon algoritması şu şekilde verilir:

$$\frac{dW_{ias}}{dt} = -\frac{\partial E}{\partial V_{ias}} \tag{112}$$

$$V_{ias} = G(W_{ias}) \tag{113}$$

Burada W_{ias}, ara değişkenlerdir ve G, 0 ile 1 arasında sınırlandırılmış monoton artan bir eşikleme fonksiyonudur.

Eşitlik (112)'nin sağ tarafını göz önüne alırsak şu şekilde yazılabilir:

$$\frac{\partial E}{\partial V_{ias}} = A\sum_{kc}\left(\frac{\partial F_{kc}^{"\ 2}}{\partial V_{ias}}\right) \tag{114}$$

Eşitlik (109)'dan

$$\frac{\partial F_{kc}^{"\ 2}}{\partial V_{ias}} = \sum_{jb}\left(\frac{\partial F_{kc}^{'\ 2}}{\partial U_{jb}}\right)\left(\frac{\partial U_{jb}}{\partial V_{ias}}\right) \tag{115}$$

ve

$$\left(\frac{\partial F_{kc}^{'\ 2}}{\partial U_{jb}}\right) = \frac{\partial F_{kc}^{2}}{\partial U_{jb}} + \sum_{mp}\left[\frac{\partial F_{kc}^{2}}{\partial\left(D_m U_{pc}\right)}\right]\left[\frac{\partial\left(D_m U_{pc}\right)}{\partial U_{jb}}\right] + \sum_{mlp}\left[\frac{\partial F_{kc}^{2}}{\partial\left(D_{ml} U_{pc}\right)}\right]\left[\frac{\partial\left(D_{ml} U_{pc}\right)}{\partial U_{jb}}\right]$$
$$+ Yüksek\ mertebeden\ terimler \tag{116}$$

yazılır. Eşitlik (103) ve (104) kullanılırsa aşağıdaki denklemler elde edilir:

$$\frac{\partial\left(D_m U_{pc}\right)}{\partial U_{jb}} = \left(\frac{1}{2h}\right)\sum_{e}\left(\delta_{c+m',e} - \delta_{c-m',e}\right)\left(\frac{\partial U_{pe}}{\partial U_{jb}}\right)$$

$$= \left(\frac{1}{2h}\right)\delta_{jp}\left(\delta_{c+m',b} - \delta_{c-m',b}\right) \tag{117}$$

ve

$$\frac{\partial\left(D_{ml} U_{pc}\right)}{\partial U_{jb}} = \left(\frac{1}{2h}\right)^2\sum_{e}\left\{\delta_{c+m'+l',e} - \delta_{c+m'-l',e} - \delta_{c-m'+l',e} + \delta_{c-m'-l',e}\right\}\left(\frac{\partial U_{pe}}{\partial U_{jb}}\right)$$

$$= \left(\frac{1}{2h}\right)^2 \delta_{jp}\{\delta_{c+m'+l',b} - \delta_{c+m'-l',b} - \delta_{c-m'+l',b} + \delta_{c-m'-l',b}\} \qquad (118)$$

Eşitlik (117) ve (118)'i eşitlik (116)'da yerine koyduğumuzda;

$$\left(\frac{\partial {F'_{kc}}^2}{\partial U_{jb}}\right) = \frac{\partial {F_{kc}}^2}{\partial U_{jb}} + \sum_{mp}\left[\frac{\partial F_{kc}^2}{\partial(D_m U_{pc})}\right]\left(\frac{1}{2h}\right)\delta_{jp}(\delta_{c+m',b} - \delta_{c-m',b}) + \sum_{mlp}\left[\frac{\partial F_{kc}^2}{\partial(D_{ml} U_{pc})}\right]\left(\frac{1}{2h}\right)^2 \delta_{jp}\{\delta_{c+m'+l',b} - \delta_{c+m'-l',b} - \delta_{c-m'+l',b} + \delta_{c-m'-l',b}\} + Yüksek\ Mertebeli\ Terimler \qquad (119)$$

$$= \frac{\partial {F_{kc}}^2}{\partial U_{jb}} + \sum_{m}\left[\frac{\partial F_{kc}^2}{\partial(D_m U_{jc})}\right]\left(\frac{1}{2h}\right)(\delta_{c+m',b} - \delta_{c-m',b}) + \sum_{ml}\left[\frac{\partial F_{kc}^2}{\partial(D_{ml} U_{jc})}\right]\left(\frac{1}{2h}\right)^2 \{\delta_{c+m'+l',b} - \delta_{c+m'-l',b} - \delta_{c-m'+l',b} + \delta_{c-m'-l',b}\} + Yüksek\ Mertebeli\ Terimler \qquad (120)$$

yazılır. Eşitlik (115) ve (120)'den eşitlik (114) şu hale gelir:

$$\frac{\partial E}{\partial V_{ias}} = A\sum_{jbkc}\left\{\frac{\partial F_{kc}^2}{\partial U_{jb}} + \sum_{m}\left[\frac{\partial F_{kc}^2}{\partial(D_m U_{jc})}\right]\left(\frac{1}{2h}\right)(\delta_{c+m',b} - \delta_{c-m',b}) + \sum_{ml}\left[\frac{\partial F_{kc}^2}{\partial(D_{ml} U_{jc})}\right]\left(\frac{1}{2h}\right)^2 (\delta_{c+m'+l',b} - \delta_{c+m'-l',b} - \delta_{c-m'+l',b} + \delta_{c-m'-l',b}) + Yüksek\ Mertebeli\ terimler\right\}\left(\frac{\partial U_{jb}}{\partial V_{ias}}\right) \qquad (121)$$

$$= A\sum_{kc}\left\{\frac{\partial F_{kc}^2}{\partial U_{ia}} + \sum_{m}\left(\frac{1}{2h}\right)(\delta_{a-m',c} - \delta_{a+m',c})\left[\frac{\partial F_{kc}^2}{\partial(D_m U_{ic})}\right] + \sum_{ml}\left(\frac{1}{2h}\right)^2 (\delta_{a-m'-l',c} - \delta_{a-m'+l',c} - \delta_{a+m'-l',c} + \delta_{a+m'+l',c})\left[\frac{\partial F_{kc}^2}{\partial(D_{ml} U_{ic})}\right] + Yüksek\ Mertebeli\ Terimler\right\}\left(\frac{\partial U_{ia}}{\partial V_{ias}}\right) \qquad (122)$$

Burada eşitlik (108) j ve b üzerinde toplamak için kullanılmış ve δ birim matrislerindeki a ve c yeniden düzenlenmiştir. Eşitlik (122)'deki (103) ve (104) bağlantıları kullanılarak diferansiyel denklemin çözümü için algoritma son olarak şu şekilde yazılır:

$$\frac{dW_{ias}}{dt} = -A\sum_{k}\left\{\frac{\partial(F_{ka}^{2})}{\partial U_{ia}} - \sum_{m} D_{m}\left[\frac{\partial(F_{ka}^{2})}{\partial(D_{m}U_{ia})}\right] + \sum_{ml} D_{ml}\left[\frac{\partial(F_{ka}^{2})}{\partial(D_{ml}U_{ia})}\right] + Yüksek\ Mertebeli\ Terimler\right\}\left(\frac{\partial U_{ia}}{\partial V_{ias}}\right) \qquad (123)$$

ve

$$V_{iam} = G(W_{iam}) \qquad (124)$$

Diferansiyel denklemlerin çözümü için ayrık formalizm, ayrık sinir minimizasyon algoritmasına dayanmaktadır. Diferansiyel denklem (100) için sonlu fark denklemi eşitlik (105) ile verilmiştir. Bununla birlikte U_{ia} değişkenlerinin ikili gösterimi, 1 ve -1 değerlerine sahip ikili değişkenler kullanılarak elde edilir. Bu nedenle ayrık algoritma için enerji fonksiyonu şu şekilde verilir:

$$E(B_{jbs}) = \sum_{ia} F_{ia}^{"}(B_{jbs})^{2} \qquad (125)$$

Burada B_{jbn}, 1 ve -1 değerlerine sahip ikili değişkenlerdir. Ayrık algoritma eşitlik (88) ve eşitlik (90) kullanılarak elde edilebilir (Lee vd 1990).

Yapay Sinir Ağı ile Lineer ve Lineer Olmayan Adi Diferansiyel Denklemlerin Çözümü

Meade ve Fernandez, ileri beslemeli yapay sinir ağı ve B_1-spline'lar kullanarak lineer ve lineer olmayan adi diferansiyel denklemlerin nümerik çözümünü elde etmiştir (Meade vd 1994).

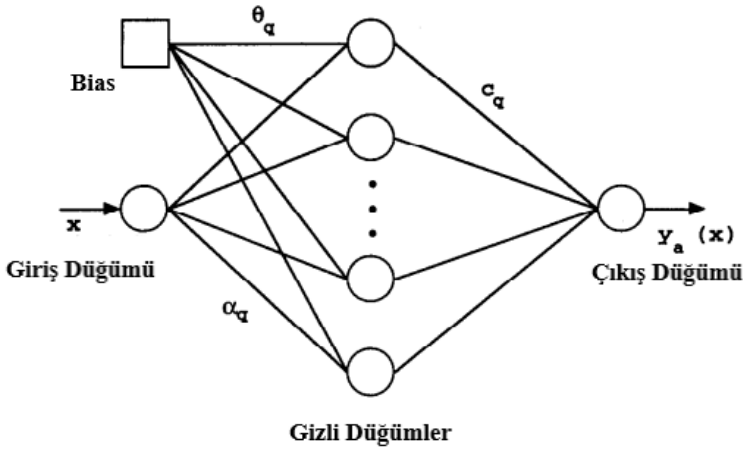
İleri beslemeli yapay sinir ağları ile lineer adi diferansiyel denklemlerin çözümü

Fonksiyon yaklaşımı

Sürekli, çok değişkenli bir fonksiyonun yaklaştırılması veya enterpolasyonu problemiyle ilgilenen fonksiyon yaklaşımı teorisi, genel olarak denetimli öğrenme ve YSA davranışını açıklamak için araştırmacılar (Omohundro 1987; Girosi vd 1989) tarafından dikkate alınan bir yaklaşımdır. Basit bir ileri beslemeli yapay sinir ağı yapısının yaklaşım performansını etkileyen dört parametre kümesi vardır:

1. Giriş ağırlıkları
2. Bias ağırlıkları
3. Transfer fonksiyonları
4. Çıkış ağırlıkları

Her bir parametre kümesinin matematiksel rollerinin yorumlanmasında rehberlik sağlamak amacıyla fonksiyon yaklaşımı teorisi incelenmiştir.



θ_q= q. gizli düğüm için bias ağırlığı
α_q= q. gizli düğüm için giriş ağırlığı
c_q= q. gizli düğüm için çıkış ağırlığı

Şekil 20. Standart ileri beslemeli yapay sinir ağı yapısı

Fonksiyon yaklaşımının en yaygın yöntemi fonksiyonel temel açılımıdır. Fonksiyonel temel açılımı, x değişkeninin fonksiyonları olan lineer bağımsız temel fonksiyonların (Υ) ağırlıklı bir bileşeni olarak keyfi bir $y(x)$ fonksiyonunu temsil eder:

$$y(x) = \sum_{q=1}^{T} \Upsilon_q(x) c_q \tag{126}$$

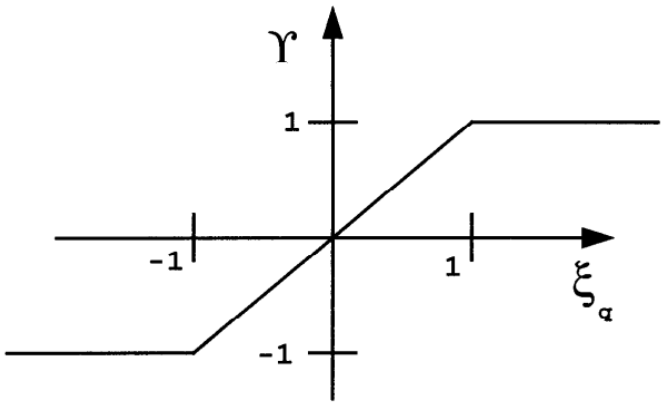
Burada T toplam temel fonksiyonların sayısı ve c_q temel katsayılardır. c_q katsayılarının değerleri seçilen temel fonksiyonlara bağlı olarak değişecektir.

Denklem (126) incelendiğinde bunun temel fonksiyonlar ve genişleme katsayılarının skaler çarpımı olduğu görülür. Şekil 20'deki basit ileri beslemeli yapay sinir ağı tarafından gerçekleştirilen matematiksel işlemler, transfer fonksiyonlarının temel fonksiyonlar olarak hareket ettiği ve çıkış ağırlıklarının temel genişleme olarak hareket ettiği bir skaler çarpım olarak yorumlanabilir. Bu anlayış, sigmoidler gibi transfer fonksiyonlarının sonlu toplamlarının keyfi sürekli fonksiyonlara yaklaşmak için kullanılabileceğini kanıtlama girişimlerinin çoğunun temelinde bulunur (Cybenko 1989; Ito1991).

İleri beslemeli yapay sinir ağı ve fonksiyonel temel genişlemesi arasındaki bağlantı, ağın performansını etkileyen daha önce bahsedilen parametre setlerinden ikisine, çıktı ağırlıklarına ve transfer fonksiyonlarına matematiksel bir rol atar.

Temel fonksiyonlar olarak transfer fonksiyonları

Transfer fonksiyonlarının temel fonksiyonlar olarak kullanışlılığı incelenmiştir. Burada transfer fonksiyonu olarak Şekil 21'deki hard limiti, transfer fonksiyonu olarak kullanılmıştır. Bu seçim, fonksiyonun basitliğine ve yazılım-donanımda uygulanabilme kolaylığına dayandırılmıştır (Chuo vd 1987).



Şekil 21. Hard limit transfer fonksiyonu

Hard limit aşağıdaki denklemlerle modellenebilir:

$$\begin{aligned} \Upsilon_q &= -1.0, \quad için\ \xi_q < -1.0 \\ \Upsilon_q &= \xi_q(x), \quad için\ -1.0 \le \xi_q \le 1.0 \\ \Upsilon_q &= +1.0, \quad için\ \xi_q > 1.0 \end{aligned} \tag{127}$$

Burada ξ_q, bağımsız değişken x'in bir lineer fonksiyonudur.

Bir fonksiyonun baz genişlemesi ile yaklaştırılmasında, genişleme için seçilen bazlar ya yaklaştırılacak fonksiyonun yaşadığı uzayı ya da bunun bir alt uzayını kapsamalıdır. Fonksiyon yaklaşımı literatüründe yaygın olarak kullanılan baz fonksiyonlarının ya yerel (sadece alanın sınırlı bir kısmında sıfır olmayan bir değer sergileyen) ya da global (alanın hemen hemen her yerinde sıfır olmayan) olarak kabul edilir.

Hard limit transfer fonksiyonları, ξ_q; dönüşüm yardımıyla parçalı bir şekilde tanımlanan global lineer polinom temel fonksiyonları olarak görülebilir. Hard limitlerin dağılımı, giriş ve bias ağırlıkları kullanılarak gerçekleştirilir.

Prenter (Prenter 1989) iyi bir polinom temelini, genişleme katsayısı problemine en uygun çözüm sağlayan bir temel olarak tanımlar:

$$f(x_i) = p(x_i), \qquad (i = 0,1, \dots, N) \tag{128}$$

Burada, yaklaştırılacak fonksiyonu ifade etmek için kullanılan N nokta genellikle düğüm olarak adlandırılır. İyi bir polinom temeli aynı zamanda düğümler arasında yaklaştırılan fonksiyonun düzgün interpolasyonunu da sağlamalıdır. Bu özellikleri sağlayan belirli bir polinom ailesi Lagrange polinomları ailesidir.

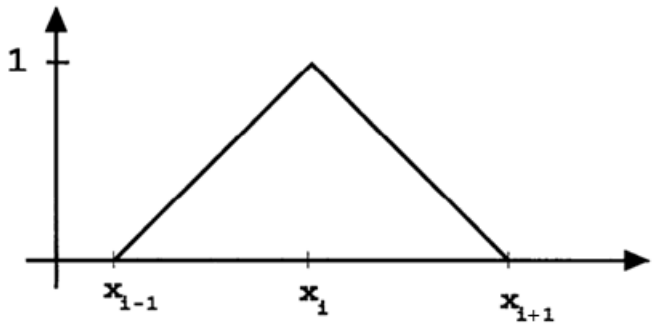
Lagrange polinom spline'ları

Global temel fonksiyonlarının dezavantajları nedeniyle, Lagrange polinomlarına dayalı parçalı polinom spline'lar sıklıkla kullanılmaktadır. Spline'lar, sınırlı sayıda düğüm üzerinde uzanan ve birlikte düğümleri arasında değişen derecelerde enterpolasyon sağlayan polinom eğrileridir. En yaygın polinom spline, Şekil 22'de gösterildiği gibi Chapeau veya "hat (şapka)" fonksiyonudur. Bu literatürde birinci dereceden Lagrange polinomu olarak adlandırılır. Formülü şöyledir:

$$\Phi_i = \frac{(x - x_{i-1})}{(x_i - x_{i-1})}, \qquad x_{i-1} \le x \le x_i$$

$$\Phi_i = \frac{(x_{i+1} - x)}{(x_{i+1} - x_i)}, \qquad x_i \le x \le x_{i+1} \tag{129}$$

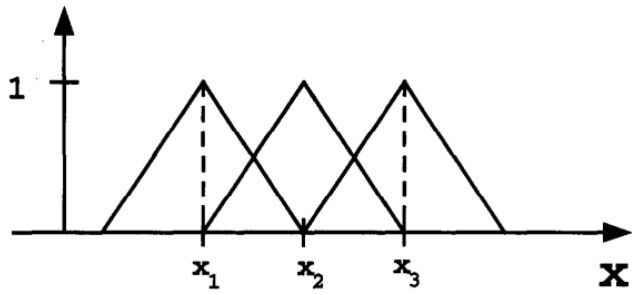
aksi takdirde $\Phi_i = 0, \quad x < x_{i-1}$ ya da $x_i > x_{i+1}$



Şekil 22. Chapeau veya "hat (şapka)" fonksiyonu

Şekil 23, $N = 3$ düğüm $(x_1, x_2\ ve\ x_3)$ ile ayrıştırılmış bir $[x_1, x_N]$ problem alanı, lineer bir enterpolasyon sağlamak için hat fonksiyonlarının nasıl dağıtılması gerektiğini göstermektedir. Etki alanındaki herhangi bir x değeri için, tüm hat fonksiyonlarının toplamının 1 değerine eşit olacağı görülür. Bu, bu şekilde dağıtılan hat fonksiyonlarının bir sabiti doğru bir şekilde temsil edeceği anlamına gelir ki bu da iyi bir enterpolasyon şemasının ilk ve en önemli

testidir. Dağılımın hesaplama verimliliği açısından ek bir faydası da hat fonksiyonlarının lineer bağımsız olmasıdır. Bu, temel açılım katsayılarının belirlenmesinde önemli olacaktır.



Şekil 23. Problem alanı boyunca hat fonksiyonlarının uygun dağılımı $(x_1 \leq x \leq x_3)$

Hat fonksiyonlarının dağılımı tamamen matematiksel gerekçelerle de doğrulanmaktadır. Hat fonksiyonlarının fonksiyonel bir temel olarak kullanıldığı ve temsil edilen fonksiyonun N düğümde ayrıklaştırıldığı düşünüldüğünde, temsil edilen fonksiyon için N serbestlik derecesi olduğu açıktır. Bu, fonksiyonu temsil etmek için kullanılan fonksiyon alt uzayında N boyut anlamına gelir ve etki alanında her düğümde bir tane olmak üzere N hat fonksiyonu gerektirir.

Ağ parametrelerinin kısıtlanması

Daha önce de belirtildiği gibi araştırmacıların YSA'ların diferansiyel denklemlerin çözümüne yönelik çalışmaları, Hopfield tipi bir ağ üzerinde diferansiyel denklemle ilgili bir enerji fonksiyonunun minimize edilmesini içermektedir. Denetimli öğrenme ve diferansiyel denklemlerin minimizasyon yoluyla çözümü, yavaş yakınsama süreleri ve optimal olmayan bir duruma yakınsama veya hiç yakınsamama olasılığı içerir. Bu durumların yalnızca kullanılan algoritmalardan kaynaklanmadığını anlamak çok önemlidir. Sorunların bir kısmı, verilen problemin hala yetersiz belirlenmiş olmasıdır: sistemdeki parametreler üzerinde yeterli sayıda çözümü kabul edecek kadar kısıtlama yoktur. Sonuç olarak, oluşturulan hata yüzeyinin karmaşık olması muhtemeldir ve algoritmalar bu yüzeyde ilerlerken sorunlar yaşayabilir.

Bu sorunların çözümü, parametrelere keyfi olarak ya da bazı gerekçeleri izleyerek kısıtlamalar getirmektir. Ancak yine de minimum hatada istenen fonksiyona yakın olan bir yüzeye dönüştürmektir. İdeal olarak, parametrelerin optimizasyonun gerekli olmadığı ve geri kalan bilinmeyen parametrelerin en iyi şekilde belirlenebileceği ölçüde kısıtlanması istenir.

Girdi ağırlıkları ve bias ağırlıkları

α_q'nun giriş ağırlığını ve θ_q'in bias ağırlığını temsil ettiği Şekil (20)'deki gösterim kullanılarak, belirli bir q işleme elemanının çıkışı (o_q) denklem formunda yazılabilir:

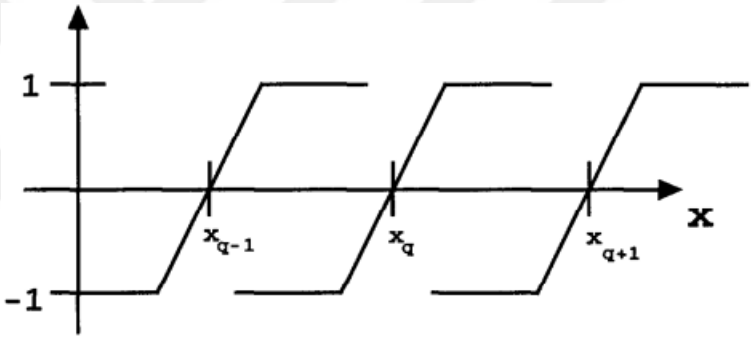
$$o_q = \Upsilon_q(\alpha_q x + \theta_q) = \Upsilon_q(\xi_q) \quad \text{burada} \quad \xi_q = \alpha_q x + \theta_q \tag{130}$$

α_q ve θ_q sabittir, ξ_q ise giriş x'e lineer bağımlı bir değişkendir. Denklem (130) incelendiğinde giriş ağırlığı α_q'nın x için bir ölçekleme katsayısı gibi davrandığı görülür. Alternatif olarak giriş ağırlığı ve biasın, giriş değişkenini q. işlem elemanına özgü yeni bir alan ξ_q'ye dönüştürmekte olarak görülebilir. Bu, x'in ξ_q'ya lineer dönüşümü x'in ölçeklenmemiş kaydırılmış dönüşümlerine benzer (Ito 1991).

x-eksenindeki bir noktayı x_q, transfer fonksiyonu Υ_q'nın "merkezi" olarak tanımlanır:

$$\alpha_q x_q + \theta_q = 0 \quad veya \quad \theta_q = -\alpha_q x_q$$

Bu denklemden, bias ağırlığının her transfer fonksiyonunun orijinini ($\xi_q = 0$) x bağımsız değişken uzayında ayarlanmasına izin verdiği görülmektedir. Giriş ve bias ağırlıkları, her transfer fonksiyonunun farklı şekilde ölçeklendirilmesine ve x bağımsız değişken uzayında farklı konumlarda merkezlenmesine izin verir (Şekil 24).



Şekil 24. Hard limitlerin x ekseni boyunca keyfi dağılımı

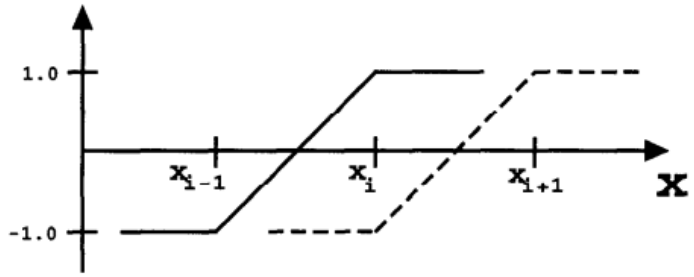
Sinir ağının transfer fonksiyonlarının spline'lara dönüştürülmesi

Hard limitin, hat fonksiyonuna dönüştürülmesi durumunda önemli bir dizi avantaj elde edilebilir. Şekil 25'de, iki hard limit fonksiyonu komşu aralıklarda merkezlenmiş olarak gösterilmiştir. x_i ve x_{i+1} arasında merkezlenmiş olan fonksiyon -1 ile çarpılır ve ikinci değiştirilmemiş fonksiyona eklenirse (Şekil 26), sonucun maksimum değeri 2 olan bir hat fonksiyonu olacağı açıkça görülür. Ayrıca, şekildeki hard limitler uygun çıktı ağırlıklarıyla -0.5 ve $+0.5$ arasında ölçeklendirilirse, sonuçta ortaya çıkan hat fonksiyonu, fonksiyon yaklaşımı literatüründe kullanılandan ayırt edilemez. Hard limitlerin hat fonksiyonlarına dönüştürülmesi, aşağıdaki denklemle tanımlanabilir:

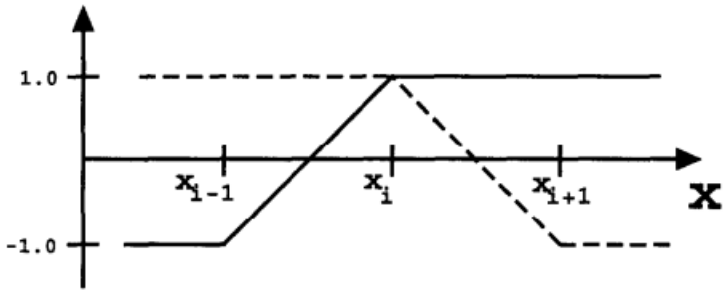
$$\Upsilon_i^A - \Upsilon_i^B = 2\Phi_i, \quad (i = 1,2, \dots, N) \tag{131}$$

Burada hard limitlerin hard simgeleri A ve B, sırasıyla bitişik aralıklar $x_{i-1} \leq x \leq x_i$ ve $x_i \leq x \leq x_{i+1}$'i ifade eder. Υ_i^A ve Υ_i^B fonksiyonları, kendi aralıklarının orta noktalarında sıfır

olarak tanımlanmıştır. Bu formülasyonun bir sonucu, hard limitlerin sayısı istenen hat fonksiyonlarının sayısına bağlanabilmesidir. Hat fonksiyonlarından iki kat daha fazla hard limite ihtiyaç duyulacaktır ($T = 2N$).



Şekil 25. Giriş ve bias ağırlıklarını kullanarak x ekseni boyunca hard limitlerin kısıtlanmış dağılımı



Şekil 26. Sağ taraftaki hard limit (kesikli çizgi), negatif çıktı ağırlığı ile çarpılarak ters çevrilir. İki hard limitin toplanması bir hat fonksiyonu oluşturur. Şekil 22'e bakınız.

Girdi, Bias ve çıktı ağırlıklarının kısıtlanması

Hard limit fonksiyonu gösterimini denklem (126) hat fonksiyonlarını kullanan bir gösterime eşitlemek için gereken özel kısıtlamalar türetilir ise;

$$y_a(x) = \sum_{q=1}^{T} \Upsilon_q(x) c_q = \sum_{i=1}^{N} \Phi_i(x) w_i \,. \tag{132}$$

yazılır. Burada Φ_i hat fonksiyonları, denklem (129)'a göre, x_i düğümünde 1 değerine sahip olarak tanımlanır ve w_i, hat fonksiyonlarıyla ilişkilendirilen temel katsayılardır.

Problem uzayını x_i düğümlerini kullanılarak aşağıdaki şekilde ayrıklaştıralım:

$$a = x_1 < x_2 < \cdots < x_{N-1} < x_N = b$$

Bu ayrıklaştırma problem ağı olarak adlandırılır. Sınırlardaki uygun hat fonksiyonlarını oluşturmak için, $x_0 < a$ ve $x_{N+1} > b$ olacak şekilde iki yardımcı düğüm x_0 ve x_{N+1} gereklidir.

$T = 2N$ kısıtlamasını kullanarak denklemi (126) şu şekilde yeniden yazılır:

$$\sum_{q=1}^{T=2N} \Upsilon_q(x)c_q = \sum_{q=1}^{N} \Upsilon_q(x)c_q + \sum_{q=N+1}^{2N} \Upsilon_q(x)c_q \tag{133}$$

Burada sağ taraftaki toplamlar, genellik kaybı olmaksızın şu şekilde yeniden yazılabilir:

$$\begin{aligned} \sum_{q=1}^{N} \Upsilon_q c_q &= \sum_{i=1}^{N} \Upsilon_i^A u_i \\ \sum_{q=N+1}^{2N} \Upsilon_q c_q &= \sum_{i=1}^{N} \Upsilon_i^B v_i \end{aligned} \tag{134}$$

Buradan

$$\sum_{q=1}^{T} \Upsilon_q(x)c_q = \sum_{i=1}^{N} \Upsilon_i^A u_i + \sum_{i=1}^{N} \Upsilon_i^B v_i \tag{135}$$

yazılır. Burada

$$\begin{aligned} \Upsilon_i^A &= -1.0, && için\ x < x_{i-1} \\ \Upsilon_i^A &= \xi_i^A(x), && için\ x_{i-1} \le x \le x_i \\ \Upsilon_i^A &= 1.0, && için\ x > x_i \\ \Upsilon_i^A &= 0, && x = \frac{x_{i-1} + x_i}{2} \end{aligned} \tag{136}$$

ve

$$\begin{aligned} \Upsilon_i^B &= -1.0, && için\ x < x_i \\ \Upsilon_i^B &= \xi_i^A(x), && için\ x_i \le x \le x_{i+1} \\ \Upsilon_i^B &= 1.0, && için\ x > x_{i+1} \\ \Upsilon_i^B &= 0, && x = \frac{x_i + x_{i+1}}{2} \end{aligned} \tag{137}$$

yazılır. Eğer çıkış ağırlıkları şu şekilde kısıtlanırsa,

$$u_i = -v_i = \frac{w_i}{2} \tag{138}$$

ve daha sonra bu bağıntı denklem (135)'de yerine konulduğunda;

$$\sum_{i=1}^{N} \Upsilon_i^A u_i + \sum_{i=1}^{N} \Upsilon_i^B v_i = \left(\sum_{i=1}^{N} \Upsilon_i^A - \sum_{i=1}^{N} \Upsilon_i^B \right) \frac{w_i}{2} = \sum_{i=1}^{N} \left(\Upsilon_i^A - \Upsilon_i^B \right) \frac{w_i}{2}$$

olarak yazılır. Ancak, denklem (131)'den, $\Upsilon_i^A - \Upsilon_i^B = 2\Phi_i$; bu nedenle,

$$\sum_{i=1}^{N}\left(\Upsilon_i^A - \Upsilon_i^B\right)\frac{w_i}{2} = \sum_{i=1}^{N}(2\Phi_i)\frac{w_i}{2} = \sum_{i=1}^{N}\Phi_i(x)w_i$$

bu da denklem (132)'den $y_a(x)$'in hat fonksiyonları cinsinden temel açılımıdır.

Denklem (134) ve (138)'den, çıktı ağırlıkları şu şekilde verilir:

$$c_i = u_i = \frac{w_i}{2}, \qquad (i = 1,2,\dots,N)$$

$$c_{i+N} = v_i = \frac{-w_i}{2}, \quad (i = 1,2,\dots,N) \tag{139}$$

İleri beslemeli yapay sinir ağındaki $T = 2N$ çıktı ağırlıklarının i. çifti, $(i = 1,2,\dots,N)$ için i. temel genişleme katsayısı w_i'nin u_i ve v_i sırasıyla $\frac{1}{2}$ ve -$\frac{1}{2}$ değerlerine ayarlanır.

Girdi ve bias ağırlığı formüllerini türetmek için, Şekil ((22),(24)-(26))'ya ve denklem ((129), (136) ve (137)) 'ye tekrar bakılırsa, Şekil 25'den de anlaşılacağı üzere, hard limitler problem uzayına, lineer davranışları (sırasıyla $\xi_i^A(x)$ ve $\xi_i^B(x)$) uygun aralıkta gerçekleşecek şekilde yerleştirilmelidir. Bu nedenle,

$$\xi_i^A(x) = \frac{2(x - x_{i-1})}{(x_i - x_{i-1})} - 1, \;\; x_{i-1} \le x \le x_i$$

$$\xi_i^B(x) = \frac{2(x - x_i)}{(x_{i+1} - x_i)} - 1, \;\; x_i \le x \le x_{i+1}$$

$$i = 1,2,\dots,N \tag{140}$$

yazılır. $\xi_i^A(x) = \alpha_i^A x + \theta_i^A$ ve $\xi_i^B(x) = \alpha_i^B x + \theta_i^B$ olduğundan, denklem (140) kullanılarak giriş ve bias ağırlıklarının değerleri hesaplanabilir:

$$\alpha_i^A = \frac{2}{(x_i - x_{i-1})}, \;\; \theta_i^A = -\frac{2x_{i-1}}{(x_i - x_{i-1})} - 1 \tag{141}$$

ve

$$\alpha_i^B = \frac{2}{(x_{i+1} - x_i)}, \;\; \theta_i^B = -\frac{2x_i}{(x_{i+1} - x_i)} - 1 \tag{142}$$

Çıktı ağırlıklarının belirlenmesi: Ağırlıklı kalanlar yöntemi (Method of weighted residuals)

Genişleme katsayılarını bulma problemi, bağlantıcılara oldukça aşina olan terimlerle ifade edilebilir: çözümü y fonksiyonu olan bir diferansiyel denklem ve çözüme bir temel genişlemesi olarak ifade edilen bir yaklaşım verildiğinde, y_a genişleme katsayıları bulunur, öyle

ki yaklaşımın hatası D alanındaki bazı normlar açısından küçük olsun.Yani alansal değişkenler s vektörü ile tanımlanıyorsa ve $L(y)$ bir diferansiyel operatör ise o zaman denklem (143);

$$y_a(s,t) = \sum_{i=1}^{N} \Phi_i(s) w_i(t), \quad (burada\ \Phi_i(s) temel\ fonksiyonlardır) \tag{143}$$

zamana bağlı diferansiyel denklem $L(y) = g(s,t)$, $L(y_a) - g(s,t) = R(w_1, \dots, w_N, s, t)$ denklemiyle sonuçlanır. Burada R, hata veya diferansiyel denklem kalanı olarak tanımlanabilecek sıfır olmayan bir değer fonksiyonudur. Ek olarak, denklem (143), diferansiyel denklemin başlangıç ve sınır koşullarına bağlı ise,

$$I(y_a) = R_I(w_1, \dots, w_N, s) \quad ve \quad B(y_a) = R_B(w_1, \dots, w_N, t)$$

yazılabilir. Temel olarak, $w_i(t)$ temel katsayıları bulunabilir, böylece R, problem alanı D üzerinde $N \to \infty$ kadar bazı normlar açısından küçük olur.

$R_I = R_B = 0$ olacak şekilde y_a'nın başlangıç ve sınır koşullarını sağlamasını isteyelim. Diferansiyel denklemi sağlayan $w_i(t)$ temel katsayıları, denklem kalanı R'nin bir $f(s)$ fonksiyonu (genellikle ağırlık fonksiyonu veya test fonksiyonu olarak adlandırılır) ile çarpılması, problem alanı D üzerinde integre edilmesi ve sıfıra eşitlenmesini gerektirerek belirlenebilir:

$$\int_{\boldsymbol{D}} fR\, dD = (f,R) = 0 \tag{144}$$

Burada (f,R), f ve R fonksiyonlarının iç çarpımıdır. Ağırlıklı kalanlar yöntemi adını (144) numaralı denklemden almaktadır. Denklem (144)'ün yönetim denkleminin zayıf formuyla yakından ilişkili olduğu belirtilebilir:

$$\int_{D} fR\, dD = \int_{D} f\big(L(y_a) - g(s,t)\big)\, dD = 0 \tag{145}$$

Zayıf formla olan bu ilişki, kesin çözümde süreksizliklere izin verme avantajına sahiptir (Lax vd 1960), bu da özellikle büyük gradyanlara veya süreksizliklere sahip problemlerin çözümüne yaklaşırken avantajlıdır.

Denklem (144)'ün temel katsayılarını çözmek için lineer bağımsız ilişkilere ihtiyaç duyulduğundan, f 'nin lineer bağımsız f_k fonksiyonlarından oluşması gerektiği açıktır. $k = 1, \dots, N$ olmak üzere, temel katsayılar için N denklem sistemi oluşturulur. Zamana bağlı bir durum için, t cinsinden bir adi diferansiyel denklem sistemi ortaya çıkar. Kararlı durum için, temel katsayılar sabittir ve bir cebirsel denklem sistemi oluşturulur.

Ağırlık fonksiyonu f_k için farklı seçimler, MWR'nin alt sınıfları olan farklı hesaplama yöntemlerine yol açar. Bu yöntemlerden bazıları şunlardır:

(1) Alt alan yöntemi. Hesaplama alanı, aşağıdaki durumlarda üst üste gelebilen N adet D_k alt alanına ayrıştırılır:

$$f_k = 1 \; içinde \; D_k \quad ve \quad f_k = 0 \; dışında \; D_k, \qquad için \;\; k = 1, \dots, N$$

Alt alan yöntemi, (144) denklemini değerlendirirken sonlu hacim yöntemiyle aynıdır. Alt alan yöntemi ile (143) ve (145) denklemleri, hem yerel hem de küresel olarak yönetim denkleminde koruma özelliklerini zorlamak için uygun çerçeveyi sağlar.

(2) Sıralama yöntemi. Ağırlık fonksiyonları şu şekilde ayarlanmıştır:

$$f_k = \delta(s - s_k), \;\; için \; k = 1, \dots, N$$

Burada δ Dirac delta fonksiyonudur. Bu bağıntının denklem (144)'de yerine konulması şunları verir:

$$\int_{\boldsymbol{D}} \delta(s - s_k) \, R \, dD = R(w_1, w_2, \dots, s_k) = 0$$

Sonlu farklar yöntemi diferansiyel denklemin sadece düğüm noktalarında çözülmesini gerektirdiğinden, sonlu farklar yöntemi yaklaşık bir çözüm y_a kullanılmadan bir kollokasyon yöntemi olarak yorumlanabilir.

(3) Momentler yöntemi. Ağırlık fonksiyonları, denklem kalıntısının ardışık olarak daha yüksek momentlerinin sıfır olması gerekecek şekilde lineer bağımsız fonksiyonlar kümesinden seçilir. Tek boyutlu bir problem için,

$$f_k = x^k, \;\; için \; k = 0, \dots, N - 1.$$

Bu bağıntının denklem (144)'ün tek boyutlu formunda yerine konulması şunu verir:

$$\int_a^b x^k \, R \, dx = 0, \;\; için \; k = 0, \dots, N - 1$$

(4) En küçük kare yöntemi. Ağırlık fonksiyonları şu şekilde ayarlanmıştır:

$$f_k = \frac{\partial R}{\partial w_k}, \;\; için \; k = 1, \dots, N$$

Burada w_k temel katsayılardır. Denklem (144)'e f_k uygulanması, problem alanı üzerinde toplanan kalıntının karesinin minimumunu bulmakla aynıdır:

$$\frac{\partial}{\partial w_k} \int_D R^2 \, dD = 0$$

(5) Genelleştirilmiş Galerkin yöntemi. Ağırlık fonksiyonları şu şekilde ayarlanmıştır:

$$f_k = \Psi_k(s) \; için \; k = 1, \ldots, N$$

Burada $\Psi_k(s)$ bazlara benzer analitik fonksiyonlardır, ancak sınır ve/veya başlangıç koşullarını karşılamak için ek terimlerle değiştirilmiştir. Sonlu elemanlar ve spektral yöntemler, genelleştirilmiş Galerkin yönteminin alt sınıfları olarak düşünülebilir. Galerkin tabanlı yöntemler, temel fonksiyonlar tam bir kümenin ilk N üyesi ise özellikle doğru olarak kabul edilir. Sonuç olarak, N sonsuza yaklaştıkça y_a yaklaşık çözümü y kesin çözümüne yakınsayacaktır.

MWR'nin başlangıç ve sınır koşullarının tam olarak sağlandığı ($R_I = R_B = 0$) durumlarla sınırlı olmadığı da unutulmamalıdır. Eğer sadece diferansiyel denklemin tam olarak sağlanması ($R = 0$) istenirse, MWR, panel ve sınır elemanı yöntemleri gibi sınır yöntemlerini türetmek için kullanabilir. Ağırlıklı kalanlar yönteminin çeşitli alt sınıfları Finlayson (Finlayson 1972) ve Fletcher (Fletcher 1984) tarafından uzun uzadıya tartışılmış ve karşılaştırılmıştır.

Çıktı ağırlıklarının belirlenmesi, ağırlıklı kalanlar yöntemiyle yaklaşarak bilimsel ve mühendislik hesaplama alanlarında en yaygın kullanılan tekniklerden hem teorik hem de hesaplama sonuçlarına ulaşılır. Her yöntemin özel uygulamaya bağlı olarak avantajları ve dezavantajları vardır. Fletcher tarafından yapılan gözlemi takiben genelleştirilmiş Galerkin tekniğini kullanacağız. "…Galerkin yöntemi sürekli olarak yüksek doğrulukta sonuçlar üretir ve ağırlıklı kalanlar yöntemi kadar geniş bir uygulama alanına sahiptir." (Fletcher 1984).

Özellikle, $f_k(x)$, y_a yaklaşımını tanımlamak için kullanılan temel fonksiyonlarla aynı olacaktır. Bu, Bubnov-Galerkin tekniği olarak bilinir (Fletcher 1984):

$$f_k(x) = \Phi_k(x), \quad \text{burada } \; k = 1,2, \ldots, N \tag{146}$$

$f(x)$ bu şekilde tanımlandığında, integrallerin değerlendirilmesi, yaklaştırılacak diferansiyel denklem lineer ise genellikle lineer olan bir cebirsel denklem sistemine yol açar. Aksi takdirde, cebirsel denklemler lineer değildir. Bu denklemler, standart teknikler (Strang 1980) aracılığıyla tek bir çözüm için değerlendirilebilir ve hem lineer hem de lineer olmayan diferansiyel denklemlerin değerlendirilmesine izin verir.

Örnek 2: Birinci Dereceden Lineer Adi Diferansiyel Denklem

$$\frac{dy}{dx} - y = 0, \qquad y(x=0) = 1, \qquad 0 \leq x \leq 1 \tag{147}$$

Bu denklem (Lee vd 1990) tarafından örnek olarak kullanılmıştır. Denklem (147)'nin çözümüne yaklaşmak için basit bir ileri beslemeli ağ oluşturulmuştur (Meade vd 1994):

(1) Bağımsız değişken x için lineer bir transfer fonksiyonunu kullanan tek bir girdi işleme elemanı.

(2) Bağımlı değişken yaklaşımı y_a için lineer bir transfer fonksiyonu kullanan tek çıkışlı bir işleme elemanı.

(3) Hard limit transfer fonksiyonunu kullanan bir dizi gizli katman işleme elemanı.

(4) Gizli katman işleme elemanlarının her birine bağlı tek bir bias düğümü.

Problem alanındaki düğüm sayısı N seçer ve sırasıyla α_i^A , α_i^B, θ_i^A ve θ_i^B giriş ve bias ağırlık değerlerini belirleriz. Hat temel fonksiyonlarını Φ kullanarak bağımlı fonksiyonun yaklaşımını temsil edilebiliriz:

$$y_a = \sum_{i=1}^{N} \Phi_i(x) w_i \tag{148}$$

Ağı tamamlamak için geriye çıkış ağırlıklarının değerleri ve dolayısıyla w_i katsayılarının değerleri kalmaktadır.

Denklem (148)'in model denkleminde yerine konulmasıyla aşağıdaki sonuçlar elde edilir:

$$\frac{dy_a}{dx} - y_a = R \tag{149}$$

Bubnov-Galerkin tekniği ($f_k = \Phi_k$) denklem (149)'a uygulanır:

$$(\Phi_k, R) = \left(\Phi_k, \frac{dy_a}{dx}\right) - (\Phi_k, y_a) = 0, \;\; için\; k = 1, \ldots, N \tag{150}$$

$$\frac{dy_a}{dx} = \sum_{i=1}^{N} \frac{d\Phi_i}{dx} w_i \tag{151}$$

Denklem (150) şu hale gelir:

$$\sum_{i=1}^{N} \left(\Phi_k, \frac{d\Phi_i}{dx} w_i\right) - \sum_{i=1}^{N} (\Phi_k, \Phi_i w_i) = 0$$

Problem sabit olduğu için w_i sabittir ve şu şekilde yazılabilir:

$$\sum_{i=1}^{N}\left(\Phi_k, \frac{d\Phi_i}{dx}\right) w_i = \sum_{i=1}^{N}(\Phi_k, \Phi_i) w_i \quad için\ k = 1, \dots, N$$

İç çarpımları kullanarak aşağıdaki lineer cebirsel denklem sistemine ulaşırız:

$$\sum_{i=1}^{N} M_{ki} w_i = b_k, \ \ için\ k = 1, \dots, N \tag{152}$$

Burada

$$M_{ki} = \left(\Phi_k, \frac{d\Phi_i}{dx} - \Phi_i\right)\ ve\ b_k = 0, \ \ için\ k = 1, \dots, N.$$

Model denklemi (147)'nin bir çözümü için başlangıç koşulu gereklidir. Yaklaşık y_a değerini kullanılarak,

$$y_a(0) = \sum_{i=1}^{N} \Phi_i(0) w_i = y(0) = 1$$

bu denklemi sağlar:

$$w_1 = y(0) = 1.$$

Bu denklem, M katsayı matrisine ve denklem (152)'deki b vektörüne dahil edilebilir ve bu nedenle,

$$M_{11} = 1 \ \ ve \ \ M_{1i} = 0, \ \ için\ i = 2, \dots, N \ \ ve \ \ b_1 = y(0) = 1.$$

Değiştirilmiş cebirsel sistemin çözümü, yaklaştırılan diferansiyel denklem için w_i temel açılım katsayılarını sağlar.

Ağırlıklı kalan yöntemi (veya özellikle Bubnov-Galerkin yöntemi) tarafından oluşturulan katsayı matrisinin türünün kullanılan temel fonksiyonların türüne bağlı olduğu daha önce belirtilmişti. Denklem (152)'nin M katsayı matrisini oluşturan integralin analizi şunu göstermektedir:

$$M_{ki} = \left(\Phi_k, \frac{d\Phi_i}{dx} - \Phi_i\right) = 0 \ \ için \ \ i < k - 1 \ \ \ ve \ \ \ i > k + 1$$

Bunun nedeni hat fonksiyonlarının problem alanındaki dağılımları ve lineer bağımsızlıklarıdır. Denklem (152)'nin M matrisi bu nedenle pozitif tanımlı ve üç köşelidir. Başlangıç koşulunun dahil edilmesiyle matrisin değiştirilmesi bu avantajları değiştirmez. Sonuç olarak, değiştirilmiş cebirsel sistem Thomas algoritması (Anderson vd 1984) ile $O(N)$ işlemde

çözülebilir. Açılım katsayıları bilindiğinde, sinir ağının çıkış ağırlıkları denklem (139) ile kolayca hesaplanabilir.

Çıkış ağırlıklarının belirlenmesi, denklem (147)'yi modelleyen ileri beslemeli ağ için parametrelerin de belirlenmesini sağlar. Çıktının doğruluğu, problem alanındaki düğüm sayısını artırarak ve böylece ek hat fonksiyonları oluşturarak kontrol edilebilir.

Örnek 3: İkinci Dereceden Lineer Diferansiyel Denklem

Pratikte ilgi çeken biraz daha zorlu bir problem de özdeğer problemidir (Trim 1990). Dinamik sistem analizi alanında çalışanların aşina olduğu özdeğer problemi, sınır koşullarıyla birlikte aşağıdaki ikinci dereceden lineer adi diferansiyel denklemle tanımlanır (Meade vd 1994):

$$\frac{d^2y}{dx^2} = \lambda_j y = 0, \;\; 0 \leq x \leq 1 \tag{153}$$

$$y(0) = y(1) = 0$$

Burada özdeğerler (λ_j) şu şekilde verilir:

$$\lambda_j = (j\pi)^2, \;\; için \; j = 1,2,3, \ldots \tag{154}$$

Özdeğerlere karşılık gelen $y(\lambda_j)$ çözümleri özfonksiyonlar olarak bilinir. Özfonksiyonlara uyguladığımız bir ek koşul da iç çarpım kullanılarak tanımlanan ortonormalleştirilmiş olmalarıdır:

$$\Big(y(\lambda_j), y(\lambda_k)\Big) = 1, \;\; eğer \; j = k \; ve \; \Big(y(\lambda_j), y(\lambda_k)\Big) = 0, \;\; eğer \; j \neq k \tag{155}$$

Özdeğer problemi, sadece yüksek türev mertebesi nedeniyle değil, aynı zamanda homojen sınır koşullarının çözümü ($y_a = 0$) zorlayabildiğinden yinelemesiz yöntem için iyi bir testtir.

Bubnov-Galerkin yönteminin probleme uygulanması sonucunda;

$$\left(f_k, \frac{d^2y_a}{dx^2}\right) + \left(f_k, \lambda_j y_a\right) = 0, \;\; için \; k = 1, \ldots, N. \tag{156}$$

Kullanılan temel fonksiyonlar lineer olduğundan ikinci türev sıfırdır. Bu nedenle parçalara ayırma yöntemini kullanarak türevlerin mertebesi düşürülmelidir. Denklem (156) şu şekilde yeniden yazabilir:

$$\left(\frac{df_k}{dx}, \frac{dy_a}{dx}\right) - \lambda_j (f_k, y_a) = f_k \frac{dy_a}{dx} \Big|_0^1, \;\; için \; k = 1, \ldots, N \tag{157}$$

Parçalarla integrasyon kullanımının sınır türev bilgisini (Neumann koşulları) getirir. Bu örnekte, bağımlı değişkenin değeri sınırlarda bilinmektedir (Dirichlet koşulları). Hat fonksiyonları kullanıldığında sınır türev terimi yalnızca $k = 1$ ve N için sıfır değildir.

Örnek 2'den farklı olarak özdeğer probleminin yaklaşık çözümü y_a, bağımsız değişken x'i kullanan bir temel genişletmeyi gerektirir. Bunun nedeni ilgili sınır koşullarıyla birlikte problemin denklem (148) gibi basit bir açılımın yerine konmasının sıfır bilinen vektöre (b) yol açacağı ve tekil olmayan bir matris için yalnızca önemsiz bir çözüm vereceği şekildedir (Strang 1980).

y_a için aşağıdaki açılım kullanılır:

$$y_a(x) = \sum_{i=1}^{N} \Phi_i(x) w_i = C1 \sum_{i=1}^{N} \Phi_i(x_i - \tau_i) \tag{158}$$

Burada

$$w_i = C1(x_i - \tau_i). \tag{159}$$

x_i düğümlerin konumları olduğundan ve Φ_i lineer enterpolasyon fonksiyonları olarak hareket ettiğinden;

$$\sum_{i=1}^{N} \Phi_i x_i = x$$

olarak yazılır. Dolayısıyla denklem (158) şu hale gelir:

$$C1\left(\sum_{i=1}^{N} \Phi_i x_i - \sum_{i=1}^{N} \Phi_i \tau_i\right) = C1\left(x - \sum_{i=1}^{N} \Phi_i \tau_i\right) \tag{160}$$

Burada $C1$, denklem (155)'de olduğu gibi yaklaşık çözüm ortonormalleştirildiğinde belirlenecek bir ölçeklendirme katsayısıdır. Bu genişleme ile,

$$y_a(0) = C1\left(0 - \sum_{i=1}^{N} \Phi_i(0)\tau_i\right) = 0 \quad ve \quad y_a(1) = C1\left(1 - \sum_{i=1}^{N} \Phi_i(1)\tau_i\right) = 0 \tag{161}$$

ve bu nedenle,

$$\tau_1 = 0 = b_1 \quad ve \quad \tau_N = 1 = b_N$$

olur. Bu sıfır olmayan bir b vektörü oluşturmaktadır. Sonuç olarak, tekil olmayan bir katsayı matrisi tek bir çözüm üretecektir.

Açılımlar denklem (157)'e uygulandığında aşağıdaki lineer sistemi elde edilir:

$$\sum_{i=1}^{N} \mathrm{M}_{ki}\tau_i = b_k, \;\; için \;\; k = 1, \dots, N \tag{162}$$

Burada

$$\mathrm{M}_{ki} = \left(\frac{d\Phi_k}{dx}, \left(1 - \frac{d\Phi_i}{dx}\right)\right) - \lambda_j\left(\Phi_k, (x - \Phi_i)\right) \qquad ve$$

$$b_k = 0, \;\; için \;\; k = 2, \dots, N-1, \tag{163}$$

$$M_{11} = 1 \;\; ve \;\; M_{1i} = 0, \qquad için \;\; i = 2, \dots, N, \;\; ve \;\; b_1 = \tau_1 = 0$$

ile

$$M_{NN} = 1 \;\; ve \;\; M_{Ni} = 0, \;\; için\, i = 1,2, \dots, N-1 \;\; ve \;\; b_N = \tau_N = 1.$$

Denklem (162)'nin çözümü τ_i değerlerini verir.

Bubnov-Galerkin çözümüne ortonormalizasyon şartı uygulandığında aşağıdaki formül elde edilir:

$$\int_0^1 \left[C1 \sum_{i=1}^{N} \Phi_i w_i \,.\, C1 \sum_{i=1}^{N} \Phi_i w_i \right] dx = 1 \tag{164}$$

veya

$$\int_0^1 \left(C1 \sum_{i=1}^{N} \Phi_i w_i \right)^2 dx = 1 \tag{165}$$

İntegral yaklaşık bir toplam ile değiştirilebilir:

$$C1^2 \sum_{r=1}^{N} h \left(\sum_{i=1}^{N} \Phi_i(x_r) w_i \right)^2 dx = 1 \tag{166}$$

Fakat

$$\Phi_i(x_r) = 1, \qquad için \;\; r = i. \tag{167}$$

Böylece $C1$ için çözdüğümüzde şunu buluruz:

$$C1 = \frac{1}{\sqrt{h \sum_{i=1}^{N} (x_i - \tau_i)^2}}$$

τ_i ve $C1$ biliniyorsa w_i belirlenebilir. Ağ çıkış ağırlıklarının w_i'den hesaplanması ağın yapısını tamamlar.

İleri beslemeli yapay sinir ağları ile lineer olmayan adi diferansiyel denklemlerin çözümü

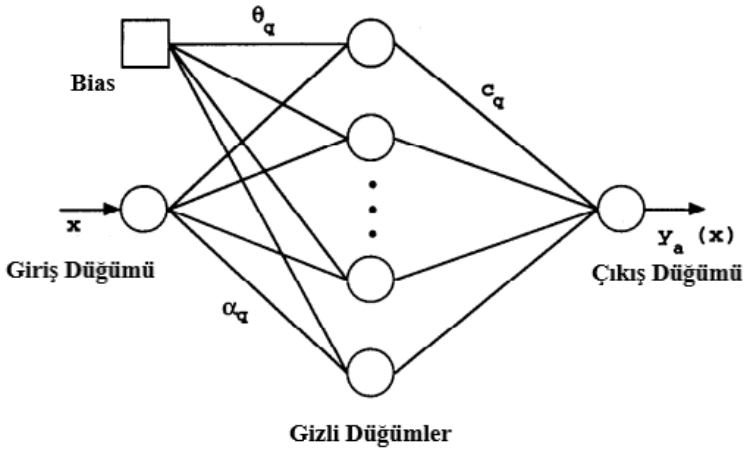
Fonksiyon yaklaşımı

Sürekli, çok değişkenli bir fonksiyonun yaklaştırılması veya enterpolasyonu problemiyle ilgilenen fonksiyon yaklaştırma teorisi, genel olarak denetimli öğrenme ve YSA davranışını açıklamak için araştırmacılar (Omohundro 1987; Girosi vd 1989) tarafından dikkate alınan bir yaklaşımdır. En yaygın fonksiyon yaklaşımı yönteminin fonksiyonel baz açılımı olduğu varsayalım. Fonksiyonel baz açılımı, keyfi bir $y(x)$ fonksiyonunu, bağımsız değişken x'in fonksiyonları olan lineer bağımsız baz fonksiyonlarının (Υ) ağırlıklı bir kombinasyonu olarak temsil eder:

$$y(x) = \sum_{q=1}^{T} \Upsilon_q(x) c_q \tag{168}$$

Burada T toplam temel fonksiyon sayısı ve c_q temel katsayılarıdır. c_q katsayılarının değerleri seçilen temel fonksiyonlara bağlı olarak değişecektir.

Denklem (168) incelendiğinde bunun temel fonksiyonlar ve genişleme katsayılarının skaler çarpımı olduğu görülür. Benzer şekilde, Şekil 27'deki basit ileri beslemeli yapay sinir ağı tarafından gerçekleştirilen matematiksel işlemler, transfer fonksiyonu çıkış değerleri ile çıkış ağırlıklarının skaler çarpımı olarak yorumlanabilir.



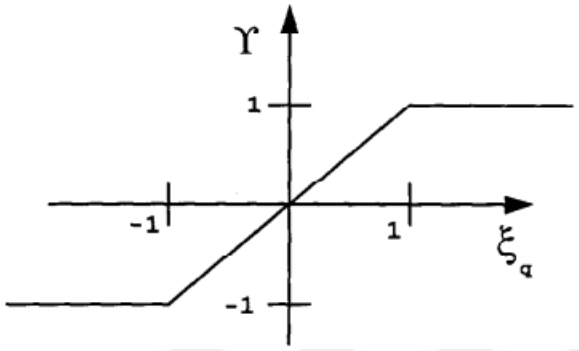
θ_q= q. gizli düğüm için bias ağırlığı
α_q= q. gizli düğüm için giriş ağırlığı
c_q= q. gizli düğüm için çıkış ağırlığı

Şekil 27. Standart ileri beslemeli yapay sinir ağı yapısı

Fonksiyon yaklaşımı teorisine başvurarak ileri beslemeli yapay sinir ağının fonksiyonel ilişkileri temsil etme performansını optimize edecek transfer fonksiyonlarını ve ağırlık değerlerini en iyi şekilde seçebiliriz. Ayrıca, ağdaki çeşitli parametrelere roller atayabiliriz.

Örneğin, çıkış ağırlıkları temel için genişleme katsayıları olarak hareket edebilir ve prensipte bir dizi geleneksel teknikle hesaplanabilir olmalıdır (Cybenko 1989).

Gizli katman için aktivasyon fonksiyonu olarak Şekil 28'deki parçalı lineer haritayı (Maren vd 1990) seçilir, çünkü hem dijital bilgisayarlarda hem de elektronik donanımda uygulanması en basit fonksiyonlardan biridir (Chua vd 1987).



Şekil 28. Aktivasyon fonksiyonu olarak kullanılan parçalı lineer harita

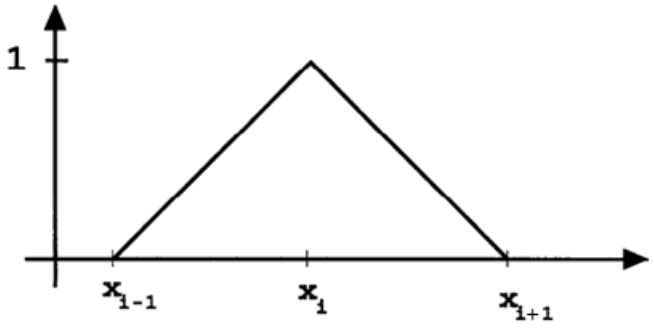
Parçalı lineer aktivasyon fonksiyonu aşağıdaki denklemlerle modellenebilir:

$$\begin{aligned} \Upsilon_q &= -1.0, \quad için\ \xi_q < -1.0 \\ \Upsilon_q &= \xi_q(x), \quad için\ -1.0 \le \xi_q \le 1.0 \\ \Upsilon_q &= +1.0, \quad için\ \xi_q > 1.0 \end{aligned} \tag{169}$$

Şekil 27 incelendiğinde transfer fonksiyonunun ağırlıklı giriş ve bias değerlerinin toplamı ($\xi_q = \alpha_q x + \theta_q$), yani bağımsız değişken x'in lineer bir dönüşümü üzerinde çalıştığı görülmektedir. Bu gösterim Ito'nun ölçeklenmemiş kaydırılmış rotasyonlarına benzer (Ito 1991).

Fonksiyon yaklaşımı literatüründe temel fonksiyonların ya yerel (sadece alanın sınırlı bir kısmında sıfır olmayan bir değer sergileyen) ya da global (yayılacakları alanın çoğunlukla her yerinde sıfır olmayan) olduğu ifade edilir. Parçalı lineer aktivasyon fonksiyonları, ξ_q 'ya dönüşüm yardımıyla parçalı bir şekilde tanımlanan global lineer polinom temel fonksiyonları olarak görülebilir. Tüm global baz fonksiyonları, genişleme katsayılarını çözerken kötü koşullandırma ve ayrıklaştırma noktaları ("düğümler") arasında zayıf yaklaşım gibi benzer zayıflıklara sahiptir (Prenter 1989).

FFANN temsili ile ilgili yaygın sorunların birçoğunun sadece matematiksel olarak uygun olmayan bir temel yaklaşımının kullanılmasından kaynaklanabileceğinden şüphelenilebilir. Fonksiyon yaklaştırma teorisinde, parçalı polinom spline'lar kullanılarak global baz fonksiyonlarının dezavantajlarından kaçınılır.



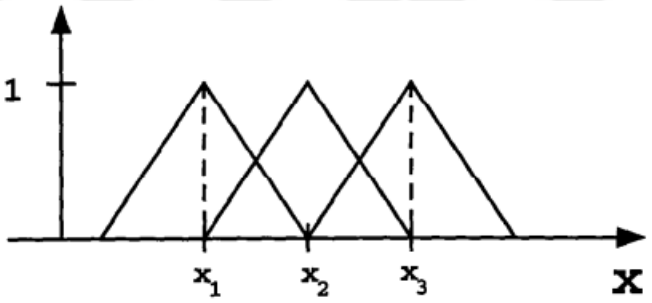
Şekil 29. Chapeau veya "hat (şapka)" fonksiyonu

En yaygın polinom spline, Şekil 29'da gösterildiği gibi Chapeau veya "hat (şapka)" fonksiyonudur. Bu fonksiyon şu şekilde tanımlanır:

$$\Phi_i = \frac{(x - x_{i-1})}{(x_i - x_{i-1})}, \quad için\ x_{i-1} \le x \le x_i$$

$$\Phi_i = \frac{(x_{i+1} - x)}{(x_{i+1} - x_i)}, \quad için\ x_i \le x \le x_{i+1} \qquad (170)$$

$$\Phi_i = 0, \quad için\ x < x_{i-1}\ veya\ x > x_{i+1}$$



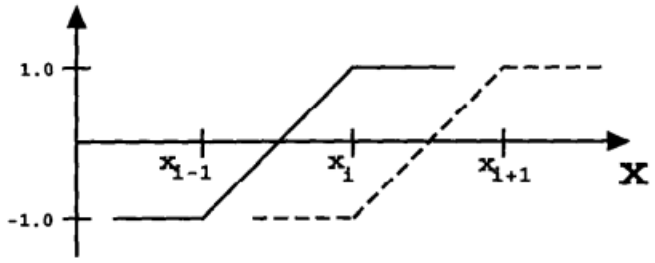
Şekil 30. Problem alanı boyunca hat fonksiyonlarının uygun dağılımı ($x_1 \le x \le x_3$)

Şekil 30, $N = 3$ düğüm ($x_1, x_2\ ve\ x_3$) ile ayrıklaştırılmış bir $[x_1, x_N]$ problem alanı, düğümler arasında lineer bir enterpolasyon sağlamak için hat fonksiyonlarının nasıl dağıtılması gerektiğini göstermektedir. Bu özel polinom spline için genişleme katsayılarını değerlendirmek üzere oluşturulan cebirsel sistem çok seyrek ve iyi koşulludur. Spline'ın maksimum değeri 1 olduğundan, genişleme katsayılarının değerleri düğüm noktalarındaki yaklaşık fonksiyonun değerleriyle aynıdır. Ancak bir dezavantajı, yaklaşımın birinci türevde süreksiz olmasıdır.

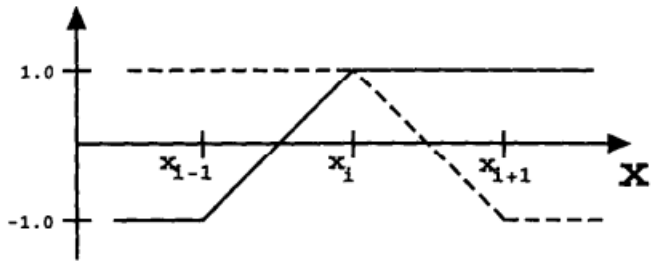
Ağırlık kısıtlamaları ile parçalı lineer aktivasyon fonksiyonlarının Spline'lere dönüştürülmesi

Hat fonksiyonlarının tüm matematiksel avantajları, parçalı lineer harita ileri beslemeli yapay sinir ağı bir şekilde hat fonksiyonu tabanlı bir temsile dönüştürmenin avantajlı olacağını

göstermektedir. Aktivasyon fonksiyonu olarak parçalı lineer haritanın seçilmesi bunu basitleştirmektedir.



Şekil 31. Giriş ve bias ağırlıklarını kullanarak x ekseni boyunca parçalı lineer aktivasyon fonksiyonlarının kısıtlı dağılımı



Şekil 32. Çıkış ağırlık işaretinin değiştirilmesiyle döndürülen sağ taraf aktivasyon fonksiyonu (kesikli). İki aktivasyon fonksiyonunun toplanması bir hat fonksiyonu oluşturur. Şekil 29'a bakınız.

Şekil 31'deki gibi tek boyutlu bir alan için x_i ve x_{i+1} arasında ortalanmış fonksiyon -1 ile çarpılır ve değiştirilmemiş ikinci fonksiyona eklenirse (Şekil 32), sonuç maksimum değeri 2 olan bir hat fonksiyonu olacaktır. Parçalı lineer aktivasyon fonksiyonlarının hat fonksiyonlarına dönüşümü aşağıdaki denklemle tanımlanabilir:

$$\Upsilon_i^A - \Upsilon_i^B = 2\Phi_i, \quad i = 1,2,\dots,N \tag{171}$$

Burada A ve B aktivasyon fonksiyonlarının üst simgeleri sırasıyla $x \le x_i$ ve $x_i \le x \le x_{i+1}$ aralıklarını ifade eder. Υ_i^A ve Υ_i^B fonksiyonları, ilgili aralıklarının orta noktalarında sıfır olarak tanımlanır. Bu formülasyonun bir sonucu, aktivasyon fonksiyonlarının sayısının istenen hat fonksiyonlarının sayısına bağlanabilmesidir. Bizim yaklaşımımız hat fonksiyonlarının iki katı kadar parçalı lineer fonksiyon gerektirmektedir ($T = 2N$).

Girdi, Bias ve Çıktı ağırlıklarının kısıtlanması

Aktivasyon fonksiyonu gösterimini (denklem (168)) hat fonksiyonları kullanarak 1'e eşitlemek için gereken diğer kısıtlamaları belirleyeceğiz. Problem alanını $[a, b]$ x_i düğümlerini kullanarak aşağıdaki şekilde ayrıklaştıralım:

$$a = x_1 < x_2 < \cdots < x_{N-1} < x_N = b$$

Bu ayrıklaştırma problem ağı olarak adlandırılır. Uygun hat fonksiyonlarını oluşturmak için sınırlarında, $x_0 < a$ ve $x_{N+1} > b$ olacak şekilde iki yardımcı düğüm x_0 ve x_{N+1} gereklidir. $T = 2N$ kısıtını kullanarak denklem (168) şu şekilde yeniden yazılabilir:

$$\sum_{q=1}^{T=2N} \Upsilon_q(x)c_q = \sum_{q=1}^{N} \Upsilon_q(x)c_q + \sum_{q=N+1}^{2N} \Upsilon_q(x)c_q \tag{172}$$

Burada sağ taraftaki toplamlar, genellik kaybı olmaksızın şu şekilde yeniden yazılabilir:

$$\sum_{q=1}^{N} \Upsilon_q c_q = \sum_{i=1}^{N} \Upsilon_i^A u_i \;\; ve \;\; \sum_{q=N+1}^{2N} \Upsilon_q c_q = \sum_{i=1}^{N} \Upsilon_i^B v_i \tag{173}$$

Dolayısıyla,

$$\sum_{q=1}^{T} \Upsilon_q(x)c_q = \sum_{i=1}^{N} \Upsilon_i^A u_i + \sum_{i=1}^{N} \Upsilon_i^B v_i \,. \tag{174}$$

Daha sonra, Şekil 27'deki FFANN ağırlıkları için aşağıdaki değerler seçilerek parçalı lineer aktivasyon fonksiyonları cinsinden temel açılımının hat fonksiyonları cinsinden bir temel açılımına dönüştürülebileceği gösterilebilir (Meade vd 1994). Giriş ağırlıkları ve bias ağırlıkları için,

$$\alpha_i^A = \frac{2}{(x_i - x_{i-1})} \;\; , \qquad \alpha_i^B = \frac{2}{(x_{i+1} - x_i)} \tag{175}$$

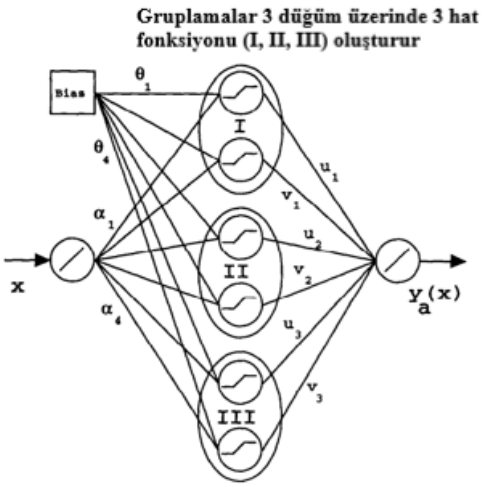
$$\theta_i^A = -\frac{2x_{i-1}}{(x_i - x_{i-1})} - 1, \qquad \theta_i^B = -\frac{2x_i}{(x_{i+1} - x_i)} - 1$$

yazılır. Burada α_i^A, θ_i^A, α_i^B ve θ_i^B denklem (171)'deki aktivasyon fonksiyonlarına karşılık gelir. Çıkış ağırlıkları için,

$$c_i = u_i = \frac{w_i}{2}, \qquad i = 1,2,\dots,N$$

$$c_{i+N} = v_i = \frac{-w_i}{2}, \qquad i = 1,2,\dots,N. \tag{176}$$

Burada ağırlıkların ağ yapısına göre numaralandırılması herhangi bir sırada olabilir.



Şekil 33. Kısıtlı ağırlıklara sahip tek gizli katmanlı ileri beslemeli yapay sinir ağı $(T = 6, N = 3)$

Denklem (171)-(176)'nın uygulanması aşağıdaki denkliği sağlar:

$$y_a(x) = \sum_{q=1}^{T} \Upsilon_q(x) c_q = \sum_{i=1}^{N} \Phi_i(x) w_i \tag{177}$$

Hat fonksiyonları Φ_i, denklem (170)'te olduğu gibi x_i düğümünde 1 değerine sahip olarak tanımlanır ve w_i, hat fonksiyonlarıyla ilişkili temel katsayılardır.

Bu yaklaşımda ağırlıklar kısıtlanmış olsa da ağın yapısı korunmuştur. Şekil 33, Şekil 30'da olduğu gibi üç düğüm ve iki yardımcı düğüm için üç hat fonksiyonu oluşturmak üzere kullanılan altı işlem elemanı için ortaya çıkan yapıyı göstermektedir.

Bu yaklaşımda "eğitim" söz konusu değildir. Girdi ve bias ağırlıklarının değerleri, problem alanındaki belirli bir düğüm sayısı için sabittir ve çıktı ağırlıklarının belirlenmesinden etkin bir şekilde ayrılmaz. Ağın yapısı, w_i katsayılarının uygun değerlerinin bulunması ve ardından w_i'yi kullanarak u_i ve v_i çıkış ağırlık değerlerini belirleme sorununa indirgenmiştir. Genişleme katsayıları w_i'nin hesaplanması çeşitli sayısal yöntemlerle yapılabilir. Ağırlıklı kalanlar yönteminden elde edilen bir prosedür uyguluyoruz (Finlayson vd 1966).

Çıktı ağırlıklarının belirlenmesi: Ağırlıklı kalanlar yöntemi (Method of Weighted Residuals)

$L(y) = g(x)$ şeklinde temsil edilen bir diferansiyel denklem verildiğinde, burada $L(y)$ bağımlı değişken y'de etkili olan bir diferansiyel operatördür, y yerine yaklaşık bir y_a açılımı yazılır:

$$y_a(x) = \sum_{i=1}^{N} \Phi_i(x) w_i \tag{178}$$

Burada $\Phi_i(x)$ temel fonksiyonlardır. Bu nedenle $L(y_a) - g(x) = R(w_1, \dots, w_N, x)$ olup R, hata veya diferansiyel denklem kalanı olarak tanımlanabilecek sıfır olmayan herhangi bir fonksiyondur.

Ağırlıklı kalanlar yönteminde, kalan bir $F(x)$ fonksiyonu (test fonksiyonu) ile çarpılır, problem alanı $[a, b]$ üzerinde entegre edilir ve sıfıra eşitlenir:

$$\int_a^b FRdx = (F, R) = 0 \tag{179}$$

Burada (F, R), F ve R fonksiyonlarının iç çarpımıdır. İntegral sıfıra eşitlenmiş olsa da, R kalan diferansiyel denkleminin sadece ortalamada ve sadece $x_{i+1} - x_i$ sıfıra yaklaştıkça sıfıra yaklaştığına dikkat edilmelidir. Bağlantıcı bir şemada bu, gizli katmandaki işlem elemanı sayısının (T) sonsuza yaklaşmasına izin vermekle eşdeğerdir. Eşitlik (179) o zaman şu anlama gelir:

$$\big(F, L(y_a) - g(x)\big) = 0 \quad veya \quad \big(F, L(y_a)\big) = (F, g) \tag{180}$$

Sonlu farklar, alt alan yöntemleri, sonlu elemanlar, kollokasyon ve spektral yöntemler gibi daha popüler fonksiyon yaklaşımı yöntemlerinin çoğu $F(x)$ fonksiyonunun uygun seçimi ile denklem (180)'den türetilebilir (Fletcher 1984). Fletcher tarafından yapılan Petrov-Galerkin tekniği (Fletcher 1984) kullanacağız; burada $F(x)$, y_a yaklaşımını tanımlamak için kullanılan temel fonksiyonların bir modifikasyonudur:

$$F(x) = F_k(x) = a(x)\Phi_k(x), \qquad k = 1,2, \dots, N \tag{181}$$

Burada $a(x)$, söz konusu problemin sınır koşullarını sağlamaya yardımcı olmak için seçilir. Çıkış ağırlıklarının belirlenmesi için Petrov-Galerkin tekniği seçilmiştir çünkü bu teknik yüksek derecede lineer olmayan veya süreksiz problemlerin ele alınmasını kolaylaştırmaktadır. Petrov-Galerkin tekniğinin yalnızca kullanılacak temel fonksiyonların hat fonksiyonları gibi lineer bağımsız olması durumunda çalışacaktır.

$F(x)$ bu şekilde tanımlandığında integrallerin değerlendirilmesi bir cebirsel denklem sistemine yol açar. Eğer yaklaştırılacak diferansiyel denklem lineer ise bu sistem genellikle lineerdir. Aksi takdirde cebirsel denklemler lineer değildir. Bu denklemler standart tekniklerle tek bir çözüm için değerlendirilebilir (Strang 1980).

Örnek 4: Falkner-Skan Denklemi

Aşağıdaki lineer olmayan adi diferansiyel denklemin çözümüne ve çözümün birinci ve ikinci türevlerine yaklaşmak için tek girişli çok çıkışlı bir FFANN oluşturulur (Meade vd 1994):

$$\frac{d^3f}{d\eta^3} + f\frac{d^2f}{d\eta^2} + \beta\left[1 - \left(\frac{df}{d\eta}\right)^2\right] = 0, \quad için\ 0 \leq \eta \tag{182}$$

ilgili sınır koşulları ile,

$$f = \frac{df}{d\eta} = 0, \ \eta = 0 \tag{183}$$

$$\frac{df}{d\eta} \to 1, \quad \eta \to \infty$$

olup sınır koşullarından;

$$\frac{df}{d\eta} \to 1, \quad \eta \to \infty, \quad o\ zaman\ \frac{d^2f}{d\eta^2} \to 0, \quad \eta \to \infty \tag{184}$$

yazılır. Denklem (182) Falkner-Skan denklemi olarak bilinir ve sabit, iki boyutlu ve sıkıştırılamaz bir momentum sınır tabakasını modelleyen kısmi diferansiyel denklem sisteminin benzerlik dönüşümünden türetilmiştir. Benzerlik dönüşümünün merkezinde aşağıdaki kısıtlamalar yer almaktadır:

$$0 \leq x\ ve\ 0 \leq y, \ \bar{u}(x,y) = \bar{U}(x)\frac{df}{d\eta}(\eta), \ \bar{U}(x) = Kx^{\frac{\beta}{(2-\beta)}}, \ \eta = y\sqrt{\frac{\bar{U}}{(2-\beta)vx}} \tag{185}$$

Burada x ve y sırasıyla akış yönünde ve yüzey normal yönlerinde alansal boyutlardır. Değişken $\bar{u}(x,y)$ boyutsal akış yönünde sınır tabaka hızı, $\bar{U}(x)$ boyutsal dış tabaka hızı, $K, +1$ değerine sahip bir orantı sabiti, v akışkanın kinematik viskozitesi ve $\eta(x,y)$ benzerlik değişkenidir. Denklem (182)'nin türetilmesi ve özellikleri hakkında kapsamlı bir tartışma (White 1974) referansında bulunabilir.

Denklem (182)'nin çözümleri, güçlü elverişli gradyanlar, sınır tabaka ayrılması, hız aşımı ve benzersizlik gibi birçok sınır tabaka fenomenini göstermektedir. β parametresi basınç gradyanının bir ölçüsüdür ve θ dönüş açılarının kamaları ve genişleme köşeleri hakkındaki sınır tabakasıyla doğrudan ilişkilendirilebilir:

$$\theta = \frac{\beta\pi}{2}$$

Denklem (182)'yi fiziksel olarak geçerli akışkan akışlarını modelleyebildiği ve $\bar{u}(x,y)$'nin y ile monoton olarak arttığı durumlarla sınırlandırırsak, o zaman $-0.19884 \leq \beta \leq +\infty$. Denklem (185) ve $\bar{u}(x,y)$'nin bir sonucu olarak $\frac{df}{d\eta}$, kısıtlı β aralığı için η ile monoton olarak artar. Stewartson'ın (Stewartson 1954) $-0.19884 \leq \beta \leq 0$ için, herhangi bir β için (182) denkleminin en az iki çözümü olduğunu ancak çözümlerden yalnızca birinin monoton

olarak artan $\bar{u}(x, y)$ değerini verdiğini gösterdiğine dikkat edilmelidir. Bu örnek problem için β 'nın üç değerini inceleyeceğiz:

$\beta = -0.15$: $\theta = 13.5^{\circ}$ dönüş açısına sahip bir genişleme köşesi etrafında akış

$\beta = 0$: düz bir plaka boyunca akış

$\beta = +0.5$: $\theta = 45^{\circ}$ yarı açılı bir kama etrafında akış

Falkner-Skan denklemini çözme yaklaşımımızda, denklem (182)'deki bağımsız değişken η'nin yarı-sonsuz olduğunu ve denklemin kendisinin sonsuzda uygulanan son koşullara sahip olduğu ve bununla birlikte, β'nin kısıtlı değerleri için (182) numaralı denklemin basit bir asimptotik analizi, monoton olarak artan $\frac{df}{d\eta}$ kısıtlamasını kullanarak, bağımlı değişken f'nin yanı sıra ikinci ve üçüncü türevinin η ile monoton olarak değiştiğini belirler. Bu nedenle denklem (182)'yi bağımlı değişkenin veya türevlerinin bağımsız değişken olarak hareket ettiği bir forma dönüştürmek mümkün olmalıdır. Daha önce de belirtildiği gibi, yaklaşık çözümde taban olarak kullanılan hat fonksiyonları lineerdir, bu da tabanların ikinci, üçüncü ve daha yüksek türevlerinin aynı şekilde sıfır olduğu anlamına gelir. Bir çözüm elde etmek için, sadece bağımsız değişkeni η'den u'ya değiştirmekle kalmıyoruz, aynı zamanda bağımlı değişkeni de f'den τ'ye değiştiriyoruz.

$$u = \frac{df}{d\eta}, \quad ile\ 0 \le u \le 1\ \ ve\ \ \tau = \frac{d^2 f}{d\eta^2} \tag{186}$$

Denklem (186) ve zincir kuralının uygulanmasıyla Falkner-Skan denklemi aşağıdakine indirgenir:

$$\frac{d\tau}{du} + f + \beta \frac{(1 - u^2)}{\tau} = 0 \tag{187}$$

Denklemdeki en yüksek dereceli türev indirgenmiş olsa da denklem (187) hala açık formda f içermektedir. f'yi ortadan kaldırmak için denklem (187)'nin u'ya göre türevini alır ve zincir kuralını uygularız bu da değiştirilmiş bir Falkner-Skan denklemi ile sonuçlanır:

$$\frac{d^2\tau}{du^2} + (1 - 2\beta)\frac{u}{\tau} + \beta(1 - u^2)\frac{d}{du}\left(\frac{1}{\tau}\right) = 0, \ \ için\ 0 \le u \le 1 \tag{188}$$

Denklem (183) ve (184)'deki sınır koşulları dönüştürülmüş değişkenler cinsinden yeniden yazılır:

$$u = 0 \ \ için\ \eta = 0\ ve\ u \to 1\ için\ \eta \to \infty,$$

$$\tau = 0 \ \ için\ u = 1$$

Denklem (179) denklem (188)'e uygulandığında zayıf form elde edilir:

$$\left(F_k, \frac{d^2\tau}{du^2}\right) = -(1-2\beta)\left(F_k, \frac{u}{\tau}\right) - \beta\left(F_k, (1-u^2)\frac{d}{du}\left(\frac{1}{\tau}\right)\right), \quad k = 1,2,\dots,N \qquad (189)$$

Burada, örneğin,

$$\left(F_k, \frac{d^2\tau}{du^2}\right) = \int_0^1 F_k \frac{d^2\tau}{du^2} du$$

τ 'nin türevinin mertebesini azaltmak için denklem (189)'un sol tarafına parçalara göre integral uygulanmalıdır. Bu da şu sonuçları verir:

$$\left(\frac{dF_k}{du}, \frac{d\tau}{du}\right) = (1-2\beta)\left(F_k, \frac{u}{\tau}\right) + \beta\left(F_k, (1-u^2)\frac{d}{du}\left(\frac{1}{\tau}\right)\right) + \left[F_k \frac{d\tau}{du}\right]_0^1 \qquad (190)$$

Şimdi de τ_a yaklaşık çözümünü inceleyelim. u bağımsız değişkeni için problem alanı N düğüme ayrılmıştır;

$$0 = u_1 < u_2 < \cdots < u_N = 1$$

ve problem alanının dışında iki yardımcı düğüm. Giriş ve bias ağırlıkları kullanılarak FFANN'nin parçalı lineer aktivasyon fonksiyonları, problem alanındaki her düğümde merkezlenen hat fonksiyonları oluşturacak şekilde kısıtlanır. Ancak, τ 'yi denklem (178)'de olduğu gibi basit bir baz açılımı ile yaklaştırmak yerine,

$$\tau_a = (1-u)\sum_{i=1}^{N} \Phi_i b1_i \qquad (191)$$

kullanacağız. τ yaklaşımı, $(u = 1)$'de sıfır ve böylece dönüştürülmüş sınır koşulunu sağlar. $\frac{1}{\tau}$ yaklaşımı için denklem (191)'in karşılığını kullanmak yerine, çarpım yaklaşımı yöntemini (Christie vd 1981) kullanacağız ve ayrı bir yaklaşım uygulayacağız:

$$\frac{1}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N} \Phi_i b2_i \qquad (192)$$

$\frac{1}{\tau}$ için temel açılımı $(u = 1)$'de sınırsızdır ve denklem (191)'de olduğu gibi, sınır koşulunu sağlamak için temel katsayılarına gerek yoktur.

Temel katsayılar kümesi arasındaki ilişki, $j = 1, \dots, N$ için her bir u_j düğümünde $\frac{1}{\tau_a}$ ve τ_a'nin tersi eşitlenerek belirlenebilir:

$$\frac{1}{\tau_a(u_j)} = \frac{1}{(1-u_j)} \sum_{i=1}^{N} \Phi_i(u_j) b2_i = \frac{1}{(1-u_j) \sum_{i=1}^{N} \Phi_i(u_j) b1_i} \tag{193}$$

Hat fonksiyonu için,

$$\Phi_i(u_j) = 1, \;\; j = i \quad ve \quad \Phi_i(u_j) = 0, \;\; j \neq i \tag{194}$$

Dolayısıyla, (193) numaralı denklemde j alt simgesini i ile değiştirdikten sonra,

$$b2_i = \frac{1}{b1_i}, \;\; için \; i = 1, \dots, N \tag{195}$$

yazılır. Bu da gerekli ek denklem sistemini sağlar.

Çarpım yaklaşımı yöntemi, her bir çarpımsal değişken grubuna ayrı temel açılımlar atayarak birden fazla bağımlı değişkene sahip problemlere de uygulanabilir. Çarpım yaklaşımı yönteminin uygulanması, problem çözümünün doğruluğunu olumsuz yönde etkilememekle birlikte hesaplama verimliliğini artırmakta ve lineer olmayan problemlerin ağırlıklı kalanlar yöntemiyle ele alınmasını büyük ölçüde basitleştirmektedir (Fletcher 1984).

Yaklaşımların ve türevlerinin zayıf formülasyonda yerine konması şu sonuçları verir:

$$\sum_{i=1}^{N} \left(\frac{dF_k}{du}, (1-u) \frac{d\Phi_i}{du} - \Phi_i \right) b1_i = (1-2\beta) \sum_{i=1}^{N} \left(F_k, \frac{u}{1-u} \Phi_i \right) b2_i$$

$$+\beta \sum_{i=1}^{N} \left(F_k, \frac{(1-u^2)}{(1-u)^2} \left((1-u) \frac{d\Phi_i}{du} + \Phi_i \right) \right) b2_i + \left[F_k \frac{d\tau}{du} \right]_0^1 \tag{196}$$

$\frac{1}{(1-u)}$ terimi mevcut olduğundan bu terim bir şekilde ortadan kaldırılmadığı sürece iç çarpımların değerlendirilmesinde zorluklara neden olacaktır. Bu, denklem (196)'ya Petrov-Galerkin tekniği uygulanarak yapılabilir. Diyelim ki,

$$F_k = (1-u)\Phi_k$$

olsun. Dolayısıyla denklem (196) şu hale gelir:

$$\sum_{i=1}^{N} \left((1-u) \frac{d\Phi_k}{du} - \Phi_k, (1-u) \frac{d\Phi_i}{du} - \Phi_i \right) b1_i = (1-2\beta) \sum_{i=1}^{N} (\Phi_k, u\Phi_i) \, b2_i$$

$$+\beta \sum_{i=1}^{N} \left(\Phi_k, (1+u) \left((1-u) \frac{d\Phi_i}{du} + \Phi_i \right) \right) b2_i + \left[(1-u)\Phi_k \frac{d\tau}{du} \right]_0^1 \tag{197}$$

Eğer aşağıdaki matrisleri tanımlarsak;

$$M1 = M1_{ki} = \left((1-u)\frac{d\Phi_k}{du} - \Phi_k, (1-u)\frac{d\Phi_i}{du} - \Phi_i\right)$$

$$M2 = M2_{ki} = (\Phi_k, u\Phi_i) \tag{198}$$

$$M3 = M3_{ki} = \left(\Phi_k, (1+u)\left((1-u)\frac{d\Phi_i}{du} + \Phi_i\right)\right)$$

o zaman denklem (197) tarafından oluşturulan cebirsel sistem şu şekilde yazılabilir:

$$\sum_{i=1}^{N} M1_{ki} b1_i = (1-2\beta)\sum_{i=1}^{N} M2_{ki}\, b2_i + \beta\sum_{i=1}^{N} M3_{ki}\, b2_i + \left[(1-u)\Phi_k\frac{d\tau}{du}\right]_0^1 \tag{199}$$

ile $k = 1, \dots, N$ için

$$b2_i = \frac{1}{b1_i}, \quad için\ i = 1, \dots, N. \tag{200}$$

Taban olarak hat fonksiyonlarının özelliği, $M1, M2$ ve $M3$ matrislerinin pozitif tanımlı ve üç köşeli olduğunu ve bu nedenle depolamada $N(O(N))$ mertebesinde olduğunu ortaya koymaktadır (Fletcher 1984). Şimdi akı terimini ele alalım:

$$\left[(1-u)\Phi_k\frac{d\tau}{du}\right]_0^1 = \left[(1-u)\Phi_k\frac{d\tau}{du}\right]_1 - \left[(1-u)\Phi_k\frac{d\tau}{du}\right]_0$$

$u = 0 = u_1$ ve $u = 1 = u_N$ olduğundan, denklem (194)'ün uygulanması şunu gösterir:

$$\left[(1-u)\Phi_k\frac{d\tau}{du}\right]_0 = \Phi_k(0)\frac{d\tau}{du}(0) = 0, \quad için\ k = 2, \dots, N \tag{201}$$

Fakat

$$\left[(1-u)\Phi_k\frac{d\tau}{du}\right]_0 = \frac{d\tau}{du}(0), \quad için\ k = 1 \tag{202}$$

ve

$$\left[(1-u)\Phi_k\frac{d\tau}{du}\right]_1 = 0 \quad için\ k = 1, \dots, N. \tag{203}$$

Denklem (187) $(u = 0)$'da değerlendirilir ve $f(0) = 0$ olduğundan,

$$\frac{d\tau}{du}(0) = -\left[f + \beta\frac{(1-u^2)}{\tau}\right]_{u=0} = -\frac{\beta}{\tau(0)}$$

olur. Bu nedenle akı terimi şu şekilde yazılabilir:

$$\left[(1-u)\Phi_k \frac{d\tau}{du}\right]_0^1 = \frac{\beta}{\tau(0)} = \beta \sum_{i=1}^{N} \Phi_i(0) b2_i = \beta b2_1 \,, \;\; için\; k = 1 \qquad (204)$$

Öyle bir $M4$ matrisi tanımlayalım ki;

$$M4_{ki} = (1-2\beta)M2_{ki} + \beta M3_{ki} + \beta \,, \;\; için \;\; k = 1 \;\; ile \;\; i = 1, \dots, N$$

ve

$$M4_{ki} = (1-2\beta)M2_{ki} + \beta M3_{ki} \,, \;\; için \;\; k = 2, \dots, N \;\; ile \;\; i = 1, \dots, N. \qquad (205)$$

Burada $M4$ pozitif tanımlı ve üç köşelidir. Denklem (199), (200), (204) ve (205)'in kullanımı cebirsel sistemle sonuçlanır:

$$\sum_{i=1}^{N} M1_{ki} b1_i - \sum_{i=1}^{N} M4_{ki} \frac{1}{b1_i} = 0 \,, \;\; için \;\; k = 1, \dots, N \qquad (206)$$

Bağımlı değişken τ 'nin yaklaşımı denklem (206)'nın çözümü ile tamamlanacaktır.

Çözüm Prosedürü

Denklem (206) ile tanımlanan sistem, standart lineer olmayan denklem çözücülerden herhangi biri ile çözülebilir. Bir hesaplama yöntemine, özellikle de Powell'ın yöntemine başvuruyoruz. Bu, IMSL sayısal kütüphanesinden DNEQNJ rutini kullanılarak uygulanmıştır (IMSL 1991).

Algoritma skaler fonksiyonu minimize eder.

$$e(b1_1, b1_2, \dots, b1_N) = \sum_{k=1}^{N} (e_k)^2$$

Burada

$$e_k = \sum_{i=1}^{N} M1_{ki} b1_i - \sum_{i=1}^{N} M4_{ki} \frac{1}{b1_i}$$

Tüm çok boyutlu algoritmalarda olduğu gibi Powell'ın yöntemi de sistemin Jacobian matrisini gerektirir. Bu örnek problemde hesaplama verimliliğini artırmak için Jacobian *J* algoritmaya analitik olarak verilmiştir. Jacobian aşağıdaki şekilde hesaplanmaktadır:

$$J = J_{kj} = \frac{de_k}{db1_j} = M1_{kj} + M4_{kj} \left(\frac{1}{b1_j}\right)^2 \qquad (207)$$

$M1$ ve $M4$ matrisleri gibi Jacobian da pozitif, belirli ve üç köşelidir.

Çözüm prosedürü şu şekilde özetlenebilir:

1. $M1$, $M2$ ve $M3$ matrislerini oluşturmak için denklem (198)'in integralleri hesaplanır.
2. β için bir değer seçilir ve $M4$ matrisinin bileşenleri hesaplanır.
3. $b1_i$ için başlangıç değerleri seçilir. Bu örnek için, $i = 1, \dots, N$ için $b1_i$= 1.
4. Her iterasyonda J_{kj} ve e_k'yı değerlendirerek e'yi minimize etmek için Powell yöntemi algoritması uygulanır.

Çözümün Son İşlenmesi

$b1_i$ değerleri bilindiğinde denklem (191)'de verilen τ açılımı tamamlanmış olur. τ ve u kullanılarak f, η ve $\frac{d^2 f}{d\eta^2}$ bilinmeyenleri için temel açılımlarının oluşturulması oldukça basittir.

f'nin yaklaşımının aşağıdaki gibi yazılabileceğini varsayalım:

$$f_a(\eta) = f_a(u) = \sum_{k=1}^{N} \Phi_k(u) w1_k \tag{208}$$

Burada N ve Φ, τ 'nin yaklaştırılması için kullanılanlarla aynıdır.

Denklem (186)'nın koordinat dönüşümünü kullanarak şunu gösteririz:

$u = \frac{df}{d\eta}$, bu yüzden $df = ud\eta$ ve $\tau = \frac{du}{d\eta}$, bu yüzden $d\eta = \frac{1}{\tau} du$

Bu nedenle şunu yazabiliriz:

$$df = \left(\frac{u}{\tau}\right) du \tag{209}$$

Düğümleri kullanarak denklem (209)'u entegre edebiliriz:

$$\int_{u_1}^{u_k} df = f(u_k) - f(u_1) = \int_{u_1}^{u_k} \left(\frac{u}{\tau}\right) du = f_a(u_k) - f_a(u_1) \tag{210}$$

Sınır koşullarından $f(u_1 = 0) = 0 = f_a(0)$ ve hat fonksiyonlarının özelliklerinden $f_a(u_k) = w1_k$, denklem (210) şu hale gelir:

$$w1_k = \int_0^{u_k} \left(\frac{u}{\tau}\right) du \tag{211}$$

Uygun açılımı yerine koyduktan sonra,

$$w1_k = \int_0^{u_k} \frac{u}{(1-u)} \sum_{i=1}^{N} \Phi_i b2_i du = \sum_{i=1}^{N} \left(\int_0^{u_k} \frac{u}{(1-u)} \Phi_i du \right) b2_i = \sum_{i=1}^{N} \mathrm{G1}_{ki} b2_i \tag{212}$$

için $k = 1,2, \dots, N-1$,

burada

$$G1_{ki} = \int_0^{u_k} \frac{u}{(1-u)} \Phi_i du \,, \quad için \quad k = 1,2,\dots,N-1. \tag{213}$$

Eğer $k = N$'de $u_N = 1$ düğümünü kullanırsak $w1_N$'in değeri sonsuz olur. Bu nedenle örneğimiz için,

$$w1_N = \sum_{i=1}^{N} \left(\int_0^{0.99} \frac{u}{1-u} \Phi_i du \right) b2_i = \sum_{i=1}^{N} \mathrm{G1}_{Ni} b2_i \tag{214}$$

yazılır. $w1_k$'nın değerlendirilmesiyle f için temel genişletme tamamlanmış olur.

Bağımsız değişken η'yi u değişkeni cinsinden yaklaşık olarak ifade etmek için şunu yazarız:

$$\eta_a = \eta_a(u) = \sum_{k=1}^{N} \Phi_k(u) w2_k \tag{215}$$

Hatırlayacak olursak:

$$\tau = \frac{du}{d\eta} \,, \quad bu\ yüzden \quad d\eta = \left(\frac{1}{\tau}\right) du$$

$d\eta$ integralini alarak ve $\eta(0) = 0 = \eta_a(0)$ olduğunu kabul ederek;

$$\eta_a(u_k) = \int_0^{u_k} \left(\frac{1}{\tau}\right) du = w2_k$$

yazılır. Bu yüzden,

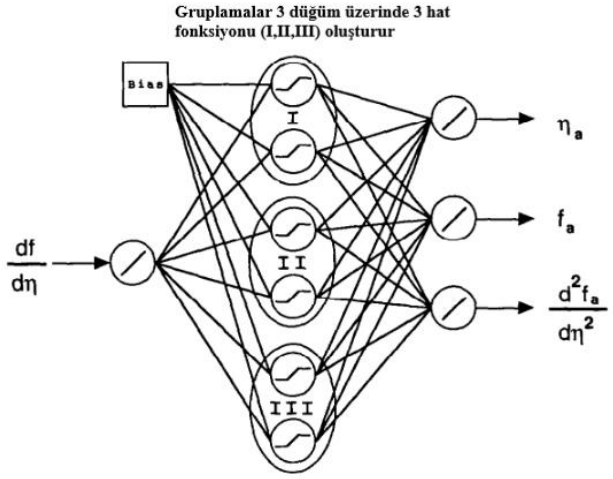
$$w2_k = \sum_{i=1}^{N} G2_{ki} b2_i \,, \quad için \quad k = 1,2,\dots,N \tag{216}$$

olup burada

$$G2_{ki} = \int_0^{u_k} \frac{1}{(1-u)} \Phi_i du \,, \quad için \quad k = 1,2,\dots,N-1 \quad ve$$

$$G2_{Ni} = \int_0^{0.99} \frac{1}{(1-u)} \Phi_i du \tag{217}$$

olarak hesaplanır.



Şekil 34. Örnek 4 için tek girişli ve çok çıkışlı ileri beslemeli yapay sinir ağı ($T = 6, N = 3$)

Geriye f'nin η'ye göre ikinci türevini u bağımsız değişkeni cinsinden yaklaşık olarak hesaplamak kalıyor. Şöyle yazıyoruz:

$$\frac{d^2 f_a}{d\eta^2}(u) = \sum_{k=1}^{N} \Phi_k(u) w3_k \tag{218}$$

$w3_k$'yi belirlemek için, τ 'nin yaklaşımını problem alanındaki düğümlerde $\frac{d^2 f}{d\eta^2}$'nin yaklaşımı ile eşitlenir:

$$\tau_a(u_k) = (1 - u_k) \sum_{i=1}^{N} \Phi_i(u_k) b1_i = \frac{d^2 f_a}{d\eta^2}(u_k) = \sum_{k=1}^{N} \Phi_i(u_k) w3_k$$

Bu da şu sonuçları doğurur:

$$w3_k = (1 - u_k) b1_k \tag{219}$$

Ayrıklaştırma tüm temel açılımları için aynı olduğundan denklem (176)'nın $w1_k$, $w2_k$ ve $w3_k$'ya uygulanması Şekil 34'deki tek girişli ve çok çıkışlı ileri beslemeli ağın oluşturulmasını sağlayacaktır. Belirli bir $\frac{df}{d\eta}$ değeri için ağ η, f ve $\frac{d^2 f}{d\eta^2}$ değerlerini çıkaracaktır.

Örnek 5: Bağlantılı Lineer Olmayan Adi Diferansiyel Denklemler

Tek girişli, çok çıkışlı bir FFANN ve aşağıdaki lineer olmayan ikinci ve üçüncü dereceden bağlantılı adi diferansiyel denklem sisteminin türevleri oluşturulur (Meade vd 1994):

$$\frac{d^3 f}{d\eta^3} - (A_1 f + A_2 g) \frac{d^2 f}{d\eta^2} = 0 \tag{220}$$

$$\frac{d^3g}{d\eta^3} - (A_1 f + A_2 g)\frac{d^2g}{d\eta^2} - A_3\frac{df}{d\eta}\frac{dg}{d\eta} - A_4\left(\frac{dg}{d\eta}\right)^2 - A_5\left(\frac{df}{d\eta}\right)^2 - A_6\lambda - A_7 = 0 \quad (221)$$

$$\frac{d^2\lambda}{d\eta^2} - Pr(A_1 f + A_2 g)\frac{d\lambda}{d\eta} - A_8\left(\frac{d^2f}{d\eta^2}\right)^2 - A_8(A_1 f + A_2 g)\frac{df}{d\eta}\frac{d^2f}{d\eta^2} = 0 \quad (222)$$

Burada $0 \leq \eta$. $A_1, \ldots, A_7$ sabitlerdir ve çeşitli fiziksel parametrelerin kombinasyonlarını temsil eder ve A_8 katsayısı Pr parametresinin bir fonksiyonudur. (220)-(222) denklemleri için ilgili sınır koşulları şunlardır:

$$f = 0,\ \frac{df}{d\eta} = 0,\ g = 0,\ \frac{dg}{d\eta} = 0,\ \lambda = 0,\ için\ \ \eta = 0\ ve\ \frac{df}{d\eta} \to 1, olarak\ \eta \to \infty \quad (223)$$

Ek olarak, sınır koşullarından dolayı;

$$\frac{df}{d\eta} \to 1,\ olarak\ \eta \to \infty,\ o\ zaman\ \frac{d^2f}{d\eta^2} \to 0,\ olarak\ \eta \to \infty. \quad (224)$$

(220)-(222) denklemleri akışkanlar mekaniği alanında pratik açıdan ilgi konusudur. Adi diferansiyel denklemler, süpersonik akışta eğimli bir dik dairesel koni etrafında hem üç boyutlu momentum hem de termal sınır tabakasını modelleyen kısmi diferansiyel denklem sisteminin benzerlik dönüşümünden türetilmiştir (Moore 1951;Reshotko 1957). Eğer (220)-(222) denklemleri anlamlı durumları modelleyecekse, $\frac{df}{d\eta}$, η ile monoton olarak artmalıdır. Bu, A_1'den A_8'e kadar kısıtlamaya eşdeğer olan parametrelerin değerlerini kısıtlayarak gerçekleştirilebilir. Yarı açısı 15° olan bir koni için serbest akış Mach sayısı 7 ve serbest akış Reynolds sayısı 5×10^5 olan havada 10°'lik bir hücum açısı için A_1'den A_7'ye kadar aşağıdaki sayısal değerlere ulaşılmaktadır:

$$\begin{aligned} &A_1 = -0.960428, \quad A_2 = -0.395063, \\ &A_3 = 0.640285, \quad\ A_4 = 0.395063, \\ &A_5 = 0.946661, \quad\ A_6 = -0.665229, \\ &A_7 = -0.665229. \end{aligned} \quad (225)$$

Ayrıca $0.7 \leq Pr \leq 1.0$ kısıtı da kullanılmalıdır, bu da $0 \leq\ A_8 \leq 0.86$ 'e karşılık gelir. (220)-(222) denklemlerinin türetilmesi, monoton olarak artan $\frac{df}{d\eta}$ gerekliliği ve A_1 ile A_8 arasındaki katsayıların değerlendirilmesine ilişkin ayrıntılı bir tartışma (Meade vd 1989) referansında bulunabilir.

Bağlantılı sistemi modelleyen FFANN'ı oluşturmak için geçerli denklemler daha uygulanabilir bir forma dönüştürülmelidir. Denklem (220)-(222)'yi çözme yaklaşımımız

Falkner-Skan problemi için kullanılan yaklaşıma oldukça benzer olacaktır. Daha önce belirtilen A_1 'den A_8'e kadar olan kısıtlamalar için $\frac{df}{d\eta}$, η ile monoton olarak değiştiğinden türev, bağımsız değişken olarak kullanılabilir. Aşağıda değişken değişimi veriliyor:

$$u = \frac{df}{d\eta}, \;\; ile \; 0 \leq u \leq 1, \quad \tau = \frac{d^2 f}{d\eta^2} \tag{226}$$

$$r = \frac{dg}{d\eta}, \;\; s = \frac{d\lambda}{d\eta}, \;\; p = \frac{d\tau}{d\eta}, \;\; q = \frac{dr}{d\eta}$$

Bağlantılı denklemler dönüştürülürken f ve g'ye olan bağlılık ortadan kaldırılmalıdır. Denklem (220) yeniden düzenlendiğinde,

$$(A_1 f + A_2 g) = \frac{d^3 f}{d\eta^3}\left(\frac{d^2 f}{d\eta^2}\right)^{-1} \tag{227}$$

olarak yazılır. Denklem (227)'nin denklem (220)'nin türevinde ve denklem (221), denklem (222)'de yerine konması sırasıyla şu sonuçları verir:

$$\frac{d^4 f}{d\eta^4} = \left(A_1 \frac{df}{d\eta} + A_2 \frac{dg}{d\eta}\right)\frac{d^2 f}{d\eta^2} + \frac{d^3 f}{d\eta^3}\left(\frac{d^2 f}{d\eta^2}\right)^{-1}\frac{d^3 f}{d\eta^3} \tag{228}$$

$$\frac{d^3 g}{d\eta^3} = \frac{d^3 f}{d\eta^3}\left(\frac{d^2 f}{d\eta^2}\right)^{-1}\frac{d^2 g}{d\eta^2} + A_3 \frac{df}{d\eta}\frac{dg}{d\eta} + A_4\left(\frac{dg}{d\eta}\right)^2 + A_5\left(\frac{df}{d\eta}\right)^2 + A_6\lambda + A_7 \tag{229}$$

ve

$$\frac{d^2 \lambda}{d\eta^2} = Pr\frac{d^3 f}{d\eta^3}\left(\frac{d^2 f}{d\eta^2}\right)^{-1}\frac{d\lambda}{d\eta} + A_8\left(\frac{d^2 f}{d\eta^2}\right)^2 + A_8\frac{d^3 f}{d\eta^3}\frac{df}{d\eta} \tag{230}$$

Dönüştürülmüş bağımlı değişkenlerin yerine konması ve zincir kuralının kullanılmasıyla, (228)-(230) denklemleri için,

$$\frac{dp}{du} = A_1 u + A_2 r + \left(\frac{p}{\tau}\right)^2 \tag{231}$$

$$\frac{dq}{du} = \left(\frac{p}{\tau}\right)\left(\frac{q}{\tau}\right) + A_3 u\left(\frac{r}{\tau}\right) + A_4\left(\frac{r^2}{\tau}\right) + A_5 u^2\left(\frac{1}{\tau}\right) + A_6\left(\frac{\lambda}{\tau}\right) + A_7\left(\frac{1}{\tau}\right) \tag{232}$$

ve

$$\frac{ds}{du} = Pr\left(\frac{p}{\tau}\right)\left(\frac{s}{\tau}\right) + A_8\tau + A_8 u\left(\frac{p}{\tau}\right) \tag{233}$$

yazılır. Denklem (226)'nın değişkenleri üzerinde zincir kuralının uygulanması, üç adet bağlantılı lineer olmayan ikinci ve üçüncü derece diferansiyel denklemden oluşan sistemi, altı adet birinci derece denklemden oluşan bir sisteme indirgeyerek dönüşümü tamamlar:

$$\frac{dp}{du} - A_1 u - A_2 r - \left(\frac{p}{\tau}\right)^2 = 0$$

$$\frac{dq}{du} - \left(\frac{p}{\tau}\right)\left(\frac{q}{\tau}\right) - A_3 u\left(\frac{r}{\tau}\right) - A_4\left(\frac{r^2}{\tau}\right) - A_5 u^2\left(\frac{1}{\tau}\right) - A_6\left(\frac{\lambda}{\tau}\right) - A_7\left(\frac{1}{\tau}\right) = 0 \quad (234)$$

$$\frac{ds}{du} - Pr\left(\frac{p}{\tau}\right)\left(\frac{s}{\tau}\right) - A_8\tau - A_8 u\left(\frac{p}{\tau}\right) = 0$$

$$\frac{d\tau}{du} - \left(\frac{p}{\tau}\right) = 0, \quad \frac{dr}{du} - \left(\frac{q}{\tau}\right) = 0, \quad \frac{d\lambda}{du} - \left(\frac{s}{\tau}\right) = 0$$

Şimdi geriye dönüştürülmüş sistem için yeni sınır koşullarını bulmak kalıyor. Denklem (220)-(222), $\eta = 0$ ve $\eta \to \infty$ 'da denklem (224) ve denklem (226) ile birleştiğinde şu sonuç ortaya çıkar:

$$p = r = \lambda = 0, \ \ için\ u = 0 \quad (235)$$

$$\tau = q = s = 0, \ \ için\ u = 1$$

Artık dönüştürülmüş bağımlı değişkenleri yaklaşık olarak hesaplamak için temel açılımlar kullanılabilir. Çarpım yaklaşımı yöntemi kullanılarak bağımlı dönüşüm değişkenleri için aşağıdaki yaklaşımlar yazılabilir:

$$\tau_a = (1-u)\sum_{i=1}^{N} \Phi_i b1_i, \qquad r_a = \sum_{i=1}^{N} \Phi_i b2_i$$

$$\lambda_a = \sum_{i=1}^{N} \Phi_i b3_i, \qquad p_a = \sum_{i=1}^{N} \Phi_i b4_i$$

$$q_a = (1-u)\sum_{i=1}^{N} \Phi_i b5_i, \qquad s_a = (1-u)\sum_{i=1}^{N} \Phi_i b6_i$$

ve çarpımsal gruplar için,

$$\frac{p_a}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N} \Phi_i b7_i, \qquad \frac{q_a}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N} \Phi_i b8_i$$

$$\frac{s_a}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N} \Phi_i b9_i, \quad \left(\frac{p_a}{\tau_a}\right)^2 = \frac{1}{(1-u)^2}\sum_{i=1}^{N} \Phi_i b10_i$$

$$\left(\frac{p_a}{\tau_a}\right)\left(\frac{q_a}{\tau_a}\right) = \frac{1}{(1-u)^2}\sum_{i=1}^{N} \Phi_i b11_i, \quad \frac{{r_a}^2}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N} \Phi_i b12_i \quad (236)$$

$$\frac{1}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N} \Phi_i b13_i, \qquad \frac{\lambda_a}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N} \Phi_i b14_i$$

$$\left(\frac{p_a}{\tau_a}\right)\left(\frac{s_a}{\tau_a}\right) = \frac{1}{(1-u)^2}\sum_{i=1}^{N} \Phi_i b15_i \ , \qquad \frac{r_a}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N} \Phi_i b16_i$$

Burada N, problem alanındaki düğüm sayısıdır.

Temel katsayılardan sadece $b1_i$'den $b6_i$'ye kadar olanlar lineer bağımsızdır. Kalan temel katsayılar kümesi arasındaki ilişki denklik argümanları kullanılarak belirlenebilir:

$$\begin{aligned} b7_i &= b4_i b13_i, & b8_i &= b5_i b13_i \\ b9_i &= b6_i b13_i, & b10_i &= (b4_i b13_i)^2 \\ b11_i &= b4_i b5_i (b13_i)^2, & b12_i &= (b2_i)^2 b13_i \\ b13_i &= \frac{1}{b1_i}, & b14_i &= b3_i b13_i \\ b15_i &= b4_i b6_i (b13_i)^2, & b16_i &= b2_i b13_i \end{aligned} \tag{237}$$

için $i = 1, \dots, N$.

Çözümü dönüştürülmüş problem denkleminin (234) açılım katsayılarını verecek cebirsel sistemleri formüle etmek için, $F_k(u) = (1-u)^2 \Phi_k(u)$ ile Petrov-Galerkin yöntemini uyguluyoruz:

$$\sum_{i=1}^{N} M1_{ki}\, b4_i - A_1 v_k - A_2 \sum_{i=1}^{N} M3_{ki}\, b2_i - \sum_{i=1}^{N} M4_{ki}\, b10_i = 0$$

$$\sum_{i=1}^{N} M7_{ki}\, b5_i - \sum_{i=1}^{N} M2_{ki}\,(A_4 b12_i + A_7 b13_i + A_6 b14_i) - \sum_{i=1}^{N} M4_{ki}\, b11_i$$
$$- A_3 \sum_{i=1}^{N} M5_{ki}\, b16_i - A_5 \sum_{i=1}^{N} M6_{ki}\, b13_i = 0$$

$$\sum_{i=1}^{N} M7_{ki}\, b6_i - Pr \sum_{i=1}^{N} M4_{ki}\, b15_i - A_8 \left(\sum_{i=1}^{N} M8_{ki}\, b1_i - \sum_{i=1}^{N} M5_{ki}\, b7_i \right) = 0 \tag{238}$$

$$\sum_{i=1}^{N} M7_{ki}\, b1_i - M2_{ki} b7_i = 0$$

$$\sum_{i=1}^{N} M1_{ki}\, b2_i - M2_{ki} b8_i = 0$$

$$\sum_{i=1}^{N} M1_{ki}\, b3_i - M2_{ki} b9_i = 0$$

Burada $M1, \dots, M8$ matrisleri ve v vektörü aşağıdaki gibi tanımlanır:

$$M1_{ki} = \left((1-u)^2 \Phi_k, \frac{d\Phi_i}{du}\right), \qquad M2_{ki} = \left((1-u)\Phi_k, \Phi_i\right)$$

$$M3_{ki} = ((1-u)^2 \Phi_k, \Phi_i), \qquad M4_{ki} = (\Phi_k, \Phi_i)$$

$$M5_{ki} = (u(1-u)\Phi_k, \Phi_i), \qquad M6_{ki} = (u^2(1-u)\Phi_k, \Phi_i) \quad (239)$$

$$M7_{ki} = \left((1-u)^2 \Phi_k, \left((1-u)\frac{d\Phi_i}{du} - \Phi_i\right)\right), \qquad M8_{ki} = ((1-u)^3 \Phi_k, \Phi_i)$$

$$v_k = ((1-u)^2 \Phi_k, u)$$

Denklem (235)'in sınır koşulları şunları verir:

$$b2_1 = b3_1 = b4_1 = 0, \ \ (u = 0) \qquad (240)$$

$$b1_N = b5_N = b6_N = 1, \ \ (u = 1)$$

Bu kısıtlamalar matrislerin uygun satırlarının değiştirilmesiyle cebirsel denklem sistemine dahil edilir.

Çözüm Prosedürü

Denklem (238)'deki altı denklem kümesi $b1_i, b2_i, \dots, b6_i$ katsayıları için çözülmelidir. Lineer olmayan denklemleri çözmek için Powell'ın yöntemine başvuruyoruz. Bu durumda algoritma aşağıdaki skaler fonksiyonu minimize eder:

$$e(b1_1, \dots, b1_N, b2_1, \dots, b2_N, \dots, b6_1, \dots, b6_N) = \sum_{k=1}^{N} \left(\left(e_k^{(1)}\right)^2 + \left(e_k^{(2)}\right)^2 + \cdots + \left(e_k^{(6)}\right)^2\right)$$

Burada

$$e_k^{(1)} = \sum_{i=1}^{N} M1_{ki}\, b4_i - A_1 v_k - A_2 \sum_{i=1}^{N} M3_{ki}\, b2_i - \sum_{i=1}^{N} M4_{ki}\, b10_i$$

$$e_k^{(2)} = \sum_{i=1}^{N} M7_{ki}\, b5_i - \sum_{i=1}^{N} M2_{ki}\, b2_i (A_4 b12_i + A_7 b13_i + A_6 b14_i) - \sum_{i=1}^{N} M4_{ki}\, b11_i$$
$$- A_3 \sum_{i=1}^{N} M5_{ki}\, b16_i - A_5 \sum_{i=1}^{N} M6_{ki}\, b13_i$$

$$e_k^{(3)} = \sum_{i=1}^{N} M7_{ki}\, b6_i - Pr \sum_{i=1}^{N} M4_{ki}\, b15_i - A_8 \left(\sum_{i=1}^{N} M8_{ki}\, b1_i - \sum_{i=1}^{N} M5_{ki}\, b7_i \right) \quad (241)$$

$$e_k^{(4)} = \sum_{i=1}^{N} M7_{ki}\, b1_i - M2_{ki} b7_i$$

$$e_k^{(5)} = \sum_{i=1}^{N} M1_{ki}\, b2_i - M2_{ki} b8_i$$

$$e_k^{(6)} = \sum_{i=1}^{N} M1_{ki}\, b3_i - M2_{ki} b9_i$$

Jacobian (J) hesaplama verimliliğini artırmak için analitik olarak değerlendirilir. Bu durumda Jacobian $(6N) \times (6N)$ boyutundadır. Her biri $N \times N$ alt-Jacobian olarak düşünülebilecek 36 bölümden oluşmaktadır. Örneğin, bir bölüm şu şekilde tanımlanmışsa;

$$J^{(AB)} = \frac{de_k^{(A)}}{dbB_j}, \quad burada\ A, B = 1, \dots, 6$$

sonra

$$J = \begin{bmatrix} J^{(11)} & \cdots & J^{(16)} \\ \vdots & \ddots & \vdots \\ J^{(61)} & \cdots & J^{(66)} \end{bmatrix}.$$

Jacobian matrisinin her bir bölümü, hat fonksiyonunun özellikleri nedeniyle üç köşelidir. Sonuç olarak, tüm Jacobian matrisi seyrektir.

Çözümün Son İşlenmesi

$b1_i, b2_i, \dots, b6_i$ katsayıları biliniyorsa, $b7_i, b8_i, \dots, b16_i$ hesaplanabilir. Bağımlı dönüşüm değişkenlerinin yaklaşımları tamamlanmıştır. Ağa giriş değişkeni olarak $u = \frac{df}{d\eta}$ kullanıldığında geriye değişkenler $\eta, f, \frac{d^2 f}{d\eta^2}, g, \frac{dg}{d\eta}, \frac{d^2 g}{d\eta^2}, \lambda$ ve $\frac{d\lambda}{d\eta}$ için çıkış ağırlıklarını belirlemek kalıyor.

Yaklaşımların temel açılımları aşağıdaki gibi yazılabilir:

$$\eta_a = \sum_{i=1}^{N} \Phi_i(u) w1_i, \qquad f_a = \sum_{i=1}^{N} \Phi_i(u) w2_i, \qquad \frac{d^2 f_a}{d\eta^2} = \sum_{i=1}^{N} \Phi_i(u) w3_i,$$

$$g_a = \sum_{i=1}^{N} \Phi_i(u) w4_i, \quad \frac{dg_a}{d\eta} = \sum_{i=1}^{N} \Phi_i(u) w5_i, \quad \frac{d^2 g_a}{d\eta^2} = \sum_{i=1}^{N} \Phi_i(u) w6_i, \qquad (242)$$

$$\lambda_a = \sum_{i=1}^{N} \Phi_i(u) w7_i, \qquad \frac{d\lambda_a}{d\eta} = \sum_{i=1}^{N} \Phi_i(u) w8_i$$

$w1_i, \dots, w8_i$ temel katsayılarını belirleme yaklaşımımız örnek 4'de kullanılanla aynı olacaktır. Bu nedenle, (209)-(219) denklemlerinden hemen şunu yazabiliriz:

$$w1_k = \sum_{i=1}^{N} G2_{ki} b13_i, \quad w2_k = \sum_{i=1}^{N} G1_{ki} b13_i, \qquad w3_k = (1 - u_k) b1_k,$$

$$w4_k = \sum_{i=1}^{N} G2_{ki} b16_i, \quad w5_k = b2_k, \qquad w6_k = (1 - u_k) b5_k, \qquad (243)$$

$$w7_k = \sum_{i=1}^{N} G2_{ki} b9_i, \quad w8_k = (1 - u_k) b6_k$$

Burada $k = 1, \dots, N$. $w1_k$'dan $w8_k$'ya kadar olan çıkış ağırlıklarının belirlenmesi ve denklem (176)'nın uygulanması ile denklem (220)-(222)'deki FFANN modeli tamamlanmış olur.

Yapay Sinir Ağı ile Adi ve Kısmi Diferansiyel Denklemlerin Çözümü

Diferansiyel denklemlerin nümerik çözümlerini elde etmek için birçok farklı yöntem geliştirilmiştir. Lagaris vd. (Lagaris vd 1998), adi ve kısmi diferansiyel denklemleri çözmek için ileri beslemeli yapay sinir ağının kullanıldığı bir yöntem öne sürmüşlerdir. Bu yöntemde, başlangıç ve sınır değer problemlerini çözmek için, yapay sinir ağı içeren bir deneme çözümü kullanılmaktadır. İleri beslemeli yapay sinir ağları fonksiyon yaklaşımı amacıyla da kullanıldığı için ağı içeren deneme çözümü gerçek çözüme yakınsayabilmektedir. Bu deneme çözümü diferansiyellenebilir ve kapalı analitik bir formdadır. Çözüm iki terimin toplamından oluşmaktadır; ilk terim başlangıç/sınır değer koşullarından oluşmaktadır. İkinci terim ise ileri beslemeli yapay sinir ağını içermektedir. Deneme çözümünü gerçek çözüme yaklaştırmak için, öğrenme algoritmaları kullanılarak ağ eğitilmektedir.

Söz edilen yöntemi açıklamak için, belirli sınır koşullarına sahip genel bir diferansiyel denklem ele alalım:

$$F\left(\vec{x}, \Psi(\vec{x}), \nabla\Psi(\vec{x}),\ \nabla^2\Psi(\vec{x}), \ldots, \nabla^n\Psi(\vec{x})\right) = 0, \qquad \vec{x} \in \mathrm{D} \tag{244}$$

Burada, $\vec{x} = (x_1, x_2, \ldots, x_n) \in R^n$ ve $\Psi(\vec{x})$ denklemin çözümünü belirtmektedir. $D \subset R^n$ tanım kümesi, ∇ ise diferansiyel operatördür.

D tanım bölgesi ve S bu bölgenin sınırı olmak üzere, tanım bölgesinin ve sınırının ayrıklaştırılmış hali $\widehat{D}$ ve Ŝ ile gösterilmektedir. O halde, denklem (244) şu formda yazılabilir:

$$F\left(\vec{x}_i, \Psi(\vec{x}_i), \nabla\Psi(\vec{x}_i),\ \nabla^2\Psi(\vec{x}_i), \ldots, \nabla^n\Psi(\vec{x}_i)\right) = 0, \qquad \forall \vec{x}_i \in \widehat{D} \tag{245}$$

Bu denklemin çözümünde kullanılacak deneme çözümü $\Psi_t(\vec{x}, \vec{p})$ ile gösterilmektedir. Burada, $\vec{p}$ vektörü yapay sinir ağının belirlenmesi gereken parametrelerinden oluşan bir vektördür.

Deneme çözümünü denklem (245)'de yerine konulup sınır koşullarını da dikkate aldığımızda, denklem (245) şu şekilde bir optimizasyon problemine dönüşmektedir:

$$\min_{\vec{p}} \sum_{x_i \in \widehat{D}} F\left(\vec{x}_i, \Psi_t(\vec{x}_i, \vec{p}), \nabla\Psi_t(\vec{x}_i, \vec{p}), \nabla^2\Psi_t(\vec{x}_i, \vec{p}), \ldots, \nabla^n\Psi_t(\vec{x}_i, \vec{p})\right)^2 \tag{246}$$

Deneme çözümü, iki terimin toplamı olarak şu şekilde ifade edilebilir:

$$\Psi_t(\vec{x}, \vec{p}) = A(\vec{x}) + f\left(\vec{x}, N(\vec{x}, \vec{p})\right) \tag{247}$$

Burada, $A(\vec{x})$ başlangıç/sınır değer koşullarını belirtir. Optimize edilmesi gereken herhangi bir parametre içermez. İkinci terim f ise, sınır koşullarına etki etmeyecek şekilde ileri beslemeli yapay sinir ağının $N(\vec{x}, \vec{p})$ çıktısını içerir. $\vec{p}$ vektörü ağırlık ve bias değerlerinden oluşmaktadır.

Problem (246)'yı minimize etmek için her bir $\vec{x}_i$ vektörüne karşılık gelen hatanın yani $F(\vec{x}_i)$'nin sıfır olması gerekmektedir. Bunun için yapay sinir ağı eğitilmelidir. Ağı eğitmek için farklı algoritmalar kullanılabilir. Örneğin, geri yayılım algoritması bu amaçla en sık kullanılan algoritmalardan biridir (Lagaris vd 1998).

Gradyan Hesaplama

$N(\vec{x}, \vec{p})$'yi içeren deneme çözümünü diferansiyel denklemde yerine koymak için deneme çözümünün x'e göre türevleri gerekmektedir. Bu nedenle ağın çıktısının x'e göre türevleri bilinmelidir. Ayrıca deneme çözümü, denklemde yerine koyulduktan sonra (246) probleminin minimize edilmesi gerekmektedir. Bu amaçla kullanılacak birçok algoritma için, deneme çözümünün ve türevlerinin, ağırlık ve bias değerlerine göre türevleri gerekmektedir.

Gradyan hesaplamalarını göstermek için n girdi birimine, bir gizli katmana ve bir çıktı birimine sahip birçok katmanlı algılayıcı ele alalım. Gizli katmandaki yapay nöron sayısını H ve çıktı biriminde kullanılan aktivasyon fonksiyonunu doğrusal fonksiyon alalım. Girdi vektörü $\vec{x} = (x_1, x_2, \dots, x_n)$ olmak üzere, çok katmalı algılayıcının çıktısı şu şekilde yazılabilir:

$$N = \sum_{i=1}^{H} v_i \sigma(z_i) \tag{248}$$

Burada, $\sigma(z_i)$, $i = 1,2, \dots, H$ gizli katmanda yer alan nöronların çıktılarını, σ ise gizli katmanda kullanılan sigmoid aktivasyon fonksiyonunu belirtmektedir. Ayrıca, gizli katmanı çıktı katmanına bağlayan ağırlık değerleri de v_i ile gösterilmiştir. Gizli katmandaki nöronların çıktıları ise şu şekilde yazılabilir:

$$\sigma(z_i) = \sigma\left(\sum_{j=1}^{n} w_{ij} x_j + u_i\right) \quad i = 1,2, \dots, H \tag{249}$$

Yukarıdaki denklemde w_{ij}, girdi katmanını çıktı katmanına bağlayan ağırlık değerlerini, u_i ise gizli katmandaki nöronların bias değerlerini göstermektedir.

İlk olarak, ağın çıktısının girdi değerlerine göre türevlerini hesaplayalım. Yapay sinir ağı çıktısının, x_j girdisine göre türevi;

$$\frac{\partial N}{\partial\, x_j} = \sum_{i=1}^{H} v_i w_{ij} \sigma_i^{(1)} \tag{250}$$

şeklindedir. Denklem (250)'de yer alan, $\sigma_i = \sigma(z_i)$ dir. $\sigma^{(1)}$ sembolü ise, aktivasyon fonksiyonunun birinci mertebeden türevini belirtmektedir. Benzer şekilde ağın çıktısının x_j girdi vektörüne göre k. mertebeden türevi ise şu şekilde gösterilir:

$$\frac{\partial^k N}{\partial x_j^k} = \sum_{i=1}^{H} v_i w_{ij}^k \sigma_i^{(k)} \tag{251}$$

Yukarıdaki denklemde $\sigma^{(k)}$, aktivasyon fonksiyonunun k. mertebeden türevini belirtmektedir. w_{ij}^k ise, w_{ij} ağırlık değerinin k. kuvvetini göstermektedir.

Genel olarak, ağın çıktısının herhangi bir girdiye göre herhangi bir mertebeden türevi de aynı şekilde ifade edilebilir:

$$\frac{\partial^{\lambda_1}}{\partial {x_1}^{\lambda_1}} \frac{\partial^{\lambda_2}}{\partial {x_2}^{\lambda_2}} \dots \frac{\partial^{\lambda_n}}{\partial {x_n}^{\lambda_n}} N = \sum_{i=1}^{n} v_i P_i \sigma_i^{(\Lambda)} = N_g(\vec{x}) \tag{252}$$

Burada,

$$P_i = \prod_{k=1}^{n} w_{ik}^{\lambda_k} \tag{253}$$

ve

$$\Lambda = \sum_{i=1}^{n} \lambda_i \tag{254}$$

dir.

Ağın herhangi girdilere göre türevi alındığında, sonucun tek gizli katmana sahip ileri beslemeli bir ağın çıktısına $N_g(\vec{x})$ eşit olduğu denklem (252) ile görülmektedir. Bu ağ, ilk başta ele aldığımız ağın parametrelerini içermektedir. Burada ilk başta ele aldığımız ağın v_i parametresinin yerini $v_i P_i$ almıştır. Ayrıca gizli katmandaki nöronların aktivasyon fonksiyonu olarak Λ. mertebeden sigmoid aktivasyon fonksiyonu alınmıştır.

Ağın çıktısının girdi değerlerine göre türevleri hesaplandıktan sonra N_g'nin ağın ağırlık ve bias değerlerine göre gradyanı;

$$\frac{\partial N_g}{\partial v_i} = P_i \sigma_i^{(\Lambda)} \tag{255}$$

$$\frac{\partial N_g}{\partial u_i} = v_i P_i \sigma_i^{(\Lambda+1)} \tag{256}$$

$$\frac{\partial N_g}{\partial w_{ij}} = x_j v_i P_i \sigma_i^{(\Lambda+1)} + v_i \lambda_j w_{ij}^{\lambda_j - 1} \left(\prod_{k=1, k\neq j} w_{ik}^{\lambda_k} \right) \sigma_i^{(\Lambda)} \tag{257}$$

şeklinde yazılabilir.

Hatanın ağırlık ve bias değerlerine göre gradyanı belirlendikten sonra (246) ifadesini optimize etmek için geri yayılım algoritması veya farklı minimizasyon yöntemleri kullanılabilir.

Adi diferansiyel denklemlerin çözümü

Yöntemi göstermek için birinci mertebeden adi diferansiyel denklemini ele alalım:

$$\frac{d\Psi(x)}{dx} = f(x, \Psi) \tag{258}$$

$x \in [0,1]$ ve başlangıç koşulu $\Psi(0) = A$ olmak üzere (258) probleminin iki kısımdan oluşan deneme çözümü şu şekilde yazılabilir:

$$\Psi_t(x) = A + xN(x, \vec{p}) \tag{259}$$

(259) eşitliğinde yer alan $N(x, \vec{p})$, ileri beslemeli yapay sinir ağının çıktısıdır. Burada, $x = (x_1, x_2, \dots, x_n)$ girdi değerlerini, $\vec{p}$ ise ağırlık değerlerinden (ağın bias değerleri de ağırlık vektörüne dahil edilmiştir) oluşan ağırlık vektörünü belirtmektedir.

Deneme çözümünü (258) denkleminde yerine koyabilmek için, (259) deneme çözümünün x'e göre birinci mertebeden türevi;

$$\frac{d\Psi_t(x)}{dx} = N(x, \vec{p}) + x\frac{dN(x, \vec{p})}{dx} \tag{260}$$

seklinde hesaplanır. Daha sonra, deneme çözümünü ve türevini denklem (258)'de yerine koyduğumuzda, hata fonksiyonunu şu şekilde yazabiliriz:

$$E(\vec{p}) = \frac{1}{2}\sum_i \left\{\frac{d\Psi_t(x_i)}{dx} - f\big(x_i, \Psi_t(x_i)\big)\right\}^2 \tag{261}$$

Burada x_i değerleri, x'in tanımlı olduğu $[0,1]$ aralığındaki noktalardır.

(261) denklemi ile verilen hata fonksiyonu elde edildikten sonra, bu fonksiyonu minimize etmek için farklı algoritmalar kullanılabilir. Burada, gradyan inişi algoritmasının uygulanışı gösterilecektir. Tek gizli katmana sahip ileri beslemeli bir yapay sinir ağı düşünelim. Girdi katmanını gizli katmana bağlayan ağırlıklar w_j ile, gizli katmanı çıktı katmanına bağlayan ağırlıklar ise v_l ile gösterilsin. O halde, ağırlık değerleri şu şekilde güncellenir:

$$\frac{\partial E(\vec{p})}{\partial w_j} = \frac{\partial}{\partial w_j}\left(\sum_i \left\{\frac{d\Psi_t(x_i)}{dx} - f\big(x_i, \Psi_t(x_i)\big)\right\}^2\right) \tag{262}$$

$$\frac{\partial E(\vec{p})}{\partial v_l} = \frac{\partial}{\partial v_l}\left(\sum_i \left\{\frac{d\Psi_t(x_i)}{dx} - f\big(x_i, \Psi_t(x_i)\big)\right\}^2\right) \tag{263}$$

$$w_j^{k+1} = w_j^k - \eta\frac{\partial E(\vec{p})}{\partial w_j^k} \tag{264}$$

$$v_l^{k+1} = v_l^k - \eta\frac{\partial E(\vec{p})}{\partial v_l^k} \tag{265}$$

Burada, k iterasyon sayısını belirtmektedir. Hata fonksiyonu minimize edilene kadar ağırlık değerleri her iterasyon adımında güncellenir (Lagaris vd 1998).

Aynı prosedür ikinci mertebeden adi diferansiyel denklem için de uygulanabilir:

$$\frac{d^2\Psi(x)}{dx^2} = f\left(x, \Psi, \frac{d\Psi}{dx}\right) \tag{266}$$

Başlangıç değer problemi için; $x \in [0,1]$, $\Psi(0) = A$ ve $\frac{d}{dx}\Psi(0) = A'$ olmak üzere deneme çözümü şu şekilde yazılabilir:

$$\Psi_t(x) = A + A'x + x^2 N(x, \vec{p}) \tag{267}$$

Burada x girdi değeri ve $\vec{p}$ agırlık vektörü olmak üzere, $N(x, \vec{p})$ yapay sinir ağının çıktısını belirtmektedir.

Deneme çözümünün x'e göre birinci ve ikinci mertebeden türevleri;

$$\frac{d\Psi_t(x)}{dx} = A' + 2xN(x, \vec{p}) + x^2 \frac{dN(x, \vec{p})}{dx} \tag{268}$$

$$\frac{d^2\Psi_t(x)}{dx^2} = 2N(x, \vec{p}) + 4x \frac{dN(x, \vec{p})}{dx} + x^2 \frac{d^2N(x, \vec{p})}{dx^2} \tag{269}$$

seklindedir. (268), (269) denklemleri ve deneme çözümü, (266) ikinci mertebeden başlangıç değer probleminde yerine yazıldıktan sonra, hata fonksiyonu şu şekilde yazılabilir:

$$E(\vec{p}) = \frac{1}{2}\sum_i \left\{\frac{d^2\Psi_t(x_i)}{dx^2} - f\left(x_i, \Psi_t(x_i), \frac{d\Psi_t(x_i)}{dx}\right)\right\}^2 \tag{270}$$

Hata fonksiyonu belirlendikten sonra, yapay sinir ağının parametreleri güncellenir. Bu güncelleme hata fonksiyonu minimize edilene kadar tekrarlanır (Lagaris vd 1998).

İkinci mertebeden diferansiyel denklem için Dirichlet sınır değer problemi:

$$\frac{d^2\Psi(x)}{dx^2} = f\left(x, \Psi, \frac{d\Psi}{dx}\right) \tag{271}$$

Dirichlet sınır değer problemi için; $x \in [0,1]$, $\Psi(0) = A$ ve $\Psi(1) = B$ olmak üzere deneme çözümü şu şekilde yazılır:

$$\Psi_t(x) = A(1 - x) + Bx + x(x - 1)N(x, \vec{p}) \tag{272}$$

Deneme çözümünü, (271) denkleminde yerine koyabilmek için, deneme çözümünün x'e göre türevleri alınır:

$$\frac{d\Psi_t(x)}{dx} = B - A + (x - 1)N(x, \vec{p}) + xN(x, \vec{p}) + x(x - 1)\frac{dN(x, \vec{p})}{dx} \tag{273}$$

$$\frac{d^2\Psi_t(x)}{dx^2} = 2N(x,\vec{p}) + 2x\frac{dN(x,\vec{p})}{dx} + 2(x-1)\frac{dN(x,\vec{p})}{dx} + x(x-1)\frac{d^2N(x,\vec{p})}{dx^2} \quad (274)$$

Deneme çözümünün x'e göre türevleri (271) sınır değer probleminde yerine koyulduktan sonra, hata fonksiyonu;

$$E(\vec{p}) = \frac{1}{2}\sum_i \left\{\frac{d^2\Psi_t(x_i)}{dx^2} - f\left(x_i, \Psi_t(x_i), \frac{d\Psi_t(x_i)}{dx}\right)\right\}^2 \quad (275)$$

seklinde hesaplanır (Lagaris vd 1998).

K tane birinci mertebeden adi diferansiyel denklem sistemi için,

$$\frac{d\Psi_i}{dx} = f_i(x, \Psi_1, \Psi_2, \dots, \Psi_K) \quad (276)$$

$x \in [0,1]$ ve $\Psi_i(0) = A_i,\ (i = 1, \dots, K)$ ile her bir $i = 1, \dots, K$ için Ψ_{t_i} , bir ileri beslemeli yapay sinir ağı içeren deneme çözümü belirtmektedir. İleri beslemeli yapay sinir ağlarının çıktıları ise, $N_i(x, \vec{p}_i), r = 1,2, \dots, K$ ile gösterilmektedir. Burada p_i her bir yapay sinir ağının ağırlık vektörünü, x ise girdiyi belirtmektedir. O halde, deneme çözümlerini şu şekilde ifade edebiliriz:

$$\Psi_{t_i}(x) = A_i + xN_i(x, \vec{p}_i) \quad \forall i = 1,2, \dots, K \quad (277)$$

Deneme çözümlerinin x'e göre türevleri;

$$\frac{d\Psi_{t_i}(x)}{dx} = N_i(x, \vec{p}_i) + x\frac{dN_i(x, \vec{p}_i)}{dx} \quad \forall i = 1,2, \dots, K \quad (278)$$

şeklindedir. Deneme çözümleri ve türevleri elde edildikten sonra hata fonksiyonu hesaplanabilir (Lagaris vd 1998):

$$E(\vec{p}) = \frac{1}{2}\sum_{k=1}^{K}\sum_i \left\{\frac{d\Psi_{t_k}(x_i)}{dx} - f_k\left(x_i, \Psi_{t_1}, \Psi_{t_2}, \dots, \Psi_{t_K}\right)\right\}^2 \quad (279)$$

Kısmi diferansiyel denklemlerin çözümü

Burada sadece iki boyutlu problemleri ele alıyoruz (Lagaris vd 1998). Ancak yöntemi daha fazla boyuta genişletmek kolaydır. Örneğin Poisson denklemini ele alalım:

$$\frac{\partial^2}{\partial x^2}\Psi(x,y) + \frac{\partial^2}{\partial y^2}\Psi(x,y) = f(x,y) \quad (280)$$

$x \in [0,1], y \in [0,1]$ ile Dirichlet sınır koşulu: $\Psi(0,y) = f_0(y), \Psi(1,y) = f_1(y)$ ve $\Psi(x,0) = g_0(x), \Psi(x,1) = g_1(x)$. Deneme çözümü şu şekilde yazılır:

$$\Psi_t(x,y) = A(x,y) + x(1-x)y(1-y)N(x,y,\vec{p}) \tag{281}$$

Burada $A(x,y)$ sınır koşulunu karşılayacak şekilde seçilir:

$$A(x,y) = (1-x)f_0(y) + xf_1(y) + (1-y)\{g_0(x) - [(1-x)g_0(0) + xg_0(1)]\} + y\{g_1(x) - [(1-x)g_1(0) + xg_1(1)]\} \tag{282}$$

Formun karışık sınır koşulları için: $\Psi(0,y) = f_0(y), \Psi(1,y) = f_1(y), \Psi(x,0) = g_0(x)$ ve $\frac{\partial}{\partial y}\Psi(x,1) = g_1(x)$ (yani sınırın bir kısmında Dirichlet ve diğer kısımlarında Neumann), deneme çözümü şu şekilde yazılır:

$$\Psi_t(x,y) = B(x,y) + x(1-x)y\left[N(x,y,\vec{p}) - N(x,1,\vec{p}) - \frac{\partial}{\partial y}N(x,1,\vec{p})\right] \tag{283}$$

ve $B(x,y)$ yine sınır koşullarını karşılayacak şekilde seçilir:

$$B(x,y) = (1-x)f_0(y) + xf_1(y) + g_0(x) - [(1-x)g_0(0) + xg_0(1)] + y\{g_1(x) - [(1-x)g_1(0) + xg_1(1)]\} \tag{284}$$

Deneme çözümünün ikinci teriminin Dirichlet sınır koşullarının uygulandığı sınırın bir kısmında yok olduğu ve sınıra normal gradyan bileşeninin Neumann sınır koşullarının uygulandığı sınırın bir kısmında yok olduğu için sınır koşullarını etkilemediğine dikkat edilmelidir.

Yukarıdaki tüm PDE problemlerinde minimize edilecek hata şu şekilde verilir:

$$E(\vec{p}) = \frac{1}{2}\sum_i \left\{\frac{\partial^2}{\partial x^2}\Psi(x_i,y_i) + \frac{\partial^2}{\partial y^2}\Psi(x_i,y_i) - f(x_i,y_i)\right\}^2 \tag{285}$$

Burada (x_i, y_i) $[0,1] \times [0,1]$ içindeki noktalardır (Lagaris vd 1998).

TARTIŞMA ve SONUÇ

Yapay sinir ağları, matematiksel olarak karmaşık bir gelişim sürecinden geçmiştir. İlk başta basit matematiksel modellemelerle başlamış ancak zaman içinde daha karmaşık yapılar ve öğrenme algoritmaları geliştirilmiştir. Yapay sinir ağlarının matematiksel gelişimi hakkında genel bir bakış:

Perceptron Modeli (1957): Yapay sinir ağlarının temelini atan ilk model, girişleri ağırlıklarla çarpar, toplar ve bir aktivasyon fonksiyonu ile çıktı üretir.

Çok Katmanlı Algılayıcı (1960-1970): Perceptron modelinin tek katmanlı olması ve sınırlı öğrenme yetenekleri nedeniyle araştırmacıları çok katmanlı yapılara yöneltti.

Geri Yayılım Algoritması (1980): Çok katmanlı yapılardaki öğrenme süreçleri için temel olan geri yayılım algoritması, matematiksel olarak daha karmaşık bir yapı sunar. Bu algoritma, hatayı geri yayarak ağırlıkları günceller ve sinir ağlarını eğitmek için kullanılır.

Radyal Tabanlı Fonksiyon Sinir ağı (1990): Yapay sinir ağlarına alternatif olarak, özellikle Radyal tabanlı sinir ağı modellerinin popülerleşmesi, matematiksel modellendirme açısından çeşitliliği artırmıştır.

Bu süreç içinde, matematiksel modellemelerin karmaşıklığı arttıkça, yapay sinir ağları daha geniş uygulama alanlarına ve daha iyi performansa sahip modellerin geliştirilmesine imkan tanımıştır.

Yapay sinir ağlarıyla ilgili dünya çapında çeşitli araştırmalar mevcuttur. Yapay sinir ağlarını eğitmek, uzun bir zaman süreci gerektirdiğinden dolayı, bu alandaki araştırmalar özellikle eğitim sürecinin etkinliği üzerinde odaklanmıştır. Bu bağlamda, daha verimli ve yeni öğrenme algoritmalarını keşfetmek amaçlanmaktadır. 1943 yılından itibaren günümüze kadar yaşanan gelişmelerde, yapay sinir ağlarının eğitimi konusu temel bir odak noktası olmuştur. Yapay sinir ağlarının evrimi, gelecekte özellikle robotların günlük hayatın kalitesini artırmada ne kadar önemli bir faktör olabileceğine işaret etmektedir (Efe ve Kaynak 2000).

Yapay sinir ağları, genellikle eğitim sırasında diferansiyel denklemlerin çözümü için kullanılan güçlü araçlardır. YSA, özellikle geri yayılım algoritması kullanılarak eğitildiğinde, diferansiyel denklemlerin çözümü konusunda etkili olabilirler. Geri yayılım algoritması, öğrenme süreci sırasında ağırlıkları güncellemek için türev hesaplamalarını içerir. Bu süreç, özellikle regresyon ve sınıflandırma problemlerinde diferansiyel denklemlerle modellemeye dayalı bir yaklaşımın temelini oluşturur.

Geri yayılım algoritması, yapay sinir ağlarının eğitiminde kullanılan bir tekniktir. Bu algoritma, ağın çıktılarının gerçek değerlerle olan farkını minimize eden bir gradyan iniş yöntemi kullanarak ağırlıkları günceller. Bu süreç, zincir kuralı gibi matematiksel konseptleri içerir. Zincir kuralı, bileşik fonksiyonların türevini hesaplamak için kullanılır. Yapay sinir ağları genellikle karmaşık fonksiyonlar olarak düşünülebilir, bu nedenle zincir kuralı, bu fonksiyonların türevlerini hesaplamak için kullanılır.

Bu bilgiler, yapay sinir ağlarının diferansiyel denklemleri çözme konusunda genel bir çerçeve sunar. Ancak bu alanda kullanılan yöntemler ve teknikler hala aktif bir araştırma konusudur ve spesifik uygulamalara bağlı olarak farklı yaklaşımlar benimsenebilir.

KAYNAKLAR

Aggarwal, C. C., 2020. Linear Algebra and Optimization for Machine Learning: A Textbook.

Anderson, D. A., Tannehill, J. C. and Pletcher, R. H., 1984. Computational Fluid Mechanics and Heat IPransfer, Hemisphere Publishing Corporation, New York.

Chakraverty, S., 2020. Mathematical methods in interdisciplinary sciences. John Wiley & Sons.

Chakraverty, S., Mall, S., 2020. Single layer chebyshev neural network model with regression-based weights for solving nonlinear ordinary differential equations. Evolutionary Intelligence, 1–8.

Chen, Y., Yu, H., Meng, X., Xie, X., Hou, M., Chevallier, J., 2021. Numerical solving of the generalized black-scholes differential equation using laguerre neural network. Digital Signal Processing, 112, 103 003.

Christie, I., Griffiths, D. F., Mitchell, A. R. and Sanz-Serna, J. M., 1981. Product approximation for non-linear problems in the finite element method, Journal of Numerical Analysis, 1 (253).

Chuo, L. O., Desoer, CA. and Kuh, E. S., 1987. Linear and Nonlinear Circuits, McGraw-Hill, New York.

Cybenko, G., 1989. Approximation by superposition of a sigmoidal function, Math. Control Signals Systems 2, 303.

Da Silva, I. N., Spatti, D. H., Flauzino, R. A., Liboni, L. H. B., dos Reis Alves, S. F., 2017. "Artificial neural networks," Cham: Springer International Publishing, 39.

Duchi, J., Hazan, E., Singer, Y., 2011. Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 12 (7).

Efe, Kaynak, O., 2000. Yapay Sinir Ağları ve Uygulamaları. Boğaziçi Üniversitesi.

Finlayson, B. A. and Striven, L. E., 1966. The method of weighted residuals-A review, Applied Mechanics Review, 19 (9), 735.

Finlayson, B. A., 1972. The Method of Weighted Residuals and Variational Principles, Academic Press, New York.

Fletcher, C. A. J., 1984. Computational Galerkin Methods, Springer-Verlag, New York.

Girosi, F. and Poggio, T., 1989. Networks for learning: A view from the theory of approximation of functions, In Proceedings of The Genoa Summer School on Neuml Networks and Their Applications, Prentice-Hall.

Hadian-Rasanan, A. H., Rahmati, D., Gorgin, S., Parand, K., 2020. A single layer fractional orthogonal neural network for solving various types of lane–emden equation. New Astronomy, 75, 101 307.

Haykin, S., 2010. Neural networks and learning machines, 3/E. Pearson Education India.

Hinton, G., Srivastava, N., Swersky, K., 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Cited on, 14 (8), 2.

Hopfield, J. J., 1982. Proc. Nat. Acad. Sci. 79, 2554.

IMSL User's Manual, 1991. MATH/LIBRARY. Version 2.

Ito, Y., 1991. Approximation of functions on a compact set by finite sums of a sigmoid function without scaling, Neuml Networks 4, 817.

Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization. ArXiv preprint arXiv:1412.6980.

Koch, C., Marroquın, J. and Yuılle, A., 1986. Proc. Nat. Acad. Sci. 83, 4263.

Kumar, M., Yadav, N., 2011. Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: A survey. Computers and Mathematics with Applications, 62 (10), 3796–3811.

Lagaris, I. E., Likas, A., Fotiadis, D. I., 1998. Artificial neural networks for solving ordinary and partial differential equations. IEEE Transactions on Neural Networks, 9 (5), 987-1000.

Lax, P. and Wendroff, B., 1960. Systems of conservation laws, Comm. Pure and Applied Mathematics 13, 217.

Lee, H., 1989. Technical Digest of the Topical Meeting on Optical Computing, Salt lake city, Utah, Feb. 27-Mar. 1**.**

Lee, H., Kang, I. S., 1990. Neural algorithm for solving differential equations. Journal of Computational Physics, 91 (1), 110-131.

Mall, S., Chakraverty, S., 2014. Chebyshev neural network based model for solving lane–emden type equations. Applied Mathematics and Computation, 247, 100–114.

Mall, S., Chakraverty, S., 2015. Numerical solution of nonlinear singular initial value problems of Emden–Fowler type using chebyshev neural network method. Neurocomputing, 149, 975–982.

Mall, S., Chakraverty, S., 2016. Application of Legendre neural network for solving ordinary differential equations. Applied Soft Computing, 43, 347–356.

Maren, A., Harston, C. and Pap, R., 1990. Handbook of Neural Computing Applications, Academic Press, New York.

Meade Jr, A. J., 1989. Semi-discrete Galerkin modelling of compressible viscous flow past a circular cone at incidence, Ph.D. Thesis, University of California, Berkeley.

Meade Jr, A. J., Fernandez, A. A., 1994. The numerical solution of linear ordinary differential equations by feedforward neural networks. *Mathematical and Computer Modelling*, 19 (12), 1-25.

Meade Jr, A. J., Fernandez, A. A., 1994. Solution of nonlinear ordinary differential equations by feedforward neural networks. *Mathematical and Computer Modelling*, 20 (9), 19-44.

Moore, F. K., 1951. Three-dimensional compressible laminar boundary-layer flow, NACA TN 2279.

Nesterov, Y., 1983. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). Doklady an ussr, 269, 543– 547.

Omohundro, S., 1987. Efficient algorithms with neural network behaviour, Complex Systems 1, 237.

Öztemel, E., 2003. Yapay Sinir Ağları. İstanbul: Papatya, 15-18.

Pao, Y. H., Phillips, S. M., 1995. The functional link net and learning optimal control. Neurocomputing, 9 (2), 149–164.

Prenter, P. M., 1989. Splines and Variational Methods, Wiley, New York.

Psaltlss, D. and Farhat, N., 1985. Opt. Lett. 10, 98.

Reshotko, E., 1957. Laminar boundary layer with heat transfer on a cone at angle of attack in a supersonic stream, NACA TN 4152.

Ruder, S., 2016. An overview of gradient descent optimization algorithms. ArXiv preprint arXiv: 1609.04747.

Stewartson, K., 1954. Further solutions of the Falkner-Skan equation, Proc. Cambr. Phil. Sot. Journal of Numerical Analysis, 50 (454).

Strang, G., 1980. Linear Algebra and Its Applications, Second Edition, Academic Press, New York.

Trim, D., 1990. Applied Partial Diflerential Equations, PWS-KENT, Boston.

Yang, Y., Hou, M., Luo, J., 2018. A novel improved extreme learning machine algorithm in solving ordinary differential equations by legendre neural network methods. Advances in Difference Equations, 2018 (1), 1–24.

Yazdi, H. S., Pakdaman, M., Modaghegh, H., 2011. Unsupervised kernel least mean square algorithm for solving ordinary differential equations. Neurocomputing, 74 (12-13), 2062–2071.

Zeiler, M. D., 2012. Adadelta: An adaptive learning rate method. ArXiv preprint arXiv:1212.5701.

White, F. M., 1974. Viscous Fluid Flow, McGraw-Hill. New York.

ÖZGEÇMİŞ

Kişisel Bilgiler **Aslıhan YILDIRIM**
Eğitim
Lise : Erzurum İmkb Anadolu Lisesi Lisans : Atatürk Üniversitesi- Fen Fakültesi- Matematik Bölümü
Yabancı Dil Bilgisi
İngilizce
Üye Olunan Mesleki Kuruluşlar
Tezden Üretilen Yayınlar