YİNELEMELİ SİNİR AĞLARI İLE FİNANSAL VERİ TAHMİNİ

FINANCIAL DATA PREDICTION WITH RECURRENT NEURAL NETWORKS

EKİN KEÇECİ

PROF. DR. TURHAN MENTEŞ
Tez Danışmanı

Hacettepe Üniversitesi
Lisansüstü Eğitim – Öğretim ve Sınav Yönetmeliğinin
İstatistik Anabilim Dalı İçin Öngördüğü
YÜKSEK LİSANS TEZİ
olarak hazırlanmıştır.

2020

ÖZET

YİNELEMELİ SİNİR AĞLARI İLE FİNANSAL VERİ TAHMİNİ

EKİN KEÇECİ

Yüksek Lisans, İstatistik Bölümü

Tez Danışmanı: Prof. Dr. Turhan MENTEŞ

Haziran 2020, 36 sayfa

Yinelemeli sinir ağları (RNN) elde edilen çıktıların her adımda ağa tekrar dahil edildiği yapay sinir ağı modelidir. Yinelemeli sinir ağlarının en büyük avantajı sıralı verilerde her bir örneğin önceki örneklere bağlı olarak değişimini göz önünde bulundurmasıdır. Yinelemeli sinir ağı modelleri geliştikçe bazı teorik engeller ortaya çıkmış ve bu engellere çözüm olarak farklı modeller geliştirilmiştir. Uzun Süreli Kısa Dönemli Hafıza (LSTM) ağlarının, bu modeller arasında en çok ilgi gören ve en iyi tasarlanmış yinelemeli sinir ağı modellerinden biri olduğu söylenebilir. Bu çalışmada, uygulama bölümünde LSTM modelinin finansal varlık fiyatlarının değerlendirilmesinde diğer sinir ağı modellerine göre başarısı ölçümlenmiştir. LSTM modelinin kıyaslanan diğer modellerden daha iyi sınıflama başarısı verdiği görülmüştür.

Anahtar Kelimeler: Yinelemeli sinir ağları, finans, veri tahmini, LSTM, zaman serileri

ABSTRACT

FINANCIAL DATA PREDICTION WITH RECURRENT NEURAL NETWORKS

EKİN KEÇECİ

Postgraduate, Department of Statistics
Supervisor: Prof. Dr. Turhan MENTEŞ
June 2020, 36 pages

Recurrent Neural Network is an artificial neural network model which the outputs are re-included to network input in every iteration. The biggest advantage of recurrent neural networks is that they consider the variation of each sample in the sequential data depending on the previous examples. As reccurent neural network models developed, some theoretical obstacles emerged and different models were developed as solutions to these obstacles. It can be said that Long term short term memory (LSTM) networks are one of the most popular and best designed reccurent neural network models among these models. In this study, the success of LSTM model is compared to other neural network models in evaluating financial asset prices. The LSTM model gave better classification accuracy than other compared models.

Keywords: Reccurent neural networks, finance, data prediction, LSTM, time series

İÇİNDEKİLER

ABSTRACT ii

İÇİNDEKİLER iii

1.Giriş 1

2. Yapay Sinir Ağları 3

2.1. İleri Beslemeli Yapay Sinir Ağları 3

2.1.1. Sinir Ağları Tarihçesi 3

2.1.2. Sinir Hücresi Modellemesi 4

2.1.3. Aktivasyon Fonksiyonu 6

2.1.4. Kayıp Fonksiyonu 7

2.1.5. Optimizasyon Algoritması 7

2.2.Yinelemeli Sinir Ağları 9

2.2.1. Yinelemeli Sinir Ağları Giriş ve Tarihçe 9

2.2.2. Çalışma Mantığı ve Teorisi 10

2.2.3. Kaybolan Gradyan problemi 11

2.3. Geçitli Tekrarlayan Birim (GRU) 12

2.3.1. GRU Giriş ve Tarihçe 12

2.3.2. Çalışma Mantığı ve Teorisi 13

2.4. LSTM 15

2.4.1. LSTM Giriş ve Tarihçe 15

2.4.2. Çalışma Mantığı ve Teorisi 16

3. Yapısal Kırılma ve Değişim Noktası Tespiti 18

3.1. CHOW Testi 18

3.2. Cezalandırılmış Değişim Noktası Tespiti (PELT) Yöntemi 19

4.Uygulama 20

4.1. Giriş 20

4.2. Verinin Hazırlanması 22
4.3. Model Kurulması 26
4.4. Sonuçlar 28
4.4.1. LSTM Modeli Sonuçları 28
4.4.2. GRU Modeli Sonuçları 30
4.4.3. İleri Beslemeli Sinir Ağı Modeli Sonuçları 32
4.4.4. Sonuçların Karşılaştırılması 33
5.Sonuç 34
KAYNAKÇA 37

1.Giriş

1940'lı yıllardan günümüze bilgisayarların gelişimiyle birlikte matematiksel işlemlerin hızlı çözülebilir hale gelmesi, yapay sinir ağları modellerinin işleyişini daha hızlı ve kullanışlı hale getirmiştir. Dijitalleşme devrimiyle birlikte resimden sese ve ısıdan hıza kadar her tür fiziksel değerin dijital verilere dönüştürülebilir hale gelmesi, bu değerlerin bilgisayar ortamındaki yapay sinir ağları modelleriyle öğrenilebilmesi ve girdi değerlerine bağlı tahminler üretilebilmesini sağlamıştır. Bilgisayar mimarileri ve hızlarının gelişmesiyle birlikte öğrenme hızları ve boyutları da artmıştır. Yapay sinir ağları herhangi bir matematiksel modelle tasvir edilemeyen veya çok karmaşık algoritmalara sahip problemlerin çözümü için tercih edilebilecek bir makine öğrenmesi modelidir. İnsan beyninin fonksiyonel özelliklerine benzer şekilde, öğrenme, ilişkilendirme, sınıflandırma, genelleme ve optimizasyon görevlerini başarıyla yapmaktadır.

Yinelemeli Sinir Ağı (RNN) ise birimler arasındaki bağlantıların yönlendirilmiş bir döngü oluşturduğu yapay sinir ağı sınıfıdır. İleri beslemeli sinir ağların aksine, yinelemeli sinir ağları kendi giriş belleğini girdilerin rastgele dizilerini işlemek için kullanabilmektedirler. [1].

Bu özellik sayesinde yinelemeli sinir ağları sıralı verilerde ve zaman serilerinde önceki değerlerden edindiği bilgiyi de ağa dahil ederek diğer sinir ağı modellerinden daha iyi sonuçlar elde edilmesini sağlamaktadır.

Bu çalışmada, yinelemeli sinir ağlarından önce temel oluşturması açısından sinir ağlarının tarihçesi ve çalışma prensipleri incelenecektir. Daha sonra yinelemeli sinir ağlarının bir çeşidi olan Uzun Süreli Kısa Dönemli Hafıza (LSTM) ağlarına ilişkin bir inceleme yapılacak ve son olarak LSTM modeli kullanılarak Borsa İstanbul vadeli işlem kontratları üzerinde finansal veri tahmin modeli geliştirilecektir.

Bölüm 2'de tipik bir yapay sinir ağının yapısından başlanarak, Yinelemeli Sinir Ağları (RNN) ve Uzun Süreli Kısa Dönemli Hafıza (LSTM) ağları açıklanmıştır. Bölümler

sıralanırken en basit sinir ağı yapısından, uygulamada kullanılacak olan karmaşık bir sinir ağı modeli olan LSTM ağlarına kadar aşamalı olarak geçilmiştir. Böylece sinir ağı mimarilerinin literatürdeki tarihsel gelişimini görmek de mümkün olmuştur.

Sinir ağı mimarileri ve çalışma prensipleri açıklandıktan sonra Bölüm 3'te uygulamanın veri ön işleme aşamasında kullanılacak yapısal kırılma ve değişim noktası tespit yöntemleri incelenmiştir. Bu yöntemler, uygulama bölümünde kullanılacak finansal veriyi makine öğrenmesi modeline en uygun hale getirmek için kullanılmıştır. Uygulama aşamasında bu yöntemlerin model başarısına olan etkisi de gösterilmiştir.

Uygulama bölümünde finansal zaman serisi verisi üzerinde fiyat hareketlerini yukarı veya aşağı yönlü olarak sınıflayan iki model kurulmuştur. İlk model basit bir mimariye sahip tipik bir yapay sinir ağı modelidir. İkinci model ise çalışmanın ana konusunu oluşturan en iyi tasarlanmış sinir ağı türlerinden biri olan LSTM ağlarıdır.

Veri seti olarak Borsa İstanbul Vadeli İşlem ve Opsiyon piyasasında işlem gören Endeks30 kontratına ait veriler kullanılmıştır. Bu verilerden belirli finansal indikatörler kullanılarak farklı değerler de elde edilmiş ve bu değerler girdilere dahil edilmiştir.

Uygulama sonucunda LSTM ağları, ileri beslemeli sinir ağı modeline göre daha yüksek sınıflama başarısı elde etmiştir. Beklendiği üzere sıralı zaman serisi verisinde yinelemeli sinir ağı modellerinin daha yüksek başarı oranı yakaladığı görülmüştür.

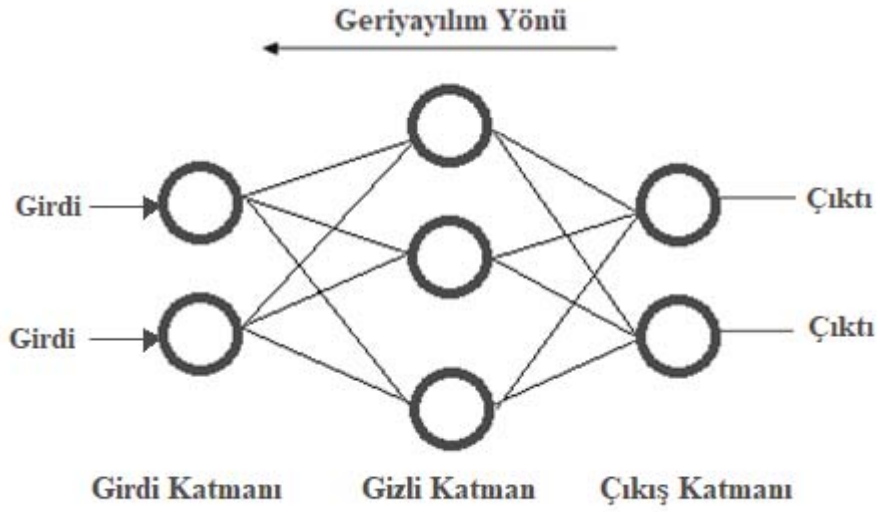
2. Yapay Sinir Ağları

2.1. İleri Beslemeli Yapay Sinir Ağları

2.1.1. Sinir Ağları Tarihçesi

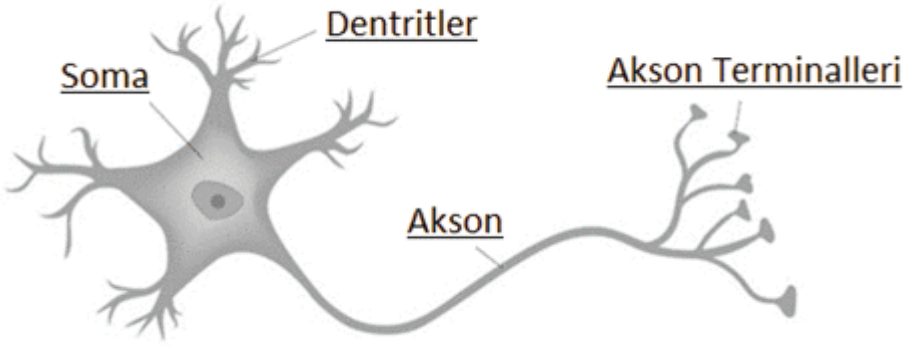
Sinir ağı teorisi Mcculloch ve Pitts [2] tarafından 1940'lı yılların başlarında geliştirilmiş olup, yöntemin farklı alanlarındaki uygulamaları ise zaman içerisinde literatüre kazandırılmış ve gelişmiştir. Yapay Sinir Ağları yaklaşımı, lineer olmayan karmaşık fonksiyonları modellemek için beynin düşünme sürecini taklit eden sistemlerin oluşturulmasında kullanılmaktadır. Bunun nedeni beynin, model tanıma, algılama ve hareket kontrolü gibi karmaşık işlemlerin gerçekleştirilmesinde bilgisayar çiplerinden daha hızlı ve daha verimli olmasıdır. Bu nedenle, beyin fonksiyonlarını taklit etmek sinir ağına, fiziksel akıl yürütme ve geleneksel matematiksel yaklaşımlarla çözülemeyen problemleri modellemek için çözümler üretme yeteneği verebilmektedir.

Araştırmacıların literatüre katkısıyla birlikte birçok farklı sinir ağı türü ortaya çıkmıştır. Bunlara örnek olarak; İleri Beslemeli Sinir Ağları (FFNN), Evrişimli Sinir Ağları (CNN) [3], Çekişmeli Üretici Sinir Ağları (GANN) [4], Uzun Süreli Kısa Dönemli Hafıza Ağları (LSTM) [5] , Nöral Turing Makinesi (NTM) [6] modelleri gösterilebilir.



Şekil 1.Yapay sinir ağlarına yönelik tipik bir geri yayılım şeması

2.1.2. Sinir Hücresi Modellemesi

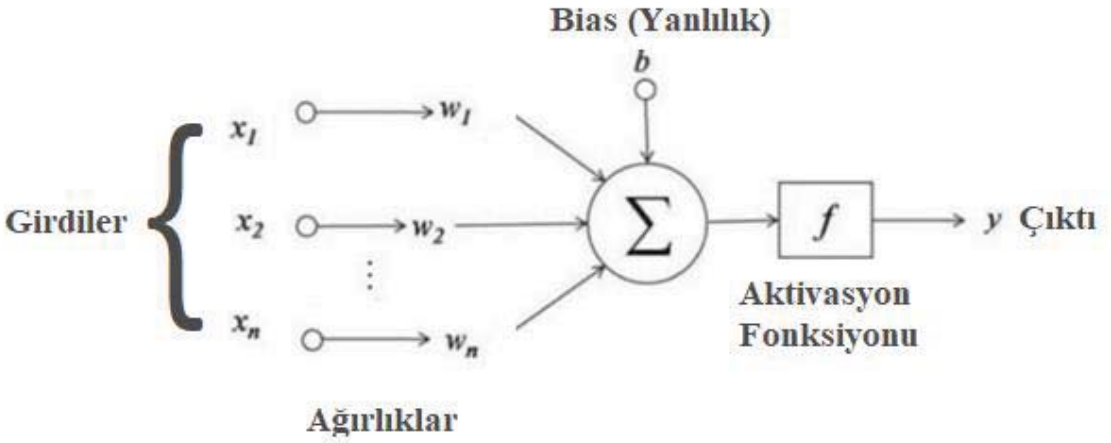


Şekil 2. Sinir hücresi anatomisi

Bir sinir hücresi sinir sistemini oluşturan en küçük yapı taşıdır ve canlılarda bilgi transferini sağlayan temel işlem birimidir. Ortalama bir insan beyninde 86 milyar sinir hücresi bulunmaktadır. Sinir hücreleri sinaps adı verilen ve elektriği ileten noktalardan birbirilerine bağlıdır. Sinir hücreleri temel olarak üzerine gelen elektrik sinyalini toplayan ve toplanan sinyal değerinin bir sonraki birime aktarılıp aktarılmayacağına karar veren birimlerdir. Elektrik sinyali ilk olarak dendritler vasıtasıyla sinir hücresine ulaşır. Bu dendritler birçok farklı sinir hücresinden gelen elektrik sinyallerini alan bağlantılardır. Dendritler vasıtasıyla sinir hücresine ulaşan elektrik sinyalleri bir toplam fonksiyonuna sokulur. Daha sonra toplam değeri bir aktivasyon fonksiyonuna verilerek, gelen sinyalin sonraki sinir hücrelerine aktarılıp aktarılmayacağı karar verilir. Aktivasyon fonksiyonu temelde belirli bir eşik değeri olan ve bu eşiğin geçilme durumuna göre açık veya kapalı (1-0) sonucunu veren bir anahtar görevi görmektedir. Sinyalin iletilmesi durumunda akson uçlarına aktarılan elektrik buradan diğer nöronlara geçer. Elektrik sinyali diğer nöronlara geçerken her akson ucunda belirli bir ağırlık değeriyle çarpılması da gerekir. Bir yapay sinir ağı modelinde amaçlanan, her bir akson ucundaki ağırlıkların optimal şekilde ayarlanarak, çıktının probleme en uygun çözümü içeren sonucu vermesini sağlamaktır.

Bir sinir ağında, katmanlar halinde organize edilmiş birbirine bağlı sinir hücreleri (veya işlem birimleri) bulunur. Tipik bir sinir ağı 3 katmandan oluşur. Bunlardan birincisi giriş katmanı, ikincisi gizli katman ve sonuncusu ise çıkış katmanı şeklinde tanımlanabilir (Şekil 1). Katmanlar arasındaki bağlantılar, giriş katmanına sunulan bilgilerin gizli katmanlardan çıkış katmanına kadar akmasına izin verir. Bu bağlantılar, sunulan girdiler

ve karşılık gelen çıktılar arasındaki ilişkinin doğasını öğrenmek için sinir ağı sistemi tarafından kullanılan ağırlıkların optimize edilmesiyle sürekli olarak güncellenir.



Şekil 3. Sinir hücresi modellemesi

Sinir ağının en temel işlem birimi olan sinir hücresini matematiksel olarak modellemek istersek, öncelikle her dendritten gelen elektrik sinyallerini bir ağırlık (w) değeriyle çarpmamız gerekir. Daha sonra bu değerler bir toplam fonksiyonuna aktarılır. Bulunan toplam değeri bir aktivasyon fonksiyonuna gönderilir ve aktivasyon fonksiyonu elektrik sinyalinin sonraki sinir hücrelerine iletilip iletilmeyeceğine karar verecektir. Aktivasyon fonksiyonundan çıkan sonuç sinir hücresinin bağlı olduğu diğer hücrelere aktarılır ve aynı işlemler tekrar edilir. Aktivasyon fonksiyonuna gönderilecek toplam değer aşağıdaki formül ile elde edilir.

$$\rightarrow f\left(b + \sum_{i=1}^{n} x_i w_i\right)$$

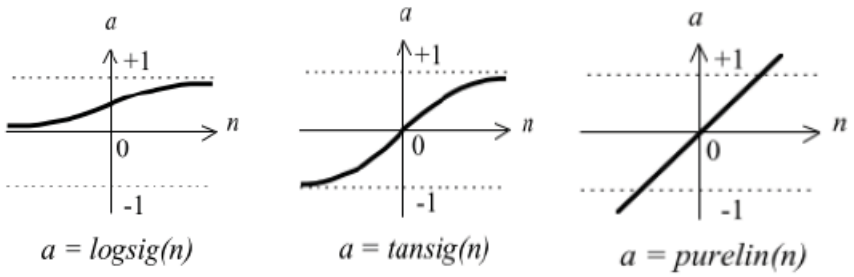
b: Yanlılık değeri, x : Girdi, w: Ağırlık, n : Girdi Sayısı

Verideki her bir nokta için sinir ağında gerçekleşen işlemler sırasıyla aşağıdaki şekilde verilmiştir.

1. Girdi nöronlarına veriden gelen değerler verilir.

2. Verilen değerler bir sonraki katmanda bulunan sinir hücrelerine ilgili bağlantının ağırlığı (w_i) ile çarpılarak gönderilir.
3. Karmandaki her bir nöron kendisine iletilen ağırlıkları toplar.
4. Her bir nöron toplanan değeri bir aktivasyon fonksiyonuna gönderir.
5. Aktivasyon fonksiyonu gelen değerin aktarılıp aktarılmayacağına karar verir.
6. Bu şekilde girdi nöronlarından gelen veri ileriye doğru tüm katmanlara gönderilir.
7. Çıkış katmanında elde edilmiş değer ile olması gereken değer karşılaştırılır ve bir kayıp değeri elde edilir. Bu kayıp değerini hesaplamak için bir kayıp fonksiyonu kullanılır.
8. Bulunan kayıp değerini azaltmak için bir optimizasyon algoritması ile sinir ağındaki tüm ağırlıklar güncellenir.
9. İstenilen kriterler gerçekleşene veya belirli bir adım sayısına gelinene kadar bu işlemler tekrar edilir.

2.1.3. Aktivasyon Fonksiyonu



Şekil 4. Yapay sinir ağlarında işlem ünitelerinde kullanılan transfer fonksiyonları

Her nöronun içinde, girilen ve toplanan verileri işleyerek nöron çıktısına çeviren ve otomatik olarak sonraki katmanda bulunan nöron için bir girdi verisi haline getiren bir **aktivasyon fonksiyonu** kullanılmaktadır. Sinir hücresinde kullanılmak üzere farklı aktivasyon fonksiyonları mevcuttur. En yaygın fonksiyonlar başlıca; Logsigmoid (Logsig), Tanjant Sigmoid (Tansig) ve doğrusal aktivasyon fonksiyonu şeklinde sıralanabilir (Şekil 4) [7]. Aktivasyon fonksiyonlarında genellikle doğrusal olmayan fonksiyonlar kullanılır. Bunun en önemli sebebi yapay sinir ağları modeline esnek bir hesaplama altyapısı sağlayabilmek ve çözüm bulunması istenen probleme doğasına yakın bir çözümle yaklaşabilmektir. Eğer aktivasyon fonksiyonu doğrusal bir fonksiyon olursa,

elde edeceğimiz çıktı, girdi vektörünün doğrusal dönüşümünden başka bir şey olamayacaktır.

2.1.4. Kayıp Fonksiyonu

Kayıp fonksiyonu sinir ağı tarafından tahmin edilen çıktı değeri ile olması gereken çıktı değeri arasındaki hatayı hesaplayan ve sinir ağının optimize edilmesi için gerekli kayıp değerini bulan fonksiyondur. Sinir ağı modelleri çözümledikleri problemin çeşidine göre farklı kayıp fonksiyonları kullanılabilir. Örneğin çözümlenen problem bir sınıflama problemi olduğunda, çıktı değeri 1 veya 0 olarak bulunmalı ve buna göre tahmin edilen sınıf değeri oluşmalıdır. Bu durumda "İkili Çapraz Entropi" kayıp fonksiyonu uygun bir seçim olacaktır. Fakat bir zaman serisi problemi ele alındığında "Ortalama Karesel Hata" gibi uzaklık bazlı bir kayıp fonksiyonu kullanılması uygun olacaktır. Aşağıda uzaklık bazlı ve kategorik kayıp fonksiyonlarına ilişkin formüller yer almaktadır.

Ortalama karesel hata formülü;

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2$$

N: Veri sayısı, f_i: Hesaplanan sonuç değeri, y_i: Gerçek sonuç değeri

Kategorik çapraz entropi formülü;

$$CCE(p,t) = -\sum_{c=1}^{C} t_{o,c}\log(P_{o,c})$$

C: Sınıf sayısı, $t_{o,c}$: Belirleyici çarpan (0-1), c: Sınıf

2.1.5. Optimizasyon Algoritması

Kayıp fonksiyonu ile hesaplanan hata değerinin düşürülebilmesi için sinir ağında bulunan ağırlık değerlerinin doğru şekilde güncellenmesi gerekir. Her bir ağırlığın hatayı azaltacak biçimde doğru yönde güncellenebilmesi için bir optimizasyon algoritması kullanılmalıdır. Bu algoritma her bir adımda hatayı azaltarak, kayıp değerini minimize etmeyi sağlayan bir optimizasyon yöntemi olmalıdır. Yapay sinir ağları genellikle geri beslemeli bir öğrenme sürecine sahiptir. Yani çıktıdan hesaplanan hata değeri ağdaki her

bir katman için geriye doğru ağırlıkların güncellenmesinde kullanılır. Geri besleme her adımda çıktı değerinden hesaplanan hatayı minimize eden bir algoritmadır.

Sinir ağındaki nöronlar arasındaki her bir bağlantı için ağırlıklar her adımda aşağıdaki formülle yeniden hesaplanır.

$$w_i^+ = w_i - n * \frac{\partial E_{toplam}}{\partial w_i}$$

E : Hata, w : Ağırlık, n : Öğrenme oranı

Her yeni ağırlık yeniden hesaplanırken amaç, ilgili ağırlıktaki (w_i) değişimin, toplam hata (E_{toplam}) değerini ne şekilde etkilediğini bulmaktır. Böylece toplam hatayı azaltacak şekilde w_i ağırlığını güncellemek mümkün olacaktır. Bunun için toplam hatanın w_i ağırlığına göre kısmi türevi bulunmalıdır.

$$\frac{\partial E_{toplam}}{\partial w_i} = \frac{\partial E_{toplam}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_i}$$

E: Hata, out: Çıktı, w: Ağırlık, net: Ağ girdisi

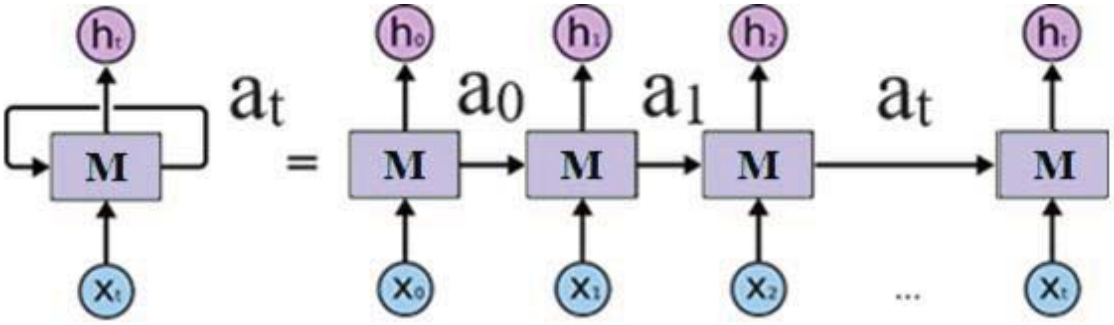
Eldeki her bir veri için veya önceden belirlenmiş bir adım sayısına erişilene kadar tüm ağırlıklar yukarıdaki formülle yeniden hesaplanır ve güncellenir. Her güncellemede kayıp fonksiyonundan elde edilen kayıp değerinin düşmesi beklenir. Kayıp değerinin düşmesi, veri seti ve beklenen çıktılar arasında bir ilişki bulunduğunu ve yapay sinir ağı ağırlıklarının bu ilişkiye uygun şekilde güncellendiğini göstermektedir. Adım sayısının sonuna gelinmesine rağmen kayıp değerlerinde bir düşüş gözlenmemesi, ver seti ve beklenen çıktı arasında bir ilişki bulunmadığı veya yapay sinir ağı mimarisinin bu ilişkiyi doğru bir şekilde yakalamaya uygun olmadığı şeklinde yorumlanabilir.

2.2.Yinelemeli Sinir Ağları

2.2.1. Yinelemeli Sinir Ağları Giriş ve Tarihçe

1980'li yıllarda temelleri atılan bir yapay sinir ağı modeli olan özyinelemeli sinir ağları (RNN) özellikle doğal dil işleme (NLP) alanında dizi işleme özelliği olmasından dolayı sıklıkla tercih edilmektedir. Konuşma, metin, müzik, video gibi sıralı akışa sahip verilerde başarılı sonuçlar üreten RNN yapısında ileri beslemeli sinir ağı yapısından farklı olarak ilerideki katmanlardan girişlere geri besleme yapılmaktadır. Ağ bu sayede önceki bilgilere bağlı bir çıkış üretmektedir. [8]

Ağın yapısı, standart birçok katmanlı sinir ağına benzemekle birlikte ağın gizli katmanları arasında geçmiş zamandan gelen verilerle ilişkili bağlantılara izin veriliyor. Bu bağlantılar aracılığıyla model geçmiş hakkında bilgi tutabiliyor ve verilerde birbirinden uzak olaylar arasındaki zamana bağlı korelasyonları keşfedebiliyor. [9]



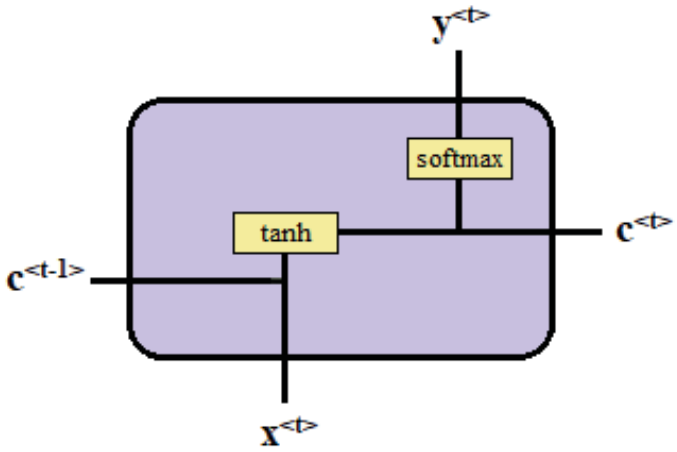
Şekil 5. Yinelemeli sinir ağı modeli

Şekil 5'te verilen yinelemeli sinir ağı modelinde dikkat edilmesi gereken, ilk bölümde verilen sinir ağı modelinden farklı olarak, burada zamana dayalı bir gösterim yapılmış ve verideki her bir noktayı temsil eden x_t girdileri ve h_t çıktıları gösterilmiştir. Yani şekilde ağın çalıştırılırken her t zamanda yapacağı işlemler kesit olarak verilmiştir. "M" ise sinir hücresini ifade eder. Her bir noktada "M" ileri beslemeli sinir ağlarında olduğu gibi bir çıktı üretir (h_t). Buna ek olarak yinelemeli sinir ağlarında bulunan öğrenilmiş bilgileri içeren bir "c" çıktısı daha üretir. Şekilde görüldüğü üzere herhangi bir x_t girdisi kendisinden önceki x_{t-1} girdisinden elde edilen "c" vektörünü hesaplamaya dahil etmektedir. "c" geçmişten öğrenilmiş bilgileri içeren bir ağırlık vektörüdür. Her yeni yeni x_t verisinde, h_t sonucunun hesaplanması için, x_t girdi vektörü yanında "c_t" vektörü de

kullanılmaktadır. Böylece veride ileri gidildikçe geçmiş verilerden elde edilmiş olan sonuçlar da tekrar girdi olarak kullanılabilmektedir. Bu durum yinelemeli sinir ağlarının sıralı verilerde ve zaman serilerinde ileri beslemeli sinir ağlarına göre daha iyi sonuç vermesini sağlamaktadır.

Örneğin, bir cümle içinde bir sonraki kelimeyi tahmin etmek için, o anki kelimeden önce hangi sözcüklerin geldiğini bilmek gerekmektedir. RNN mimarisinin yinelenen olarak adlandırılmasının sebebi, bir dizinin her öğesi için (cümledeki kelimeler gibi) aynı görevi önceki çıktılara bağlı olarak yerine getirmesidir. [10]

2.2.2. Çalışma Mantığı ve Teorisi



Şekil 6. Yinelemeli sinir ağı modeli

Yukarıdaki şekilde "x" girdi vektörü, "y" çıktı vektörü ve "c" yinelemeli sinir ağlarına özel olan geçmiş verilerden elde edilen bilgileri saklayan bir vektördür. Yalnızca "x" ve "y" vektörleri düşünüldüğünde bu model ileri beslemeli bir yapay sinir ağıdır. Burada farklı olan her bir x_t girdisi için önceki verilerden elde edilen sonuçları içeren c_{t-1} vektörü ağa dahil edilmektedir. Aşağıdaki formül ile her bir x noktası için "a" vektörünün ve "y" çıktı vektörünün değerleri hesaplanır.

$$c^{<t>} = g(w_a . c^{<t-1>} + w_a . x^{<t>} + b_a)$$

Formülde $c^{<t>}$ vektörü t zamandaki ve t zamandan önce öğrenilen bilgileri içeren vektördür. $c^{<t>}$ vektörünü hesaplamak için t-1 verisinden edilen $c^{<t-1>}$ vektörü ve güncel

girdi olan $x^{<t>}$ vektörü kullanılır. Böylece önceki tüm veriden elde edilen bilgileri içeren yeni bir $c^{<t>}$ vektörü elde edilmiş olur.

$$y^{<t>} = g\left(w_y . c^{<t>} + b_y\right)$$

Daha sonra $y^{<t>}$ çıktısının elde edilebilmesi için önceki formülde hesaplanan $a^{<t>}$ bilgi vektörü formüldeki yerine yazılır. Bu şekilde ağ çıktısı (y) her bir x girdisi için hesaplanmış olur.

Formüllerde "g" fonksiyonu aktivasyon fonksiyonunu ifade etmektedir. Formüllerde aynı gösterilmiş olsa da genellikle $a^{<t>}$ vektörünün hesaplanması için "tanh" aktivasyon fonksiyonu tercih edilir. Bunun en önemli sebebi öz yinelemeli sinir ağlarının en büyük problemlerinden biri olan kaybolan gradyan durumunu önlemeye uygun olmasıdır. Bu probleme sonraki bölümde detaylı olarak değinilecektir. Çıktı değeri olan $y^{<t>}$ hesaplaması için ise çözülmesi istenen probleme uygun herhangi bir aktivasyon fonksiyonu seçilebilir.

2.2.3. Kaybolan Gradyan problemi

Teoride yinelemeli sinir ağları basit ve güçlü bir model olsa bile, pratikte doğru bir şekilde eğitilmesi zordur. Modelin bu hantallığının başlıca nedenleri Bengio ve diğerleri (1994) tarafından da açıklandığı gibi kaybolan gradyan ve patlayan gradyan problemleridir. [9] [11]

Bu problemin ardında yatan sebebi anlamak için basit bir örnek vermek gerekirse; "w" skalar bir ağırlık değişkeni olsun ve modelimiz "w" değişkeninin birçok kez çarpımını içeren aşağıdaki eşitlik olsun;

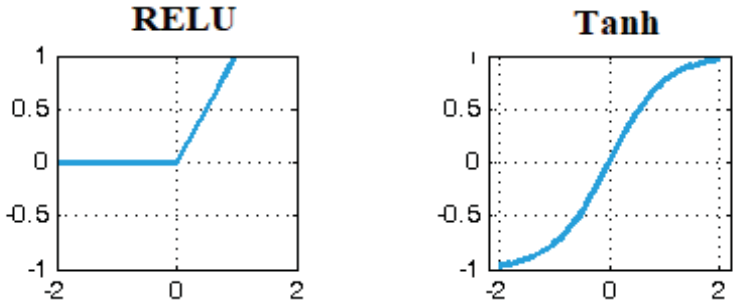
$$x_{t+1} = w . x_t$$

Buna göre "t" zaman sonraki sonuç, modelin ilk haline (t_0) göre w^t ile çarpılmış olacaktır. Bu durumda "t" zaman içinde ileri gidildikçe $|w|>1$ ise ifade çok büyük bir sayıya gidecek ve "patlayan" olarak tanımlanacaktır. Aynı şekilde $|w|<1$ olması durumunda da ifade sıfıra yakınsayacak ve "kaybolan" olarak tanımlanacaktır.

Bu durum yinelemeli sinir ağlarında geçmişten gelen bilgileri içeren matristeki ağırlıkların her t zamanda çarpılmasıyla ortaya çıkmaktadır. Sonuç olarak ağırlıklar ya sıfıra yakınsayacak ya da çok büyük değerlere ulaşacak ve model doğru bir şekilde eğitilemeyecektir.

Kaybolan gradyan ve patlayan gradyan sorunu yinelemeli sinir ağlarının teoride basit ve başarılı bir model olmasına karşın pratikte ciddi bir soruna sahip olmasına yol açmaktadır. Bu sorunun çözümü için yinelemeli sinir ağlarında bazı modifikasyonlar yapılmış ve daha sonra bu sorunlara çözüm olan Uzun Süreli Kısa Dönemli Hafıza (LSTM) ve Geçitli Tekrarlayan Birim (GRU) gibi modeller ortaya çıkmıştır.

Bölüm 3.2.'de de bahsedildiği gibi yinelemeli siniri ağlarında geçmişten gelen bilgileri içeren vektörün hesaplanmasında "tanh" aktivasyon fonksiyonu kullanılmaktadır. Tanh aktivasyon fonksiyonunun alabileceği değerler 1 ve -1 arasında sabitlenmiş olduğu için "t" zamanda ileri gidildikçe ağırlıkların çarpımlarından elde edilen değerlerin çok büyük veya sıfıra yakın hale gelmesi diğer aktivasyon fonksiyonlarına göre daha zor olmaktadır. Aşağıdaki görselde tanh aktivasyon fonksiyonu ve relu aktivasyon fonksiyonları verilmiştir.



Şekil 7. Relu ve Tanh Aktivasyon Fonksiyonları

2.3. Geçitli Tekrarlayan Birim (GRU)

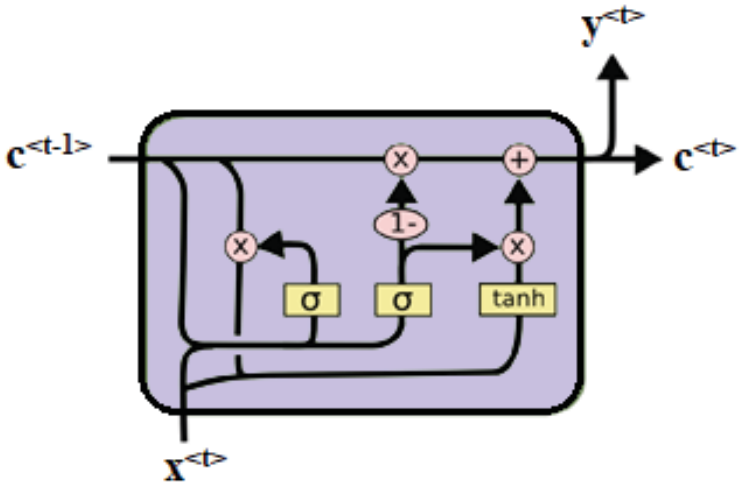
2.3.1. GRU Giriş ve Tarihçe

Yinelemeli sinir ağlarının en büyük problemi olan kaybolan veya patlayan gradyan sorunu, yinelemeli sinir ağlarının verideki uzun süreli ilişkileri öğrenebilmesini

oldukça zor hale getiriyor. Bu nedenle verideki uzun süreli ilişkileri başarıyla belleğinde tutabilecek ve bu öğrenilmiş bilgiyi verideki diğer noktalara uygulayabilecek modellerin geliştirilmesi gerekiyordu. LSTM modeli bu soruna çözüm getirerek yinelemeli sinir ağlarının sıralı zaman serilerindeki başarısını oldukça arttırdı. LSTM modelinden daha sonra ortaya çıkmasına rağmen daha basit ve hesaplama süresi daha kısa bir yapıya sahip olan GRU (Geçitli Tekrarlayan Birim) modeli ise oldukça popüler bir model haline geldi. İlk olarak 2014 yılında Cho ve diğerleri tarafından önerilen model [12], aynı yıl yayınlanan "Geçitli Tekrarlayan Birimle Dizi Modellemenin Deneysel Değerlendirilmesi" gibi birçok uygulamada başarısını kanıtladı. [13]

GRU tipik bir yinelemeli sinir ağından farklı olarak "t" zamanda öğrenilen verinin, öğrenilmiş bilgiyi saklayan vektöre yazılıp yazılmayacağına karar veren bir kapıya (gate) sahip olmasıdır. Böylece işe yarayabilecek bilgileri belleğinde daha uzun süre tutabilirken, işe yaramayacak verileri çarpıma dahil etmeyerek kaybolan gradyan sorununun da önüne geçmiş olacaktır. Bunun sonucunda zaman serisindeki uzun süreli ilişkileri öğrenebilen ve daha iyi performans veren bir model ortaya çıkmış olur.

2.3.2. Çalışma Mantığı ve Teorisi



Şekil 8. GRU hücresi

Tipik bir yinelemeli sinir ağında olduğu gibi her GRU hücresi girdi olarak $x^{<t>}$ girdi vektörünü ve geçmiş bilgileri içeren $c^{<t-1>}$ vektörünü alır. Çıktı olarak ise $y^{<t>}$ ve $c^{<t>}$ vektörlerini bir sonraki hücreye gönderir. Tipik bir yinelemeli sinir ağından farklı olarak $c^{<t>}$ çıktısı hesaplanırken $c^{<t-1>}$ vektöründen gelen bilginin güncellenip

güncellenmeyeceğine karar veren bir fonksiyon bulunmaktadır. Tipik yinelemeli sinir ağlarında $c^{<t>}$ vektörü herhangi bir koşul olmaksızın güncellenmekte ve bu durum çarpım sayısını arttırdığı için kaybolan gradyan sorunu ortaya çıkmaktadır.

$$h^{<t>} = c^{<t>}$$

Öncelikle GRU formülü için $c^{<t>}$ vektörünü $h^{<t>}$ olarak ifade edeceğiz. Şu an bu iki değişken birbirine eşit olacak fakat LSTM bölümüne geçildiğinde bu değişkenler farklı olacağı böyle bir ifade kullanalım.

$$\tilde{h}^{<t>} = \tanh\left(w_c[h^{<t-1>}, x^{<t>}] + b_c\right)$$

İlk olarak girdi vektörü ($x^{<t>}$) ve eski bilgi vektörünü ($h^{<t-1>}$) kullanarak $h^{<t>}$ vektörünün yeni "aday" değerini hesaplayalım. Aday değeri olmasının sebebi, ileri beslemeli yapay sinir ağlarındaki gibi hesaplanan değerin hemen yeni bilgi vektörü olarak eşitlenmemesidir. Bunun yerine bu aday değer bir kapı fonksiyonuna gönderilir ve bu fonksiyon bilgi vektörünün aday değerle güncellenip güncellenmeyeceğine karar verir. Bu aşamada önceki bölümlerde açıklandığı gibi yine tanh aktivasyon fonksiyonu kullanılır

$$g_u = \sigma\left(w_u[h^{<t-1>}, x^{<t>}] + b_u\right)$$

GRU arkasındaki en büyük fikir olan "kapı" veya "geçit" fonksiyonu, 0 ile 1 arasında bir sonuç döndürür ve yeni $h^{<t>}$ vektörünün hesaplanmasında aday $h^{<t>}$ vektörünün ne oranda kullanılacağına karar verir. Başka bir ifadeyle geçmişten öğrenilmiş bilgileri saklayan vektörün, yeni veriden öğrenilmiş bir bilgiyle ne oranda değiştirileceğine karar verir. Böylece uzun süreli ilişkiler içeren bir veride geçmiş bilgileri saklayan vektörün doğru bir şekilde tutulmasını ve ileriye aktarılmasını sağlar. Aktivasyon fonksiyonu olarak 0 ile 1 arasında değer döndüren Sigmoid aktivasyon fonksiyonu kullanılır.

$$h^{<t>} = g_u . \tilde{h}^{<t>} + (1 - g_u) . h^{<t-1>}$$

Son olarak yeni bilgi vektörünü oluşturmak için kapı fonksiyonundan elde edilen değer, aday $h^{<t>}$ vektörü ile çarpılır ve (1-Kapı değeri) ile toplanır. Aday değer çarpımı ve (1- Kapı değeri) toplandığı için çarpımdan dolayı ortaya çıkabilecek çok büyük ve çok küçük sayılar engellenmiş olur ve kaybolan gradyan sorunu ortaya çıkmaz. Aynı zamanda $h^{<t>}$ içinde yalnızca işe yarayacak bilgilerin tutulması sağlanır.

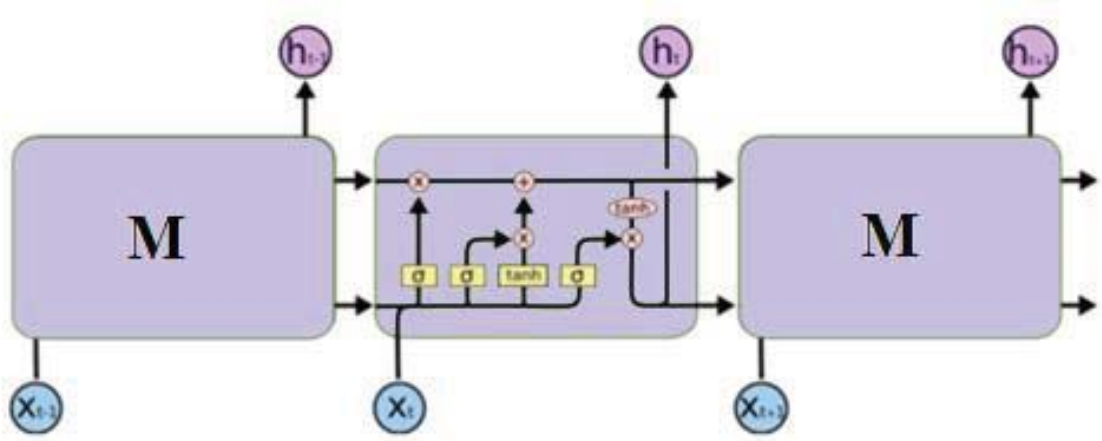
2.4. LSTM

2.4.1. LSTM Giriş ve Tarihçe

Uzun Süreli Kısa Dönemli Hafıza (LSTM) kavramı ilk olarak 1997 yılında Sepp Hochreiter ve Jürgen Schmidhuber tarafından yayınlanan makale ile ortaya çıkmıştır. Makalede yinelemeli sinir ağlarında bulunan geçmiş verilerden öğrendiği bilgileri belleğine alarak ağa dahil etme özelliğinin her ne kadar oldukça iyi bir fikir olduğunu belirtse de, pratikte bu algoritmaların ileri beslemeli bir sinir ağına göre avantaj sağlamadığını ifade etmiştir.

Yinelemeli sinir ağlarında kısa süreli belleğe ne konulacağını öğrenmek için en yaygın olarak kullanılan algoritmalar, özellikle girdiler ve karşılık gelen öğretici sinyaller arasında zaman gecikmesi uzun olduğunda çok fazla zaman alır veya hiç iyi çalışmaz. Teorik olarak büyüleyici olmasına rağmen, mevcut yöntemler, örneğin, sınırlı zaman aralıklı ileri beslemeli ağlarda geri-beslemeli sinir ağlarına kıyasla açık pratik avantajlar sağlamamaktadır. [5]

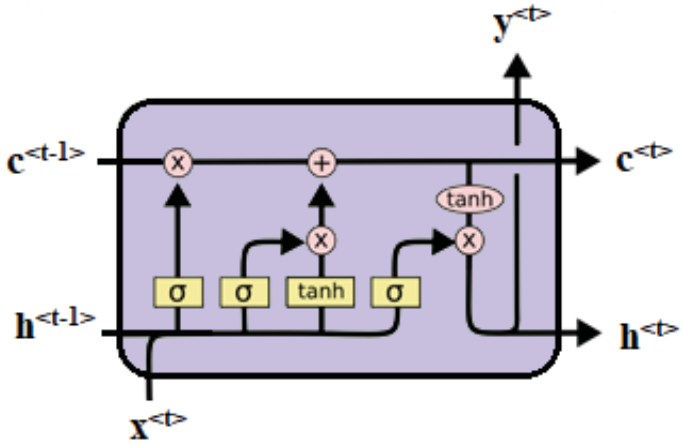
Hochreiter ve Sepp makalesinde LSTM modelini tanıtırken; “Gürültülü, sıkıştırılamaz giriş dizileri durumunda bile 1000 adımı aşan zaman aralıklarında kısa süreli ilişkileri kaybetmeden bağlantı kurmayı öğrenebilir.” olarak ifade etmiştir. Bu durumda yinelemeli sinir ağlarının en büyük problemi olan kaybolan veya patlayan gradyan sorunu çözülmüş ve uzun zaman serilerinde beklenen şekilde geçmiş dönemde elde edilen bilgileri ağa başarılı bir şekilde dahil eden bir model elde edilmiştir.



Şekil 9. LSTM Modeli

Şekil 8'de LSTM modelindeki ilk modüle bakıldığında aslında Yinelemeli Sinir Ağları bölümünde bulunan şekil ile neredeyse aynı olduğu görülmektedir. Şekilde, x_t girdi vektörü, "M" sinir hücresi ve h_t sinir hücresinin çıktısıdır. Burada farklı olarak ortada bulunan modüle bakıldığında sinir ağı modelinin çıktıyı elde ederken birçok farklı yol kullandığı göze çarpacaktır. Bu yollar ilk olarak karışık gözükse de yinelemeli sinir ağlarındaki kaybolan gradyan problemine çözüm olmaktadır. Sonraki bölümde her bir LSTM ağında çıktı değerlerinin nasıl hesaplandığı incelenecektir.

2.4.2. Çalışma Mantığı ve Teorisi



Şekil 10. LSTM hücresi

LSTM ağları, basit yinelemeli sinir ağları ve GRU modeline göre daha karmaşık bir yapıya sahiptir. Bu modelde 3 farklı kapı fonksiyonu bulunur. Kapı fonksiyonları

öğrenilmiş bilginin ne kadarının unutulacağını, yeni bilginin öğrenilmiş bilgiyi ne kadar değiştireceğini ve bu değişimlerin bir sonraki hücreye ne şekilde aktarılacağına karar veren fonksiyonlardır. Bu sayede karmaşık uzun süreli ilişkiler ağ tarafından başarılı bir şekilde öğrenilebilir.

GRU aksine bu modelde $c^{<t>}$ ve $h^{<t>}$ iki farklı vektördür ve LSTM hücresinden iki farklı vektör olarak bir sonraki hücreye aktarılırlar.

$$\tilde{c}^{<t>} = \tanh\left(w_c[h^{<t-1>}, x^{<t>}] + b_c\right)$$

Aday $c^{<t>}$ vektörü GRU modeline benzer şekilde yukarıdaki formülle hesaplanır.

$$g_u = \sigma\left(w_u[h^{<t-1>}, x^{<t>}] + b_u\right)$$

Aday $c^{<t>}$ vektörünün, geçmiş bilgileri içeren $c^{<t>}$ vektörünü ne oranda değiştireceğine karar veren güncelleme kapısı fonksiyonu yukarıdaki formülle hesaplanır.

$$g_f = \sigma\left(w_f[h^{<t-1>}, x^{<t>}] + b_f\right)$$

Geçmiş veriden gelen bilgilerin ne oranda unutulması gerektiğine karar veren unutma kapısı fonksiyonu, $c^{<t-1>}$ vektörünün yeni değerinin hesaplanmasında kullanılır.

$$g_o = \sigma\left(w_o[h^{<t-1>}, x^{<t>}] + b_o\right)$$

Son kapı olan çıktı kapısı, bir sonraki hücreye aktarılacak $h^{<t>}$ vektörünün hesaplanmasında kullanılır.

$$c^{<t>} = \left(g_u . \tilde{c}^{<t>}\right) + \left(g_f . c^{<t-1>}\right)$$

Yeni $c^{<t>}$ vektörü hesaplanırken, öncelikle aday $c^{<t>}$ vektörü ile güncelleme kapısı çarpımı bulunur. Daha sonra $c^{<t-1>}$ vektörü ile unutma kapısı çarpımı bulunur ve iki çarpım toplanarak yeni $c^{<t>}$ değeri elde edilir.

$$h^{<t>} = g_o . \tanh\left(c^{<t>}\right)$$

Son olarak yeni $h^{<t>}$ vektörünün hesaplanması için yeni $c^{<t>}$ vektörünün tanh aktivasyon fonksiyonu çıktısı ve çıktı kapısı çarpımı yukarıdaki formülle yapılır.

Sonuçta elde edilen model gelişmiş uzun süreli bellek kapasitesi, unutma yeteneği ve uzun süreli ilişkileri yorumlayabilme özelliği bulunan, sıralı verilerde üstün performans gösteren sağlam bir model halini almıştır.

3. Yapısal Kırılma ve Değişim Noktası Tespiti

Uygulamaya geçmeden önce sinir ağlarından bağımsız bir konu olan yapısal kırılma ve değişim noktası testlerine değinilecektir. Bunun sebebi, uygulamada kullanılacak verilerin borsa verisi olmasından dolayı ortaya çıkan ani fiyat hareketlerinin ve dış müdahalelerin modelin başarısını olumsuz etkilememesini sağlamaktır.

3.1. CHOW Testi

Chow testi 1960 yılında Gregory C. Chow tarafından bulunan, verideki herhangi bir noktada yapısal bir kırılmanın veya değişimin olup olmadığını test edilmesini sağlayan bir ekonometri modeli olarak ortaya çıkmıştır. Bu yöntem özünde test edilecek noktadan önceki ve sonraki veriler için iki regresyon doğrusu oluşturur ve bu iki doğrunun arasındaki eğim farkını inceleyerek bir değişim noktası olup olmadığını test eder.

$$CHOW = \frac{(RSS_p - (RSS_1 + RSS_2)) \,/\, k}{(RSS_1 + RSS_2)/(N_1 + N_2 - 2k)}$$

RSS_p = Tüm veriye ait regresyon doğrusu

RSS_1 = Kırılma öncesi regresyon doğrusu

RSS_2 = Kırılma sonrası regresyon doğrusu

Genellikle iki ilişkinin tamamen aynı olduğunu varsaymak için ekonomik bir gerekçe yoktur. İki periyotta veya iki grup için ilişkilerin sadece bir kısmının aynı olduğunu varsaymak daha mantıklı olabilir. Belki belirli bir gıda ürününe olan talebin fiyat esnekliği II. Dünya Savaşı'ndan bu yana değişmemişken, gelir esnekliği değişmiştir. Belki

iki grup firmanın yatırımı aynı şekilde kârdan etkilenir, ancak likit varlıklardan etkilenmez. İstatistiksel olarak, iki regresyondaki katsayı alt kümelerinin eşit olup olmadığını sorguluyoruz. [14]

3.2. Cezalandırılmış Değişim Noktası Tespiti (PELT) Yöntemi

Değişim noktası tespitinde kullanılan yöntemler genellikle bir optimizasyon algoritmasına dayanmaktadır. Turong, Oudre ve Vayatis tarafından 2019 yılında yapılan bir çalışmada bu yöntemlerden bazıları incelenmiştir. İncelenen yöntemler ile ilgili Python yazılım dilinde kodlanmış bir kütüphane de yazarlar tarafından sunulmuştur.

Bir değişim noktası tespit algoritması temelde üç bölümden oluşur. Bu bölümler; Maliyet fonksiyonu, arama metodu ve kısıtlardır. Maliyet fonksiyonu bir "homojenlik" ölçüsü olarak da tanımlanabilir. Homojen kelimesi burada bir veride bulunan değişim noktası sayısını ifade etmektedir. Eğer veride değişim noktası bulunmuyorsa bu "homojen" bir veri olduğu anlamına gelir. Veride bir veya daha fazla değişim noktası bulunuyorsa bu "heterojen" olarak tanımlanabilir. Maliyet fonksiyonu homojenlik değerini maksimize eden bir fonksiyon olacaktır.

Arama algoritması aşağıdaki problemleri çözümleyecek optimizasyon algoritmasıdır.

Problem 1: Değişim noktası sayısı (K) bilindiğinde;

$$\min_{|T|=K} V(T)$$

Problem 2: Değişim noktası sayısı (K) bilinmediğinde;

$$\min_{T} V(T) + pen(T)$$

$\tilde{T}$ en iyi segmentasyon kabul edildiğinde V(T) kriterinin minimize edicisidir. K ise değişim noktası sayısıdır. Değişim noktası sayısı bilinmediğinde pen(T) segmentasyonun karmaşıklık derecesi ölçüsüdür.

PELT algoritması ise değişim noktası sayısı bilinmediğinde arama problemine çözüm bulan bir algoritmadır. Uygulamada kullanılacak veride değişim noktaları önceden belirlenmeyeceği için PELT optimizasyon yöntemi tercih edilmiştir.

PELT algoritması adımları aşağıdaki gibidir;

Girdiler: $\{y_t\}_{t=1}^{T}$ verisi, c(.) maliyet fonksiyonu, β penaltı değeri

Tanımla: Z bir (T + 1) uzunluğunda dizi; Z[0] ← −β

Tanımla: L[0] ←∅

Tanımla: χ ←{0}

for t = 1,...,T do

$\hat{t} \leftarrow argmin_{s \in x} [Z(s) + c(y_{s \ldots t}) + \beta]$

$\mathrm{Z(t)} \leftarrow [Z(\hat{t}) + c(y_{\hat{t} \ldots t}) + \beta]$

$\mathrm{L(t)} \leftarrow L(\hat{t}) \cup \{\hat{t}\}$

$x \leftarrow \{s \in x : Z(s) + c(y_{s \ldots t}) \leq \mathrm{Z(t)}\} \cup \{t\}$

[15]

4.Uygulama

4.1. Giriş

Bu çalışmada LSTM, GRU ve ileri beslemeli yapay sinir ağı modelleri kullanılarak Borsa İstanbul Vadeli İşlem ve Opsiyon piyasalarında işlem gören Endeks 30 vadeli kontratında yükseliş ve düşüş trendlerinin tahmini yapılacaktır. Veri olarak Endeks 30 vadeli kontratının seçilmesinin nedeni, bu kontratın Borsa İstanbul 30 endeksinde işlem gören hisselerle hesaplanıyor olmasıdır. Yani tek bir hisse senedine dayalı değil, Bist30 endeksi içindeki tüm hisselere dayalı olarak fiyatı belirlenmektedir. Bu durum tek bir hissede meydana gelebilecek dış etkiler ve manipülasyonlardan daha az etkileneceği anlamına gelmektedir. Bu nedenle bir makine öğrenmesi modeli için daha uygun bir veri olduğuna karar verilmiştir.

Şekil 11. Endeks 30 Vadeli Kontratı Grafiği

Yukarıdaki görselde Endeks 30 Vadeli Kontratına ait günlük periyotta bir mum grafiği yer almaktadır. Her bir mum çubuğu kendi içinde Açılış, Kapanış, Yüksek ve Düşük değerlerine sahiptir. Her bir mum çubuğu içinde belirlenen periyotta kontratın aldığı ilk değer, son değer, en yüksek değer ve en düşük değeri ifade etmektedir. Modelde girdi olarak 5 dakikalık periyottaki veriler kullanılacaktır. Bunun nedeni uzun periyotlarda meydana gelebilecek dış etkilerin kısa periyotlarda modeli daha az etkilemesidir. Dolayısıyla daha kısa vadeli periyotlar kantatif bir model için doğru bir seçim olacaktır.

Bu çalışmada çıktı değerleri ise ALIM veya SATIM anlamına gelen bir sınıf değeri olacaktır. Eğitim verisinde bu sınıf değerlerinin nasıl hesaplandığına aşağıdaki bölümlerde yer verilmiştir. Veri hazırlama ve ön işleme aşamasından sonra LSTM modeli, GRU modeli ve ileri beslemeli bir yapay sinir ağı modeli kurulacaktır. Bu modellere ait konfigürasyonlara da aşağıdaki bölümlerde değinilmiştir. Kurulan modeller eğitim verisi ile eğitildikten sonra test verisi ile sınanacak ve modellerin performansları karşılaştırılacaktır.

Uygulama yapılırken Python yazılım dili kullanılmıştır. Sinir ağları, GRU ve LSTM ağları için Google tarafından desteklenen Keras ve Tensorflow kütüphaneleri kullanılmıştır. Bunun dışında verideki değişim noktalarının çıkarılması için de iki farklı kaynaktan yararlanılmıştır. Chow testi için Joshua Loong tarafından geliştirilmiş python kütüphanesi kullanılmıştır. PELT değişim noktası tespiti algoritması için Charles Turong,

Laurent Oudre ve Nicolas Vayatis tarafından geliştirilen Ruptures kütüphanesi kullanılmıştır.

4.2. Verinin Hazırlanması

Veri seti olarak Endeks 30 yakın vade kontratında Ekim 2015 ile Mart 2020 arasındaki 5 dakikalık veriler elde edilmiştir. Bu veri setinde toplamda 5 dakikalık 115876 adet mum grafiği verisi bulunmaktadır. Bu verilerin %80'i eğitim ve %20'si test verisi olarak ayrılmıştır. Eğitim verisi için gerekli olan çıktı değerleri yükseliş ve düşüş sınıflarını temsil edecek şekilde 1 ve 0 olarak ayrılacaktır. Bunu yaparken aşağıda algoritması verilen, "ZigZag" yöntemi adı verilen bir hesaplama ile verideki düşüş trendleri ve yükseliş trendleri işaretlenmiştir.

ZigZag algoritması;
Girdi (Salınım Yüzdesi): swp = 0.01
Tanımla: H bir t uzunluğunda boş dizi
Tanımla: L bir t uzunluğunda boş dizi
Tanımla: Z bir t uzunluğunda boş dizi
oH = 0
oL = 0
sH = false
sL = false

for t = 1,...,T do

$h_t > h_{oH} \rightarrow$ oH = t

$sL = false$ & $[(h_{oH} - l_{oL}) \,/\, l_{oL}] * 100 \geq$ swp $\rightarrow$

sH = false

sL = true

$Z_t = l_{oL}$

$L_t = l_{oL}$

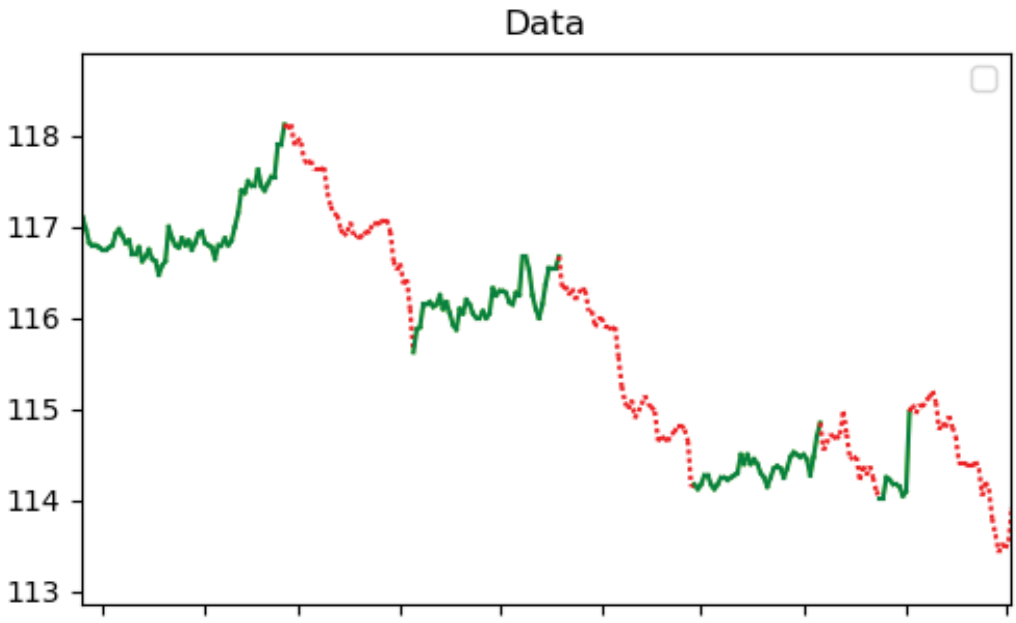
$l_t > l_{oL} \rightarrow$ oL = t

$sH = false$ & $[(h_{oH} - l_{oL}) \,/\, l_{oL}]] * 100 \geq$ swp $\rightarrow$

sL = false

$$sH = true$$
$$Z_t = h_{oH}$$
$$H_t = h_{oH}$$

Bu yönteme göre düşüş yaşanan bir piyasada, son noktadan "%x" oranında bir artış olana değin tüm veriler düşüş olarak işaretlenir. Bu durumun tersi için de yükseliş olarak tüm veriler yükseliş olarak işaretlenir. Veri oluşturulurken bu oran %1 olarak alınmıştır.



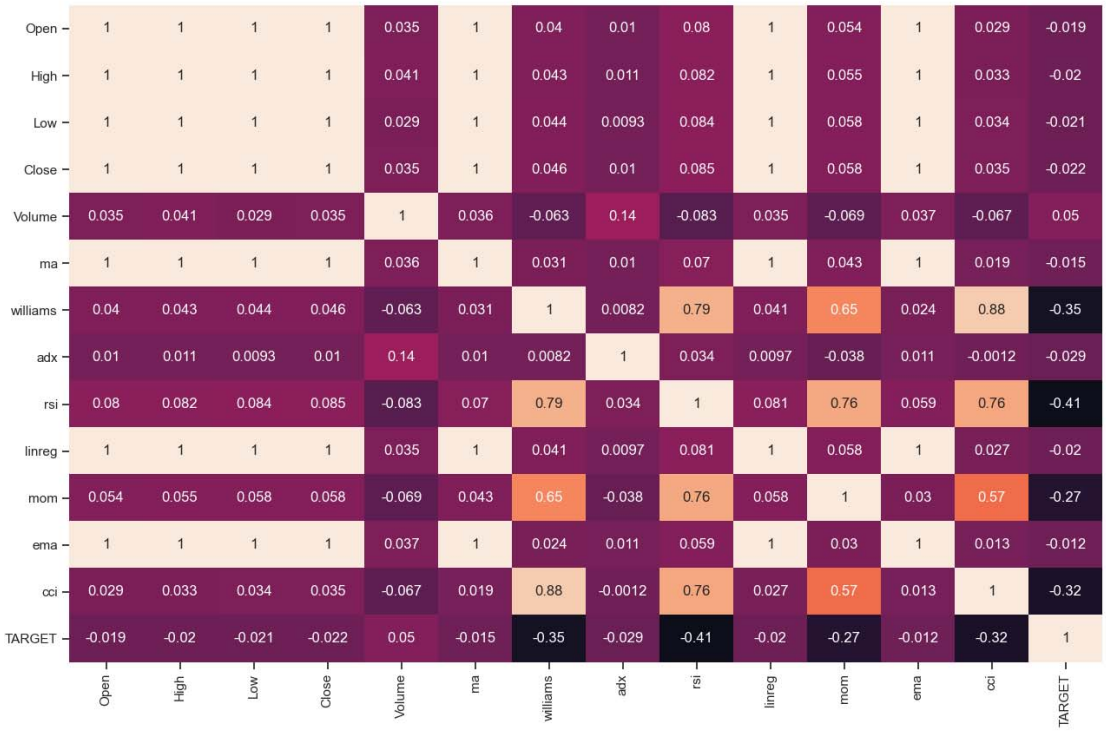
Şekil 12. Yükseliş ve Düşüş Trendlerinin İşaretlenmesi

Yukarıdaki grafikte kesikli çizgiler düşüş trendlerini, kesiksiz çizgiler ise yükseliş trendlerini göstermektedir. Böylece eğitim verisindeki çıktı değerleri olacak yükseliş ve düşüş sınıfları işaretlenmiş olur. Modelimizden beklentimiz burada öğreneceği yükseliş ve düşüş sınıflarını daha önce görmediği bir test verisinde de başarılı bir şekilde tahmin etmesidir.

Çıktı değerleri elde edildikten sonra veri seti toplamda 5 girdiye ve 1 çıktıya sahip bir veri haline gelmiştir. Girdiler içinde mum grafiği verisine ait açılış, kapanış, yüksek, düşük ve hacim değerleri bulunmaktadır. Bu 5 girdi varlığın fiyatı hakkında bilgi içerse de elde edilmek istenen çıktı hakkında fazla bir bilgi içermemektedir. Öyle ki korelasyon tablosu oluşturulduğunda çıktı değeri ile fiyat bazlı girdiler arasında bir ilişki

bulunamamıştır. Aşağodaki tabloda Açılış (Open), Kapanış (Close), Yüksek (High) ve Düşük (Low) değerlerine bakıldığında çıktı ile arasında bir korelasyon bulunmadığı görülüyor. Bu nedenle bu girdilerden farklı indikatör verileri de türetilerek ve girdi olarak modele sağlanarak model başarısının arttırılması mümkündür. Bunun için finansta sıkça kullanılan indikatörler arasından seçilmiş 8 farklı indikatör kullanılacak ve girdi sayısı 13'e çıkarılmış olacaktır. Kullanılan indikatörlere ait açıklama sonraki bölümde verilmiştir.

İndikatörler arasında korelasyon incelendiğinde bazı indikatörler için daha yüksek korelasyon değerleri gözlemlenmiştir. Fiyat verilerinin kendi arasındaki korelasyonu 1 olmasına rağmen çıktı değeri açısından korelasyon bulunmamaktadır. Öte yandan bazı indikatörler ve çıktı değeri arasında korelasyon diğerlerine oranla daha yüksektir. Örneğin WilliamsR, RSI ve CCI indikatörleri çıktı değeri üzerinde daha yüksek korelasyona sahiptir. Yine de bu değerler ±0.40 üzerine çıkamadığı için güçlü bir korelasyondan söz etmek mümkün değildir.

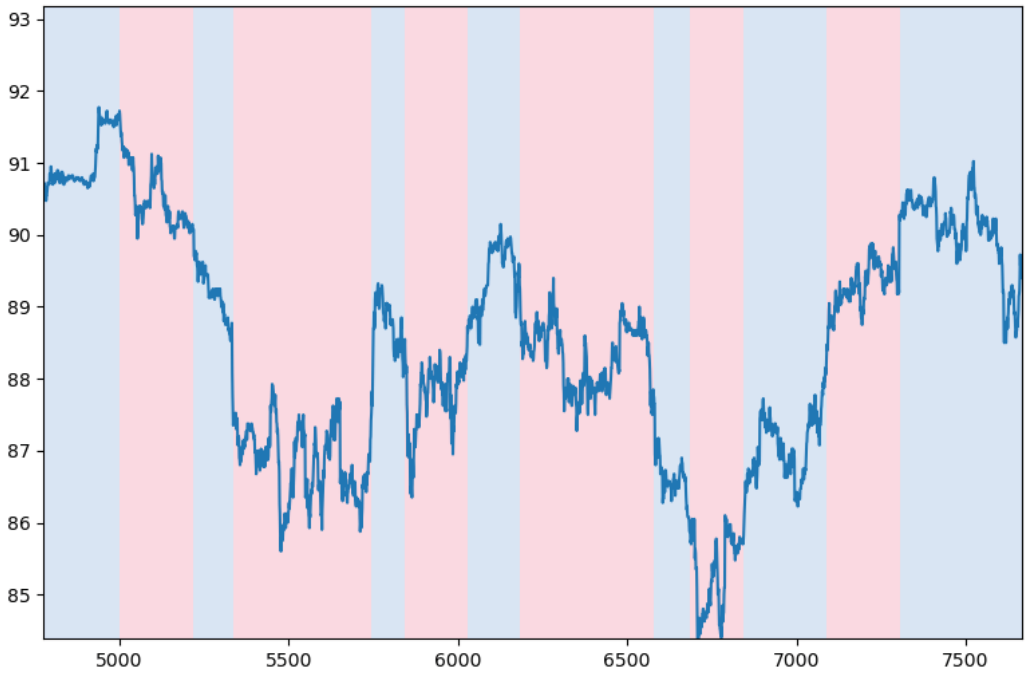


Şekil 12. Model girdileri ve çıktısı korelasyon tablosu

Aşağıdaki listede verideki yüksek, düşük, açılış ve kapanış değerleri kullanılarak türetilmiş finansal indikatörlerin listesi verilmiştir; Tüm indikatörler için son 14 periyot kullanılmıştır, yani her bir değer kendisinden önceki 14 değer kullanılarak hesaplanmıştır.

- MA (Hareketli Ortalama): Son 14 kapanış değerinin ortalaması alınır.
- WILLIAMSR: Yüksek ve düşüklerin farkını kullanarak hesaplanan indikatör 0 ile 100 arasında değer alan bir momentum göstergesidir.
- ADX: Belirli bir yönde oluşan fiyat hareketini gösteren indikatördür.
- RSI: Fiyat hareketlerinin gücünü ölçmek için kullanılan indikatördür.
- LINREG: Son 14 kapanış değerine ait lineer regresyon hesaplanır.
- MOMENTUM: Zamana bağlı fiyattaki değişimi hesaplayan indikatördür.
- EMA (Üssel Hareketli Ortalama): Klasik hareketli ortalamada değişimlerin yumuşatılmış olarak hesaplandığı indikatördür.
- CCI: Ortalama fiyat değişimi ve standart sapma kullanılarak elde edilen momentum göstergesidir.

Model kurma aşamasına geçmeden önce Bölüm 6'da bahsedilen yapısal kırılma ve değişim noktalarının model başarısını ne yönde etkilediğiyle ilgili testlerin yapılabilmesi için de veri hazırlanması gerekmektedir. Bunun için öncelikle elde ettiğimiz nihai veri setinin bir kopyası alınır. Daha sonra veri üzerinde Bölüm 6'da verilen CHOW testi ve PELT testleri uygulanır. Bulunan değişim noktaları işaretlenir ve eğitim verisinde herhangi bir aralık içinde bu noktalardan biri bulunuyorsa, ilgili veri girdilerden çıkarılır. Sonuç olarak elde değişim noktalarından arındırılmış bir veri seti kalır.



Şekil 13. Değişim noktalarının işaretlenmesi

Son olarak verinin yapay sinir ağlarına uygun bir girdiye dönüştürülmesi için normalize edilmesi gerekmektedir. Veriyi 1 ve -1 arasına normalize ettiğimiz zaman özellikle fiyat verilerinde yeni gelecek verideki fiyatların aralık dışında kalma ihtimalinden dolayı öncelikle verideki her bir noktanın önceki noktaya göre yüzdelik değişimi alınmıştır. Daha sonra yeni değerler üzerinden tüm girdiler 1 ve -1 arasına normalize edilmiştir.

4.3. Model Kurulması

LSTM modeli, GRU modeli ve ileri beslemeli yapay sinir ağı modellerinin karşılaştırılması için üç farklı model kurulacaktır. Daha sonra bu üç model değişim noktaları çıkarılmış ve değişim noktaları çıkarılmamış iki veri setiyle eğitilecektir. Son olarak elde edilen 6 sonuca ait metrikler karşılaştırılarak performans karşılaştırması yapılacaktır.

Öncelikle sinir ağı modellerinin kurulumu için ağ mimarisi seçilmeli ve başlangıç parametreleri belirlenmelidir. Aşağıdaki tabloda modellerin konfigürasyonları verilmiştir.

İsim	Değer
Eğitim Veri Sayısı	87527

Test Veri Sayısı	23175
Gizli Katman Sayısı	2
Katman Nöron Sayısı	100
Çıktı Katmanı Nöron Sayısı	1
Aktivasyon Fonksiyonu	Relu
Kayıp Fonksiyonu	İkili Çapraz Entropi
Öğrenme Oranı	0.0005
Örnek Seri Uzunluğu	50
Devir Sayısı	10

Tablo 1. Sinir Ağı Konfigürasyonu

Eğitim ve test verisi belirlenirken makine öğrenmesi uygulamalarında sıkça kullanılan %80 eğitim ve %20 test verisi olarak dağılım yapılmıştır. Ağların gizli katman sayısı belirlenirken verinin karmaşıklık derecesi düşünüldüğünde ve yapılan testler sonucunda 2 gizli katmanın, tek gizli katmana göre daha iyi sonuç verdiği görülmüştür. Bu nedenle ağlara 2 gizli katman eklenmiştir. Her bir katmanda 100 nöron bulunmaktadır. Bu sayı verinin boyutu göz önünde bulundurularak belirlenmiştir. Çıktı katmanında ise problemin bir ikili sınıflama problemi olması nedeniyle 1 nöron bulunmaktadır. Aynı şekilde önceki bölümlerde açıklandığı gibi ikili sınıflama problemine uygun olarak "Relu" aktivasyon fonksiyonu ve "İkili Çapraz Entropi" kayıp fonksiyonu modellerde kullanılmıştır.

Öğrenme oranı her bir ağırlığın yeniden hesaplanmasında kullanılan ve öğrenme adımlarının yeterince küçük ve kesin hale gelmesini sağlayan katsayıdır. Bu katsayı seçilirken farklı öğrenme oranları test edilmiş ve probleme en uygun katsayı bulunmuştur. Öğrenme oranının yüksek olduğu durumlarda model hızlı bir şekilde eğitim verisine yakınsamış ve "overfitting" adı verilen, aşırı öğrenmeden kaynaklı ezberleme sorunu ortaya çıkmıştır.

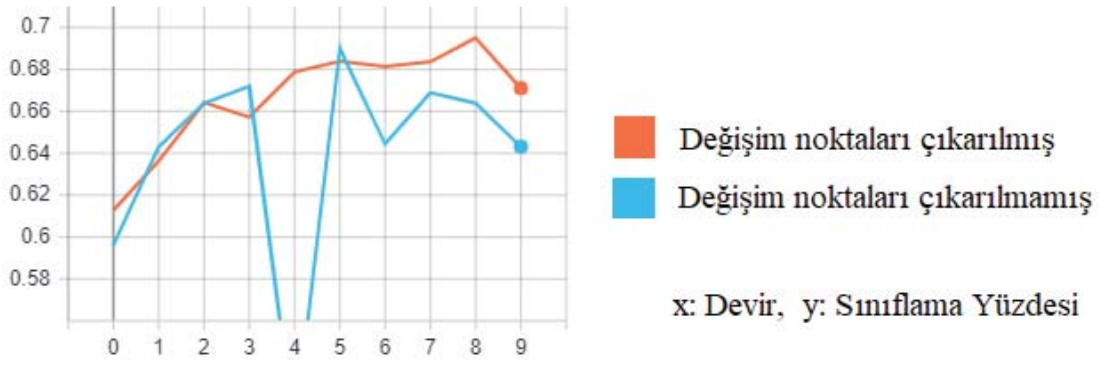
Çalışmada özellikle LSTM ağlarının uzun süreli ilişkileri tanıma konusundaki başarısının gösterilmesi hedeflendiği için her bir örnek için seri uzunluğu 50 olarak seçilmiştir. Yani modele verilen girdiler verideki sıralı 50 değerdir. Bu değer literatürde "window size" parametresi olarak da bilinmektedir. Değer olarak 50 seçilmesinin nedeni

sinir ağı mimarilerinin nöron sayıları ve hesaplama süreleri göz önünde bulundurulduğunda, 50 sıralı girdi uzunluğu hem uzun süreli ilişkileri içermesi hem hesaplama süresi kısa olmasından dolayı seçilmesine karar verilmiştir.

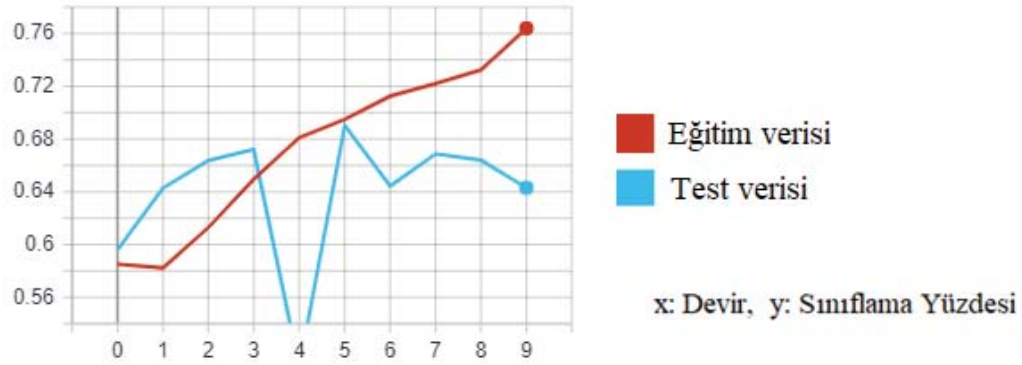
4.4. Sonuçlar

4.4.1. LSTM Modeli Sonuçları

Kurulan LSTM modeli, GRU modeli ve ileri beslemeli yapay sinir ağı modeli önceki bölümde verilen parametreler ile eğitilmiş ve her devir sonunda test verisi ile sınanarak, sınıflama doğruluğu elde edilmiştir. Her model değişim noktaları çıkarılmış ve değişim noktaları çıkarılmamış iki veri setiyle ayrı ayrı çalıştırılmıştır.



Şekil 14. LSTM ağları test verisi doğru sınıflama oranları



Şekil 15. LSTM ağları eğitim ve test verisi doğru sınıflama oranları

Yukarıdaki grafikte görüldüğü üzere ilk devirden itibaren test verisi üzerindeki sınıflama başarısının genellikle arttığı görülmektedir. En yüksek başarının elde edildiği devrin, değişim noktaları çıkarılmış veride 0.69 sınıflama oranı başarısı ile 8. devir olduğunu ve bu noktaya kadar sınıflama yüzdelerinin düzenli bir artış içinde olduğu gözlemlenmektedir. Dolayısıyla model gerçekten veriden öğrenerek birtakım bilgiler elde etmiş ve bu bilgileri test verisinde başarılı olarak kullanabilmiştir. Son devire gelindiğinde ise sınıflama başarısı oranının 0.67 olduğu görülmektedir. Model, 9. Devirde eğitim hatasını düşürmüş fakat test hatasını arttırmıştır. Bu durum ağırlıkların fazla yakınsayarak verinin gerçek doğasından çok eğitim verisine uygun bir hale geldiğini yani aşırı öğrenme (overfitting) gerçekleştiğini göstermektedir.

Değişim noktaları çıkarılmayan veriye baktığımızda, daha düzensiz bir artışın olduğu ve 4. devirde büyük bir düşüş yaşandığı görülmektedir. Yapay sinir ağlarında eğitim yapılırken eğitim verisi için hatanın azaldığı bazı noktalarda test verisi için yüksek hatalar elde edilebilir. Bu durum öğrenme süreci içindeki adımları atarken karşılaşılabilecek doğal bir sonuçtur. 4. Devirde yaşanan büyük düşüş bu şekilde yorumlanabilir. Fakat burada asıl önemli nokta değişim noktaları çıkarılmamış veride, başarı oranlarındaki düzensizlik, değişim noktaları çıkarılmış veriye göre daha fazladır. Yani değişim noktaları çıkarılan veri setinde, doğru bilginin model tarafından öğrenilebilmesi kolaylaşmıştır. Bu durum makine öğrenmesinde doğru veri seçimi ve doğru veri ön işlemenin önemini de göstermektedir.

	Tahmin: 0	**Tahmin: 1**
Gerçek: 0	6589	3731
Gerçek: 1	2705	6538

Tablo 2. LSTM değişim noktaları çıkarılmış veri seti hata matrisi

Yukarıdaki tabloda değişim noktaları çıkarılmış veri seti ile eğitilmiş LSTM modeli hata matrisi bulunmaktadır. Görüldüğü üzere gerçek değer ve tahminler arasında dengeli bir dağılım bulunmaktadır. Model 67.10% oranında doğru tahmin elde etmiştir. Sınıflar arasındaki hata oranı karşılaştırıldığında "ALIM" sınıfına karşılık gelen "0" değerinde daha yüksek başarı elde edildiği görülüyor. Fakat inceleyeceğimiz diğer hata

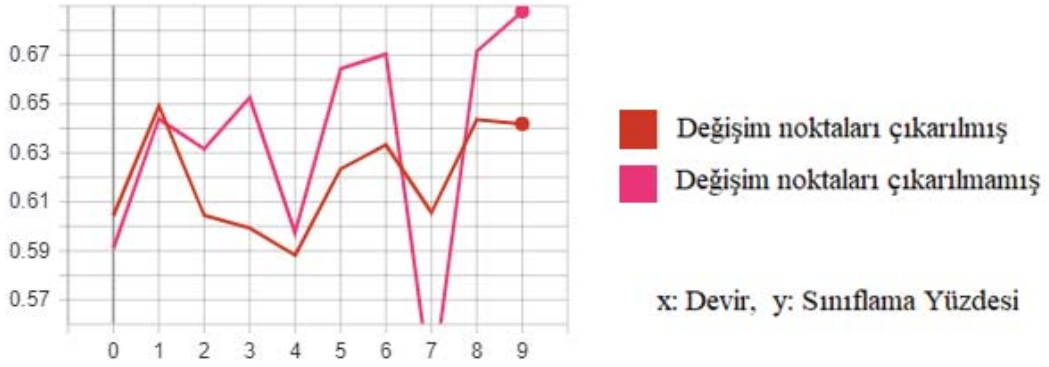
matrislerine bakıldığında her zaman "ALIM" kategorisinde yüksek başarı elde edilemediğinden, bu durum için herhangi bir yargıya varılamaz. Sınıflara ait verilerin neredeyse eşit olduğu bir veri setinde 67.10% başarı oranı modelin başarılı olarak eğitildiğini göstermektedir.

	Tahmin: 0	**Tahmin: 1**
Gerçek: 0	7786	3990
Gerçek: 1	3937	6500

Tablo 3. LSTM değişim noktaları çıkarılmamış veri seti hata matrisi

Değişim noktaları çıkarılmamış veri setinde hata matrisine bakıldığında ise sınıflama başarısının 64.31% değerine düştüğü görülüyor. Değişim noktaları çıkarılmış veri setiyle karşılaştırıldığında, burada en büyük kaybın "ALIM" sınıfına karşılık gelen 0 değerinde yaşandığı görülüyor. Sınıfların tek başına tahmin başarısı birbirine oldukça yakın hale gelmiştir.

4.4.2. GRU Modeli Sonuçları



Şekil 16. GRU modeli test verisi doğru sınıflama oranları

Yukarıdaki grafikte GRU modeli kullanarak iki ayrı veri seti ile eğitilmiş modellerin, test verisindeki başarı oranları verilmiştir. Değişim noktaları çıkarılmış veride ilk devirdeki %61 başarı oranı ile son devirdeki %64 başarı oranı arasında önemli bir fark

bulunmamaktadır ve devirler arasında düzensiz bir artış yaşandığı görülüyor. Değişim noktaları çıkarılmamış veride ise devirler arasındaki sınıflama başarısının oldukça düzensiz olduğu görülüyor fakat son devirdeki başarı oranının beklenenin aksine değişim noktaları çıkarılmış veriden yüksek olduğu görülmektedir. Değişim noktaları çıkarılmamış veride son devirde %68'lik bir başarı oranı elde edilmiştir.

	Tahmin: 0	**Tahmin: 1**
Gerçek: 0	8305	3471
Gerçek: 1	4389	6048

Tablo 4. GRU değişim noktaları çıkarılmış veri seti hata matrisi

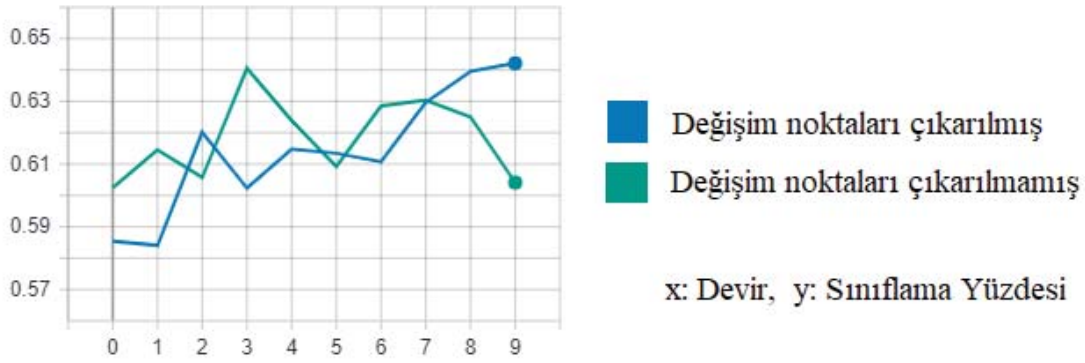
İleri beslemeli yapay sinir ağları ile değişim noktaları çıkarılmış verideki hata matrisi incelendiğinde sınıflar arasında dengeli bir dağılım görülüyor. Genel başarı oranına bakıldığında ise 64.62% sınıflama doğruluğu gözlemlenmiştir.

	Tahmin: 0	**Tahmin: 1**
Gerçek: 0	7917	3859
Gerçek: 1	3075	7362

Tablo 5. GRU değişim noktaları çıkarılmamış veri seti hata matrisi

Değişim noktaları çıkarılmamış veri incelendiğinde ise beklenenin aksine hata oranının azaldığı görülmektedir. Sınıflama başarısı son devirde 68.78% olmuş ve LSTM modelini bile geride bırakmıştır. Fakat sınıflama başarısının devirler arasındaki düzensizliği göz önünde bulundurulduğunda bu sonucun LSTM modelinde elde edilen sonuç kadar sağlam olmadığı söylenebilir.

4.4.3. İleri Beslemeli Sinir Ağı Modeli Sonuçları



Şekil 17. İleri beslemeli sinir ağları test verisi doğru sınıflama oranları

Yukarıdaki grafikte ileri beslemeli sinir ağlarını kullanarak iki ayrı veri seti ile eğitilmiş modellerin, test verisindeki başarı oranları verilmiştir. Grafikte görüldüğü üzere ileri beslemeli sinir ağları ile yapılan eğitimlerde başarı oranlarının daha yüksek oynaklığa sahip olduğu ve LSTM modeli kadar doğrusal bir biçimde artmadığı görülüyor. Yine de özellikle değişim noktaları çıkarılmış veride ilk devirdeki %58 başarı oranı ile son devirdeki %64 başarı oranı arasında önemli bir fark bulunuyor ve devirler arasında bir artış yaşandığı görülüyor. Fakat değişim noktaları çıkarılmamış veriye bakıldığında devirler arasındaki sınıflama başarısının oldukça düzensiz olduğu görülüyor. 3. Devirde elde edilen en yüksek başarı oranı %64 olsa da daha sonraki devirlerde bu oranı hiçbir zaman yakalayamıyor.

	Tahmin: 0	**Tahmin: 1**
Gerçek: 0	7238	4538
Gerçek: 1	3351	7085

Tablo 6. Sinir ağları değişim noktaları çıkarılmış veri seti hata matrisi

İleri beslemeli yapay sinir ağları ile değişim noktaları çıkarılmış verideki hata matrisi incelendiğinde sınıflar arasında dengeli bir dağılım görülüyor. LSTM modelinin aksine bu modelde “SATIM” yönlü 1 sınıfında daha yüksek hata oranı olduğu görülüyor.

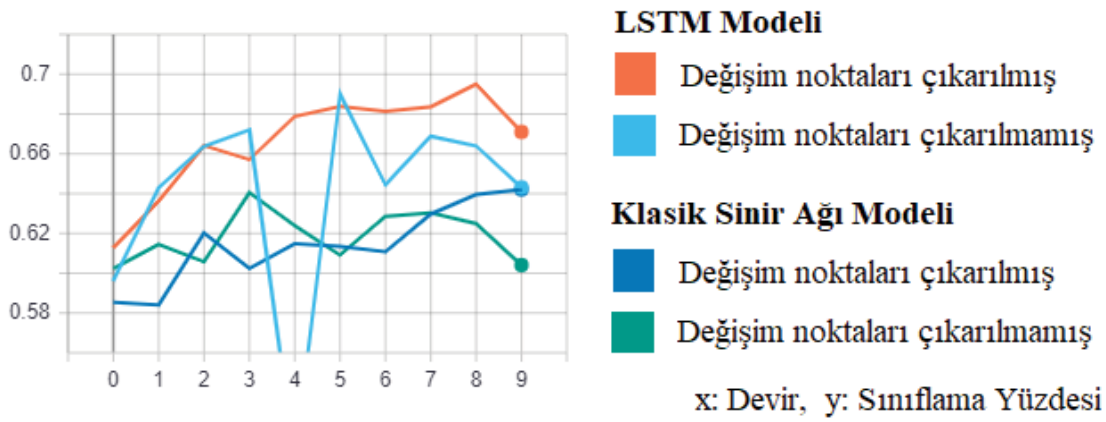
Genel başarı oranına bakıldığında ise 64.48% sınıflama doğruluğu ile değişim noktaları çıkarılmamış LSTM modeli ile neredeyse aynı başarı oranına sahip olduğu görülüyor.

	Tahmin: 0	**Tahmin: 1**
Gerçek: 0	9911	1865
Gerçek: 1	6587	3849

Tablo 7. Sinir ağları değişim noktaları çıkarılmamış veri seti hata matrisi

Değişim noktaları çıkarılmamış veri incelendiğinde ise hata oranının oldukça arttığı ve tahmin edilen sınıfların homojen bir şekilde dağılmadığı görülüyor. Değişim verilerinin çıkarılmaması durumu özellikle ileri beslemeli sinir ağlarını daha çok etkilemiş görünüyor. Sınıflama başarısı son devirde 61.95% olmuş ve özellikle "ALIM" yönlü 0 sınıfına ait tahminlerin oldukça fazla olduğu göze çarpıyor.

4.4.4. Sonuçların Karşılaştırılması



Şekil 18. Tüm model ve veri setleri sınıflama başarısı karşılaştırması

Yukarıdaki grafikte her iki veri seti ile eğitilmiş LSTM ve ileri beslemeli sinir ağı modellerinin sınıflama başarıları verilmiştir. Görüldüğü üzere LSTM ağları ileri beslemeli sinir ağlarına göre daha iyi bir performans sergilemiştir. Aradaki fark yaklaşık 4% olsa da modelde kullanılan finansal verinin tahmin zorluğu düşünüldüğünde kayda değer bir farklılıktır. Değişim noktalarının veriden elenmesinin de model başarısında olumlu bir fark yarattığı görülüyor. Hatta değişim noktası elenmemiş LSTM modeli ile

değişim noktası elenmiş bir ileri beslemeli sinir ağı modeli neredeyse aynı sınıflama başarısına sahip. Bu durum veri seçiminin ve veri ön işlemenin makine öğrenmesinde neredeyse modelin kendisi kadar önemli olduğunu gösteriyor.

Uygulamadan çıkan sonuçlar yorumlandığında sıralı bir zaman serisi verisinde LSTM ağlarının, ileri beslemeli sinir ağlarına göre bir miktar üstünlüğü bulunmaktadır. Bu durum LSTM ağlarının zaman serisindeki uzun ve kısa süreli ilişkileri belleğinde tutabilme özelliğini ortaya çıkarmaktadır. Finans verisi yerine genellemeye daha uygun bir zaman serisi seçilirse modeller arasındaki farklılık da artabilir fakat seçilen veri göz önünde bulundurulduğunda ortaya çıkan fark küçük olsa bile model seçiminin başarıyı etkilediği görülmüştür.

5.Sonuç

Bu çalışmada bir yinelemeli sinir ağı türü olan LSTM modeli, GRU modeli ve tipik bir yapay sinir ağı modeli, finansal bir zaman serisi verisi üzerinde, performans kriterleriyle karşılaştırıldı. İlk olarak yapay sinir ağlarının ortaya çıkışı ve tipik bir sinir ağında bulunan aktivasyon fonksiyonu, kayıp fonksiyonu, optimizasyon algoritması gibi bölümler ele alındı. Daha sonra bu yapının üzerine inşa edilmiş yinelemeli sinir ağları, GRU, LSTM gibi modellere değinildi ve bu modellerin çalışma prensipleri açıklandı. Çalışmada görüldüğü üzere yapay sinir ağları tarihi boyunca birçok farklı alanda ve birçok farklı geliştirme ile yeni özellikler kazanmış ve bu özellikler farklı problemlerin çözümünde kullanılmıştır. Temelinde bir optimizasyon problemini çözen sinir ağları, probleme konu olan veri setinin doğasına uygun bir mimariyle yapıldığında daha iyi çözümler elde edilebileceği görüldü.

Çalışmada değinilen bir diğer önemli konu ise yinelemeli sinir ağlarında ortaya çıkan kaybolan gradyan sorunu ve bu soruna getirilen çözümlerdir. Aslında çalışmanın ana konusu olan LSTM ağları bu sorunu oldukça başarılı bir şekilde çözen bir model olduğu için yinelemeli sinir ağları üzerine inşa edilmiş gelişmiş bir model olduğu söylenebilir.

Çalışmada LSTM ağları ile ileri beslemeli bir sinir ağı arasında, sıralı zaman serilerindeki tahmin başarısının farkı gösterilmesi amaçlandı. Bu nedenle uygulamada Borsa İstanbul Vadeli İşlem Piyasasında bulunan bir kontrata ait fiyatlar veri olarak kullanıldı. Daha sonra bu fiyat değerlerinden türetilen indikatör verileri ile 13 girdiye sahip bir veri seti elde edildi. Çalışmada finansal veri kullanıldığından dolayı ekonometride sıkça kullanılan değişim noktası tespiti konusuna da değinildi. Buna göre dış etkilerden kaynaklı olan varlık fiyatındaki değişim noktaları veri setinden çıkarılarak, bir makine öğrenmesi modeline daha uygun, temizlenmiş bir veri seti edildi.

Modeller kurulurken başlangıç parametreleri ve sinir ağı mimarileri olabildiğince aynı seçilmeye çalışıldı. Model parametrelerinden doğabilecek farklılıkların böylece minimuma indirilmesi amaçlandı. Modeller, değişim noktaları çıkarılmış ve değişim noktaları çıkarılmamış iki farklı veri seti ile eğitildi. Eğitimlerde her devirde ortaya çıkan sınıflama başarıları kaydedildi. Bunun dışında her modelin eğitimi tamamlandığında, test verisi ile bir hata matrisi oluşturuldu.

Sonuçlar incelendiğinde LSTM ağlarının varlık fiyatındaki yukarı yönlü ve aşağı yönlü hareketleri bir miktar daha iyi tahmin edebildiği görüldü. İleri beslemeli sinir ağlarında her devir sonundaki sınıflama başarılarına bakıldığında genellikle düzensiz bir artışın olduğu ve modelin verinin doğasını anlamakta güçlük çektiği görüldü. LSTM ağları bu anlamda geçmiş veriden öğrenilmiş bilgiyi daha iyi sakladığı için ileriki tahminlerinde daha %4 kadar daha doğru sonuçların elde edildiği görüldü. Bunun dışında değişim noktası tespit edilen örneklerin veriden çıkarılmasının model başarısını olumlu yönde etkilediği görüldü.

Sonuç olarak LSTM ağlarının sıralı zaman serilerinde, ileri beslemeli sinir ağlarına göre daha iyi performans verebileceği için bu tip problemlerde sıklıkla kullanılmaktadır. Çalışmada kullanılan modeller bir finansal varlığın t+1 zamandaki hareketini tahmin ettiği için başarı oranı 70% üzerine çıkmamıştır ve modeller arasındaki başarı farkı da %10'u geçmemiştir fakat bu tip karmaşık ve tahmin edilmesi zor olan bir veri setinde bile başarı farkı modeller arasında görülebilir derecededir. Eğer farklı zaman serisi problemleri ele alınırsa modeller arasındaki farkın artması da mümkün görünmektedir. Yine de model başarıları ele alındığında, model parametreleri, verinin ön

işlemesi, model girdileri gibi değişkenlerde farklı kombinasyonlar denendiğinde daha iyi sonuçlar ile de karşılaşmak mümkündür.

KAYNAKÇA

[1] A. Graves ve N. Jaitly, Towards End-To-End Speech Recognition with Recurrent Neural Networks, ICML, 2014, p. 1764–1772.

[2] W. P. Warren S. MCCULLOCH, A Logical Calculus Of The Ideas Immanent In Nervous Activity, Bulletin of Mathematical Biophysics Vol. 5, 1943.

[3] Y. LeChun, P. Haffner, L. Bottou ve Y. Bengio, Object Recognition With Gradient Based Learning, 1998.

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville ve Y. Bengio, Generative Adversarial Nets, 2014.

[5] S. Hochreiter ve J. Schmidhuber, LONG SHORT-TERM MEMORY, 1997.

[6] A. Graves, G. Wayne ve I. Danihelka, Neural Turing Machines, 2014.

[7] M. H. Beale, M. T. Hagan ve H. B. Demuth, Neural network Toolbox user's guide, 2009.

[8] M. A. KIZRAK ve B. BOLAT, Derin Öğrenme ile Kalabalık Analizi Üzerine Detaylı Bir Araştırma, BİLİŞİM TEKNOLOJİLERİ DERGİSİ, CİLT: 11, SAYI: 3, 2018.

[9] R. Pascanu, T. Mikolov ve Y. Bengio, «On the difficulty of training recurrent neural networks,» *Universit´e de Montr´eal, 2920, chemin de la Tour, Montr´eal, Qu´ebec, Canada, H3T 1J8,* 2012.

[10] D. B. Şeker, Derin Öğrenme Yöntemleri ve Uygulamaları Hakkında Bir İnceleme, 2017.

[11] Y. Bengio, P. Simard ve P. Frasconi, «Learning Short Term Dependencies with Gradient Descent is Difficult,» *IEEE Transactions on Neural Networks,* 1994.

[12] K. Cho, B. v. Merrienboer ve D. Bahdanau, On the Properties of Neural Machine Translation: Encoder–Decoder Approaches, 2014.

[13] J. Chung, C. Gulcehre, K. Cho ve Y. Bengio, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, 2014.

[14] G. Chow, Tests of Equality Between Sets of Coefficients in Two Linear Regressions, 1960.

[15] C. Truong, L. Oudre ve N. Vayatis, Selective review of offline change point detection methods, 2019.