Öğrenen ve Öngören Varlık Yönetimi

Learning and Predicting Asset Management

Kağan KÜÇÜK
Yapay Zekâ Yazılım Geliştirme Bölümü
32Bit Bilgisayar Hizmetleri
İstanbul, Türkiye
kagan.kucuk@32bit.com.tr
ORCID: 0000-0003-4286-1848

Fatih KAHRAMAN
Yapay Zekâ Yazılım Geliştirme Bölümü
32Bit Bilgisayar Hizmetleri
İstanbul, Türkiye
fatih.kahraman@32bit.com.tr
ORCID: 0000-0001-5495-4991

Mustafa Ersel KAMAŞAK
Bilgisayar Mühendisliği Bölümü
İstanbul Teknik Üniversitesi
İstanbul, Türkiye
kamasak@itu.edu.tr
ORCID: 0000-0002-5050-3357

Eşref ADALI
Bilgisayar Mühendisliği Bölümü
İstanbul Teknik Üniversitesi
İstanbul, Türkiye
adali@itu.edu.tr
ORCID: 0000-0002-1561-8255

Öz

Müşterilere yapılan anlık kur teklifleri, bankacılık sektöründeki en kritik konular arasında yer almaktadır. Verilen tekliflerin uygun seviyede olması hem banka hem de müşteri açısından oldukça önemlidir. Bu çalışmada, müşteriye verilen kur tekliflerinin tahmini için yaklaşık 8 aylık veri kullanılmış ve tahmin modelleri tasarlanmıştır. Toplamda 18 farklı kur üzerinden çalışma yürütülmüştür. Çalışmada bağımlı değişkenler müşteri segmenti, anlık kur değeri, gün bilgisi, saat bilgisi ve volatilite değeri olarak belirlenmiştir. Bağımsız değişken ise kur marjıdır. Eğitimler günlük verilerle ve Rastgele Ağaçlar (RF), Gradyan Arttırma Makinesi (GBM), Yapay Sinir Ağları (ANN), Derin Sinir Ağları (DNN), Evrişimli Sinir Ağları (CNN) ve Elman Sinir Ağı algoritmaları kullanılarak gerçekleştirilmiştir. Algoritmaların hiper-parametrelerini bulmak için rastgele arama algoritması kullanılmış ve model eğitimlerinin sonuçları karşılaştırılarak en düşük hata değerine sahip modeller tahmin aşamasında kullanılmak üzere seçilmiştir. Başarım ölçümü için Ortalama Kare Hata (MSE) ve Ortalama Mutlak Hata (MAE) hata fonksiyonları kullanılmıştır. Üç farklı veri modeli ve altı farklı yapay zekâ algoritması üzerinden gerçekleştirilen eğitimlere göre yapay sinir ağları ve evrişimli sinir ağları algoritmalarının diğer algoritmalara göre daha iyi sonuçlar verdiği gözlemlenmiştir. 18 kur için tahmin süresi yaklaşık 3s'dir.

Gönderme ve kabul tarihi: 03.11.2021 – 29.11.2021
Makale türü: Araştırma

Anahtar Sözcükler — *Marj Tahmini, Yapay Sinir Ağları, Tahmin Modelleri, Derin Sinir Ağı, Evrişimli Sinir Ağı, Elman Sinir Ağı*

Abstract

Instant exchange rate offers to customers are the most critical issues in the banking industry. It is very important for both the bank and the customer that the offers given are at the appropriate level. In this study, approximately 8 months of data were used and estimation models were designed for the estimation of the currency offers given to the customer. In total, the study was conducted over 18 different currencies. In the study, dependent variables were determined as customer segment, instant exchange rate, day information, time information and volatility value. The independent variable is the exchange rate margin. The training was carried out with daily data and using Random Forest (RF), Gradiant Boosting Machine (GBM), Artificial Neural Network (ANN), (Deep Neural Network)DNN, (Convolutional Neural Network)CNN and Elman Neural Network algorithms. Random search algorithm was used to find the hyperparameters of the algorithms and the results of the model training were compared and the

models with the lowest error values were selected to be used in the estimation phase. Mean Square Error (MSE) and Mean Absolute Error (MAE) error functions were used to measure performance. It has been observed that artificial neural networks and convolutional neural network algorithms give better results than other algorithms according to the trainings carried out on three different data model and six different artificial intelligence algorithms. Estimation time for 18 currencies is about 3s.

Keywords *— Margin Estimation, Artificial Neural Networks, Estimation Models, Deep Neural Network, Convolutional Neural Network, Elman Neural Network*

1. Giriş

Bankaların kurlar için müşterilerine sunduğu fiyat alım-satım oranları ve bunlar arasındaki denge gerek banka gerekse müşteriler açısından günümüz dünyasında büyük önem arz etmektedir. Bu durum, kur fiyatlarının alım ve satım yönündeki değerlerinin tahminine yönelik, farklı yöntemler kullanılmasına sebep olmuştur. Son zamanlardaki makine öğrenme ve yapay zekâ teknolojisindeki hızlı gelişim, bu tarz problemlerin çözümlenebilmesini hızlandırmıştır.

Marjin, finansal ürünlerin alış ile satış fiyatı arasındaki farkı ifade eden bir terimdir. Başka bir deyişle finans piyasalarında karşılaştığımız dolar, euro, sterlin vb. kurların alış fiyatı ile satış fiyatı arasındaki fiyat farkı marjin olarak ifade edilmektedir. Marjin hesaplaması, piyasalarda dikkat edilmesi gereken çok önemli bir konudur. Döviz Büroları ve Bankalar, merkez bankasının günlük olarak açıkladığı döviz kurlarına, serbest piyasada bulunan kurları da öngörerek kendi alım satım fiyatlarını belirlerler. Vereceği fiyatlama konusunda her kurum özgürdür.

Müşterilere önerilen kur fiyat tekliflerine, volatiliteli bir piyasada anlık olarak analist ve uzmanlar tarafından karar verilmesi zor ve yorucu bir işlemdir. Çalışmanın motivasyonu, bu durumu makine ve yapay zekâ algoritmaları ile gerçekleştirmek ve volatilitenin etkisini en aza indirmektir.

Bu çalışmada 8 aylık günlük banka verileri kullanılmış olup 18 adet kur üzerinde eğitimler gerçekleştirilmiştir. Kur alım ve satım fiyatlarının tahmini için 6 farklı makine öğrenme ve yapay zekâ algoritması kullanılmıştır. Bu algoritmaların en iyi hiper-parametrelerini tespit edebilmek için, rastgele orman arama optimizasyon algoritması kullanılmıştır. Her kur için ayrı ayrı eğitimler yapılmış ve modeller oluşturulmuştur.

Bu çalışmanın, bankacılık sistemlerindeki müşterilere verilecek olan kurun alış ve satış fiyatlarının, volatilite, yani piyasa oynaklığı değişimi karşısında otomatik olarak tahmin yapabilmesi yönünden bilimsel yazında önemi büyüktür.

Çalışmanın ilerleyen bölümlerinde kullanılan veri setleri, ön işleme operasyonları ve kur tahminleme algoritmaları hakkında bilgiler ve bunların sonuçları sunulmuştur.

2. Terminoloji

2.1 Veri Seti

Çalışma, 8 ay boyunca saniyelik olarak gelen günlük banka verileri kullanılarak gerçekleştirilmiştir. Verilerin anlık olarak kayıt dosyaları üzerinden çekilmesi ve saklanabilmesi için bir ELK(Elasticsearch-Kibana) sistemi oluşturulmuştur.

Çalışmada kullanılan verilerin öznitelikleri,

- Müşteri Segmenti,
- Anlık Kur Değeri,
- Gün Bilgisi,
- Saat Bilgisi,
- Volatilite Değeri

olarak belirlenmiştir. Kurlar üzerindeki oynaklık bilgisi kur fiyat değişimi açısından önemli bir etkendir. Bu etken yapay zekâ modellerinde, girdi verisine bir öznitelik olarak aktarıldığı takdirde kur için verilen fiyat teklifinde oynaklığın etkisi minimuma indirilebilir.

Volatilite riski, çeyrek asırdan fazla bir süredir yaşanmış birçok finansal felaketin (örneğin; Barings Bank, Long-Term-Capital Management) meydana gelmesinde başlıca rol oynamıştır. Volatilite riski kısaca finansal piyasalarda bulunan varlıkların fiyat ve değerlerinde meydana gelen dalgalanmalardan kaynaklanan volatilitenin belirsiz olması olarak tanımlanabilmektedir [1]. Söz konusu risk yönetiminin finansal sektörde bu denli önemli yer tutmasının nedenlerinin başında piyasalardaki volatilitenin 2000'li yıllardan sonra hızlı artış göstermesi gelmektedir [2].

Genellikle bir finansal güvenlik veya piyasa endeksinden elde edilen getirilerin standart sapması veya varyansı ile ölçülen oynaklık, varlık tahsisi, risk

yönetimi ve türev fiyatlandırmanın önemli bir bileşenidir[19].

Volatilite özniteliğinin çıkarımı aşamasında, kurun t anındaki yüzdesel değişimi bulunmaktadır. Yüzdesel değişimin standart sapması ise bize volatiliteyi vermektedir. Herhangi bir t anındaki volatilite değerini bulmak için o andan itibaren 15 dk. geriye gidilerek veriler alınır ve volatilite hesaplamaları gerçekleştirilir. Kurdaki her veri için volatilite hesabı yapılmaktadır ve çıkan değer ilgili veriye öznitelik olarak eklenmektedir. Bunun yanı sıra volatilite tahmini makine öğrenmesi algoritmaları aracılığı ile de gerçekleştirilebilmektedir.

Volatilite özniteliğine sahip olan ve olmayan veriler ile ilgili makine öğrenme ve yapay zekâ algoritmaların eğitimi gerçekleştirilmektedir ve bu eğitimlerin sonuçları karşılaştırılarak volatilite özniteliğinin sonuçlara etkisi bulunmaktadır. Veriler makine öğrenmesi ve yapay zekâ algoritmalarına girdi olarak verilmeden önce belirli veri ön-işleme algoritmalarından geçirilmiştir.

Bu algoritmalar ile

- Eksik verilerin giderilmesi,
- Metinsel verilerin sayısallaştırılması,
- Verilerin normalize edilmesi

adımları gerçekleşmektedir.

Eksik verilerin giderilmesi her kur için ayrı ayrı yapılmaktadır. Bu aşamada eksik verilerin giderilmesi için,

- Medyan değeri,
- Ortalama değer,
- Maksimum değer,
- Minimum değer,
- Sabit değer

eksik veri giderme yöntemleri hazırlanmıştır. Veri bilimi alanında yapılan birçok çalışmada bu yöntemler tercih edilmektedir. Bu yöntemlerin dezavantajı ise ilgili değişkenin varyans değerini düşürmesidir. Bu yöntemlerden ön-işleme için medyan değeri ile doldurma algoritması kullanılmıştır.

Veri sayısallaştırması aşaması, makine öğrenme ve yapay zekâ algoritmalarının anlayabileceği verileri elde etmek için metinsel olan verilerin sayısal olan verilere dönüştürülmesi işlemini kapsamaktadır. Müşteri Segmenti maksimum 20 segment olacak şekilde seçilmiştir. Segment1 ve Segment20 aralığındaki bütün veriler 1-20 aralığındaki sayısal verilere dönüştürülmüştür. Gün bilgisinde ise Pazartesi – Pazar aralığındaki bütün veriler 0-6 aralığındaki sayısal verilere dönüştürülmüştür.

Projedeki veri normalizasyonu aşamasındaki temel amaç verilerin 0-1 aralığında olmasını sağlamaktır. Bu sayede gerek eğitim başarılarının daha iyi sonuçlanması gerekse eğitim zamanının kısaltılması hedeflenmektedir. Bu aşamada minimum ve maksimum normalizasyonu kullanılmıştır. Veri özniteliği olarak seçilen saat özniteliğindeki bütün değerler 24'e bölünerek veriler 0-1 aralığına indirgenmiştir. Gün verisindeki bütün değerler 6'ya bölünerek veriler 0-1 aralığına indirgenmiştir. Segment verisindeki bütün değerler, 20'e bölünerek veriler 0-1 aralığına indirgenmiştir. Her kur verisinin sayısal olarak kendine özgü bir aralığı bulunmaktadır. Kur verilerinin ilgili veri bloğu içerisindeki maksimum değerine bölünmesi ile veriler 0-1 aralığına normalize edilmektedir.

3. Uygulamalar

Çalışmada veri özniteliklerinin başarım üzerindeki etkisini test edebilmek için 3 farklı uygulama hazırlanmıştır. 3 farklı uygulama 6 farklı makine öğrenme ve yapay zekâ algoritması ile eğitilmiştir.

1. Uygulamanın Öznitelikleri:

- Müşteri Segmenti
- Anlık Kur Değeri
- Saat Bilgisi
- Çıktı:
- Marj Değeri

2. Uygulamanın Öznitelikleri:

- Müşteri Segmenti
- Anlık Kur Değeri
- Gün Bilgisi
- Saat Bilgisi
- Çıktı:
- Marj Değeri

3. Uygulamanın Öznitelikleri:

- Müşteri Segmenti
- Anlık Kur Değeri
- Gün Bilgisi
- Saat Bilgisi

- Volatilite Bilgisi
- Çıktı:
- Marj Değeri

Buradaki temel amaç, öznitelikleri birbirinden farklı olarak eğitilmiş modellerin validasyon verileri ile test edilip doğruluk oranlarını karşılaştırmaktır.

4. Makine Öğrenmesi ve Yapay Zekâ Algoritmaları

Tahmin problemlerinde güncel olarak kullanılan 6 farklı yapay zekâ algoritması seçilmiştir.

Bu algoritmalar:

- Rastgele Orman
- Gradyan Arttırma
- Yapay Sinir Ağları
- Derin ve Evrişimli Sinir Ağları
- Elman Sinir Ağları

4.1 Rastgele Orman Algoritması

Marj değerinin belirlenmesinde Rastgele Orman Regresyonu olarak bilinen karar ağacı tabanlı bir regresyon aracı ele alınmıştır. Rastgele orman, rastgele seçilen basit karar ağaçlarının bir koleksiyonu veya topluluğu olarak düşünülebilir. Tahmini bir öngörü fonksiyonundaki varyansı azaltmayı amaçlayan sözde önyükleme toplama veya torbalama tekniği sınıfına aittir[19]. Breiman'ın (2001) geliştirdiği karar ağaçlarının bir kombinasyonu olan rastgele orman algoritması, torbalama yönteminin geliştirilmiş bir versiyonudur. Rastgele orman, tüm değişkenler arasından en iyi dalı kullanarak her bir düğümü dallara ayırmak yerine, her bir düğümde rastgele olarak seçilen değişkenler arasından en iyisini kullanarak her bir düğümü dallara ayırır. Her bir veri seti, orijinal veri setinden yer değiştirmeli olarak üretilir. Daha sonra rastgele olacak şekilde özellik seçimi kullanılarak ağaçlar geliştirilir. Geliştirilen ağaçlar budanmaz [3]. Hızlı, istenilen sayı kadar regresyon ağacı geliştiren ve çalıştıran rastgele orman, bir veriyi tahmin etmek için, girdi verisini ormandaki her ağaca yerleştirir. Her bir veri seti orijinal veri setinden yeniden örnekleme kullanılarak üretilir. Şekil-1'de Rastgele Orman Algoritmasının yapısı gösterilmiştir. Şekil-1'de yapısal olarak gösterilen Rastgele Orman, çoklu karar ağaçlarının kullanımıyla hem regresyon hem de sınıflandırma görevlerini gerçekleştirebilen bir topluluk tekniği ve yaygın olarak torbalama olarak bilinen Ön-yükleme ve Toplama adlı bir tekniktir. Bunun arkasındaki temel fikir, tek tek karar ağaçlarına güvenmek yerine, nihai çıktının belirlenmesinde çoklu karar ağaçlarının birleştirilmesidir. Rastgele orman, temel öğrenme modelleri olarak birden fazla karar ağacına sahiptir. Her model için örnek veri kümelerini oluşturan veri kümesinden rastgele satır örnekleme ve özellik örnekleme gerçekleştiririz. Bu bölüme Bootstrap denir.

Sonuç olarak rastgele orman regresyonu, birden fazla karar ağacını oluşturmaktadır ve daha doğru bir tahmin elde etmek için oluşturduğu ağaçları birleştirmektedir. Bu nedenle, rastgele orman regresyonunun doğru olarak uygulanabilmesi, Karar Ağaçları algoritmasının mantığının çözülmesi ile mümkün olabilmektedir.

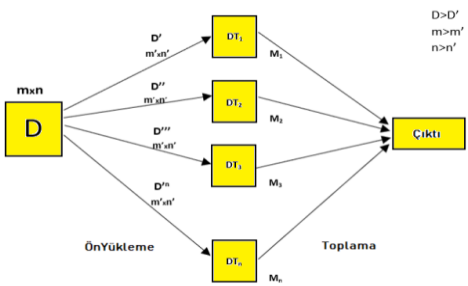
Rastgele Orman algoritması için arama optimizasyon algoritmasında kullanılan hiper-parametreler,

- Maksimum Derinlik: 5, 10, 15, 20, 50, 100
- Maksimum Özellik: n_features, sqrt (n_features)
- Maksimum Yaprak Düğümü: 2, 5, 10, 20
- Minimum Yaprak Sayısı: 2, 5, 10
- Karar Ağacı Sayısı: 200, 400, 600, 800, 1000, 1200

4.2 Gradyan Arttırma Algoritması

Gradyan arttırma algoritması, birbirinden farklı uygulamalarda yüksek başarılar gösterebilen önemli makine öğrenmesi tekniklerindendir. Farklı kayıp fonksiyonları ile öğrenebilme kapasitesi olduğu gibi, uygulamaların istenen ihtiyaçlarına göre şekillendirilebilir. Gradyan artırma algoritması, tahmin modelleri oluşturmak için en güçlü tekniklerden biridir.

Gradyan artırma algoritması, genellikle karar ağaçları



Şekil 1: Rastgele Orman Algoritması Ön Yükleme ve Toplama (Bootstrap, Aggregation)

olmak üzere zayıf tahmin modelleri topluluğu şeklinde bir tahmin modeli üreten bir makine öğrenimi tekniğidir. Güçlendirme kavramı, Y sonucunu (zayıf öğrenen) tahmin etmede güçlü olmayan bir öze lliği güçlü bir öğreniciye dönüştürmektir. Her bir regresyon ağacındaki zayıf öğrenicileri belirledikten sonra, güçlendirme, zayıf öğrenici tarafından tahmin edilen uygun değerler ile gerçek değerler arasındaki hatalara odaklanırken her gözleme eşit ağırlık verecektir. Gradyan artırma ile bu model, hata fonksiyonunun gradyanını hesaplamak için hata oranını kullanır ve her yinelemede hatayı azaltmak için model parametrelerinin nasıl ayarlanacağını belirlemek için gradyanı benimser. Bu işlem, veri bilimcilerin belirttiği şekilde m kez gerçekleştirilecektir [20]. Rastgele Orman algoritması, bağımsız karar ağaçları oluşturabilir ve oluşturduğu ağaçlardan ürettiği tahminlerin ortalamasını nihai tahmin olarak sonuçlandırır. Öte yandan, gradyan arttırma algoritması ağaçları üretir daha sonra oluşturduğu ağaçları sıralı bir şekilde topluluğa ekler. İlk üretilen ağaç, rastgele orman algoritmasındaki gibi üretilir. Temel fark ilk ağacın çıkardığı tahmin hatalarını en aza indirmek için ikinci bir ağacın üretilmesidir. Burada hem birinci hem de ikinci ağaç topluluğa eklenir ve her birine farklı ağırlıklar verilir. Bu süreç birden çok kez tekrarlanır. Her adımda, tüm topluluk üzerinden öğrenilen hatalara göre yeni bir ağaç eğitilir ve daha sonra bu ağaç topluluğa eklenir. Son topluluk, yeni girdilerin sonucunu kestirmek için bir model olarak kullanılır. Gradyan artırma makinelerinde veya basitçe GBM, öğrenme prosedürü, yanıt değişkeninin daha doğru bir tahminini sağlamak için art arda yeni modellere uygulamaktır [4].

Rastgele orman algoritmasının aksine, gradyan arttırma algoritmasında öğrenici sadece regresyon ağaçları değil, sinir ağları veya doğrusal regresyon gibi diğer regresyon öğrenme algoritmaları da olabilir.

Gradyan Arttırma algoritması için arama optimizasyon algoritmasında kullanılan hiper-parametreler,

- Maksimum Derinlik: 5, 10, 15, 20, 50, 100, 120, 150
- Maksimum Özellik: n_features, sqrt(n_features)
- Maksimum Yaprak Düğümü: 2, 5, 10, 20, 25, 30
- Minimum Yaprak Sayısı: 2, 5, 7, 10, 15
- Karar Ağacı Sayısı: 200, 400, 600, 800, 1000, 1200, 1500, 5000

4.3 Yapay Sinir Ağları

Son zamanlarda, ekonomik-finans endüstrisinde, Yapay Sinir Ağları veya basitçe sinir ağları olarak adlandırılan, insan beynini yeniden üretme eğiliminde olan bir yapıya sahip olan ve doğrusal olmayan model sınıfına geniş bir ilgi olmuştur[23].

Yapay sinir ağları; insan beyninden esinlenerek, öğrenme sürecinin matematiksel olarak modellenmesi uğraşı sonucu ortaya çıkmıştır. Bu nedenle konu üzerindeki çalışmalar ilk olarak beyni oluşturan biyolojik üniteler olan nöronların modellenmesi ve bilgisayar sistemlerinde uygulanması ile başlamış, daha sonraları bilgisayar sistemlerinin gelişimine de paralel olarak birçok alanda kullanılır hale gelmiştir.

Yapay sinir ağları genellikle aşağıdan yukarıya doğru nöronlar, katmanlar ve mimari olmak üzere üç düzeyde bileşenden oluşur. Mimari, çok sayıda yapay nörondan oluşan farklı katmanların bir kombinasyonu ile belirlenir. Öğrenilebilir ağırlıklar ve önyargılar içeren bir nöron, yapay sinir ağlarının temel birimidir. Bitişik katmanların nöronlarını bağlayarak, önceki katmanın çıkış sinyalleri, giriş sinyalleri olarak bir sonraki katmana girer. Katmanları üst üste istifleyerek, sinyaller giriş katmanından gizli katmanlar aracılığıyla potansiyel olarak döngüsel veya tekrarlayan bağlantılar yoluyla çıkış katmanına gider ve YSA, girdi-çıktı çiftleri arasında bir eşleme oluşturur[18].

İnsan beyninin çalışma prensibini taklit eden bu sistemler, bilgisayar teknolojisi gelişmekte olmasına rağmen, insan beyni bir yana, ilkel canlı bir beynin işlevleri dikkate alınsa bile, o organizmaya göre hala çok ilkeldir. Hızla ve işlem hızı nanosaniyeler mertebesinde azaltılır. Nanosaniye cinsinden işlem hızları ile RNA'lar (Ribonucleic Acid), milisaniye aralığındaki işlem hızlarıyla işleyen insan beyninin işlevselliğinden hala uzaktır.

4.3.1 Avantajları

- Eğitim sırasında kullanılmayan girdiler için iyi sonuçlar çıkarabilmektedir.
- Bilgi işleme yöntemleri geleneksel programlanın üstündedir.
- Girdi katmanında diğer katmanlara aktarılan bilgiler bütün ağlara dağılmaktadır. Ağ içerisindeki sinir hücrelerinden kayıp veya

kayıplar anlamlı bilginin kaybolmasına neden olmaz.

- Algılamaya yönelik uygulamalarda başarılıdır.
- Regresyon, sınıflandırma, tahmin ve örüntü anlamlandırma problemlerinde oldukça başarılıdır.
- Eksik bilgiler ile çalışabilme özelliği bulunmaktadır.

4.3.2 Dezavantajları

- Donanıma bağımlı yapılardır.
- Ağın eğitiminin bitirilmesi için birçok yöntem kullanılmakta fakat kesin bir yöntem bulunmamaktadır.
- Ağın davranışı açıklanamamaktadır.

Çalışmada tahmin işlemleri için Çok Katmanlı Algılayıcı (MLP) yapay sinir ağı modeli kullanılmıştır. Yapay sinir ağı modelleri yapay sinir hücrelerinin birbirlerine olan bağlantısıyla gerçekleştirilen yapılardır. Yapay sinir ağlarında 3 katman bulunmaktadır:

Giriş Katmanı, Ara Katman ve Çıkış Katmanı.

Katmanlar içerisinde kullanılan matematiksel fonksiyonların çeşitliliği sebebiyle farklı sinir ağı modelleri literatürde yer almaktadır. Şekil-1'de 3 adet gizli katmana sahip birçok katmanlı bir yapay sinir ağı yapısı görselleştirilmiştir. Günümüzde en çok bilinen ve yaygın biçimde kullanılan yapay sinir ağı modellerinden bir tanesi çok katmanlı algılayıcılardır [5].

Dışarıdan gelen girdiler nöronlara gelen verilerdir. Girdiler yapay sinir hücrelerine dışarıdan gelebileceği gibi başka hücrelerden de gelebilir. Bu girdilerden gelen veriler toplanmak üzere nöron çekirdeğine gönderilir. Yapay sinir hücresine gelen bilgiler girdiler üzerinden çekirdeğe ulaşmadan önce geldikleri bağlantıların ağırlığıyla çarpılarak çekirdeğe iletilir. Böylece girdilerin diğer nöronlardaki çıktılara etkisi ayarlanabilmektedir. Hücrelerde arasındaki bu ağırlık değerleri pozitif, negatif veya sıfır değerini alabilir. Ağırlığın sıfır olması hücrenin diğer sinir hücrelerine hiçbir etki yapamayacağı anlamına gelmektedir.

Toplama fonksiyonu diğer hücrelerden gelen ağırlık ile çarpılmış hücre girdilerinin toplanması işlevini sağlamaktadır.

Aktivasyon fonksiyonu olarak çoğunlukla doğrusal olmayan fonksiyonlar seçilmektedir. Yapay sinir ağlarındaki doğrusal olmama durumu doğrusal olmayan aktivasyon fonksiyonları sayesinde gerçekleşmektedir. Aktivasyon fonksiyonu seçilirken dikkat edilmesi gereken bir diğer nokta da fonksiyonun türevinin kolaylıkla hesaplanabilmesidir. Aktivasyon fonksiyonunun türevi geri beslemeli ağlarda da kullanıldığından hesaplamanın yavaşlamaması için türevi hesaplaması kolay bir fonksiyon seçilmelidir. Günümüzde daha yaygın olarak kullanılan "çok katmanlı algılayıcı" modelinde aktivasyon fonksiyonu olarak "sigmoid fonksiyonu" kullanılmaktadır. Tetikleme işlevinden gelen değer, hücrenin çıkış değeridir. Bu değer yapay sinir ağından çıktı olarak dış dünyaya iletilir veya ağ içinde tekrar kullanılabilir. Her hücrenin birden fazla girişi olmasına rağmen, yalnızca bir çıkışı vardır. Bu çıktı herhangi bir sayıda hücreye bağlanabilir. Ağı eğitmek için örnekler toplandığı gibi, ağı test etmek için de örneklerin toplanması gerekir. Eğitim seti örnekleri, ağın olay hakkında bilgi edinebilmesi için sırayla görüntülenir. Ağ olayı öğrendiğinde, test kitindeki örnekler gösterilerek ağın performansı ölçülür [11].

Daha önce hiç görmediğiniz örneklerle elde ettiğiniz başarı, ağın iyi öğrenip öğrenmediğini gösterir. Öğrenilecek olay için oluşturulacak ağın topolojik yapısı belirlenir. Sinir ağlarının eğitimi, yaygın olarak "hiper parametreler" olarak adlandırılan çok sayıda seçenek içerir. Bunlar, katman sayısı, nöronlar ve spesifik aktivasyon fonksiyonu vb. parametrelerdir. İlk olarak, yapay sinir ağlarının derinliğini (gizli katman sayısı) ve genişliğini (nöron sayısı) belirlemek zorlu bir problemdir[18]. Bu adım, her bir ara katmanda kaç tane girdi birimi, kaç tane ara katman, kaç tane işlem elemanı ve kaç tane çıkış elemanı olması gerektiğini tanımlar. Bu adımda ağ öğrenme katsayısı, işlem elemanlarının toplama ve aktivasyon fonksiyonları ve momentum katsayısı gibi parametreler belirlenir. Proses elemanlarını birbirine bağlayan ağırlık değerlerinin başlangıç değerleri ile birim ağırlıkların eşik değeri atanır. İlk olarak rastgele değerler atanır. Ağ daha sonra öğrendikçe kendisine uygun değerleri belirler. Ağın öğrenmeye başlaması ve öğrenme kuralına göre ağırlıkları değiştirmesi için belirli bir mekanizmaya göre örnekler ağa gösterilir. Görüntülenen giriş için ağ çıkış değerleri hesaplanır. Bu adımda ağın oluşturduğu hata değerleri hesaplanır. Geriye dönük hesaplama yöntemi kullanılırken oluşan hatayı azaltmak için ağırlıklar değiştirilir.

İleri beslemeli sinir ağı öğrenmeyi tamamlayana

kadar devam eder, yani gerçekleşen ve beklenen iterasyonlar arasındaki hataları kabul edilebilir seviyelere düşer. Momentum katsayısı, geri beslemeli sinir ağının yerel sonuçlarda çıkmaza girmesini önlemek için geliştirildi. Eğitim sürecindeki bir diğer önemli sorun da öğrenme süresinin çok uzun olmasıdır. Ağırlık değerleri ilk başta büyükse, ağın yerel sonuçlara düşmesine ve bir yerel sonuçtan diğerine atlamasına neden olacaktır. Ağırlıklar küçük bir aralıkta seçilirse ağırlıkların doğru değerleri bulması uzun zaman alacaktır.

İleri beslemeli bir sinir ağında, bağlantı ağırlıklarının ilk tespiti ağ performansı ile yakından ilgilidir. Genel olarak ağırlıklar belirli aralıklara atanır. Bu aralık büyük tutulursa yerel çözümler arasında sürekli dolaştığı, küçükse öğrenmenin geç gerçekleştiği tespit edilmiştir. Bu değerleri atamak için henüz standart bir yöntem yoktur. Deneyimler, 0.1 ile 1.0 arasındaki değerlerin tatmin edici sonuçlara yol açtığını göstermektedir. Ama hepsi öğrenilmekte olan problemin türüne bağlıdır. Başlangıç değerlerinin yanı sıra öğrenme ve momentum katsayılarının belirlenmesi ağın öğrenme performansı ile yakından ilgilidir. Öğrenme katsayısı, ağırlıklardaki değişim miktarını belirler. Büyük değerler seçilirse, ağ yerel çözümler arasında dolaşabilir ve salınım yapabilir. Küçük değerlerin seçilmesi öğrenme süresini uzatır. Deneyimler, 0.20. arasındaki değerlerin normal olarak kullanıldığını göstermiştir. Bazı uygulamalar, öğrenme katsayısı 0.6 olduğunda en tatmin edici sonuçların elde edildiğini göstermiştir [10].

Benzer şekilde, momentum katsayısı da öğrenme performansını etkiler. Momentum katsayısı, önceki yinelemedeki değişikliğin yeni değişiklik miktarına belirli bir oranının toplamıdır. Bu, özellikle yerel çözümlere bağlı ağların atlama başına daha iyi sonuçlar elde etmesine izin vermek amacıyla önerilmiştir. Bu değerler çok küçük seçilirse global çözümlerin çıkarılmasını zorlaştırabilir. Çok büyük değerler çözüme ulaşmada sorunlara neden olabilir. Tecrübeler momentum katsayısının 0.6 ve 08 arasında seçilmesinin uygun olacağını göstermiştir [10]. Sorunun niteliğine bağlı olarak, kullanıcının farklı değerler kullanması faydalıdır. Sinir ağlarının çalışması, ağdan elde edilen sonuç değeri ile istenen sonuç değeri arasındaki hata değerinin en aza indirilmesi esasına dayanır. Elde edilen hata değerleri eğitim aşamasında geri beslenir. Bu sayede elde edilen hata değerini en aza indirecek şekilde katmanlar arasındaki bağlantının ağırlık değerleri güncellenir.

Yapay sinir ağları için arama optimizasyon algoritmasında kullanılan hiper-parametreler,

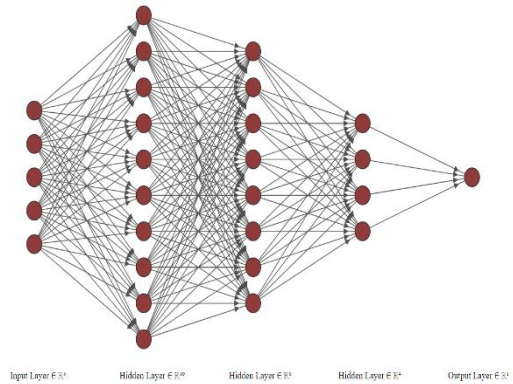
- Ara Katman Nöron Sayısı: 3, 5, 10, 12, 15, 20, 50, 75, 100, 250
- Ara Katman Sayısı: Maksimum 3 Katman
- Maksimum İterasyon Sayısı: 5000, 10000, 15000, 50000
- Aktivasyon Fonksiyonları: Relu (Rectified Linear Unit), tanh (hyperbolic tangent), logistic
- Optimizasyon Algoritması: lbfgs (Limited Memory Broyden-Fletcher-Goldfarb-Shanno), Sgd (Stochastic Gradiant Descent), Adam (Adaptive Moment Estimation)
- Öğrenme Katsayısı: constant, invscaling, adaptive
- Alfa: 0.0001, 0.001, 1e-5
- Momentum: 0.8, 0.9
- Batch size: 1, 4, 16, 64, 128, 256

4.4 Derin ve Evrişimli Sinir Ağları

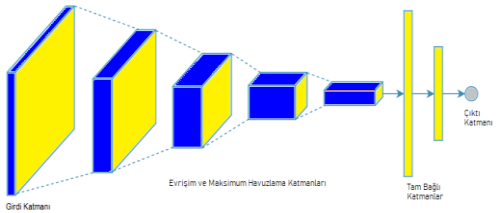
Derin sinir ağı, giriş ve çıkış katmanları arasında birden çok katman bulunan bir yapay sinir ağıdır. Son yıllarda yapay sinir ağları alanındaki araştırmalarda evrişimli sinir ağı modelleri önemli bir gelişme göstermiştir. Bunun en önemli sebeplerinden biri diğer sinir ağı modellerine göre evrişimli sinir ağı modellinin çok miktarda veri ile çalışabilmesidir. Derin sinir ağlarının eğitimi, yaygın olarak "hiper parametreler" olarak adlandırılan çok sayıda seçenek içerir. Bunlar, katman sayısını, nöronları ve spesifik aktivasyon fonksiyonunu içeren hiper-parametrelerdir[18].

ESA mimarisi temel olarak girdi katmanı, evrişim katmanı, havuzlama katmanı, tam bağlantılı katman ve çıktı katmanından oluşmaktadır. Evrişim katmanı ilk olarak Yann LeCun tarafından sunulmuştur [7]. Evrişimli sinir ağları sınıflandırma problemlerinin yanı sıra regresyon problemlerini çözmek için de kullanılır [8]. Şekil-2'de 3 evrişim katmanı, 2 tam bağlantılı katmandan oluşan bir evrişimli sinir ağı yapısı gösterilmektedir.

Evrişimli sinir ağının yerel algısı ve ağırlık paylaşımı, parametre sayısını büyük ölçüde azaltabilir, böylece öğrenme modellerinin verimliliğini artırabilir[21].



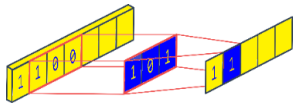
Şekil 2: YSA yapısı



Şekil 1: Evrişimli Sinir Ağı Yapısı

Evrişim özelleştirilmiş bir doğrusal işlemdir. Bu ağlar basitçe, en az bir katmanında matris çarpımı yerine evrişim işlemi gerçekleştiren ağlardır [12]. Evrişim işlemi, diğer makine öğrenmesi algoritmalarına göre parametre paylaşımı, seyrek etkileşim ve eş değişken kullanması nedeniyle avantajlar sunmaktadır [12]. Evrişim katmanında gerçekleşen seyrek etkileşimler, daha az parametre tanımlamasını sağlamaktadır. Bu durum istatiksel verimliliği arttırmakta ve parametre depolama alanında avantaj sağlamaktadır. Evrişim katmanında filtre olarak çoğunlukla 3x3, 5x5 ve 7x7 'lik filtreler kullanılmaktadır.

Çalışmamızda kullandığımız konvolüsyon katmanı tek boyutludur. Kullanılan evrişim filtresindeki nöron sayısı 3'dür. Evrişim filtresi girdi bilgisi üzerinde belirtilen derecede kayma sağlayarak çıktı oluşturmaktadır. Her konumda filtre ile girdinin üst üste çakıştığı bölgeler çarpılmakta ve bu çarpımlar toplanarak yeni yeni birer matrise aktarılmaktadır. Bu matrise öznitelik haritası denir. 1D evrişim işlemi Şekil-4'te gösterilmektedir.



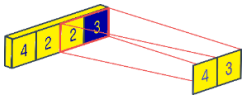
Şekil 4- 1D Evrişim İşlemi

En iyi filtreyi tasarlamak aynı zamanda en verimli filtre parametrelerini seçmeyi gerektirir. Bu parametreler filtre ağırlık değerleri ve filtre boyutudur. Bu parametrelerin düzenlenmesi ağ çıktısının verimini artırmayı ve hafıza kullanımını azaltmayı sağlasa da eniyileme için sadece bu alanlara odaklanmak yeterli olmamaktadır. Bazı araştırmacılar filtre boyutunun ağın çıktısına etkisini de araştırmıştır. Szegedy ve arkadaşları [13] 5×5 n tane filtre ile 3×3 aynı sayıda filtreyi karşılaştırmış ve 5×5 filtre ile yapılan hesaplamanın 2,78 kat daha maliyetli olduğu sonucuna varmıştır.

Havuzlama işlemlerinde, alt bölgeleri özetlemek için ortalama veya maksimum değer alma gibi bazı fonksiyonlar kullanılarak çıktının boyutu düşürülmektedir. Havuzlama işlemi belirlenen bir alan içindeki değerlerin ortalamasını alarak veya maksimumunu hesaplayarak bir değer çıkarır. Girdide havuzlama işlemi kayan pencere yöntemiyle uygulanmaktadır. Kayan pencere her seferinde belirlenen havuzlama yöntemine göre denk geldiği girdi alanından bir değer oluşturur ve bunu çıktı katmanına ekler [14].

Tam bağlı katmanlar yapay sinir ağı gibi çalışmaktadır. Evrişim ve havuzlama işlemleri sonucunda üretilen değerler bu katman tarafından girdi olarak alınarak işleme sokulmakta ve çıkış katmanında sınıf sayısı kadar sonuç üretilmektedir.

Bir evrişimli sinir ağında havuzlama katmanları sayesinde, daha önce bahsedilen girdinin küçük parçaları, tercih edilen yönteme göre tek bir sabit değere indirgenmiş olur. Buna göre havuzlama katmanı hesaplamaları, evrişim katmanı hesaplamalarına göre daha az maliyetlidir. Şekil 5'de listelenen havuzlama türlerinden en yaygın olarak kullanılan türü, girdinin parçalara ayrılıp her bir parçadaki en yüksek değerin alındığı maksimum havuzlamadır. Havuzlama türlerinden sıklıkla kullanılan ortalama havuzlamada, girdi yine maksimum havuzlamadaki gibi parçalara ayrılarak her bir parçadaki değerlerin ortalaması alınır.



Şekil 5: Havuzlama

Girdi katmanında veri girişi sağlanmaktadır. Başarım için bu katmandaki veri boyutu önemlidir. Bu nedenle ağ derinliği, donanımsal hesaplama maliyeti ve ağ başarısı için uygun bir veri boyutu seçilmelidir.

Evrişim katmanı ESA'nın temelini oluşturan katmandır. Evrişim katmanı, evrişim işlemlerini gerçekleştiren filtreleri, I girişini boyutlarına göre tararken kullanır. Hiper-parametreleri F filtre boyutunu ve S adımını içerir. Elde edilen çıktı O, öznitelik haritası veya aktivasyon haritası olarak adlandırılır [15].

Evrişim katmanından aktivasyon fonksiyonu olarak RELU kullanılmıştır. Ortalama katmanı, tipik olarak uzamsal değişkenlik gösteren bir evrişim katmanından sonra uygulanan bir örnekleme işlemidir. Özellikle, maksimum ve ortalama, sırasıyla maksimum ve ortalama değerin alındığı özel ortalama türleridir. Tam bağlı katman, her girişin tüm nöronlara bağlı olduğu bir giriş üzerinde çalışır.

Derin ve Evrişimli Sinir Ağları için arama optimizasyon algoritmasında kullanılan hiper-parametreler:

- Epoch Sayısı: 50, 100, 150, 250, 500
- Batch Size: 2, 4, 8, 16, 32, 64,128
- Optimizasyon Algoritması: Rmsprob (Root Mean Squared Propagation), Adam, Sgd
- Dropout: 0,0.2, 0.4

4.5 Elman Sinir Ağları

Geri dönüşümlü ağlarda, sinir hücrelerinin çıktıları yine ağa belirli kurallara döndürülerek girdi olarak kullanılmaktadır. Geri dönüşümlü sinir ağları 2'ye ayrılmaktadır:

- Tam Geri Dönüşümlü Sinir Ağları
- Kısmi Geri Dönüşümlü Sinir Ağları

Tam geri dönüşümlü sinir ağlarında ileri ve geri bağlantılı olan ağlar bulunmaktadır ve bu ağların hepsi eğitilebilir. Kısmi geri dönüşümlü sinir ağları ise içerik katmanı bulunmaktadır. Temelde bu ağlar ileri beslemeli sinir ağıdır.

Geri dönüşüm aşaması sadece içerik katmanında gerçekleştirilir. İçerik katmanı ile girdi katmanı arasındaki bağlantılar eğitilemezler. İçerik katmanındaki her nöron bir önceki iterasyondaki durumları içermektedir ve geçmiş durumları hatırlamak için kullanılmaktadır. Geçmiş durumları hatırlamalarında dolayı dinamik bir hafızaya sahiptirler.

Elman sinir ağı, geri dönüşümlü bir sinir ağıdır. Çok katmanlı algılayıcıdaki öğrenme kuralı ile aynı öğrenme kuralını kullanarak öğrenme sağlar. Elman sinir ağı, tipik bir geri beslemeli sinir ağıdır. Yaygın olarak kullanılan model, genellikle dörde bölünmüştür [22]. Bir Elman sinir ağında,

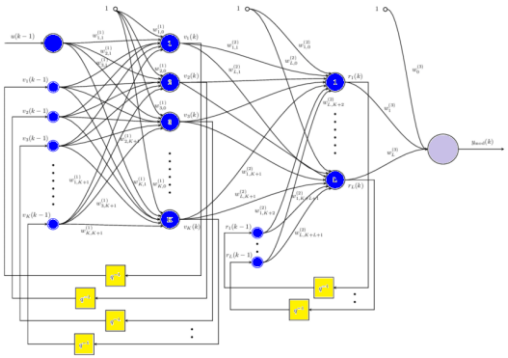
- Girdiler
- Ara Katmanlar
- Çıktılar
- İçerik Katmanları

olmak üzere 4 işlem birimi bulunmaktadır.

Girdi nöronlarına gelen bilgileri ilk ara katmana aktarmaktadır. Ara katmanlar arasında bilgiler iletilerek çıktı katmanına sonuç bilgileri iletilmektedir. Çıktı bölümünde bulunan fonksiyonlar doğrusal fonksiyonlardır. Ara katmanlarda hem doğrusal hem de doğrusal olmayan fonksiyonlar bulunmaktadır.

Elman sinir ağı, sistemin zamanla değişen özelliklere uyum sağlama, ağın kararlılığını geliştirme yeteneğine sahip olması ve bellek amacına ulaşmak için tek adımlı bir gecikme operatörü olarak gizli katmana bir yatak katmanı ekler [22].

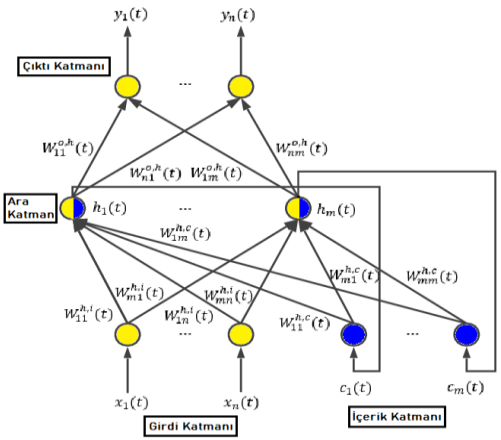
Şekil-6'te gösterilen yapı 2 ara katmana sahip bir elman sinir ağıdır. Her bir ara katmanın kendine ait içerik ünitesi bulunmaktadır. İçerik katmanında bulunan nöronlar her bir ara katman nöronundan gelen net bilgi değerini alır bir sonraki iterasyon işleminde bu bilgiyi işleyerek ilgili katmana iletirler. Bu işlem bir one step time delay olarak adlandırılmaktadır. İçerik nöronları ile ilgili ara katman nöronları arasındaki ağırlıklar sabittir, değiştirilemez. Bu prosesde herhangi bir optimizasyon fonksiyonu kullanılmaz. Bu yüzden elman sinir ağına kısmi dönüşümlü sinir ağı da denilebilir [10].



Şekil 6 - Elman Sinir Ağı [17]

Elman sinir ağında girdi değerleri ile ara katmanlardan gelen bir önceki iterasyondaki aktivite değerleri ağa girdi olarak verilmektedir. Bu bilgiler toplandıktan sonra ağ ileri beslemeli çok katmanlı algılayıcı olarak çalışmaktadır. İleri doğru hesaplama aşamasında ağın ara katmanlarındaki nöronlarından gelen aktivasyon değerleri içerik nöronlarına aktarılarak bir sonraki iterasyonda kullanılmak üzere saklanır[17].

Elman sinir ağlarında içerik nöronlarının başlangıç değerlerini atama kısmında kesin bir yöntem bulunmamaktadır. Genellikle sigmoid fonksiyonu kullanılacaksa 0.5, hiperbolik tanjant fonksiyonu kullanılacaksa 0 olarak başlangıç girdi değerleri verilebilir.



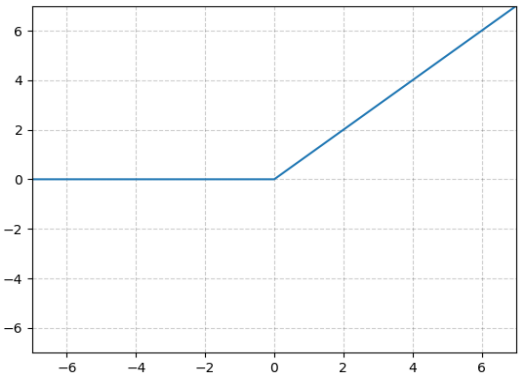
Şekil 7: Detaylı Elman Sinir Ağı [16]

Şekil-7'de tek ara katmanlı elman sinir ağının basit bir yapısı gösterilmiştir. Şekildeki x(t)'ler t. zaman diliminde sinir ağına gelen girdi bilgilerini, y(t)'ler t. zaman diliminde üretilen çıktıyı, h(t)'ler t. zaman biriminde üretilen ara katman nöronlarının çıktısını c(t)'ler ise t. zaman birimindeki içerik katmanı nöronlarının değerlerini göstermektedir.

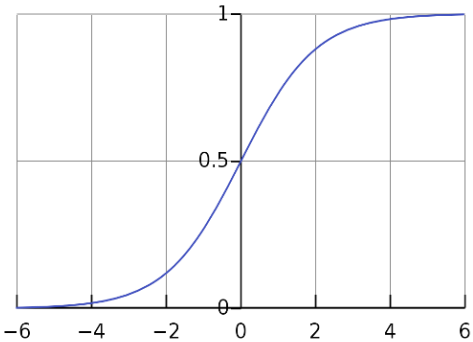
Ara katmanlar ile girdi katmanı arasındaki ağırlıklar $W^{h,i}(t)$ ile ara katmanlar ile içerik katmanı arasındaki ağırlıklar ise $W^{h,c}(t)$ ile gösterilmiştir. $W^{h,c}(t)$ ağırlıkları değiştirilemez.

Elman sinir ağının öğrenmesi delta öğrenme kuralına göre gerçekleşmektedir. Ara katmanda bulunan nöronların net girdi değerleri girdi katmanındaki nöronların net girdi değerleri ile $W^{h,i}$ ağırlık değerlerinin çarpılıp toplanması sonucu bulunan değerlere içerik nöronlarından gelen bağlantı değerlerinin ara katmanların bir önceki aktivite değerleri ile çarpılıp toplanması sonucu bulunur. Bu çıktı bir aktivasyon fonksiyonundan geçirilmektedir.

Çıktımız ara katmandaki ilgili nöronun çıktısı olacaktır. Ara katman elemanında doğrusal ve doğrusal olmayan uyarı fonksiyonları vardır ve uyarı fonksiyonu genellikle sigmoid doğrusal olmayan fonksiyonunu alır[22]. Bu çalışmada hem türevinin kolay alınması hem de güncel çalışmalarda çoğunlukla kullanılan sigmoid ve RELU aktivasyon fonksiyonları kullanılmaktadır. Şekil-8 ve 9'da RELU ve sigmoid aktivasyon fonksiyonlarının grafikleri gösterilmiştir.



Şekil 8: RELU Aktivasyon Fonksiyonu



Şekil 9: Sigmoid Aktivasyon Fonksiyonu

$$\frac{1}{1+e^{-Net(t)}} \tag{1}$$

Formül 1'de sigmoid aktivasyon fonksiyonu gösterilmiştir. Burada Net(t) ifadesi belirttiğimiz ara katmandaki ilgili nörondan çıkan net çıktıdır. Buradaki Net girdiyi ifade edecek olursak,

$$NET(t) = W_n^{h,i}(t) * x_n(t) + c_m * W_m^{h,c}(t-1) \tag{2}$$

Formül-2 olarak gösterebiliriz. Formül tam olarak yazacak olursak,

$$NET(t) = \sum_{i=1}^{n} W_n^{h,i}(t) * x_n(t) + \sum_{i=1}^{m} W_m^{h,c}(t) * c_m(t-1) \tag{3}$$

Burada n ile gösterilen değer girdi katmanındaki nöron sayısı m ile gösterilen değer ise ara katmanda bulunan nöron sayısıdır. Çok katmanlı sinir ağlarındakine benzer şekilde aktivasyon fonksiyonları türevi kolay alınabilecek şekilde seçilebilir. Genelleştirilmiş delta kuralına göre hem çıktı değerleri hesaplanır hem de ağırlıkların ilgili optimizasyon fonksiyonları ile güncellemesi sağlanır [16]. Çıktı nöronlarındaki değer son katmandaki nöronlardan gelen net çıktıların doğrusal fonksiyonlardan geçirilmesi ile bulunmaktadır. t. zaman dilimindeki çıktı katmanında bulunan nöronun çıktısı

$$y_n(t) = \sum_{i=1}^{m} W_{nm}^{o,h}(t) * h_m(t) \tag{4}$$

Formül-4 ile gösterilebilir. Bu durumda ağın ileri besleme aşaması tamamlanmış olacaktır. Çalışmada bir adet çıktı bulunmaktadır. Bu yüzden çıktı katmanı bir nörondan oluşur. Ağın çıktı katmanına aktarılan her bilgi için t. zaman dilimindeki hesaplanan hata değeri Formül 5'teki gibi gösterilebilir.

$$E_i = pi(t) - yi(t) \tag{5}$$

Sinir ağlarındaki temel amaç buradaki hata değerini en aza indirebilmektir. Bu hata değeri ile ağın ağırlıklarının güncellenme işlemi başlatılmaktadır. Bu aşama geri yayılım aşamasıdır. Toplam hata değerini hesaplamak için,

$$ToplamHata = \frac{1}{2} * \sum_{n} E_m^2 \tag{6}$$

formülünden yararlanılmaktadır. Buradaki bulunan kare ifadesinin amacı bazı gelen hata değerlerinin negatif olmasıdır. Çalışmamızda çıktı katmanında tek bir nöron bulunduğu için buradaki n değeri 1'dir. Toplam hatayı en aza indirebilmek için bu hataya neden olan katmanlara dağıtılması ve bu katmanlardaki ağırlıkların güncellenmesi gerekmektedir.

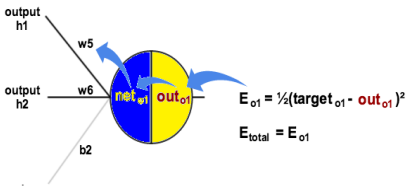
Bu aşamada iki kısım bulunmaktadır:

- Ara katman ve çıktı katmanı arası ağırlık güncellemeleri
- Ara katmanlar arası ve ara katman girdi katmanı arasındaki ağırlıkların güncellemeleri

Daha önceden belirttiğimiz üzere içerik katmanındaki nöronlar ile girdi katmanındaki nöronlar arası ağırlıklar değiştirilmemektedir.

Şekil–10'da w5 ile gösterdiğimiz değer bir ara katman nöronu ile çıktı katmanı nöronu arasındaki ağırlıktır. Bu kısımda w5 ağırlık değerinin total hatayı ne kadar etkilediğini ($\frac{\partial Etotal}{\partial w5}$) göz önünde bulunduruyoruz. Zincir kuralına göre,

$$\frac{\partial Etotal}{\partial w5} = \frac{\partial Etotal}{\partial outo1} + \frac{\partial outo1}{\partial neto1} + \frac{\partial neto1}{\partial w5} \tag{7}$$



Şekil 10- Geri Yayılım Ara-Çıktı Katmanı

Öncelikle bu denklemi hesaplamak için bu Formül-8'deki her parçayı bulmamız gerekir. Toplam hata Formül-6'da gösterilmiştir. Total hatanın outo1'e göre kısmi türevi,

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) \quad (8)$$

İkinci aşamada ise toplam girdiye göre o1 değişimini bulmamız gerekmektedir. Lojistik fonksiyonun kısmi türevi çıktının 1 eksiği ile çarpımıdır.

$$out_{o1} = \frac{1}{1-e^{-net_{o1}}} \quad (9)$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) \quad (10)$$

Son olarak w5 ağırlık değerine göre net o1 değerinin değişimini bulmamız gerekmektedir.

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 \quad (11)$$

$$\frac{\partial net_{o1}}{\partial w_5} = out_{o1} \quad (12)$$

Bulduğumuz bütün denklemleri birleştirirsek Formül-13'te gösterilen w5 ağırlık değerinin toplam hataya etkisini bulmuş oluruz.

$$\frac{\partial Etotal}{\partial w5} = (out_{o1} - target_{o1}) * out_{o1}(1 - out_{o1}) * out_{o1} \quad (13)$$

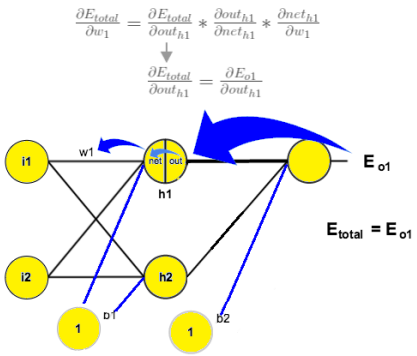
Alternatif olarak $\frac{\partial E_{total}}{\partial net_{o1}}$ olarak yazılan $\frac{\partial E_{total}}{\partial out_{o1}}$ ve $\frac{\partial out_{o1}}{\partial net_{o1}}$ ifadelerini δ_{o1} aka düğüm noktası olarak gösterebiliriz. Hatayı azaltmak için bulduğumuz Formül-14'teki değerini belirleyeceğimiz öğrenme katsayısı ile çarparak ağırlık değerimizden çıkarmamız gerekmektedir.

$$w_5^{+} = w_5 - \eta * \frac{\partial Etotal}{\partial w5} \quad (14)$$

Ara katman ile çıktı katmanı arasındaki ağırlık değerlerinin hesabından sonra ara katman ile ara katmanlar arasındaki ağırlık değerleri hesaplanabilir. Çıktı katmanı için yaptığımıza benzer bir süreç kullanacağız, ancak her bir gizli katman nöronunun çıktısının, birden fazla çıktı nöronunun çıktısına (ve dolayısıyla hataya) katkıda bulunduğu gerçeğini hesaba katmak için biraz farklı durumdur.

out_{h1}'in out_{o1} etkilediğini biliyoruz, bu nedenle $\frac{\partial E_{total}}{\partial out_{h1}}$' in çıktı nöronu üzerindeki etkisini de dikkate alması gerekiyor:

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} \quad (15)$$



Şekil 11 – Geri Yayılım Ara Katman

Öncelikle outh1 sonucunun Eo1 değerine etkilemişini hesaplamak gerekmektedir. Formül 11'den $\frac{\partial out_{o1}}{\partial net_{o1}}$ değerini biliyoruz.

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} \quad (16)$$

net_{o1} değerini Formül-11'den biliyoruz. $\frac{\partial net_{o1}}{\partial out_{o1}}$ değeri buradan w5 olarak hesaplanmaktadır. Bu durumda $\frac{\partial E_{o1}}{\partial out_{h1}}$ değeri Formül-17'de gösterildiği gibi bulunmaktadır.

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial out_{h1}} \quad (17)$$

Bu işlemlerden sonra Formül-15'te gösterdiğimiz

$\frac{\partial E_{total}}{\partial out_{h1}}$ değeri hesaplanabilmektedir. $\frac{\partial out_{o1}}{\partial net_{h1}}$ değerini bulmak için net_{h1} değerinin outh1 değerini etkilemesini hesaplamalıyız. Formül-9'da o1 için lojistik fonksiyonu belirtmiştik. Aynı şekilde h1 için,

$$out_{h1} = \frac{1}{1-e^{-net_{h1}}} \quad (18)$$

olarak gösterilebilir.

Kısmi türevin sonucu ise Formül-10'da o1 için gösterdiğimizle benzer şekilde,

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) \quad (19)$$

Buradan $\frac{\partial net_{h1}}{\partial w1}$ değeride neth1 = w1*i1 + w3*i2 + b1 'den i1 olarak hesaplanmaktadır. Bütün bu hesaplamaları dikkate alarak w1 ağırlığının toplam hatayı etkileme durumunu,

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w1} \quad (20)$$

formülünden hesaplayabilmekteyiz. Aynı işlemler diğer ara katman – ara katman arası ağırlık değerlerini hesaplamak için kullanılmaktadır. İleri besleme aşamasında her ara katmana aktarılan girdi net değeri içerik katmanındaki ilgili nöronda kaydedilmektedir.

Optimizasyon yöntemleri olarak çalışmada Momentumlu Stokastik Gradyan İnişi, Mini Parti Stokastik Gradyan İnişi ve Adam algoritmaları kullanılmıştır.

Gradyan inişi, amaç fonksiyonunun dışbükey olmamasına rağmen sinir ağlarının eğitiminde küresel bir minimum bulur [24]. Gradyan inişi en basit ve en kötü yöntem olarak nitelendirilebilir. Buradaki problem $\min_w f(w)$'yi hesaplamaktır. Tekrarlamalı çözüm ise Formül-14'te bahsettiğimiz ile aynıdır. Burada f fonksiyonunun sürekli ve türevlenebilir olduğu varsayılmaktadır. Amacımız optimizasyon fonksiyonunun en düşük noktasını (vadi) bulmaktır. Ancak, bu vadiye giden kesin yön bilinmemektedir. Yalnız bölgesel olarak bakılınca negatif gradyanın yönü sahip olduğumuz en iyi bilgidir. Ancak bu yönde küçük bir adım atmak bizi minimuma yaklaştırır. Bir kez küçük bir adım attıktan sonra, tekrar yeni gradyanı hesaplarız ve tekrar bu yönde küçük bir miktar hareket ederiz, ta ki vadiye ulaşıncaya kadar. Aslında temel olarak gradyan inişinin yaptığı şey en dik iniş (negatif gradyan) yönünü takip etmektir.

Tekrarlamalı güncelleme denklemindeki η parametresine adım boyutu denir. Genelde en uygun adım boyutunun değeri bilinmez; bu yüzden farklı değerler denemek zorunda kalırız. Standart uygulama, bir log ölçeğinde bir grup değeri denemek ve en iyisini kullanmaktır. Bu durumda oluşabilecek birkaç farklı senaryo vardır. Yukarıdaki görüntü 1 boyutlu ikinci dereceden (quadratic) bir fonksiyon için bu senaryoları göstermektedir. Öğrenme hızı çok düşük olursa minimum seviyeye doğru istikrarlı bir ilerleme kaydedebiliriz. Ancak bu ideal olandan daha fazla zaman alabilir. Bizi doğrudan minimuma indirecek bir adım boyutu elde etmek genellikle çok zordur. Bu durumda ideal olarak isteyeceğimiz şey, optimalden biraz daha büyük bir adım boyutudur. Pratikte bu en hızlı yakınsamayı sağlar. Bununla birlikte çok büyük bir öğrenme oranı kullanırsak iterasyonlar sonunda minimumdan uzaklaşırız ve sapmış oluruz. Dolayısıyla uygulamada bizi saptıracak değerden biraz daha düşük bir öğrenme hızı kullanmak isteriz.

Stokastik gradyan inişinde ise gerçek gradyan vektörü stokastik tahmin ile değiştirilmektedir. Sinir ağlarında, stokastik tahmin tek bir örnek kayıp değerinin gradyanı anlamına gelmektedir. Gradyan karışıklığı yüksek olduğunda, farklı veri örnekleri tarafından üretilen stokastik gradyanlar, yakınsamayı yavaşlatarak negatif korelasyon gösterebilir. Ancak gradyan karışıklığı düşük olduğunda, veri örnekleri uyumlu bir şekilde etkileşime girer ve eğitim hızla ilerler[25]. i-inci örnek için kayıp değeri gösterilmektedir.

$$f_i = loss(x_i, y_i, w) \quad (21)$$

Sonuç olarak en aza indirmek istediğimiz fonksiyon tüm örneklerin toplam kayıp değer olan f'dir.

$$f = \frac{1}{m}\sum_i^m f_i \quad (22)$$

Stokastik gradyan inişinde fi üzerindeki gradyana göre ağırlıklar güncellenmektedir. Burada en önemli noktalardan biri ise i'nin rastgele biçimde seçilmesidir.

$$w_{k+1} = w_k - \eta_k * \nabla f_i(w_k) \quad (23)$$

Stokastik gradyan inişinde fi üzerindeki gradyana göre ağırlıklar güncellenmektedir. Eğer i rastgele seçilirse fi f gürültülü fakat tarafsız bir tahminleyici haline gelmektedir. Bunun sonucu olarak, SGD'nin beklenen k. adımı, tam gradyan inişin k. adımı ile aynıdır:

$$E[w_k + 1] = w_k - \eta_k * E[\nabla f_i(w_k)] \quad (24)$$

Mini parti aşamasında tek bir örnek üzerinden gitmek

yerine rastgele seçilmiş çoklu örnekler ile kayıplar bulunmaktadır. Bu işlem gürültü azaltıcı bir işlemdir ve donanımı genellikle daha az yorar.

Momentumlu yaklaşım ise her iterasyonda öğrenme katsayısına ek olarak bir momentum değişkeni işleme katılmaktadır. Momentumu p olarak gösterirsek her iterasyonda momentum Formül-25'teki şekilde değişecektir.

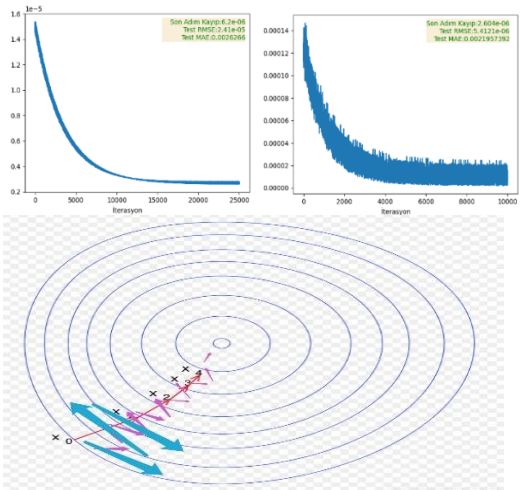
$$p_{k+1} = \beta_k p_k + \nabla f_i(w_k) \quad (26)$$

Her iterasyondaki ağırlık değişimi:

$$w_{k+1} = w_k - \eta_k * p_{k+1} \quad (27)$$

olarak ifade edilmektedir. ρ değişkeni SGD momentumu olarak ifade edilmektedir. Her iterasyonda gradyan β_k değeri ile sönümlendikten sonra momentumun eski değerine eklenir. ρ gradyanların her iterasyonda hesaplanan ortalaması olarak ifade de edilebilir.

Momentum fizikteki momentum ile aynı mantıkta çalışır. Optimizasyon süreci tepeden aşağıya yuvarlanan bir topa benzetilebilir. Burada momentum yönü topun gittiği yönde hareket ettirilmektedir. Gradyan, topu farklı aksislere çekmektedir. Momentum burada optimize yöndeki salınımları azaltmaktadır. Şekil-12'de momentumun kullanımının etkisi gösterilmiştir.



Şekil 12- SGD Momentum Etkisi

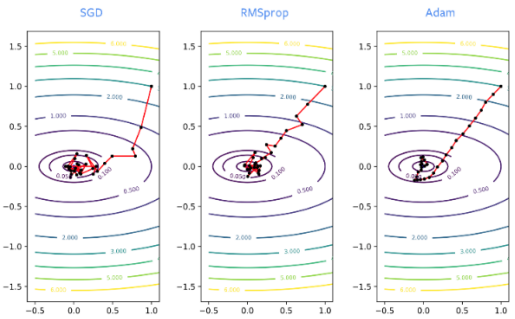
Momentumlu yaklaşım ise her iterasyonda öğrenme katsayısına ek olarak bir momentum değişkeni işleme katılmaktadır. Momentumu p olarak gösterirsek her iterasyonda momentum Formül-25'teki şekilde değişecektir. ADAM, RMSProp'a momentum eklenerek elde edilen ve daha çok kullanılan bir yöntemdir. Momentum güncellemesi üssel hareketli ortalama ile yapılmaktadır yani gradyanın karesinin üssel ortalaması alınır. m_{t+1} değeri momentumun üstel hareketli ortalamasıdır.

$$m_{t+1} = \beta_{mt} + (1-\beta)\nabla f_i(w_t) \quad (28)$$

$$v_{k+1} = \eta_k v_k + (1-\eta_k\ )\nabla f_i({w_t}^2) \quad (29)$$

$$w_{k+1} = w_t - \nabla f_i\left(\frac{m_t}{E+\sqrt{v_{t+1}}}\right) \quad (30)$$

Sinir ağlarının eğitim sürecinin başlangıcında SGD genellikle yanlış yöne giderken, RMSProp'a doğru yönde ilerler. Bununla birlikte, RMSProp da klasik SGD gibi gürültüden etkilenir, yani yerel bir minimum bulduğunda optimumun etrafında yaptığı sıçramalar kritik sonuçlara yol açar. Tıpkı SGD'ye momentum eklediğimizde olduğu gibi, ADAM ile de aynı iyileştirmeyi elde ederiz. Bu çözümün gürültülü olmayan iyi bir tahminidir, bu nedenle ADAM genellikle RMSprop üzerinden önerilir. Şekil-13'te SGD, RMSProp ve ADAM algoritmalarının karşılaştırmaları gösterilmiştir.



Şekil 13- Adam, Sgd, RMSProp global minimum iniş

Sonuçlar

5.1. Sayısal Sonuçlar

Aşağıda Tablo-1'de birinci uygulama için 18 farklı kur değerindeki en düşük MSE değeri veren makine öğrenme ve yapay zekâ algoritmaları gösterilmiştir.

Tablo-1

			ASK-MODEL
KUR	ALGORİTMA	(MSE)	(MAE)
AUD.TRY	RF	8.613605311166703e-07	0.000300765134477457 04
AUD.USD	ANN	4.379945333544657e-09	1.53E-06
CAD.TRY	CNN	1.3192231962091462e-07	0.000262249645050 20
EUR.GBP	DNN	8.35041474928907e-08	0.000192970455295497 88
EUR.TRY	RF	5.697261279845799e-06	0.000993360610704001 62
EUR.USD	RF	2.0928051270214633e-07	1.843543099309346 7e-05
GBP.TRY	ANN	2.777484993952163e-06	0.000290283533818512 5
GBP.USD	DNN	9.383370712423344e-07	0.000961387783200277
JPY.TRY	CNN	7.82E-07	0.000273054460547965
NOK.TRY	ANN	4.038155308871388e-08	9.33E-05
SEK.TRY	ANN	1.762347954227498e-07	6.153669692319692e-05
TRY.SAR	DNN	6.873833771934793e-07	0.000328595801571 57
USD.CHF	RF	2.1145257963599476e-07	0.000227809238021816 6
USD.JPY	RF	4.957411989869195e-11	1.728568361052433 2e-06
USD.TRY	CNN	4.033896781868719e-06	0.000139279137249925 28
XAR.TRY	GBM	1.063513957916152e-05	0.002112787391953182
XAR.USD	CNN	3.780160652229881e-07	0.000301417490213544
ORTALAMA		1.63E-06	
			BİD-MODEL
KUR	ALGORİTMA	(MSE)	(MAE)
AUD.TRY	CNN	7.92431338513482e-07	0.000304897508623144 1
AUD.USD	DNN	2.673633092121959e-09	1.528669054410898 4e-05
CAD.TRY	CNN	7.864322649629613e-07	0.000302589292768087 2
EUR.GBP	DNN	8.701286588400785e-08	0.000170242063868344 1
EUR.TRY	RF	5.149423575886132e-06	0.001112714552378589 6
EUR.USD	RF	2.417851891785245e-07	0.000235299997539426 91
GBP.TRY	RF	2.59245129878044e-06	0.000560190622739375
GBP.USD	CNN	7.440709923330879e-07	0.000593187115422989 8
JPY.TRY	CNN	2.589751439496704e-07	0.000271169284078663 9
NOK.TRY	ANN	2.701672879470 53e-07	0.000359486086938939 55
SEK.TRY	CNN	5.183868213832636e-07	0.000549136090536145 4
TRY.SAR	ANN	1.151779954276793 7e-06	0.000571885172926527 8
USD.CHF	GBM	2.021645681801459e-07	0.000219351568256629 18
USD.JPY	RF	6.718192284 5181e-11	2.131318330777757 5e-06
USD.TRY	GBM	1.142672237030 1057e-06	0.000689859176042530 2
XAR.TRY	RF	1.557469274933711e-05	0.002165202060569 3175
XAR.USD	DNN	4.229777942074444e-07	0.000395925911398092
ORTALAMA		1.76E-06	

Aşağıda Tablo-2'de ikinci uygulama için 18 farklı kur değerindeki en düşük MSE değeri veren makine öğrenme ve yapay zekâ algoritmaları gösterilmiştir.

Tablo-2

			ASK-MODEL
KUR	ALGORİTMA	(MSE)	(MAE)
AUD.TRY	CNN	1.942067738109 27e-06	0.001172797219020358
AUD.USD	CNN	3.021890349339 15e-09	4.149890701408703e-05
CAD.TRY	CNN	1.226604198736 93e-07	0.000241363136686 8355
EUR.GBP	ANN	5.491110028147064e-08	6.477135110240639e-05
EUR.TRY	GBM	5.615873760292 35e-06	0.000908637157095 6814
EUR.USD	DNN	2.146770741648e-08	4.214067108378151 5e-05
GBP.TRY	ANN	2.7308607500328e-06	0.000492365268782 9344
GBP.USD	ANN	1.362450675151 53e-07	0.000322562336 4430756
JPY.TRY	CNN	9.939144111621553e-07	0.000905928747994 5589
NOK.TRY	CNN	5.69710888805 28e-07	0.000400430388 5996342
SEK.TRY	ANN	8.799822034926 2e-08	5.657538058475481e-05
TRY.SAR	GBM	1.088399261563 15e-06	0.000599549127866747
USD.CHF	ANN	2.097733894444e-07	0.000231622693515 03038
USD.JPY	ANN	6.919645777523507e-11	2.384578484992844e-06
USD.TRY	CNN	5.181509505552 4e-06	0.002854272754134 5674
XAR.TRY	GBM	2.787054183016 7e-05	0.003874578415531 1256
XAR.USD	CNN	3.056284198262 4e-07	0.000141852939013 02577
ORTALAMA		2.76E-06	
			BİD-MODEL
KUR	ALGORİTMA	(MSE)	(MAE)
AUD.TRY	CNN	1.425210518165 95e-06	0.000888825128 9098387
AUD.USD	CNN	3.756197176100 6e-09	3.030121362772331e-05
CAD.TRY	ANN	8.611709822303 7e-07	0.000297955413162 6389
EUR.GBP	CNN	7.634528944114e-08	7.751725610305262e-05
EUR.TRY	CNN	8.025560169393 7e-06	0.000815742207969 6782
EUR.USD	DNN	4.135324286989e-08	0.000152622114766 9521
GBP.TRY	ANN	2.612281683166 66e-06	0.000510708217344 0091
GBP.USD	CNN	6.451831551556 0e-08	0.000141852939013 02577
JPY.TRY	CNN	2.747991959887 2e-07	0.000214306122591 6556
NOK.TRY	ANN	1.387759180524 6e-06	0.000586179541 6572602
SEK.TRY	DNN	5.192432523252 1e-07	0.000595554122619 7573
TRY.SAR	CNN	1.257124997694 6e-06	0.000407837198 7531059
USD.CHF	ANN	1.837049522584 9e-07	0.000125863277402 63026
USD.JPY	DNN	7.085017949360 7e-09	8.390312433533456e-05
USD.TRY	ANN	7.676072207563 5e-07	0.000378220804 8019435
XAR.TRY	CNN	2.610427172199 879e-05	0.002341184167 7161115
XAR.USD	CNN	4.203705010554 0e-07	0.000282719816514 06926
ORTALAMA		2.59E-06	

Aşağıda Tablo-3'de üçüncü uygulama için 18 farklı kur değerindeki en düşük MSE değeri veren makine öğrenme ve yapay zekâ algoritmaları gösterilmiştir.

Tablo-3

			ASK-MODEL
KUR	ALGORİTMA	(MSE)	(MAE)
AUD.TRY	CNN	7.167096611927945e-07	0.000368927930193812 3
AUD.USD	DNN	3.460376555796772e-09	7.404522414219116e-08
CAD.TRY	CNN	1.221876189328017 1e-06	0.000869306345485516 7
EUR.GBP	DNN	7.404522414219116e-08	6.36E-05
EUR.TRY	CNN	2.987702752387773e-07	0.000162353494429105 17
EUR.USD	RF	3.953747770347137 3e-07	0.000214489390700364 8
GBP.TRY	CNN	4.045728259825118e-06	0.001515249414263592
GBP.USD	CNN	6.031241995441731e-07	0.000409215388218331 7
JPY.TRY	DNN	8.669551288778886e-07	0.000346206477021590 3
NOK.TRY	CNN	1.482653014945428 2e-07	0.000145551984280347 4
SEK.TRY	CNN	4.553894562208959 6e-07	0.000391380516298653 4
TRY.SAR	ANN	6.362039285688263e-08	0.000523083248659874
USD.CHF	DNN	2.315504002876090 6e-07	0.000128258258560262
USD.JPY	RF	4.871927198697006e-11	1.61E-06
USD.TRY	CNN	3.143324476232398e-07	0.000527646024867 3725
XAR.TRY	ANN	1.560319049232 49558e-05	0.001640892297726826 6
XAR.USD	DNN	4.916763528153419e-07	0.000192763079494541 5
ORTALAMA		1.50E-06	
			BİD-MODEL
KUR	ALGORİTMA	(MSE)	(MAE)
AUD.TRY	GBM	6.876829328942773e-07	0
AUD.USD	DNN	7.362635606248235e-09	4.547241015946467e-05
CAD.TRY	DNN	1.288289310466 7317e-06	0.000275252407330669 4
EUR.GBP	CNN	1.230388283212310 6e-07	0.000277325935059153 4
EUR.TRY	ANN	1.132071751735833e-06	0.000366633966455276 8
EUR.USD	RF	2.40E-07	0.000413632779774411 62
GBP.TRY	GBM	3.733693081714001e-06	0.000479625761373468 3
GBP.USD	ANN	7.795424610051722e-07	0.001343517979475578
JPY.TRY	DNN	7.912261772798176e-07	0.000383507617378357 4
NOK.TRY	DNN	1.812528455352641e-07	0.000312383826390092 6
SEK.TRY	ANN	1.368939631233177 9e-07	0.000101691448744534 2
TRY.SAR	ANN	7.884933598389471e-07	0.000688726579105623 4
USD.CHF	RF	2.70E-07	0.000465336877212130 8
USD.JPY	RF	8.632060310347304e-11	2.854646856406587e-06
USD.TRY	CNN	1.195155180745300 5e-06	0.000406204739212989 7
XAR.TRY	GBM	1.604099972542742 3e-05	0.002280232977137072
XAR.USD	RF	4.918379925395 62e-07	0.000209551612411892 13
ORTALAMA		1.64E-06	

Tablo 4'de ise Elman sinir ağları ile yapılan uygulamanın sonuçları gösterilmektedir.

Tablo-4

			ASK-MODEL
KUR	ALGORİTMA	(MSE)	(MAE)
AUD.TRY	Elman	5.6194e-06	0.00033301826
AUD.USD		3.7459e-08	5.8952e-05
CAD.TRY		1.51e-08	4.91e-05
EUR.GBP		1.726522e-07	9.91e-05
EUR.TRY		3.84213-05	0.71797511
EUR.USD		6.3678e-06	0.00287879276
GBP.TRY		1.899446e-05	0.0043262184
GBP.USD		1.3616e-06	0.0013286207
JPY.TRY		1.22489e-05	0.002891365
NOK.TRY		2.440124e-05	0.005596107
SEK.TRY		1.41493e-05	0.0050567098
TRY.SAR		9.7475e-07	0.00013151968
USD.CHF		7.4856e-06	0.0037479222
USD.JPY		2.604-06	0.0021957392
USD.TRY		1.63e-05	0.0038681
XAR.TRY		1.81e-05	0.0031184
XAR.USD		6.7e-06	0.0033833
ORTALAMA		9.69E-06	
			BİD-MODEL
KUR	ALGORİTMA	(MSE)	(MAE)
AUD.TRY	Elman	5.546e-07	0.0002761
AUD.USD		1.24e-08	3.74855e-05
CAD.TRY		8.3282e-06	0.0035644329
EUR.GBP		9.208e-07	0.0012744188
EUR.TRY		3.57249e-05	0.0065932955
EUR.USD		5.0642e-06	0.0026268416
GBP.TRY		1.83667e-05	0.0050686929
GBP.USD		5.0165e-06	0.0023492922
JPY.TRY		1.08401e-05	0.0044227682
NOK.TRY		2.853e-07	0.0006671146
SEK.TRY		6.24e-08	0.000238203
TRY.SAR		2.0936e-06	0.0019723405
USD.CHF		7.4093e-06	0.0033885091
USD.JPY		1.4048e-06	0.0013633933
USD.TRY		1.1233e-06	0.00376299737
XAR.TRY		0.0002322896	0.017962233
XAR.USD		6.2831e-06	0.0034650269
ORTALAMA		6.09E-06	

5.2. Değerlendirme, Tartışma ve Gelecek Araştırması

Algoritmalar artırımlı öğrenme olacak şekilde günlük olarak gelen veriler ile eğitilmiştir. Tablolarda her kur için en düşük hata değerini veren algoritma seçilmiştir.

Eğitimler sonucunda, 18 kur için makine öğrenme ve yapay zekâ algoritmalarının tümünün hata ortalamasına bakıldığında, en düşük hata değerini veren model, volatilitenin de öznitelik olarak veriye verildiği üçüncü modeldir. Sonuç olarak kurlar üzerindeki oynaklık bilgisi olan volatilitenin kur fiyatının değişimi açısından önemi bu çalışma ile kanıtlanmıştır.

Bu çalışmada en düşük hata değerlerini, kullanılan makine öğrenme ve yapay zekâ algoritmaları arasından yapay sinir ağları ve evrişimli sinir ağları çıkarmıştır. Algoritmaların en iyi hiper-parametreleri rastgele arama algoritması ile bulunmuştur.

Gelecekte yapılacak çalışmalarda şu anki yapılan volatilite hesabının geliştirilmesine yönelik çalışmalar planlanmakta ve daha optimize bir fiyat tahmini hedeflenmektedir. Müşteri alım-satım kur tahmini için ilerleyen çalışmalarda hibrit yapay zeka algoritmaları kullanılarak sonuçlar üzerindeki etkisinin mevcut çalışmalar ile karşılaştırılması hedeflenmektedir.

Kaynakça

[1] Niyazi Telçeken & Murat Kıyılar & Eyüp Kadıoğlu, "Volatilite Endeksleri: Gelişimi, Türleri, Uygulamaları ve Trvıx Önerisi", Ekonomi, Politika & Finans Araştırmaları Dergisi, 2019, 4(2): 204-228

[2] Özkan Şahin & Mehmet Akif Öncü "Volatilite Alanında Yapılmış Lisansüstü Tezelere Yönelik Bir İçerik Analizi" Muhasebe ve Finansman Dergisi

[3] Breiman L., Cutler A., Random forest, http://www.stat. berkeley.edu/~breiman/RandomForests/cc_home.html , (2005)

[4] Alexey Natekin, Alois Knoll, Gradient boosting machines, a tutorial, Neurorobot., 04 December 2013

[5] Silva F M and Almeida L B 1990, Acceleration techniques for the backpropagation algorithm Neural Networks ed L B Almeida andCJWellekens (Berlin: Springer)

[6] Ayşe ARI,Murat Erşen Berberler, Yapay Sinir Ağları ile Tahmin ve Sınıflandırma Problemlerinin Çözümü İçin Arayüz Tasarımı, Acta Infologıca – 2017

[7] Y. LeCun et al., "Backpropagation applied to handwritten zip code recognition," Neural computation, vol. 1, no. 4, pp. 541-551, 1989.

[8] Stephane Lathuili ´ ere, Pablo Mesejo, Xavier Alameda-Pineda, ` Member IEEE, and Radu Horaud "A Comprehensive Analysis of Deep Regression " https://arxiv.org/abs/1803.08450, v3 [cs.CV] 24 Sep 2020

[9] Antoni Wysocki and Maciej Ławryńczuk, Two- and Three-Layer Recurrent Elman Neural Networks as Models of Dynamic Processes, February 2016, Challenges in Automation, Robotics and Measurement Techniques (pp.165-175)

[10] Ercan Öztemel, Yapay Sinir Aglari Paperback – January 1, 2003

[11] Vidushi Sharma, Sachin Rai, Anurag DevA Comprehensive Study of Artificial Neural Networks, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10, October 2012, ISSN: 2277 128X

[12] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning (Adaptive computation and machine learning). Cambridge, Massachusetts: The MIT Press, pp. xxii, 775 pages, 2016.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818-2826, 2016.

[14] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," CoRR, vol. abs/1603.07285, 2016.

[15] Evrişimli Sinir Ağları el kitabı, CS 230 - Derin Öğrenme,https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-convolutional-neural-networks

[16] Ren, G., Cao, Y., Wen, S., Huang, T., & Zeng, Z. (2018). A modified Elman neural network with a new learning rate scheme. Neurocomputing, 286, 11–18. doi: 10.1016/j.neucom.2018.01.046

[17] Wysocki, A., & Lawryńczuk, M. (2016). Two- and Three-Layer Recurrent Elman Neural Networks as Models of Dynamic Processes. Advances in Intelligent Systems and Computing, 165–175. doi:10.1007/978-3-319-29357-8_15

[18] Shuaiqiang Liu, Cornelis W. Oosterlee and Sander M. Bohte Pricing Options and Computing Implied Volatilities using Neural Networks, Modern Numerical Techniques and Machine-Learning in Pricing and Risk Management, https://doi.org/10.3390/risks7010016, 2019

[19] Chuong Luong and Nikolai Dokuchaev, Forecasting of Realised Volatility with the Random Forests Algorithm, Journal of Risk and Financial Management, 2018

[20] Winky K.O. Ho, Bo-Sin Tang Predicting property prices with machine learning algorithms, Journal of Property Research, 2020

[21] Wenjie Lu, Jiazheng Li, Jingyang Wang, Lele Qin, A CNN-BiLSTM-AM method for stock price prediction, Neural Computing and Applications, 2020

[22] Bo Liu , Qilin Wu, and Qian Cao, An Improved Elman Network for Stock Price Prediction Service, Volume 2020, Article ID 8824430, 9 pages

[23] Fang Wang, Sai Tang and Menggang Li, Advantages of Combining Factorization Machine with Elman Neural Network for Volatility Forecasting of Stock Market, Volume 2021 |Article ID 6641298

[24] Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, Xiyu Zhai ¸Gradient Descent Finds Global Minima of Deep Neural Networks, arXiv:1811.03804, 2019

[25] Yuanzhi Li, Yingyu Liang, Learning Overparameterized Neural Networks via Stochastic Gradient Descent on Structured Data, arXiv:1808.01204, 2018