FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

ZAMAN SERİLERİ TAHMİNİNDE MELEZ BİR YAKLAŞIM

YÜKSEK LİSANS TEZİ

NURBANU IŞIK DELİBALTA

İSTANBUL, 2023

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

ZAMAN SERİLERİ TAHMİNİNDE MELEZ BİR YAKLAŞIM

YÜKSEK LİSANS TEZİ

NURBANU IŞIK DELİBALTA
(210221010)

Danışman
(Prof. Dr. Burhanettin Can)
Eş Danışman
(Dr. Öğr. Üyesi Gönül Uludağ)

İSTANBUL, 2023

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ
TEZ ONAY FORMU

31/07/2023

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Enstitümüz Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans programı 210221010 numaralı öğrencisi Nurbanu Işık-Delibalta'nın hazırladığı "ZAMAN SERİLERİ TAHMİNİNDE MELEZ BİR YAKLAŞIM" konulu Yüksek Lisans tezi ile ilgili Tez Savunma Sınavı, 31/07/2023 Pazartesi günü saat 10'da yapılmış, sorulara alınan cevaplar sonunda adayın tezinin **Kabulüne Oy Çokluğu/Birliği** ile karar verilmiştir.

Tez adı değişikliği yapılması halinde: Tez adının .....
.....
şeklinde değiştirilmesi uygundur.

	Jüri Üyesi	Karar
1.	(Danışman) Prof. Dr. Burhanettin CAN	Kabul
2.	Dr. Öğr. Üyesi Fatma CORUT ERGİN	Kabul
3.	Dr. Öğr. Üyesi Shabaan SAHMOUD	kabul
4.	.....	.....
5.	.....	.....
6.	(İkinci Danışman)*.....	.....

**2. Danışman varsa doldurulması gerekmektedir.*

ETİK BİLDİRİM

Bu tezin yazılmasında bilimsel ahlak kurallarına uyulduğunu, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, tezin herhangi bir kısmının bağlı olduğum üniversite veya bir başka üniversitedeki başka bir çalışma olarak sunulmadığını beyan ederim.

Nurbanu Işık Delibalta

TEŞEKKÜR

Tez konusunun belirlenmesinden tamamlanmasına kadar her aşamasında değerli tecrübelerini, zamanını ve motivasyon arttırıcı konuşmaları ile desteklerini esirgemeyen değerli danışman hocalarım Prof. Dr. Burhanettin CAN'a ve Dr. Öğr. Üyesi Gönül ULUDAĞ'a teşekkürlerimi sunuyorum.

Eğitim hayatım boyunca desteğini ve inancını hiç esirgemen anneme, babama ve kız kardeşlerime tüm fedakarlıkları için teşekkür ederim. Tüm bu desteklere ek olarak sabır, anlayış ve sevgisini hiç eksiltmeyen oğlum Asaf'a ve eşim Vahdettin'e teşekkür ediyorum.

Nurbanu Işık Delibalta

ZAMAN SERİLERİ TAHMİNİNDE MELEZ BİR YAKLAŞIM

Nurbanu Işık Delibalta

ÖZET

Bu çalışma, farklı karakteristik örüntülere sahip zaman serilerinin analizi için derin öğrenme (ing: deep learning) ve meta-sezgisel (ing: meta-heuristic) yöntemlerin birlikte kullanıldığı melez bir çerçeve uygulamasıdır. Bu melez çerçeve, Parçacık Sürü Optimizasyonu (ing: Particle Swarm Optimization) ve Uzun Kısa Süreli Bellek (ing: Long Short-Term Memory) yöntemlerini bir araya getirmektedir. Geleneksel Uzun Kısa Süreli Bellek modellerinde kullanılan stokastik gradyan iniş (ing: Stochastic Gradient Descent) optimizasyon algoritmasının öğrenme oranı Parçacık Sürü Optimizasyonu meta-sezgiseli ile adaptif bir şekilde kullanılmasını içerir. Bu yaklaşım, farklı dillere ait Wikipedia web trafiği zaman serileri için etkin bir gelecek tahmini yaklaşımı sunmaktadır.

Uzun Kısa Süreli Bellek, derin öğrenme yöntemlerinden biridir ve zaman serilerinin bağlamsal bağımlılıklarını modellemek için kullanılır. Parçacık Sürü Optimizasyonu, toplum tabanlı bir optimizasyon algoritmasıdır ve derin öğrenme mimarilerinin eğitimi ve hiper-parametre ayarlarında başarılı bir şekilde yaygın olarak kullanılmaktadır. Bu yöntem, gelecek tahmini için en iyi parametre değerlerini bulmak için kullanılan bir optimizasyon süreci sağlamaktadır.

Uzun Kısa Süreli Bellek modeli için hiper-parametreler arasında birim sayısı, öğrenme oranı, iterasyon sayısı gibi değerler bulunurken, Parçacık Sürü Optimizasyonu algoritması için ise kendilik faktörü, sosyal faktör, sürü ağırlık faktörü, parçacık sayısı, iterasyon sayısı gibi parametreler önemlidir. Bu hiper-parametrelerin doğru bir şekilde seçilmesi, modelin veri kümesine uyum sağlaması ve aşırı öğrenme (ing: overfitting) ya da eksik öğrenme (ing: underfitting) gibi sorunların önlenmesi açısından hayati önem taşır.

Bu çalışmada, zaman serisi analizi için farklı Uzun Kısa Süreli Bellek yapıları ve optimizasyon yöntemleri kullanılarak performans değerlendirmesi yapılmıştır.

Yöntemler web trafik zaman serisi verisi üzerinde uygulanmıştır. İlk olarak, Tek Yönlü Uzun Kısa Süreli Bellek modeliyle deneyler gerçekleştirilmiştir. Ardından, Çift Yönlü ve çok katmanlı Uzun Kısa Süreli Bellek modelleri de değerlendirilmiştir. Elde edilen sonuçlar incelendiğinde, Tek Yönlü Uzun Kısa Süreli Bellek modelinin en düşük Ortalama Kare Hatası (ing: Mean Squared Error) değerini verdiği gözlemlenmiştir.

Uzun Kısa Süreli Bellek modeli deneyinde Stokastik Gradyan İniş (ing: Stochastic Gradient Descent) optimizasyon algoritmasının yüksek iterasyonlarda iyi sonuçlar verdiği sonucu elde edilmiştir. Ancak düşük iterasyonlarda daha iyi sonuçlar elde etmek için Parçacık Sürü Optimizasyonu-Uzun Kısa Süreli Bellek melez yöntemi kullanılarak öğrenme oranı değiştirilmiştir. Parçacık Sürü Optimizasyonu algoritması, hiper-parametre optimizasyonu ile en uygun öğrenme oranı belirlenmiştir. Elde edilen sonuçlar, Parçacık Sürü Optimizasyonu ve Uzun Kısa Süreli Bellek kullanarak yapılan analizin diğer geleneksel yöntemlere kıyasla daha iyi performans gösterdiğini göstermektedir.

Özellikle, MSE hata metrikleri açısından, melez Parçacık Sürü Optimizasyonu-Uzun Kısa Süreli Bellek yönteminin daha düşük hata değerleri elde ettiği gözlenmiştir. Bu durum, melez yöntemin zaman serisi verilerinin modellemesi ve gelecek tahminleri için daha etkili bir kombinasyon sunduğunu göstermektedir. Bu çalışma, farklı Uzun Kısa Süreli Bellek yapıları ve optimizasyon yöntemlerinin zaman serisi analizindeki performanslarının değerlendirilmesi açısından önemli bir katkı sağlamaktadır. Farklı hiper-parametre değerlerinin denendiği bir yaklaşım, hem Uzun Kısa Süreli Bellek hem de Parçacık Sürü Optimizasyonu ile elde edilen sonuçların daha güvenilir ve daha iyi performanslı olmasını sağlayabilir. Bu şekilde, modelin genel performansını iyileştirmek ve veri kümesine en uygun modeli elde etmek için daha kapsamlı bir araştırma yapılabilir.

Anahtar kelimeler: Derin Öğrenme, LSTM, PSO, Optimizasyon, Yapay Sinir Ağı, Zaman Serisi

A HYBRID APPROACH TO TIME SERIES FORECASTING

Nurbanu Işık Delibalta

ABSTRACT

This study presents a hybrid framework that combines deep learning and metaheuristic methods for the analysis of time series with different characteristic patterns. The hybrid framework integrates Particle Swarm Optimization (PSO) and Long Short-Term Memory (LSTM) methods. It involves the adaptive use of the stochastic gradient descent optimization algorithm used in traditional Long Short Term Memory models with the learning rate Particle Swarm Optimization meta-heuristic. This approach provides an efficient future forecasting approach for Wikipedia web traffic time series of different languages.

LSTM is one of the DL methods used to model contextual dependencies in time series. PSO, on the other hand, is a metaheuristic optimization method widely used for time series analysis. This method provides an optimization process to find the best parameter values for future predictions.

For the LSTM model, hyperparameters such as the number of units, learning rate, and number of iterations are important, while for the PSO algorithm, parameters like individuality factor, social factor, swarm weight factor, number of particles, and number of iterations are crucial. Properly selecting these hyperparameter values is of vital importance to ensure the model fits the dataset and prevent issues such as overfitting or underfitting.

In this study, performance evaluation was conducted using different LSTM architectures and optimization methods for time series analysis. The methods were applied to web traffic time series data. Firstly, experiments were carried out using the Unidirectional LSTM model. Subsequently, Bidirectional LSTM and Multilayer LSTM models were also evaluated. Upon examining the results, it was observed that the Unidirectional LSTM model provided the lowest Mean Squared Error (MSE) values.

In the experiments with the LSTM model, it was observed that the Stochastic Gradient Descent (SGD) optimization algorithm yielded good results in high iterations. However, to achieve better results in low iterations, the PSO-LSTM hybrid method was employed by adjusting the learning rate. The PSO algorithm was used for hyperparameter optimization to determine the optimal learning rate. The obtained results demonstrate that the analysis performed using PSO and LSTM outperformed other traditional methods in terms of performance.

Especially in terms of MSE error metrics, it has been observed that the hybrid PSO-LSTM method achieves lower error values. This finding indicates that the hybrid approach provides a more effective combination for modeling time series data and making future predictions. This study makes a significant contribution to the evaluation of different LSTM architectures and optimization methods for time series analysis. The approach of trying different hyperparameter values can lead to more reliable and better-performing results for both LSTM and PSO. By doing so, a more comprehensive research can be conducted to improve the overall performance of the model and obtain the most suitable model for the dataset.

Anahtar kelimeler: Deep Learning, LSTM, PSO, Optimization, Artificial Neural Network, Time Series

ÖNSÖZ

Tez çalışmasında, zaman serisi verilerinde derin öğrenme (DÖ) yöntemlerinin kullanıldığı bir araştırmanın sonuçları sunulmaktadır. Araştırma kapsamında, Uzun Kısa Süreli Bellek (ing: Long Short Term Memory [LSTM]) modeli zaman serisi analizi için uygulanmış ve modelin hiper-parametreleri optimize edilerek en uygun yapıya ulaşılmıştır. Ayrıca, eniyileyicinin (ing: optimizer) öğrenme oranının en iyi değerini bulmak amacıyla Parçacık Sürü Optimizasyonu (ing: Particle Swarm Optimization [PSO]) algoritması kullanılmış ve PSO algoritmasının parametreleri optimize edilmiştir.

Zaman serisi analizinde DÖ yöntemlerinin başarısı, doğru hiper-parametre ayarlamalarına (ing: tunning) bağlıdır. Bu çalışma, LSTM modelinin zaman serisi verilerinde nasıl kullanıldığını, hiper-parametrelerin nasıl optimize edildiğini ve elde edilen sonuçların ne kadar etkileyici olduğunu incelemektedir. Ayrıca, PSO algoritmasının eniyileyicinin öğrenme oranı için nasıl bir iyileştirme sağladığını ve PSO algoritmasının parametrelerinin nasıl ayarlandığını ele almaktadır.

Bu tezde sunulan sonuçlar, zaman serisi analizinde DÖ yöntemlerinin etkinliğini ve PSO algoritmasının hiper-parametre optimizasyonunda nasıl kullanılabileceğini vurgulamaktadır. Bu çalışma, gelecekteki araştırmalara ışık tutmak amacıyla gerçekleştirilmiştir.

Temmuz, 2023 Nurbanu Işık Delibalta

İÇİNDEKİLER

ÖZET ..... **v**
ABSTRACT ..... **vii**
ÖNSÖZ ..... **ix**
SEMBOLLER ..... **xii**
ŞEKİL LİSTESİ ..... **xiv**
TABLO LİSTESİ ..... **xvi**
KISALTMALAR ..... **xvii**
GİRİŞ **1**
BİRİNCİ BÖLÜM **3**
1. TEMEL KAVRAMLAR **3**
1.1. ZAMAN SERİSİ 3
1.1.1. Zaman Serilerinin Sınıflandırılması **5**
1.1.2. Veri Ön İşleme **7**
1.1.3. Hata Ölçütleri **8**
1.2. DERİN ÖĞRENME 9
1.2.1. Yapay Sinir Ağları **10**
1.2.2. Tekrarlayan Sinir Ağları **22**
1.2.3. Uzun Kısa Süreli Bellek Derin Öğrenme Ağları **24**
1.2.3.1. Gözetleme Açıklıklı LSTM **28**
1.2.3.2. Kapılı Tekrarlayan Hücre **28**
1.2.3.3. Çift Yönlü LSTM **30**
1.2.3.4. Dikkat Temelli LSTM **30**
1.2.3.5. Hiyerarşik LSTM **30**
1.2.3.6. Derinlik Kapılı LSTM (ing: Depth Gated) **31**
1.3. META-SEZGİSEL ALGORİTMALAR 31
1.3.1. Parçacık Sürü Optimizasyonu **31**
İKİNCİ BÖLÜM **34**
2. PSO-LSTM MELEZ ÇERÇEVE **34**
2.1. LSTM HİPER-PARAMETRELERİNİN ÖNEMİ 34
2.2. PSO HİPER-PARAMETRE VE ÖĞRENME ORANININ ÖNEMİ 37
ÜÇÜNCÜ BÖLÜM **40**
3. LİTERATÜR ÇALIŞMALARI **40**
3.1. LSTM MİMARİSİ İLE İLGİLİ YAPILAN ÇALIŞMALAR 40
3.2. MELEZ LSTM YAKLAŞIMLARI İLE YAPILAN ÇALIŞMALAR 41

DÖRDÜNCÜ BÖLÜM **44**
4. DENEYSEL ÇALIŞMALAR **44**
4.1. DENEYSEL TASARIM 44
4.2. VERİ SETİ 44
4.3. DEĞERLENDİRME METRİKLERİ 46
4.4. VERİ ÖN İŞLEME 46
4.5. HİPER-PARAMETRE DEĞERLERİ 47
BEŞİNCİ BÖLÜM **50**
5. BULGULAR VE TARTIŞMALAR **50**
5.1. LSTM ve BiLSTM MİMARİLERİ 50
5.1.1. LSTM Mimarileri İçin Hiper-Parametre Ayarı ve Analizleri **51**
5.1.2. LSTM Ait Sapma-Değişkenlik Dengelemesi Analizi **58**
5.2. PSO-LSTM MELEZ ÇERÇEVE 59
5.2.1. PSO-LSTM İçin Hiper-Parametre Ayarı **59**
5.2.2. PSO-LSTM Sapma-Değişkenlik Dengelemesi Analizi **61**
5.2.3. PSO-LSTM'nin Diğer Veri Setleri Üzerinde Başarımı **62**
SONUÇ VE GELECEK ÇALIŞMA **65**
KAYNAKÇA **66**

SEMBOLLER

t : Beklenen değer
a : Elde edilen sonuç değeri
f : Hata fonksiyonu (mse)
N : Toplam veri noktalarının sayısı
i : Veri noktalarının indisleri
e_i : Gerçek değer (t_i) ile tahmin edilen değer (a_i) arasındaki hata
t_i : Gerçek değer (hedef değer) veri setindeki i. veri noktası
a_i : Tahmin edilen değer (modelin çıkışı) veri setindeki i. veri noktasını
t : Zaman
y_t : Gözlem
T_t : Eğilim varyasyonu
S_t : Mevsimsel varyasyonu
C_t : Döngüsel varyasyonu
I_t : Düzensiz varyasyonu
σ(z) : Sigmoid fonksiyonu
z : Giriş değeri veya ağırlıklı toplam değeri
e : Euler sayısı
f(z) : Hiperbolik tanjant (tanh) fonksiyonu
∂L / ∂wt : Modelin mevcut türevi
a : Öğrenme oranı
w_t : Mevcut ağırlık
L : Kayıp fonksiyonu
$\hat{S}_t$: w'nin t zamanındaki ikinci moment tahmininin düzeltilmiş hali
S_t : w'nin t zamanındaki ikinci momentin tahmini
V_t : w'nin t zamanındaki birinci moment tahmini
V_t^ : w'nin t zamanındaki gradyan tahmininin düzeltilmiş hali
∈ : Küçük bir değer
β_1 : Birinci momentin eksponansiyel düşürme oranı
β_2 : İkinci momentin eksponansiyel düşürme oranı
β_1^t : Birinci moment (gradyan) tahmininin eksponansiyel düşürme oranı (ing: exponential decay rate)
β_2^t : İkinci moment (gradyan) tahmininin eksponansiyel düşürme oranı (ing: exponential decay rate)
u_t : Unutma kapısı
r_t : Sıfırlama kapısı

$\mathbf{h_t}$	**:** Gizli durum vektörü
$\mathbf{v_i[t]}$	**:** Parçacığın mevcut durumunu belirten hız vektörü
$\mathbf{w}$	**:** Parçacığın kendi yönündeki hareketinin katsayısı
$\mathbf{x_i[t]}$	**:** Parçacığın mevcut durumunu belirten konum vektörü
$\mathbf{x_{i,best[t]}}$	**:** Parçacığın geçmiş yinelemeler çerçevesinde elde ettiği en iyi konum
$\mathbf{x_{gbest[t]}}$	**:** Tüm parçacıklar kapsamında elde edilen en iyi konum
$\mathbf{c_1}$	**:** Parçacığın kendi geçmişinden gelen öğrenme katsayısı
$\mathbf{c_2}$	**:** Parçacığın toplumun geçmişinden gelen öğrenme katsayısı
$\mathbf{r_1}$ **ve** $\mathbf{r_2}$	**:** (0,1) aralığında atanan rastgele değerleri ifade eden değişkenleri

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1. Durağan olmayan zaman serisi grafiği 6

Şekil 1.2. Durağan zaman serisi grafiği 6

Şekil 1.3. Biyolojik sinir hücresi (Diamantaras ve Kung, 1996; Haykin, 2009) 11

Şekil 1.4. YSA bileşenleri (Öztemel, 2006)..... 12

Şekil 1.5. Sigmoid fonksiyonu görseli (Anderson ve McNeill, 1992)..... 13

Şekil 1.6. Hiperbolik tanjant fonksiyonu görseli (Anderson ve McNeill, 1992)..... 14

Şekil 1.7. ReLu fonksiyonu görseli (Nair ve Hinton, 2010)..... 15

Şekil 1.8.Tek katmanlı algılayıcılar (Öztemel, 2006)..... 18

Şekil 1.9. Çok katmanlı algılayıcılar (Öztemel, 2006) 18

Şekil 1.10. Bir YSA'nın ileri beslemeli topolojisi (Krenker vd., 2011) 19

Şekil 1.11. Bir YSA'nın geri beslemeli topolojisi (Krenker vd., 2011) 19

Şekil 1.12. Denetimli öğrenme algoritmalarının işleyişi (Jahnke, 2015)..... 20

Şekil 1.13. Denetimsiz öğrenme algoritmalarının işleyişi (Jahnke, 2015) 21

Şekil 1.14. Destekleyici öğrenme algoritmalarının işleyişi (Jahnke, 2015)..... 22

Şekil 1.15. RNN yapısı ve tekrarlayan kısmın açık hali (Url-5, 2023)..... 23

Şekil 1.16. LSTM ağın iç yapısı (Url-5, 2015)..... 25

Şekil 1.17. LSTM modeli tahmin adımları (Url-5, 2015)..... 26

Şekil 1.18. GRU bellek hücresi yapısı (Shen ve ark., 2018)..... 29

Şekil 1.19. PSO akış diyagramı 32

Şekil 2.1. Çalışmamızın iş akışı..... 39

Şekil 4.1. Farklı dillerde sayfaların günlük görüntülenme sayıları..... 46

Şekil 5.1. SGD eniyileyicisinin test ve eğitim süreçlerine ait gerçek ve tahmin edilen değerleri..... 53

Şekil 5.2. Farklı eniyileyicilere ait iterasyon adımına karşı kayıp (loss) değerleri.... 59

Şekil 5.3. İngilizce web trafik için PSO-LSTM eğitim ve test tahminleri..... 61

Şekil 5.4. SGD için PSO-LSTM ve LSTM iterasyon adımına karşı kayıp (loss) değerleri..... 62

Şekil 5.5. Japonca web trafik veri seti için PSO-LSTM eğitim ve test tahminleri 64

Şekil 5.6. Almanca web trafik veri seti için PSO-LSTM eğitim ve test tahminleri... 64

TABLO LİSTESİ

Sayfa

Tablo 4.1. Sayfa diline göre makale sayısı 45

Tablo 4.2. Ölçeklendirmeler göre LSTM'in en iyi MSE değerleri..... 47

Tablo 4.3. LSTM ve BiLSTM mimarileri için hiper-parametre değerleri..... 49

Tablo 4.4. Çalışmamızda kullanılan LSTM hiper-parametreleri..... 49

Tablo 5.1. LSTM mimarisinde farklı eniyileyici metotlarına göre başarımı en yüksek hiper-parametre değerleri..... 52

Tablo 5.2. BiLSTM mimarisinde farklı eniyileyici metotlarına göre en iyi sonuç veren hiper-parametre kombinasyonu..... 53

Tablo 5.3. SGD eniyileyicisi için birim sayısı ve yığın boyutuna göre MSE değerleri 54

Tablo 5.4. Birim sayısı 8, iterasyon sayısı 100 olması durumunda farklı yığın boyutuna göre ADAM ve RMSPROP'un MSE değerleri..... 55

Tablo 5.5. Çok katmanlı LSTM SGD eniyileyicisi için eğitim ve test MSE değerleri 57

Tablo 5.6. Çok katmanlı LSTM ADAM ve RMSPROP eniyileyicisi için eğitim ve test MSE değerleri..... 57

Tablo 5.7. LSTM'nin iterasyon sayısı 450 olması durumunda farklı eniyileyicilere göre en düşük test MSE değerlerine göre bulunan hiper-parametrelere değerleri..... 58

Tablo 5.8. Birim sayısı 8, iterasyon sayısı 100, yığın boyutu 16 olması durumunda eniyileyicilerin MSE değerleri 60

Tablo 5.9. PSO-LSTM'nin PSO'ya ait optimum hiper-parametre ve literatür kabul değerleri..... 60

Tablo 5.10. PSO-LSTM'nin farklı veri setleri için PSO'ya ait optimum hiper-parametre değerleri..... 63

Tablo 5.11. LSTM ve PSO-LSTM sonuçları..... 63

KISALTMALAR

ADAM	Adaptive Moment Estimation (Uyarlanabilir Moment Tahmini)
ANOVA	Analysis of Variance (Varyans Analizi)
BiLSTM	Bi-directional LSTM (Çift Yönlü LSTM)
ark.	Arkadaşları
CNN	Convolutional Neural Network (Evrişimsel Sinir Ağı)
DÖ	Derin Öğrenme
ed. veya haz.	Editör/yayına hazırlayan
Globalbestpso	Global Best Particle Swarm Optimization (Genel En İyi Parçacık Sürü Optimizasyonu)
GRU	Gated Recurrent Unit (Kapılı Tekrarlayan Hücre)
ing	İngilizce
LSTM	Long Short-Term Memory (Uzun Kısa Süreli Bellek)
MAE	Mean Absolute Error (Ortalama Mutlak Hatası)
MÖ	Makine Öğrenmesi
MSE	Mean Squared Error (Ortalama Kare Hatası)
PSO	Particle Swarm Optimization (Parçacık Sürü Optimizasyonu)
ReLu	Rectified Linear Unit (Doğrultulmuş Doğrusal Birim)
RMSprop	Root Mean Square Propagation (Kök Ortalama Kare Yayılımı)
RNN	Recurrent Neural Network (Tekrarlayan Sinir Ağı)
s.	Sayfa/sayfalar
SGD	Stochastic Gradient Descent (Stokastik Gradyan İniş)
YSA	Yapay Sinir Ağı

GİRİŞ

Günümüzde, zaman serisi verilerinin analizi ve tahmini birçok alanda büyük önem taşımaktadır. Borsa, havacılık, istatiksel alanlarda, derin öğrenme yöntemleri, zaman serisi analizinde etkili sonuçlar elde etmek için yaygın olarak kullanılan yaklaşım/yöntemlerden biridir. Özellikle Uzun Kısa Süreli Bellek (ing: Long Short Term Memory [LSTM]) modeli, zaman serisi analizi için özel olarak tasarlanmış bir model olarak kabul edilir. Ancak, LSTM modelinin performansı, doğru hiper-parametre ayarlamalarına bağlıdır. Bu nedenle, hiper-parametrelerin optimize edilmesi, modelin başarısını artırmak için kritik öneme sahiptir.

Bu çalışmada, başlangıçta tek yönlü LSTM modeli kullanılarak veri üzerinde gerçekleştirildi. Ardından, çift yönlü LSTM ve çok katmanlı tek yönlü LSTM modelleri de değerlendirildi. Elde edilen sonuçlar incelendiğinde, tek yönlü LSTM modelinin en düşük MSE değerini verdiği gözlemlendi. Bu sonuçlar, tek yönlü LSTM modelinin zamana bağlı veri içerisindeki bağlamsal bağımlılıkları yakalama ve modellenme konusunda diğer modellere kıyasla üstün performans gösterdiğini vurgulamaktadır. Bu nedenle, bu çalışmada ileri analiz ve deneyler için en uygun model olarak seçildi.

LSTM modelinin performansını artırmak ve daha düşük iterasyonlarda SGD eniyileyicisinin iyi sonuçlar vermesini sağlamak için Parçacık Sürü Optimizasyonu (ing: Particle Swarm Optimization [PSO]) ile bir melez çerçeve geliştirilmiştir. ADAM ve RMSPROP gibi eniyileyiciler, adaptif öğrenme oranı sağladığı için daha az iterasyonda düşük MSE değerleri elde edebilirken, SGD yüksek iterasyonlarda daha iyi sonuçlar vermektedir. SGD'nin düşük iterasyonlarda iyi performans göstermesini sağlamak için hiper-parametrelerin doğru bir şekilde ayarlanması zordur. Bu melez çerçeve sayesinde, PSO algoritması kullanılarak LSTM modelinin hiper-parametreleri optimize edilmiştir. PSO, hiperparametre uzayında en iyi değerleri arayarak ve optimize ederek LSTM modelinin daha iyi tahmin performansı elde etmesini sağlamıştır. Böylece, daha düşük iterasyonlarda SGD'nin iyi sonuçlar vermesi ve daha yüksek iterasyonlarda diğer adaptif optimizasyon yöntemlerinin avantajlarından

faydalanılması mümkün olmuştur. **Bu çalışmanın amacı, LSTM modelinin zaman serileri tahmin performansını artırmak için PSO ile melez bir yaklaşımın geliştirilmesi ve bu melez çerçevenin diğer geleneksel eniyileyici yöntemlerine kıyasla daha iyi sonuçlar vermesinin gösterilmesidir. Bu melez çerçeve, zaman serileri analizinde ve diğer sıralı veri uygulamalarında daha iyi performans elde etmek için kullanılabilir. Ayrıca, bu çalışmanın sonuçları, DÖ modelinin optimizasyonunda PSO gibi meta-sezgisel algoritmaların kullanılmasının önemini vurgulayacaktır.**

Bu tez çalışması, DÖ yöntemlerinin zaman serisi verilerinde kullanılmasına odaklanmıştır. Araştırma kapsamında, LSTM modeli zaman serisi analizi için uygulanmış ve modelin hiper-parametreleri optimize edilerek en uygun yapıya ulaşılmıştır. Ayrıca, PSO algoritması kullanılarak optimizerin **en iyi öğrenme oranı** belirlenmiş ve PSO algoritmasının parametreleri optimize edilmiştir.

Yapılan bu çalışmanın ilk bölümü, zaman serisilerine ve DÖ'ye ilişkin bilgilendirmeleri içermektedir. Bu bölümde zaman serilerinin sınıflandırılması, verilerin ön işlenmesi, eniyileyicilerin çeşitleri ve hata ölçütleri açıklanmıştır. Zaman serilerinde DÖ yöntemlerin LSTM'nin çeşitleri incelenerek yöntemlerin süreçleri hakkında bilgiler yer almaktadır. DÖ'nün hiper-parametrelerini optimizasyonu için meta-sezgisel yöntemlerden PSO algoritmasına ilişkin açıklamalar içermektedir.

İkinci bölümünde, birinci bölümde bahsedilen yöntemlerden LSTM ve PSO algoritmalarını kullanarak melez bir çerçeve sunulmuştur. LSTM ve PSO için hiper-parametre optimizasyonu ve öğrenme oranı hakkında bilgilendirmeler yapılmıştır.

Üçüncü bölümde, YSA, PSO-LSTM melez yöntemine ilişkin literatürdeki çalışmalardan bahsedilmiştir.

Dördüncü bölümde, web trafik veri seti üzerinde LSTM ve PSO-LSTM melez yönteminin deneysel çalışmaları yapılmıştır.

Beşinci bölümde ise dördüncü bölümde yapılmış olan deneylerin bulguları ve tartışılması olası durumları anlatılmaktadır.

BİRİNCİ BÖLÜM

1. TEMEL KAVRAMLAR

Bu bölümde zaman serisine ilişkin temel kavramlar ve literatürdeki çalışmalar incelenmiştir.

1.1. ZAMAN SERİSİ

Zaman serisi verileri, düzenli zaman aralıklarında bir değere ait gözlem sonuçlarından oluşmaktadır. Düzenli, ardışık zaman aralıklarında belirli bir tipte ölçüm yapılır. Toplanan veri noktalarının sıklığı saatlik, haftalık, aylık, üç aylık veya yıllık olabilmektedir. Geçmiş veriler analiz edilerek zaman serisi tahmini yapılabilir. Örneğin, istatistik, sinyal işleme, ekonometri ve finans alanlarında gözlem değerlerine bakılarak gelecekte beklenen değerler tahmin edilmektedir (Wei, 2006; Franses ve ark., 2000; Liao, 2005; Granger, 1981; Hamilton, 1989; Scharf ve Demeure, 1991). Aynı zamanda politik bilimler, ekonomi, psikoloji, sosyoloji, biyomedikal istatistik ve meteoroloji alanları için zaman serisi verileri analiz edilerek geleceğe dair öngörülerde (ing: prediction, forecasting) bulunmak mümkündür (Yaffee ve McGee, 2000).

Zaman serisi analizi, belirli bir zaman dilimi içerisinde gerçekleşen olayların ve işlemlerin analiz edilmesine yarayan, ardından bu durumun bir öngörüyle sonuçlanmasını sağlayan sistemdir. Anlık kısa vadeli tahminler, birkaç saat ileriyi, kısa vadeli tahminler, birkaç gün ileriyi ve uzun vadeli tahminler ise çok daha ileriki günleri tahmin edebilmektedir (Seddighi ve ark., 2000).

Genel olarak kısa vadeli tahminler yapılır. Uzun süreli tahminler daha zor olduğu için nispeten daha az tercih edilir. Verilerin zaman içerisinde ardışık bir biçimde olması gerekli bir koşul değildir. Fakat düzenli zaman aralıklarında dizinin gelişimini takip etmesi ve doğru analiz açısından önemlidir (Seddighi ve ark., 2000).

Zaman serisi analizi, belirli bir zaman diliminde verilerin örüntüsünün uygun bir model ile öğrenilmesi ve gelecekteki olası değerlerinin tahminin esasına dayanır (Martinez, ve ark., 2015). Zaman serisi analizi, verinin doğasını anlamaya çalışmakta

ve geleceğe yönelik tahminleri ve simülasyonlar için fayda sağlayacak yöntemleri ortaya çıkarmaktadır (Chatfield, 2016).

Zaman serisi bileşenleri uzun dönem eğilimi (ing: long-term), devresel (ing: cyclical) hareketler, mevsimsel (ing: seasonal) hareketler ve düzensiz (ing: irregular) hareketler olmak üzere 4 temel unsurdan meydana gelmektedir (Chatfield, 2000).

Uzun dönem eğilimi, verinin bir zaman içinde ard arda yukarı doğru büyüme veya aşağı doğru düşüş gösterilmesi durumunda oluşur. İlgili değişkende büyüme veya azalma doğrusal olmayan bir şekilde de gerçekleşebilmektedir. İlgili değişken için zamana bağlı olarak artan oranlarda büyüme veya azalma olur iken, bazen de belirli bir noktadan sonra azalan oranlarda büyüme ve azalma olabilmektedir (Chatfield, 1995).

Devresel hareketleri, zaman içerisinde serinin gözlem değerlerinde periyodik olmayan dalgalanmalar olarak adlandırılır. Bazı zamanlar için uzun dönem eğiliminin üzerinde, bazen de altında olmaktadır. Finansal kriz sebebiyle üretimin azalması, işsizliğin artması örnek verilebilir (Özmen ve ark., 2006).

Mevsimsel değişmeler, birbirini takip eden yılların, mevsimlerin, çeyrek yılların, ayların aynı zaman içinde gözlem değerlerinde düzenli olarak artma veya azalma olması durumudur. Doğa olayları ya da insan yapımı durumlar nedeniyle meydana gelmektedir. Çeşitli mevsimler veya iklim koşulları, mevsimsel değişikliklerin bu örüntülerin ortaya çıkmasında önemli rolü vardır. Yaz aylarında ülkemize gelen turist sayılarında artış, kış aylarında azalış durumu mevsimsel değişmelere örnek verilir (Özmen ve ark., 2006). Mevsimsel dalgalanmalar grafiğindeki dalga uzunluğu ve dalga şiddeti Şekil 1.3'de gösterilmiştir.

Mevsimsel dalgalanmalar veri içerisinden arındırılması gerekmektedir. Eğer mevsimsel dalgalanmarlar zaman serisi içerisinden arındırılmaz ise modelin varyansı yüksek çıkmaktadır (Tüzen, 2012).

Düzensiz değişmeler, rassal sebepler veya geçici olarak meydana gelen hareketlerdir. Deprem, su baskını gibi doğal nedenler ve siyasi karışıklıklar düzensiz değişmelere örnek verilebilir. Düzenli değişim göstermedikleri için hangi zaman ve oranda ortaya çıkacakları önceden tahmin edilememektedir (Serper, 2000).

Zaman serilerindeki dört bileşenin etkileri dikkate alındığında, genellikle toplamsal ve çarpımsal olmak üzere iki farklı model kullanılır. Toplam modeli, zaman

serisi verilerinde yer alan 4 temel unsurun toplanmasından meydana gelmektedir (Folk, 2012; Adhikari ve Agrawal, 2013). Toplam modeline ilişkin formülasyon aşağıda verilmiştir **(1.1.)**.

$$y_t = T_t + C_t + S_t + I_t \quad \textbf{(1.1.)}$$

Burada $\boldsymbol{t}$ anındaki bileşenleri için $\boldsymbol{y_t}$: gözlemi, $\boldsymbol{T_t}$: eğilim, $\boldsymbol{S_t}$: mevsimsel, $\boldsymbol{C_t}$: döngüsel ve $\boldsymbol{I_t}$: düzensiz varyasyonu temsil eder.

Çarpım modeli, zaman serisi verilerinde yer alan 4 temel unsurun çarpılmasından meydana gelmektedir (Folk, 2012; Adhikari ve Agrawal, 2013). Çarpım modeli formülü aşağıda gösterilmiştir **(1.2.)**.

$$y_t = T_t \times C_t \times S_t \times I_t \quad \textbf{(1.2.)}$$

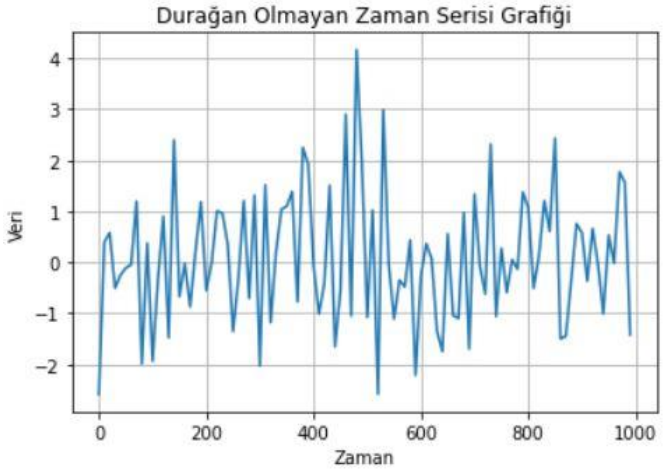
Zaman serilerine ilişkin çalışma adımları aşağıda bahsedilmektedir.

1.1.1. Zaman Serilerinin Sınıflandırılması

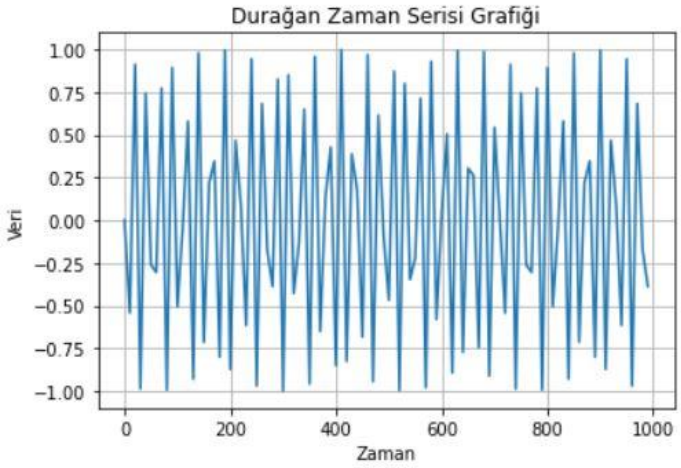
Gözlem değerlerine göre zaman serileri, sürekli ve kesikli zaman serileri olmak üzere ikiye ayrılır. Gözlem değerleri zaman içinde sürekli ise, bu serilere sürekli zaman serileri adı verilir. Genellikle eşit olmayan zaman aralıkları içirisinde alınan gözlem değerleridir. Eğer gözlem değerleri sadece belirli zaman aralıklarında alınıyor ise, bu serilere kesikli zaman serileri denir. Genellikle eşit olan zaman aralıkları içerisinden alınan gözlem değerleridir (Chatfield, 1989). Gözlemlerin sürekli yapıldığı hallerde bile, belirli zaman aralıkları için gözlem değerlerinin ya toplamı alınarak ya da örnekleme yoluyla sürekli seriler kesikli hale dönüştürülebilir (Kayım, 1985).

Zaman serisinde önemli kavramlardan biri de "durağanlık (ing: stationarity)" kavramıdır. Çünkü zaman serisi analizinde kullanılan birçok yöntem ancak seri durağan ise uygulanabilmektedir. Durağanlık, serinin ortalamasında sistematik bir değişme olmaması (trendin olmaması), varyansın sistematik değişiminin olmaması ve periyodik değişmelerin olmaması anlamına gelmektedir. Zaman serisinde, yukarıda sayılan özellikler mevcut ise, durağan olduğundan bahsedilir. Zaman serisinin karakteri zaman içerisinde değişiklik gösteriyorsa diğer bir ifadeyle durağanlık koşullarını sağlamıyorsa bu tür seriler durağan olmayan seriler olarak adlandırılmaktadır (Pindyck ve Rubinfeld, 1991). Durağan olmayan serilerin, durağan hale getirilmesi daha güvenilir tahminler yapmak ve istatistiksel analizlerin doğruluğunu artırmak için önemlidir. Serinin durağan olup olmadığı, serinin zaman

grafiklerinin çizilmesi ile gözlemlenebilir. Şekil 1.1.'de durağan olmayan zaman serisi varyansında, durağanlık görülmemiştir. Şekil 1.2.'de gösterilen durağan bir zaman serisi grafiğinde varyansın durağan olduğu gözlemlenir. Eğer serinin varyansında durağanlık görülmüyorsa, yani varyansta sistematik bir değişme varsa ve bu değişme ortalama ile beraber ise, serinin dönüştürülmesi gerekmektedir. Varyansta durağan olmaması durumunda logaritmik dönüşüm önerilmektedir. Eğer sadece ortalamada durağan olmaması durumu var ise fark-alma dönüşümleri önerilmektedir (McCleay ve Hay, 1983).



Şekil 1.1. Durağan olmayan zaman serisi grafiği



Şekil 1.2. Durağan zaman serisi grafiği

Bir zaman serisinde birbirini takip eden yılların aynı aylarında benzer periyodik hareketler gözlemlenmesi durumunda seri mevsimsel seridir (Kayım, 1985).

1.1.2. Veri Ön İşleme

Zaman serilerinde gelecek tahminin (ing: prediction) etkin ve doğru, yapılabilmesi için uygulanacak öğrenme algoritmalarının yanı sıra, verilerin kaliteli olması yani, veri setinde eksik verilerin olmaması, aykırı değerlerden, gürültüden temizlenmiş olması ve gerekiyorsa normalizasyon işlemlerinin yapılması önem arzetmektedir. İşlem görmemiş (ham) veriden kaliteli veri elde edilebilmesi için probleme ve veri setinin doğasına uygun bir süreçten geçirilmesi gerekmektedir. Yapılan uygulamalarda genellikle bu adım ihmal edilmektedir (Garcia ve ark., 2015). Veri temizleme, veri entegrasyonu, veri dönüşümü – normalizasyon, veri azaltma adımları veri ön işleme için gereklidir (Garcia ve ark., 2015; Awad ve ark., 2015).

Veri temizleme, veri seti içerisinde yer alan gürültü, eksik veya gerekli olmayan verilerin düzenleme adımıdır. Veri setinde yer alan hatalı verilere gürültülü veri denir. Gürültülü veri öncelikle tespit edilmelidir. Düzenlenebilecek veriler düzenlenir. Eğer veri düzeltilemiyor ise veri setinden çıkarılmalı veya gürültüye duyarlılığı az olan algoritmalar kullanılmalıdır. Eksik verinin bulunduğu tarih veri setinden çıkarılabilir, fakat bu yöntem veri kaybına sebep olur (Garcia ve ark., 2015; Awad ve ark. 2015; Kotsiantis ve ark., 2006). Eksik olan verilerin yerine 0 (sıfır) yazılarak doldurma işlemi de yapılabilmektedir. Eksik veri sayısının az olduğu durumlar için başarılı sonuçlar elde edilir. Eksik veriler, bir önceki ve/veya sonraki veri ile de doldurulabilmektedir (Hsu, 2021). Verileri temizlemek için bir diğer yöntem ise aritmetik ortalamadır. Eksik verinin bulunduğu elemanların değerlerinin tümü toplanır ve eleman sayısına bölünerek hesaplanır. Doğrusal enterpolasyon yöntemi ile veri setindeki eksik veriler doldurulabilir. Bilinen iki nokta arasındaki bir değerin bulunmasını sağlar (Gilik ve ark., 2022).

Veri entegrasyonu, birden fazla veri setini birleştirme adımıdır. Birleştirme kısmında tekrar eden veriler veya tutarsız veriler oluşabilmektedir. Veri setinin genişlemesi avantaj sağlarken, maliyetin artması dezavantaj sağlamaktadır (Garcia ve ark., 2015; Awad ve ark., 2015).

Veri dönüşümü – ölçeklendirme (normalizasyon), veri setini oluşturan orijinal değerler ile derin öğrenme yöntemleri uygulandığında doğru sonuçlar vermeyebilmektedir. Bu durumda tahmin başarısını arttırmak için yeni özellikler oluşturmadan uygun özelliklerin yer aldığı yeni değer dizisi meydana getirilir (Garcia

ve ark., 2015; Awad ve ark., 2015). Veri setindeki dağılımın düzenli olması ölçeklendirme ile sağlanır. Veri setindeki değerler çok büyük veya çok küçük olabilmektedir. Bu değerler, ağın yanlış yönlendirilmesine sebep olabilmektedir. Tüm verileri ölçeklendirildiğinde, aykırı değerlerin etkisi azalır ve aynı ölçekte yer alınır. Farklı ölçeklendirme yöntemleri kullanılabilir. Bunlardan bazıları aşağıda verilmiştir (Öztemel, 2003).

- **Min-Max ölçeklendirmesi,** her bir özellik için ayrı çalışır. Veri kümesin her bir özelliği belirli bir aralığa ölçeklendirilir. Ölçeklendirme, minimum ve maksimum değerler arasında uygulanır. Varsayılan olarak minimum değer 0, maksimum değer 1 olarak belirlenir (Url-1, 2023).
- **Standard ölçeklendirme,** özelliklerin ortalamasını 0 ve standard sapmasını 1 olmasını sağlar. Ölçeklendirme ile veri özellikleri arasındaki farklı ölçeklendirme ve dağılım problemleri ortadan kalkar. Veri kümesindeki her bir özellik için ayrı uygulanabilir (Url-2, 2023).
- **Robust ölçeklendirme,** özellikleri ölçeklendirilmesi için medyanı ve yüzdelik aralıkları kullanılır. Özelliklerin ortanca ve çeyreklerine baz alınır, böylelikle aykırı değerlerin etkisi azalır. Her bir özellik için ayrı uygulanabilir (Url-3, 2023).

Veri azaltma, çok büyük veri kümeleri için kullanılır. Çok büyük veri kümelerinde analiz süreleri çok uzun olmaktadır. Orijinal veri seti özellikleri ve bütünlüğü korunarak veri miktarı azaltılmaktadır (Toprak, 2023). Ayrıca, tüm veri setini kullanmak yerine, veri setinden alınan örneklem (ing: sampling) ile model oluşturmak da mümkündür (Estabrooks, 2004).

1.1.3. Hata Ölçütleri

Tahmin yöntemlerinin gerçek hayatta hatasız sonuçlar vermesi oldukça güçtür. Bu sebeple algoritmanın sağlam (ing: robust) ve güvenilir çıktılar üretmesi durumu sinir ağlarını sınırlamaktadır. Eğitim verilerine benzer veriler ile test edildiğinde doğru sonuçlar vermektedir. Fakat yeni veriler işleme girdiğinde beklenen sonuç ile çıkış sonucu arasında önemli farklar olmaktadır. Modelin sonuçlarının uyumunu ölçmek için hata fonksiyonları kullanılır (Bishop, 1994).

Ortalama kare hatası (ing: Mean Square Error [MSE]), sinir ağın eğitimi kısmında, ağların bağlantı ağırlık değerlerinin ayarlanması ile sağlanmaktadır. Bu hata hesaplanırken, modelin beklenen çıktı sonucu ile elde edilen çıktı sonucu arasındaki farkların kareleri toplanır. Bu sayede, sinir ağın başarımı incelenir. Denklem **(1.3.)**'de formülasyonu verilmiştir (Demuth ve ark., 2009).

$$\boldsymbol{f = mse = \frac{1}{N}\sum_{i=1}^{N} e_i^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2} \qquad \textbf{(1.3.)}$$

Burada, $\boldsymbol{t}$ beklenen değeri, $\boldsymbol{a}$ elde edilen sonuç değerini ifade eder. $\boldsymbol{f}$: Hata fonksiyonu (MSE), $\boldsymbol{N}$: Toplam veri noktalarının sayısı, $\boldsymbol{i}$: Veri noktalarının indisleri, $\boldsymbol{e_i}$: Gerçek değer ($\boldsymbol{t_i}$) ile tahmin edilen değer ($\boldsymbol{a_i}$) arasındaki hata, $\boldsymbol{t_i}$: Gerçek değer (hedef değer) veri setindeki i. veri noktası, $\boldsymbol{a_i}$: Tahmin edilen değer (modelin çıkışı) veri setindeki i. veri noktasını göstermektedir.

Negatif değerlerden kurtulmak amacı ile elde edilen fark değerlerinin karesi alınır. Beklenen değer ile elde edilen değerlerin farklarının toplamını en aza indirmek temel amaçlarındandır (Hagan ve ark., 1996).

Ortalama mutlak hatası (ing: Mean Absolute Error [MAE]), sinir ağın eğitimi sırasında, tahmin edilen sonuç ile gerçek sonuç değerleri kıyas edilmelidir. Denklem **(1.4.)**'de formülü ile ortalama mutlak hatası hesabı gösterilmiştir (Wang ve ark., 2016).

$$\boldsymbol{f = mae = \frac{1}{N}\sum_{i=1}^{N}|t_i - a_i|} \qquad \textbf{(1.4.)}$$

Burada, $\boldsymbol{t}$ beklenen değeri, $\boldsymbol{a}$ elde edilen sonuç değerini ifade eder. $\boldsymbol{f}$: Hata fonksiyonu (MAE), $\boldsymbol{N}$: Toplam veri noktalarının sayısı, $\boldsymbol{i}$: Veri noktalarının indisleri, $\boldsymbol{t_i}$: Gerçek değer (hedef değer) veri setindeki i. veri noktası, $\boldsymbol{a_i}$: Tahmin edilen değer (modelin çıkışı) veri setindeki i. veri noktasını göstermektedir.

1.2. DERİN ÖĞRENME

DÖ, makine öğrenmesinin (MÖ) bir alt başlığıdır. Derinlik ifadesi ile ardışık katmanlar anlatılmak istenmiştir. DÖ modelleri, verileri bir süzgeçten geçirir gibi art arda artacak şekilde tasarlanmış ve iyileştirilmiş veri filtreleri uygulanacak katmanlardan oluşmaktadır. Katmanlar, verilerin ifade ettiği örüntüleri öğrenmek için birarada kullanılır (LeCun ve ark., 2015).

DÖ, denetimli ve denetimsiz, özellik çıkarma, dönüştürme, görüntü analizi, sınıflandırma ve tahmin gibi doğrusal olmayan analizi, gizli katmanlardan faydalanarak analiz eder. Bu sebeple MÖ sınıfına dahil edilmektedir (Deng ve Yu, 2014).

MÖ, fonksiyonları içerisinde DÖ fonksiyonları da yer alır. Verinin etiketlenmemiş ve yapılandırılmamış olmalıdır. Bu verilerden denetimsiz öğrenebilen ağlara uygun olan MÖ yapısını kullanır (Bengio, 2009).

DÖ algoritmaları, çok fazla hiper-parametrelere sahiptir. En iyi sonucu elde edebilmek için çok yönlü hiper-parametreler kullanarak başarılı tahminler elde eder. Bu modeller sayesinde büyük veri setlerini işlemesi daha kolay olmaktadır (Awad, 2015; Subasi, 2020).

DÖ modelleri, hatırlanması gerekli olmayan verilerin unutulması, öğrenilen bilgiyi anlamlandırılması ve kullanılması, geçmiş bilgileri gelecekte ilişkilendirmesi yetkinliklerine sahip olarak tasarlanmıştır. DÖ'de yer alan katmanlar ve katmanların sahip olduğu nöronlar ile öğrenme işlemi yapılır. Katmanlar girdi bilgisinden yeni bilgiler elde eder. Bu bilgilerin değerli olup olmaması durumuna karar verir. Bu özellikleri sebebiyle daha doğru sonuçlara ulaşmaktadır (Subasi, 2020; LeCun ve ark., 2015). DÖ algoritmaları, doğal dil işleme, sosyal medya, görüntü işleme alanlarında aktif rol alır (LeCun ve ark., 2015).

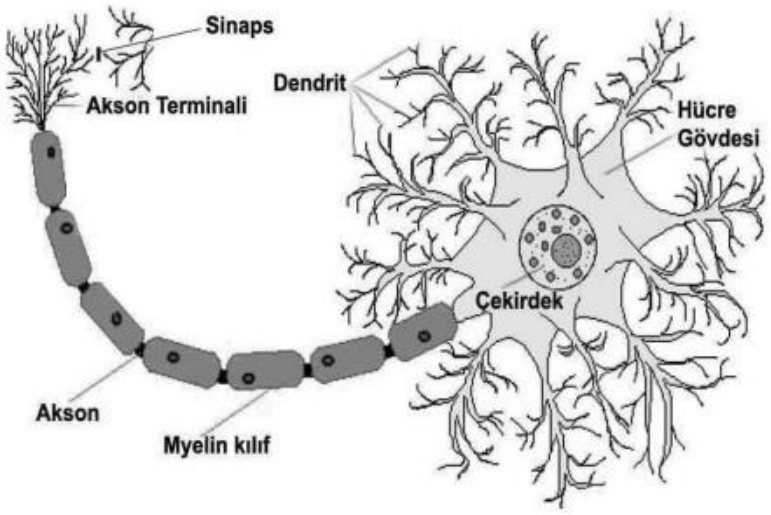
DÖ yaklaşımlarının alt mimarisi yapay sinir ağlarıdır (ing: artificial neural network) (Schmidhuber, 2015).

1.2.1. Yapay Sinir Ağları

YSA'ların, insan beyni ile benzer özellikleri bulunmaktadır. Basit birimlerin paralel çalışması ile hesaplanan bir modeldir (Patterson ve ark., 2017). İnsan beyninde olduğu gibi paralel işlemler yapabilmektedir. Nöron adı verilen yapılar ile hızlı bir şekilde hesaplamalar gerçekteştirmektedir (Haykin, 2009).

YSA modelleri örnekler üzerinden öğrenme yapabilmekte, veriler arasındaki ilişkileri bulabilmekte ve eksik gözlemleri doğru olarak tespit edebilmektedir. YSA, geçmiş dönem verilerin modellerini öğrenerek gelecekteki verilerin tahmin edilmesine imkan sağlayan ve çok değişkenli doğrusal olmayan bir yöntemdir (Cheng – Titterington, 1994).

İnsan sinir sisteminin ana merkezinde beyin yer alır. Birbirlerine ağlar ile bağlanırlar. Bu alt ağlara bağlı nöronlar (sinir hücresi) yer almaktadır. Nöronlar farklı şekil ve büyüklüklere sahiptirler. Hücre gövdesi, dentrit, akson ve akson terminalleri ortak olarak bulunur. Şekil 1.3.'de biyolojik sinir hücresine ait akson, dendrit, çekirdek kısımları gösterilmiştir (Diamantaras ve Kung, 1996; Haykin, 2009).



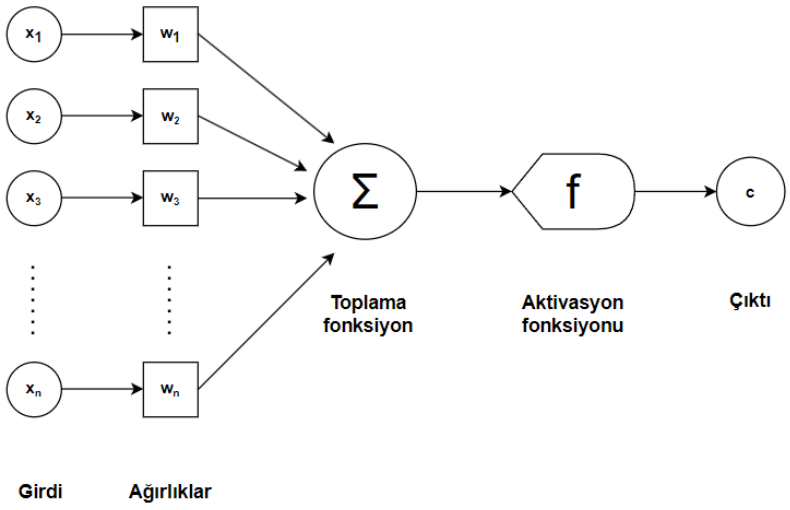
Şekil 1.3. Biyolojik sinir hücresi (Diamantaras ve Kung, 1996; Haykin, 2009)

Dendritler, hücrenin gövde kısmında dışa doğrudur. Diğer hücrelerin akson sonlarından sinyal almaktadır (Guyton ve Hall, 2006). Dendritlere bir hücrenin girdi kanalları adı verilir (Anderson ve McNeill, 1992). Hücre gövdesine dendritler sayesinde gelen sinyali işler. Bu sinyali çıktıya dönüştürür. Çıktılar başka bir hücrenin girdisi olabilir (Anderson ve McNeill, 1992). Sinaps, iki hücre arasındaki fonksiyonel birimdir (Guyton ve Hall, 2006).

YSA'larda, bilginin işlenmesi nöronlar ile sağlanır. Hücreler arası bağlantı ile sinyaller geçiş yapar. Her bağlantının bir ağırlığı vardır. İşlem sonuçlarında çıktı değerleri elde edilebilmesi için transfer fonksiyonları bulunur (Fausett, 1994). Biyolojik sinir ağlarında nöronlar, işlem elemanına (düğüm), sinapslar, işlem elemanları arasındaki bağlantı ağırlıklarına, dendritler birleştirme fonksiyonuna, hücre gövdesi transfer fonksiyonuna, akson ise işlem elemanları çıkışına karşılık gelmektedir (Elmas, 2003).

YSA'da bağlantıları sağlayabilmek için katmanlar kullanılmaktadır. Sinir ağı, girdi katmanı, ara katmanı ve çıktı katmanından oluşur. Girdi katmanı hücrelere gelen verileri işlem uygulamadan ara katmana iletmektedir. Giriş değişkenleri katmandaki hücre sayısı ile aynı sayıda olup her hücre verileri sonraki katman hücrelerine iletimini sağlamaktadır. Ara katmanda işlenen veriler çıktı katmanına iletilir. Ara katmanda hücrelerin ağırlıkları bilinmemektedir. Bu sebeple gizli katman olarak ifade edilir. İdeal ara katman sayısı verilere göre belirlenmektedir. Ara katmanda nöron sayısı az olması durumu çıktığının güvenirliğini etkilerken fazla olması ise modelin ezberlenmesine neden olmaktadır. Ara katmandan gelen veriden işlenen çıktının dışarı iletildiği katmana çıktı katmanı adı verilir. Modelin değişken sayısına göre katmandaki hücre sayısı belirlenir (Öztemel, 2006).

Bileşenleri Şekil 1.4.'da gösterilen YSA'ı (Öztemel, 2006), girdi, ağırlıklar, toplam fonksiyonları, aktivasyon fonksiyonları ve çıktı bölümlerinden oluşmaktadır.



Şekil 1.4. YSA bileşenleri (Öztemel, 2006)

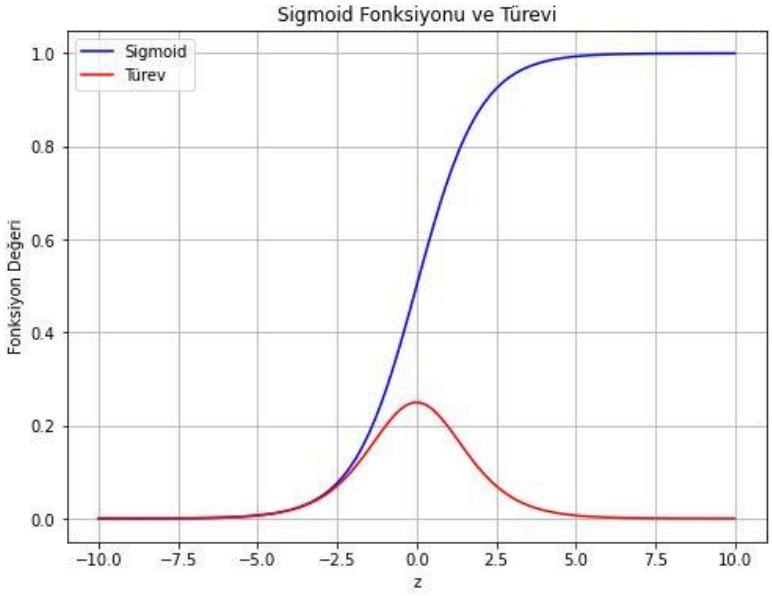
"Girdi" katmanına ilk olarak veriler gönderilir ve çıktılar diğer bir nöronun girdisi olmaktadır (Elmas, 2003). Girdilerin katsayılarına ağırlık denir. Bir nörondan çıkan veri diğer bir hücrenin nöronuna girdi oluyor ise ağırlıkları olmalıdır. Ağırlık değerlerine göre yorum yapılmamalıdır. Küçük ağırlık değerleri bir model için değersiz bir etkene sahip olduğunu gösterebilir ya da daha değerli bir etkene sahip olabilmektedir. Toplama fonksiyonu, yapay sinir hücresine net olarak giren veriyi hesaplamaktadır. Modelin durumuna uygun bir toplam fonksiyonu ancak deneme

yanılma yöntemi ile bulunabilir (Öztemel, 2006). YSA'da aktivasyon fonksiyonu, toplam fonksiyonu aracılığıyla hücreye gelen değerleri işleme alarak hücrenin çıktısını [0,1] veya [-1,1] aralığına getirmektedir (Yıldız, 2009). Çıktı değerlerinin eğitim sırasında büyük değerlere varmaması ise aktivasyon fonksiyonunun amaçlarındandır. YSA hücrelerinin hepsine aynı aktivasyon fonksiyonu kullanılabilirken hepsine farklı aktivasyon fonksiyonları da kullanılabilmektedir. Farklı amaçlara yönelik farklı aktivasyon fonksiyonu kullanılmaktadır (Öztemel, 2006). Yaygın olarak kullanılan aktivasyon fonksiyonları bir kısmı ana hatları ile aşağıda anlatılmaktadır.

Sigmoid Fonksiyonu, en sık kullanılan aktivasyon fonksiyonudur. Diğer fonksiyonlardan farklı olan özelliği sayıları 0 ile 1 arasına sınırlandırmasıdır. Fakat çıktı değeri 0 veya 1 olmamaktadır. Denklemi (1.5.)'de sigmoid fonksiyonu formülü verilmiştir. Şekil 1.5.'de görüldüğü üzere 0 ve 1'e yakınsamaktadır.

$$\boldsymbol{\sigma(z) = \frac{1}{1+ e^{-z}}} \qquad \textbf{(1.5.)}$$

Burada, ***σ(z)***: Sigmoid fonksiyonu, giriş değeri (z) üzerinden bir aktivasyon değeri hesaplamak için kullanılır. ***z***: Giriş değeri veya ağırlıklı toplam değeri, ***e***: Euler sayısı ifade eder.



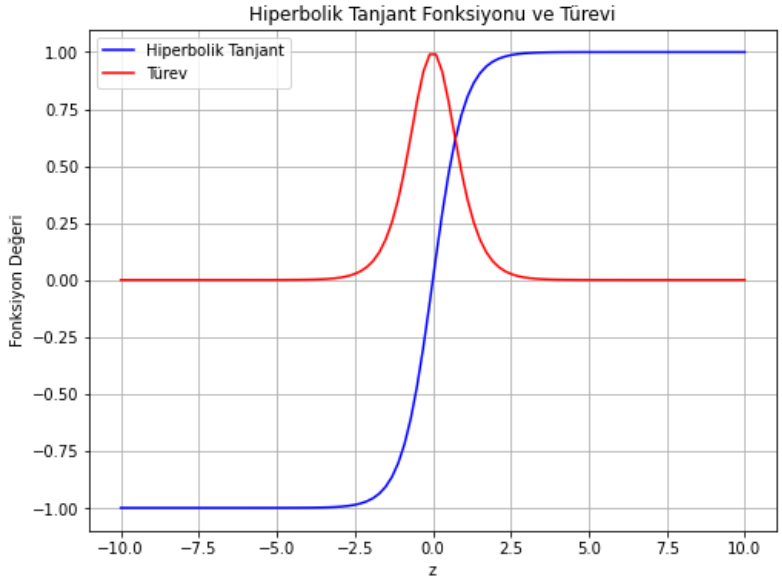
Şekil 1.5. Sigmoid fonksiyonu görseli (Anderson ve McNeill, 1992)

Hiperbolik Tanjant Fonksiyonu, sigmoid fonksiyonuna benzer yapıya sahiptir. Toplam girdiler tanjant fonksiyonu ile işleme alınır (-1,1) arasında değerler

elde edilir. Türev değerleri daha fazla değer alabildiği Şekil 1.6.'da gösterilmiştir. Hiperbolik tanjant fonksiyonuna ait denklem (**1.6.)**'da gösterilmiştir (Haykin, 1999; Demuth ve ark. 2014).

$$\boldsymbol{f(z) = \frac{e^{z} + e^{-z}}{e^{z} - e^{-z}}} \qquad \textbf{(1.6.)}$$

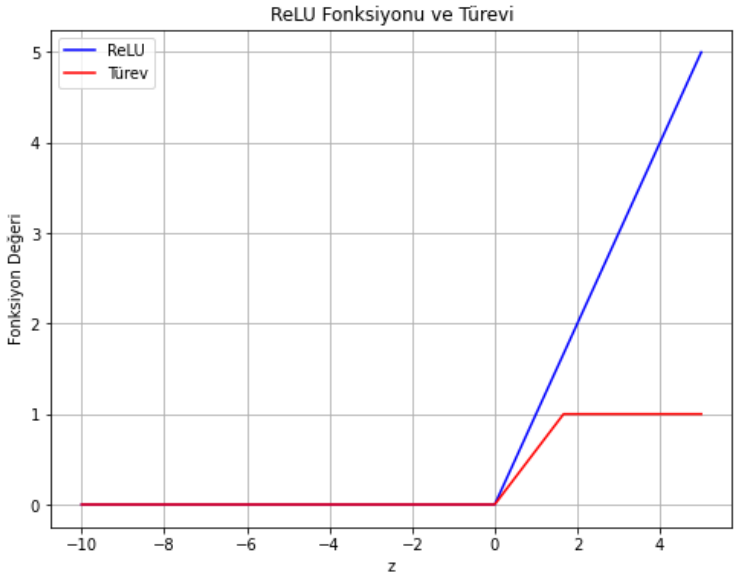
Burada, ***f(z)***: Hiperbolik tanjant (tanh) fonksiyonu, giriş değeri (z) üzerinden bir aktivasyon değeri hesaplamak için kullanılır. ***z***: Giriş değeri veya ağırlıklı toplam değeri, ***e***: Euler sayısı ifade eder.



Şekil 1.6. Hiperbolik tanjant fonksiyonu görseli (Anderson ve McNeill, 1992)

Doğrutulmuş Lineer Birim Fonksiyonu (ing: Rectified Linear Unit [ReLu]), [0,+∞) arasında değer alır (Jin ve ark., 2015). Bu fonksiyon nöron sayısının fazla olduğu ve çok katmanlı ağlarda daha çok kullanılır (LeCun ve ark., 2015). Şekil 1.7.'de gösterildiği üzere $x < 0$ olduğunda değeri 0 iken, $x \geq 0$ olması durumunda doğrusal bir fonksiyon olmaktadır. Kayıp fonksiyonunun mimimum değerine gradyan inişinin yakınsamasının hızlı olmasını sağlar. ReLu, ağ yapısının son katmanında da kullanılabilir. Burada sınıflandırma fonksiyonu olarak görev yapar (Agarap, 2018). Sınıflandırma fonksiyonu Denklem **(1.7.)**'de formüle edilmiştir (Nair ve Hinton, 2010).

$$\boldsymbol{f(z) = \max(0, z)} \qquad \textbf{(1.7.)}$$



Şekil 1.7. ReLu fonksiyonu görseli (Nair ve Hinton, 2010)

DÖ, daha sağlam ve tutarlı sonuç elde etmek amacı ile aktivasyon fonksiyonlarının yanı sıra yinelemeli olarak çalışması gerekmektedir. Çünkü, ağın ağırlık çeşitleri içerisinde hangisinde daha iyi sonuç elde edildiği bulunmalıdır. Bu sebeple, farklı kombinasyonlar için model denemeleri yapılmalıdır (Le ve ark., 2011). Bu sebeple **eniyileyici yöntemler** kullanılmaktadır. Stokastik gradyan inişi (ing: Stochastic Gradient Descent [SGD]), adaptif moment tahmini (ing: Adaptive Moment Estimation [ADAM]), kök ortalama kare yayılımı (ing: Root Mean Square Propagation [RMSprop]) **eniyileyici yöntemlerden (ing: Optimizer)** bazılarıdır. Bu yöntemlere ilişkin açıklamalar aşağıda yer almaktadır.

Stokastik gradyan inişi, modeli ağırlık güncellemelerini gerçekleştirirken tüm türev değerlerini hesaplamak yerine rastgele bir alt kümesini kullanır (Seyyarer ve ark., 2022; Yazan ve Talu, 2017). Türev hesabı ile stokastik gradyan inişini Denklem **(1.8.)**'de gösterilmiştir (Zeiler, 2012; Ruder, 2016; Lydia ve ark., 2019).

$$w_t = w_t - a\,\frac{\partial L}{\partial w_t} \qquad \textbf{(1.8.)}$$

Burada, $\boldsymbol{L}$ kayıp fonksiyonu, $\boldsymbol{a}$ öğrenme oranı (ing: learning rate), $\mathbf{w_t}$ mevcut ağırlığı göstermektedir.

Adaptif moment tahmini, 2014 yılında King ve Ba tarafından önerilen bir yöntemdir. Momentum ve RMSprop'un birleşimini temsil eder. Bu yöntem, iki temel

hesaplama farklılığını bir araya getirir (Kingma ve Ba, 2014). Bias düzeltmeleri Denklem **(1.9.)** de gösterilmiştir.

$$w_{t+1} = w_t - \frac{a}{\sqrt{\hat{S}_t + \epsilon}} \ .V_t \qquad \textbf{(1.9.)}$$

Burada $\boldsymbol{w_{t+1}}$: w'nin (t+1) zamanındaki değeri, yani güncellenmiş ağırlık değeri, $\boldsymbol{w_t}$: w'nin t zamanındaki değeri (mevcut ağırlık değeri), $\boldsymbol{a}$: Öğrenme oranı, $\boldsymbol{\hat{S}_t}$: w'nin t zamanındaki ikinci moment tahmini (ing: second moment estimate) veya gradyanın karesel ortalamasının tahmini ifade eder. Burada, $\boldsymbol{\epsilon}$: Küçük bir değer, genellikle sayısal istikrar ve sıfıra bölme hatası önlemek için kullanılır. V_t: w'nin t zamanındaki gradyanın tahminini ifade eder. Denklem **(1.10) - (1.13.)** $\boldsymbol{V_t^{\wedge}}$ ve $\boldsymbol{\hat{S}_t}$ değerlerinin hesaplamaları verilmiştir.

$$V_t^{\wedge} = \frac{V_t}{1 - \beta_1^t} \qquad \textbf{(1.10.)}$$

Burada, $\boldsymbol{\beta_1^t}$: Birinci moment (gradyan) tahmininin eksponansiyel düşürme oranı (ing: exponential decay rate), $\boldsymbol{V_t^{\wedge}}$: w'nin t zamanındaki gradyan tahmininin düzeltilmiş halini ifade eder.

$$\hat{S}_t = \frac{S_t}{1 - \beta_2^t} \qquad \textbf{(1.11.)}$$

Burada, $\boldsymbol{\beta_2^t}$: İkinci moment (gradyan) tahmininin eksponansiyel düşürme oranı (ing: exponential decay rate), $\boldsymbol{S_t}$: w'nin t zamanındaki ikinci momentin tahmini, $\boldsymbol{\hat{S}_t}$: w'nin t zamanındaki ikinci moment tahmininin düzeltilmiş hali ifade eder.

$\boldsymbol{V_t}$ ve $\boldsymbol{S_t}$ aşağıdaki gibi hesaplanabilmektedir:

$$V_t = \beta_1 V_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial wt} \qquad \textbf{(1.12.)}$$

$$S_t = \beta_2 S_{t-1} + (1 - \beta_2) \left[\frac{\partial L}{\partial w_t}\right]^2 \qquad \textbf{(1.13.)}$$

Burada, $\boldsymbol{\beta_1}$: Birinci momentin eksponansiyel düşürme oranı, $\boldsymbol{V_t}$: w'nin t zamanındaki birinci moment tahmini, $\boldsymbol{\beta_2}$: İkinci momentin eksponansiyel düşürme oranını göstermektedir.

Başlangıç değerleri $\boldsymbol{V}$ ve $\boldsymbol{S}$'nin 0'dır. Varsayılan değer olarak, $\boldsymbol{a}$ = 0.001, $\boldsymbol{\beta_1}$ = 0.9, $\boldsymbol{\beta_2}$ = 0.999, $\boldsymbol{\varepsilon}$ = 10^{-8}'dir (Kingma ve Ba, 2014).

Kök ortalama kare yayılımı, adaptif moment tahmine çok benzerdir. Momentumlu ve momentumsuz versiyonları vardır. Momentumlu versiyonunda parametre güncellemelerini yeniden ölçeklenmiş gradyan üzerinde momentum

kullanarak yapar (Kingma ve Ba, 2014). Denklem **(1.14.)**, Denklem **(1.15.)** ile RMSprop, ağırlık güncellemelerinde ikinci moment tahminini kullanarak gradyanın büyüklüğünü normalize eder. İkinci moment tahmini, gradyanın ikinci momentini hesaplayarak gradyanın dalgalanmasını ve büyüklüğünü ölçer. Ağırlık güncellemesi, gradyanın ağırlıklandırılmış bir versiyonu ile ikinci moment tahmini arasındaki çarpımın öğrenme hızı ve normalizasyon terimiyle çarpılmasıyla yapılır. Bu güncelleme, gradyanın büyüklüğüne bağlı olarak öğrenme hızının ayarlanmasını sağlar.

$$w_{t+1} = w_t - \frac{a}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t} \quad \textbf{(1.14.)}$$

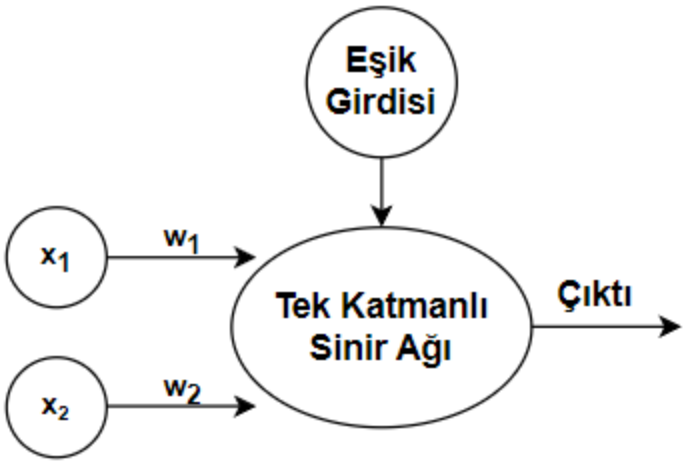
$$S_t = \beta S_{t-1} + (1 - \beta) \left[\frac{\partial L}{\partial w_t}\right]^2 \quad \textbf{(1.15.)}$$

Burada $\boldsymbol{w_t}$: t zamanındaki ağırlık (weight) değeri, $\boldsymbol{w_{t+1}}$: (t+1) zamanındaki ağırlık değeri, $\boldsymbol{a}$: Öğrenme hızı, $\boldsymbol{S_t}$: t zamanındaki ikinci moment tahmini, $\boldsymbol{S_{t-1}}$: (t-1) zamanındaki ikinci moment tahmini, $\boldsymbol{\beta}$: İkinci momentin eksponansiyel düşürme oranı, $\boldsymbol{(\partial L / \partial w_t)}$: t zamanındaki gradyan, $\boldsymbol{\varepsilon}$: Düzeltme terimini ifade eder.

YSA'larda çeşitli veri setleri için farklı modeller uygulanabilmektedir. Ağ modelleri ***katman sayısına, bağlantı şekillerine, öğrenme şekillerine*** göre sınıflandırılmaktadır (Çaparoğlu, 2021). Bu sınıflandırmalar aşağıda kısaca özetlenmektedir.

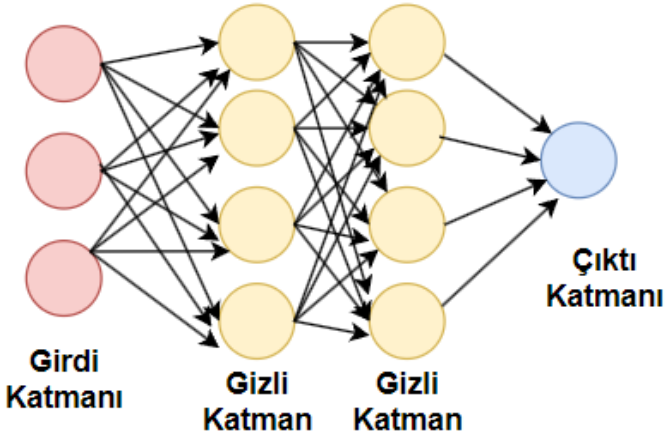
Katman sayılarına göre tek katmanlı algılayıcılar ve çok katmanlı algılayıcılar olarak sınıflandırılabilir.

Tek katmanlı algılayıcılar, girdi ve çıktı katmanından oluşmaktadır. Bu tür algılayıcılarda gizli katman bulunmamaktadır (Çaparoğlu, 2021; Öztürk ve Şahin, 2018). Şekil 1.8.'de tek katmanlı YSA'ya ait bileşenler gösterilmiştir.



Şekil 1.8.Tek katmanlı algılayıcılar (Öztemel, 2006)

Çok katmanlı algılayıcılar, giriş katmanı ve çıkış katmanı arasındaki bir veya birden fazla gizli katman bulundurur (Fausett, 1994). Tek katmanlı yapıdan farklı olarak çok katmanlı yapılarda da gizli katmanları bulunmaktadır (Smith, 2002). Çok katmanlı ağlar, tek katmanlı ağlara göre daha zor problemleri çözebilmektedir. Fakat tek katmanlı ağlara göre verinin eğitimi oldukça güçtür (Fausett, 1994). Şekil 1.9.'da çok katmanlı YSA'ya ait girdi, gizli ve çıktı katmanları görseli verilmiştir.

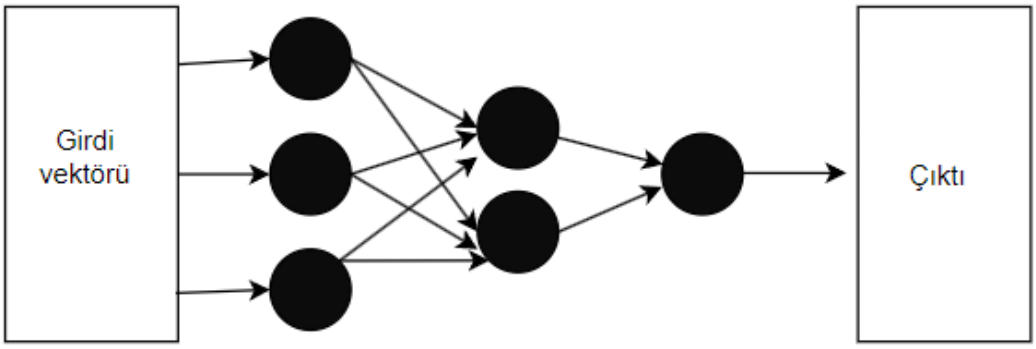


Şekil 1.9. Çok katmanlı algılayıcılar (Öztemel, 2006)

YSA, bağlantı şekillerine göre ileri beslemeli ve geri beslemeli ağlar olarak sınıflandırılabilir.

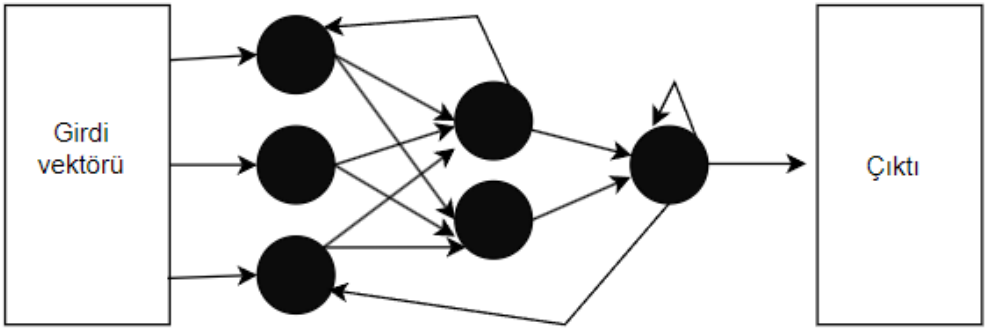
İleri Beslemeli Ağlar, verilerin bilgi akışları giriş katmanı, gizli katman ve çıkış katmanına doğru tek yönlü ilerleyen ağlara denir. Sadece sıralı olarak

katmanlardan ileri doğru akış bulunur (Öztürk ve Şahin, 2018; LeCun ve ark., 2015). Şekil 1.10.'da verilen ileri beslemeli sinir ağları topolojisi gösterilmiştir.



Şekil 1.10. Bir YSA'nın ileri beslemeli topolojisi (Krenker vd., 2011)

Geri Beslemeli Ağlar, her katman sadece bir sonraki katman ile bağlantılı değildir. Bir katman kendinden önce gelen bir katman ile de bağlantı kurabilmektedir. Hata oranlarına göre ağırlıklar tekrar tekrar güncellenir ve en iyi sonuçlar elde edilir (Gilik ve ark. 2022; LeCun ve ark., 2015). Şekil 1.11.'de verilen geri beslemeli sinir ağları topolojisi gösterilmiştir.



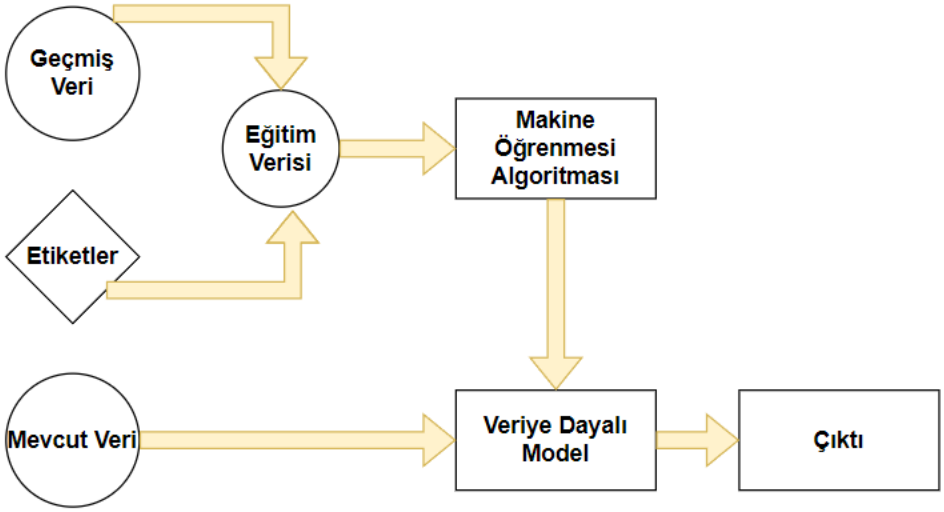
Şekil 1.11. Bir YSA'nın geri beslemeli topolojisi (Krenker vd., 2011)

Geri beslemeli YSA'lar, ileri beslemeli YSA'lara kıyasla daha zengin yapıya sahip modeller geliştirmek için kullanılabilir. Ancak geri beslemeli ağların uygulanması akademik ve pratik alanda daha zordur. Bunun sebebi, geri beslemeli ağların uygulandıktan sonra kullanılabilirliğinin güç olmasıdır. Geri beslemeli ağlar farklı yapılarla oluşturulabilir. Bu da belirli bir model yapısına odaklanmayı zorlaştırabilir ve eğitim algoritmalarının birbiri ile uyumlu olmaması sebebiyle eğitiminin zorluğuna

yol açabilir. Bu nedenle, genellikle ileri beslemeli ağlar daha yaygın olarak kullanılmaktadır (Zhang, 2003).

Öğrenme şekillerine göre YSA'lar, denetimli öğrenme (ing: supervised learning), denetimsiz öğrenme (ing: unsupervised learning) ve destekleyici öğrenme (ing: reinforcement learning) olmak üzere sınıflandırılmaktadır.

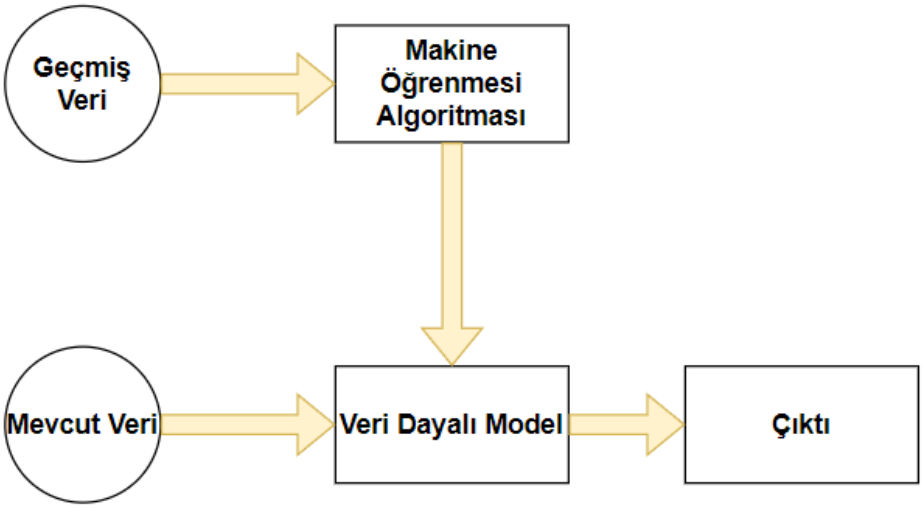
Denetimli Öğrenme, YSA'ya örnek girdi verileri ve hedef çıktı verilerini içeren veri seti eğitilmesi için verilir (MacKay, 2003). Genellikle, hataların hesaplanması için MSE ve MAE gibi performans ölçütleri kullanılır. Bu ölçütler, YSA tarafından üretilen çıktılar ile hedef çıktılar arasındaki hata paylarını hesaplayarak ağın performansını değerlendirir. Ağın ürettiği çıktılar ile hedef çıktılar arasındaki hata oranlarını minimize etmek için bağlantı ağırlıkları düzeltilir. Denetimli öğrenme yöntemiyle YSA, verileri işleyerek kendi çıktılarını üretir ve gerçek çıktılarla karşılaştırır. Öğrenme süreci sayesinde, hataları en aza indirmek için bağlantı ağırlıkları yeniden ayarlanarak YSA'nın denetimli öğrenmeye benzemesi hedeflenir (Haykin, 1999). Şekil 1.12.'de, denetimli öğrenme algoritmasının işleyişi gösterilmektedir (Bayramoğlu, 2007).



Şekil 1.12. Denetimli öğrenme algoritmalarının işleyişi (Jahnke, 2015)

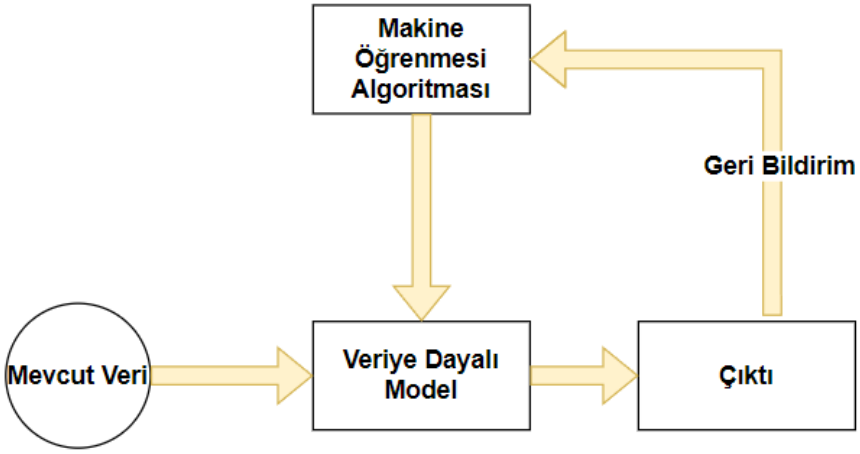
Denetimsiz Öğrenme, öğrenme sürecinde, YSA yalnızca giriş verileri sunulurken hedef çıktılar ağa sunulmaz. Bu sebeple, ağın kontrol fonksiyonunu gerçekleştirecek bir denetim mekanizması yer almamaktadır. Denetimsiz öğrenme

yöntemiyle, ağ giriş verilerinin özelliklerine göre gruplamalar yapmak için ağırlık değerlerini ayarlar. Ardından, ağ her grup için bir örnek vektör üretir. Bu şekilde, ağ, verilerin özelliklerine dayanarak kendi içinde yapılandırma ve kümelenme yapabilir (Fausett, 1999). Şekil 1.13.'de, denetimsiz öğrenme algoritmasının işleyişi gösterilmektedir (Jahnke, 2015).



Şekil 1.13. Denetimsiz öğrenme algoritmalarının işleyişi (Jahnke, 2015)

Destekleyici Öğrenme, ağa bir danışman eşlik eder ve her giriş verisi için beklenen çıktı setini doğrudan sunmak yerine, sistemin kendi çıktılarını üretmesini ve bu çıktıların doğru veya yanlış olduğunu gösteren bir sinyal üretmesini bekler. Danışman, sistemin ürettiği çıktıları değerlendirir ve geri bildirimde bulunur. Sistem, danışmandan gelen bu geri bildirimi dikkate alarak öğrenme sürecini sürdürür. Bu yöntemde, ağ kendi hatalarını tespit ederek öğrenme yapar ve danışmanın yardımıyla daha doğru sonuçlar üretmeyi hedefler (Öztemel, 2003). Şekil 1.14.'de, destekleyici öğrenme algoritmasının işleyişi gösterilmektedir (Jahnke, 2015).



Şekil 1.14. Destekleyici öğrenme algoritmalarının işleyişi (Jahnke, 2015)

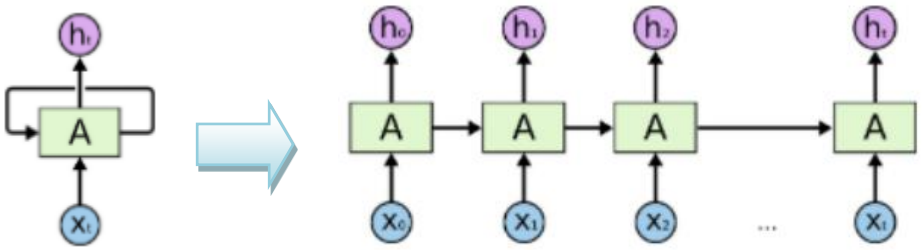
1.2.2. Tekrarlayan Sinir Ağları

Tekrarlayan sinir ağı (ing: Recurrent Neural Network [RNN]), sıralı veriler ile işlem yapmak için kullanılan DÖ yöntemidir. Sıralı bilgiden yararlanır ve sinir ağına bir önceki katmandan gelen çıktıları girdi olarak kullanmaktadır (Lipton ve ark., 2015). Sıralı bilgi bir katmana girdi olarak gelmektedir. Bu katmanda işlenir ve bu adım öncesinde hangi özellik geldi ise onun bilgisi bir sonraki katmanda hatırlanabilecek biçimde hafızada tutulur. Bir sıralı bilgi her elemanı için bu görevi tekrar ederek gerçekleştirmektedir. Bu, RNN'lerin, önceki girdilerin modelleme sonuçlarını bir sonraki adımda kullanarak, ardışık veriler arasındaki ilişkileri anlamalarına olanak sağlamaktadır (Gilik ve ark., 2022; Kushwah ve ark., 2022).

Geleneksel ileri beslemeli sinir ağları, karmaşık yapıya sahip verilerde yetersiz kalabilmektedir. Bu nedenle, bellek içeren bir model olan RNN'ler geliştirilmiştir. RNN modelleri, geri yayılımlı ve gerçek Zamanlı öğrenme algoritmalarıyla eğitilmektedir, bu da RNN modellerinin daha dinamik hale gelmesini sağlamaktadır. Ancak, RNN'deki belleğin kısa süreli bellek olduğu unutulmamalıdır. Bu yüzden, RNN modelleri kısa süreli tahminlerde daha iyi sonuçlar verirken, uzun süreli tahmin problemlerine göre daha zorluklar yaşayabilmektedirler (Kumar ve Ningombam, 2018; Staudemeyer ve Morris, 2019).

RNN yapısı sebebiyle zaman serileri problemlerinde kullanılmaktadır. Zaman serisi tahminlerinde kullanılmasının en önemli gerekçelerinden biri, diğer ağlardan

farklı olarak değişken boyutlu ve esnek girişleri kabul edebilme yetenekleridir. Bu özellik, RNN'lerin çeşitli veri tiplerini işleme kabiliyetini sağlar. Ayrıca, RNN modelleri sabit uzunlukta çıktılar üretebilirler, bu da tahmin sonuçlarının tutarlı olmasını sağlar. Öğrenme adımında sabit hesaplamalar kullanarak çalışır, bu da veri üzerinde tutarlı bir şekilde işlem yapılmasını sağlar. Aynı zamanda vektör dizileri üzerinde işlem yapma özelliğine sahiptir ve giriş ile çıkışlar diziler halinde olabilir (Url-4, 2015). Şekil 1.17.'de basit RNN yapısını ve tekrarlayan kısmının açık hali görülmektedir. Şekildeki x_t girdi vektörünü, h_t ise çıktı vektörünü ifade etmektedir. A olarak ifade edilen kısım ise RNN modelinin iç yapısını ifade etmektedir. $x_0 \ldots . x_t$ şeklindeki t uzunluğundaki yapı RNN içerisinde¸ Şekil 1.15.'deki gibi açılır ve her x_i için bir h_i çıktısı elde edilmektedir (Url-5, 2023).



Şekil 1.15. RNN yapısı ve tekrarlayan kısmın açık hali (Url-5, 2023)

RNN, iç yapılarının tekrarlı olması sebebiyle uzun vadeli geçmiş örüntüleri hatırlayabilmektedir. Bu hatırlama kapasitesine erişebilmek için parametreleri dikkatlice seçilmesi gerekmektedir. Fakat gerçekleşmesi bu şekilde olmamaktadır. Parametre seçimi mümkün olmaması sebebiyle RNN çok eski geçmişi hatırlayamazlar (Bengio ve ark., 2015).

RNN modelinin önemli problemi *kaybolan gradyan problem*dir. Kaybolan gradyan problemi, model karmaşıklaştıkça (ard arda eklenen RNN katman sayısı attıkça) daha belirgin olmaktadır. İleri besleme işlemi çalıştırıldıktan sonra, modelin hesaplanan tahmin hatası modelin iç katmanlarında yer alan nöronlara dağıtılması gerekmektedir. Dağıtımı olacak hata, zincir kuralı uygulanacak ve modelin ağırlıkları hesaplanan gradyanlarına göre dağıtımı yapılacaktır. Geri besleme işlemi bu şekilde gerçekleşir. Zincir kuralı uygulanırken RNN modeli önce açılır ve açıldığında kullanılacak olan zaman serisi büyüklüğü alınan gradyanı ya çok büyük ya da çok küçük olmaktadır.

Yok olan ya da patlayan gradyan etkisi seri boyunca uzun bağımlılıkları olan problemleri çözmeyi imkansız hale getirir. Bu problemi çözmek için Hochreiter ve Schmidhuber LSTM modelini geliştirmişlerdir (Hochreiter ve Schmidhuber, 1997).

1.2.3. Uzun Kısa Süreli Bellek Derin Öğrenme Ağları

Hochreiter ve Schmidhuber tarafından kaybolan ve patlayan gradyan (eğim) probleminin çözülmesi için LSTM modeli geliştirilmiştir (Hochreiter ve Schmidhuber, 1997).

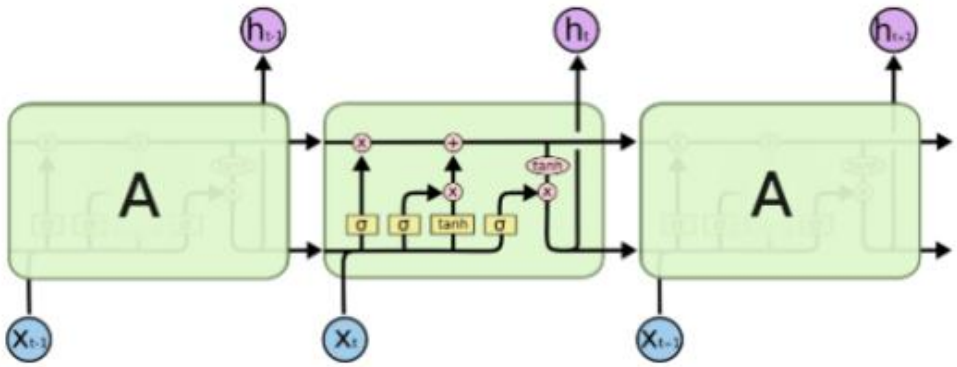
Kaybolan gradyan problemi, çoğunlukla büyük zaman serisi verilerinde meydana gelmektedir. Zaman içerisinde geri yayılma mimarisinden faydalanarak eğitilen ve kaybolan gradyan problemini çözen tekrarlayan yapay bir sinir ağı modelleri arasında yer alır. Çok fazla tekrarlayan ağlar oluşturur. Böylelikle MÖ'de yer alan zor dizi problemlerini çözmektedir (Url-5, 2015).

MÖ'de sıralı verilerin öğrenmesi, öğrenilmesi ve örüntü tanıma oldukça zordur. Standart ileri beslemeli sinir ağlarından farklı olarak LSTM yapısında geri bildirim bağlantılar yer almaktadır. Tekrarlayan döngüler kullanması sebebiyle LSTM daha kalıcı bilgiler sunmaktadır. LSTM aynı zamanda tüm veri dizilerini işlemektedir. Bu bilgileri belirli bir süre saklama özelliğine sahiptir. Bu özellikleri barındırdığından zaman serileri veya sıralı verilerin analizinde kullanışlı olmaktadır (Url-5, 2015).

LSTM modelinin ana prensibi bilgiyi uzun süre hafızada tutmasıdır. Unutulması ve hatırlanması gereken bilgilerin kontrolü sırasında hafızanın ne zaman ve nasıl bir dönüşüm yapacağına karar vermelidir. Uzun süreli belleğin depolanmasına yardımcı olmakla beraber diğer sinir ağlarına göre daha hızlı çalışmaktadır. Modelin oluşturulması kısmında hangi bilgilerin depolanıp, hangi bilgilerin unutulacağı kararında herhangi bir bağımlılık bulunmamaktadır (Url-5, 2015).

LSTM modelinde birbiri ile etkileşimli dört katman yer almaktadır. Diğer YSA'lardan ayıran özelliği katman sayısıdır. LSTM'nin iç yapısı Şekil 1.16.'da gösterilmiştir. Katmanlarda yer alan kapılar aracılığıyla hücrelerin bellekte tutacağı veriler seçilir, okunur ya da unutulmaktadır. Kapılar içlerinde ağ yapısı ve aktivasyon fonksiyonu olması sebebiyle ağa gelen verilerin ağırlıklarını güncelleyerek filtreleme yapılabilmektedir. Kapılarda değişiklikler yapılırken bilgiyi iletmesi için yollar

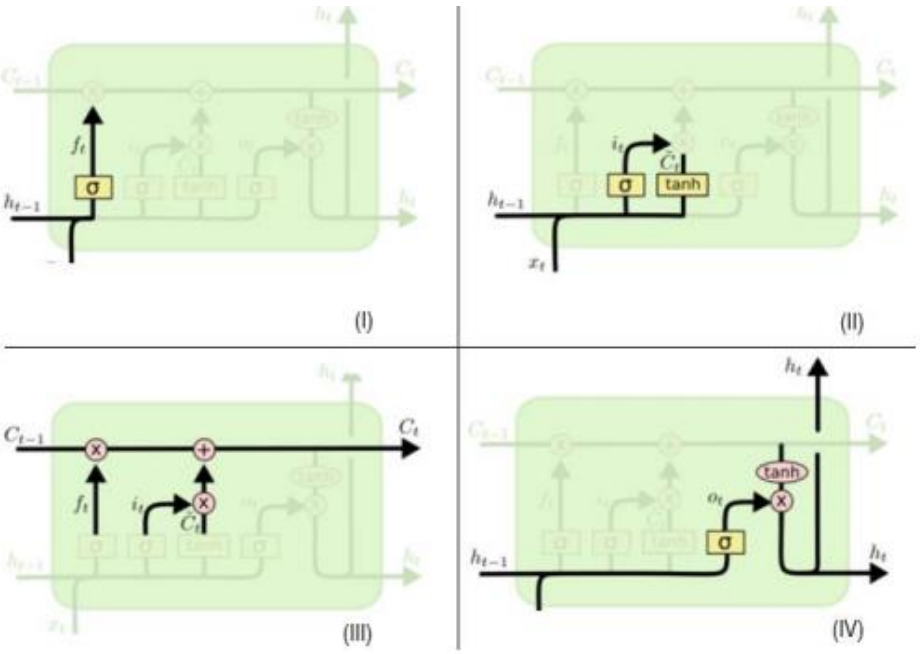
yapılmaktadır. Bu yollar da sigmoid veya relu kullanılarak 0 ile 1 arasında değerler almaktadır. Bilgi girişini sınırlandırmak için 0, bilgi geçişine izin vermek için 1 değerlerine sahip olurlar. Kapı kullanımı, modelin parametre sayısını arttırır ve uzun dizilerin işlenmesinde katkı sağlamaktadır (Url-5, 2022)



Şekil 1.16. LSTM ağın iç yapısı (Url-5, 2015)

LSTM ağı, **w** ağırlık matrisi, **b** yanlılık (bias) belirten terim, **σ** sigmoid işlevi, tanh işlevi gibi bileşenlerden oluşur. Hücre durumu (c_t), kısa-uzun süreli hafızaları depolamaktadır. Hücrenin dahili hafızasını oluşturur. Gizli durum (h_t), mevcut girdinin, önceki gizli durum (h_{t-1}) ve mevcut hücre girdisine göre hesaplanan çıktı durum bilgisidir. Bir sonraki tahmini yapabilmek için hücre durumunda depolanan sadece kısa veya uzun süreli ya da her iki bellek türünü geri alma kararını vermektedir. Giriş kapısı (i_t), girişten hücre durumuna hangi bilgilerin akacağına karar vermektedir. Unutma kapısı (f_t), girdilerden ve önceki hücre durumundan hangi bilgilerin mevcut hücre durumuna geçeceğine belirler. Çıkış kapısı (o_t), mevcut hücre durumunda hangi verilerin gizli duruma geçeceğinin kararını verebilmektedir (Url-5, 2015).

LSTM modelinin tahmin adımları Şekil 1.17.'de gösterilmiştir. LSTM bloğu sadece gizli durum (h_t) kullanmamaktadır. Zaman içinde bilgiyi hatırlayan bellek hücresi (c_t) kullanarak uzun süreli bilgiyi koruma ve kısa süreli girdileri unutma işlemlerini kapsayan bir tasarımı bulunur (Url-5, 2015).



Şekil 1.17. LSTM modeli tahmin adımları (Url-5, 2015)

Şekil 1.19. (I)'de gösterilen modelde ilk olarak hangi bilgilerin atılacağı kararı verilir. Unutma kapısı olarak verilen sigmoid katmanında h_{t-1} ve x_t değerleri kullanılarak her bir y_{t-1} için 0 ile 1 arasında değer üretmektedir. 0 atılması gereken, 1 ise korunması gereken değerleri temsil etmektedir. Şekil 1.19. (II)'de LSTM modelinin tahmininde bir sonraki adımı göstermektedir. Giriş kapısı katmanı, bilgileri güncellemekte ve sonrasında tanh tabakası ile eklenebilecek yeni aday değer vektörünü oluşturmaktadır. Şekil 1.19. (III)'de LSTM modelinin bu aşamasında hücrenin eski değerlerini tutmakta olan c_{t-1} vektörü ile giriş kapısından çıkan sonuç c_t vektörlerinden hangisinin saklanacağının kararını unutma kapısı vermektedir. Unutma kapısından gelen f_t sonucu ile eski vektör çarpılır. Şekil 1.19. (IV)'te LSTM modelinin son aşamasında verilen çıktı kapısı, tahmin çıktısı olarak kullanılabilir veya bir sonraki LSTM hücresine gönderilmek için çıktımızın ne olduğuna karar verebilmektedir. İlk olarak bu durumda, sigmoid katman çıktı değerlerinin önemine karar verilir. Sonraki adımda ise mevcut hücre durumu bir tanh katmanından geçirilerek (değerleri -1 ile 1 aralığında kalması için) çıktı değerleri elde edilir (Url-5, 2015).

Denklem **(1.16.)**'de, mevcut zaman adımında önceki hücre çıkışı h_{t-1} ve mevcut giriş x_t'yi birleştirerek $[h_{t-1}, x_t]$ vektörü oluşturulur. Ardından, bu vektörü W_f ağırlık

matrisiyle çarpılır ve bias terimi $\boldsymbol{b_f}$ ile toplanır. Sonuç üzerinden sigmoid (σ) aktivasyon fonksiyonu uygulanır. Bu işlem, unutma geçişi olarak adlandırılan $\boldsymbol{f_t}$'yi hesaplar.

$$f_t = \sigma(W_f \,.\, [h_{t-1}, x_t] + b_f) \qquad \textbf{(1.16.)}$$

Denklem **(1.17.)**'de görüldüğü üzere, $[h_{t-1}, x_t]$ vektörü $\boldsymbol{W_i}$ ağırlık matrisiyle çarpılır ve $\boldsymbol{b_i}$ bias terimiyle toplanır. Sonucu sigmoid aktivasyon fonksiyonundan geçirerek i_t değeri hesaplanır.

$$i_t = \sigma(W_i \,.\, [h_{t-1}, x_t] + b_i) \qquad \textbf{(1.17.)}$$

Denklem **(1.18.)**'da, $[h_{t-1}, x_t]$ vektörünü $\boldsymbol{W_C}$ ağırlık matrisiyle çarpılır ve $\boldsymbol{b_C}$ bias terimiyle toplanır. Elde edilen sonucu tanh aktivasyon fonksiyonundan geçirerek $\tilde{C}$t değeri hesaplanır. $\tilde{C}$t, güncellenmiş hücre durumunu temsil etmektedir.

$$\tilde{C}t = tanh\,(WC \,.\, [ht\text{-}1, xt] + bC) \qquad \textbf{(1.18.)}$$

Denklem **(1.19.)**'de, önceki hücre durumu $\boldsymbol{C_{t-1}}$, unutma geçişi $\boldsymbol{f_t}$ ile çarpılır ve $\boldsymbol{i_t}$ ile $\boldsymbol{C_t}$'nin çarpımı eklenir. Bu, hücre durumunun güncellenmiş bir sürümünü hesaplar.

$$C_t = f_t \,.\, C_{t-1} + i_t \,.\, \tilde{C}_t \qquad \textbf{(1.19.)}$$

Denklem **(1.20.)**'de, $[h_{t-1}, x_t]$ vektörünü $\boldsymbol{W_o}$ ağırlık matrisiyle çarpılır ve $\boldsymbol{b_o}$ bias terimiyle toplanır. Sonucu sigmoid aktivasyon fonksiyonuna geçirerek $\boldsymbol{o_t}$ değeri hesaplanmaktadır. Bu adım, mevcut hücre durumunun ne kadarının çıkışa dönüştürüleceğini belirler.

$$ot = \sigma(Wo \,.\, [ht\text{-}1, xt] + bo) \qquad \textbf{(1.20.)}$$

Denklem **(1.21.)**'da, $\boldsymbol{o_t}$ çıkış geçişini, tanh aktivasyon fonksiyonu tarafından güncellenmiş hücre durumuna $\boldsymbol{C_t}$ çarpılır. Bu, mevcut zaman adımında elde edilen hücre çıkışı $\boldsymbol{h_t}$ değeri hesaplanır (Fischer ve Krauss, 2018; Sagheer ve Kotb, 2019)

$$h_t = o_t \,.tanh(C_t) \qquad \textbf{(1.21.)}$$

Literatürde farklı karakteristik yapılara sahip LSTM mimarisi geliştirilmiştir:

1.2.3.1. Gözetleme Açıklıklı LSTM

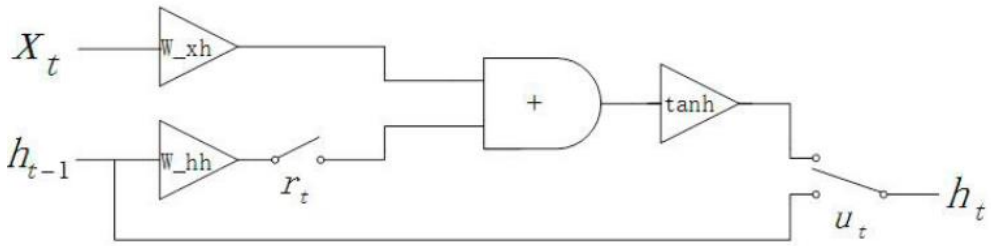
Gözetleme Açıklıklı LSTM (ing: Peephole LSTM), geleneksel (ing: Vanilla) LSTM mimarisinin geliştirilmiş halidir. Hücrelerde unutma, giriş ve çıkış kapıları bulunur. Bu kapılar, veri akışını düzenleyerek hücrelerin uzun süreli bağımlılıkları daha iyi yakalamasını sağlar. Zaman serileri analizinde ve doğal dil işleme gibi çeşitli sıralı veri uygulamalarında kullanılan bir LSTM varyasyonudur. Geleneksel LSTM modelleri, zaman serisi verilerini veya metinleri tek bir yönde (önceki girdilerden sonraki girdilere doğru) işlerken, Gözetleme Açıklıklı LSTM, modelin hücre durumunu güncellerken ekstra bilgi ekleyen gözetleme açıklığı bağlantılarını içerir. Gözetleme Açıklıklı LSTM, geleneksel LSTM'den farklı olarak hücre durumunu, gizli birimlerin iç durumunu dikkate alarak günceller. Bu bağlantılar, hücre durumunu, gizli birimlerin iç durumunu kullanarak ve girdi, çıktı ve unutma kapılarından elde edilen bilgilerle birleştirir. Böylece, modelin daha fazla bağlantı ve bilgi ile hücre durumunu güncellemesi sağlanır.

1.2.3.2. Kapılı Tekrarlayan Hücre

Kapılı Tekrarlayan Hücre (ing: Gated Recurrent Unit) mimarisi, uzun süreli ilişkilerin öğrenmesindeki zorluklara çözüm olarak sunulmuştur. Gradyan kaybolma veya patlama problemi için dahili hafıza kullanmaktadır. Güncelleme ve sıfırlama kapıları aracılığıyla verileri saklar ve filtrelemektedir (Cho ve ark., 2014).

GRU modeli, bilgi geçişini gizli katmanlarda gerçekleştirerek ayrı bir hafızaya ihtiyaç duymaz. Önceki zaman adımından gelen bilgiyi iletebilme ve engelleyebilme yetkinliğine sahip olan bir sıfırlama kapısı bulunur. Bu kapı, önceki zaman adımındaki bilginin artık önemsiz olduğu durumlarda bu bilgileri sıfırlar. Güncelleme kapısı, önceki hafızadaki verinin ne kadarının unutulacağını ve yeni hafızaya ne kadarının eklenip güncelleneceğini tespit etmektedir. Böylelikle uzun vadeli bağımlılık kontrolünü sağlar. Eğer hafıza içeriğinin daha sonra kullanılması gerektiği sonucu çıkarsa, güncelleme kapısı kapatılır ve hafıza içeriği birden çok zaman adımında taşınır. Son hafıza verileri, yeni hafıza verileri ile önceki hafıza verilerinin ağırlıklı

olarak bir araya gelmesi ile oluşur (Lewis, 2017). GRU ağının iç yapısı Şekil 1.18.'de gösterilmiştir.



Şekil 1.18. GRU bellek hücresi yapısı (Shen ve ark., 2018)

GRU'nun matematiksel adımları denklem **(1.22.)-(1.25.)** de gösterilmiştir (Shen ve ark., 2018).

$$u_t = s\,(W_u\,[h_{t-1}, x_t]) \qquad \textbf{(1.22.)}$$

Burada $\boldsymbol{u_t}$: t zamanındaki güncelleme (ing: update) kapısı vektörüdür. Bu vektör, geçmiş gizli durumun ne kadarını koruyacağımızı kontrol eder.

$$r_t = s\,(W_r\,[h_{t-1}, x_t]) \qquad \textbf{(1.23.)}$$

Burada $\boldsymbol{r_t}$: t zamanındaki sıfırlama (ing: reset) kapısı vektörüdür. Bu vektör, geçmiş gizli durumun ne kadarını unutacağımızı kontrol eder.

$$h_t^{\wedge} = tanh\,(W\,[r_t * h_{t-1}, x_t]) \qquad \textbf{(1.24.)}$$

$\boldsymbol{h_t^{\wedge}}$: t zamanındaki tahmin edilen gizli durum vektörüdür.

$$h_t = (1 - u) * h_{t-1} + u_t * h_t^{\wedge} \qquad \textbf{(1.25.)}$$

$\boldsymbol{h_t}$: t zamanındaki gizli (ing: hidden) durum vektörüdür.

Denklem **(1.22.)**'deki matematiksel gösterim sıfırlama kapısıdır. Sıfırlama kapısı, $\boldsymbol{h_{t-1}}$'in $\boldsymbol{x_t}$ ile olan etkisini kontrol etmek için kullanılır. Eğer $\boldsymbol{h_{t-1}}$ $\boldsymbol{x_t}$ için önemli değilse, $\boldsymbol{r_t}$ kapısı açılarak $\boldsymbol{h_{t-1}}$'in $\boldsymbol{x_t}$'yi durumdan etkilemez. Denklem **(1.23.)**'deki matematiksel gösterim güncelleme kapısıdır. Güncelleme kapısına gelindiğinde $\boldsymbol{h_{t-1}}$'den $\boldsymbol{h_t}$'ye bir kısa devre ilişkisi oluşur. Böylelikle $\boldsymbol{x_t}$ dikkate alınmaz. Denklem **(1.24.)** ve **(1.25.)**'deki matematiksel gösterimler GRU modeline aittir. Denklemler dikkate alındığında her **t** anındaki girdi verileri işlenir. Son girdi verisini işlenmesi aşamasında

son çıktı verisine dönüşür. Ağ eğitiminde amaç parametrelerin ayarlanması ve kayıp fonksiyonlarını en aza indirmektir (Shen ve ark., 2018).

1.2.3.3. Çift Yönlü LSTM

Çift Yönlü LSTM (ing: Bi-directional [BiLSTM] normal LSTM hücrelerinin bir adım ileri ve bir adım geriye doğru çalışan iki katmanının birleştirilmesiyle oluşturulur. Bu sayede, mevcut zamandaki örüntülerin yanı sıra geçmiş zamandaki örüntülere de erişim sağlanır, bu da modelin daha iyi bir bağlam anlayışı kazanmasına yardımcı olur. Tek yönlü LSTM, veriyi sadece ileri yönde işler ve geçmiş verilere dayanarak gelecekteki verileri tahmin eder. Ancak, bazı durumlarda, gelecekteki verilerin tahmin edilmesi için sadece geçmiş verilere değil, aynı zamanda gelecekteki verilere de ihtiyaç duyulabilir. Bu noktada Çift Yönlü LSTM devreye girer.

1.2.3.4. Dikkat Temelli LSTM

Dikkat Temelli LSTM (ing: Attention-based) mimarisi, giriş dizisindeki farklı öğelere farklı ağırlıklar vererek önemli bilgi parçalarının vurgulanmasını sağlar. Böylece model, önemli bilgilerin odaklanmasını ve daha az önemli olanların ihmal edilmesini öğrenir. LSTM mimarisi ile dikkat mekanizmasının birleştirildiği bir DÖ modelidir. Dikkat mekanizması, modelin girdi verilerinin farklı bölgelerine odaklanarak önemli bilgileri vurgulama ve çıktı oluşturma yeteneğine sahip bir mekanizmadır. LSTM, uzun sıralı verilerin analizinde bazı zorluklarla karşılaşabilir. Özellikle, uzun bir zaman serisinin ortasındaki bilgiler, modelin başlangıç ve son kısımlarına göre daha az etkili olabilir. Bu durumda, dikkat mekanizması devreye girerek modelin önemli bilgilere odaklanmasını sağlar. Dikkat temelli LSTM, her bir girdi zaman adımında dikkat ağırlıklarını hesaplayarak, önemli girdi zaman adımlarına daha fazla ağırlık verir. Bu sayede, model, önemli bilgileri vurgular ve daha kısa hafıza birimlerine odaklanarak daha etkili bir zaman serisi analizi yapar.

1.2.3.5. Hiyerarşik LSTM

Bu model, birden fazla LSTM katmanının hiyerarşik bir yapıda düzenlendiği bir yapıdır. Normal bir LSTM modelinde, tek bir LSTM katmanı çeşitli giriş verilerini işleyebilir ve geçmiş bilgileri hatırlayabilir. Ancak, bazı zaman serileri veya görevler, daha karmaşık bağımlılık yapılarına sahip olabilir ve tek bir LSTM katmanı bu

bağımlılıkları tam olarak yakalayamayabilir. Bu durumda, hiyerarşik bir yapıya sahip olan LSTM, uzun süreli bağımlılıkları daha iyi kavrayabilir. Hiyerarşik LSTM, birden çok LSTM katmanını kullanarak verilerin farklı seviyelerdeki örüntülerini işleyebilir. Örneğin, üst katmandaki LSTM, daha uzun süreli bağımlılıkları yakalamak için genel bir anlayışa sahip olabilirken, alt katmandaki LSTM, daha kısa süreli bağımlılıkları ve daha detaylı örüntüleri işleyebilir.

1.2.3.6. Derinlik Kapılı LSTM (ing: Depth Gated)

Derinlik Kapılı (ing: Depth Gated) LSTM modelinde, her bir gizli katman, LSTM hücreleri ve geleneksel sinir ağı katmanları arasında birleştirilir. Geleneksel LSTM'de, giriş, unutma ve çıkış kapıları için her bir zaman adımında aynı ağırlıklar kullanılır ve bu ağırlıklar sabit kalır. Ancak Derinlik Kapılı LSTM'de, her bir kapının ağırlıkları zamanla değiştirilir ve optimize edilir. Bu, modelin veri içindeki farklı derinliklerdeki ilişkileri daha iyi yakalayabilmesini sağlar.

1.3. META-SEZGİSEL ALGORİTMALAR

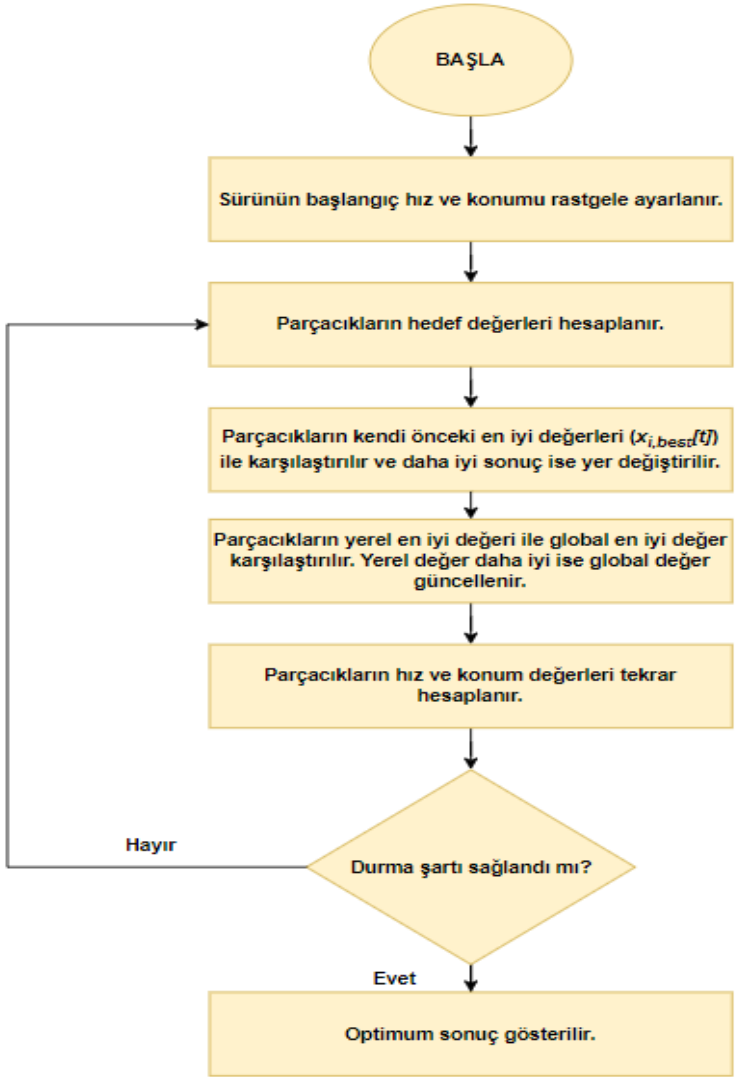
LSTM çok fazla hiper-parametrelere sahiptir. Hiper-parametreler ise modelin başarısına direkt etki etmektedir. Çok çeşitli hiper-parametre kombinasyonları olması sebebiyle her birinin denenmesi uzun zaman almaktadır. Modelin eğitimi ve optimize edilmesi için Genetik Algoritmalar (ing: Genetic Algorithms [GA]), Yapay Arı Kolonisi Algoritması (ing: Artificial Bee Colony [ABC]), Karınca Kolonisi optimizasyonu (Ant Colony Optimization [ACO]) meta-sezgisel algoritmalara örnek verilebilir. Bu çalışmada en optimum sonuçları elde edebilmek için meta-sezgisel yöntemlerden Parçacık Sürü Optimizasyonu (ing: Particle Swarm Optimization [PSO]) yöntemi kullanılmıştır.

1.3.1. Parçacık Sürü Optimizasyonu

Bu meta-sezgisel yöntemi sürülerin davranışlarının gözlemlenmesi sonucu Dr. Eberhart ve Dr. Kennedy tarafından 1995 yılında önerilmiştir (Eberhart ve Kennedy, 1995). Esinlenen algoritma, sürü halinde yaşamlarını devam ettiren kuş ve balık türlerinin tehlike halinde beraber hareket etmeleridir. Algoritmadaki her bir parçacık, sürüdeki tek bir canlıya eş gelmektedir. Parçacıklar, en iyi parçayı taklit ederek hayatta kalırlar. Her parçacık öncesindeki hareketleri hafızasında kaydeder. Böylelikle kendi

ve sürünün geçmiş kararlarını değerlendirerek bir sonraki hareketine karar verir. Yeni atılan adım ile önceki adımları karşılaştırarak hayatta kalmayı ve daha optimum değerler almayı hedefler (Eberhart ve Kennedy, 1995; Kennedy,2011).

PSO algoritması başlangıçta sürüdeki tüm parçacılar arama alanı (ing: landscape) içerisinde rastgele değerlere sahiplerdir. Her parçacığa ait hız ve konum bileşenlerini bulunur ve bu değerler başlangıçta oluşturulmalıdır. Her algoritma adımında, bu parçacıkların çözümleri (ing: objective/fitness value) değerlendirilir. Son olarak parçacıkların hızları ve konumları iyilik değerlerine göre güncellenmektedir (Houssein ve ark., 2021). Şekil 1.19.'da PSO'nun çalışmasına ilişkin akış diyagramı gösterilmiştir.



Şekil 1.19. PSO akış diyagramı

PSO algoritması, toplum tabanlı iteratif bir algoritmadır. İterasyonlar süresince en iyi sonucu bulmayı hedefler. Parçacıklara ait güncellenen hız ve konum bilgileri denklem **(1.26.)** ve denklem **(1.27.)** de verilmiştir (Houssein ve ark., 2021).

$$V_i[t+1]=wv_i[t]+c_1\, r_1(x_{i,best}[t]-x_i[t])+c_2\, r_2\, (x_{gbest}[t]-x_i[t]) \quad \textbf{(1.26.)}$$

$$X_i[t+1]= x_i[t]+v_i[t+1] \quad \textbf{(1.27.)}$$

Denklemlerdeki semboller aşağıda tanımlanmıştır.

$\boldsymbol{v_i[t]}$:Parçacığın mevcut durumunu belirten hız vektörüdür.

$\boldsymbol{w}$:Parçacığın kendi yönündeki hareketinin katsayısıdır.

$\boldsymbol{x_i[t]}$:Parçacığın mevcut durumunu belirten konum vektörü.

$\boldsymbol{x_{i,best}[t]}$:Parçacığın geçmiş yinelemeler çerçevesinde elde ettiği en iyi konum.

$\boldsymbol{x_{gbest}[t]}$:Tüm parçacıklar kapsamında elde edilen en iyi konum.

$\boldsymbol{c_1}$:Parçacığın kendi geçmişinden gelen öğrenme katsayısıdır.

$\boldsymbol{c_2}$:Parçacığın toplumun geçmişinden gelen öğrenme katsayısıdır.

$\boldsymbol{r_1}$ ve $\boldsymbol{r_2}$:(0,1) aralığında atanan rastgele değerleri ifade eden değişkenlerdir.

PSO algoritmasının en temel parametreleri $\boldsymbol{c_1}$ ve $\boldsymbol{c_2}$ öğrenme faktörleridir. Her parçacığı $\boldsymbol{x_{i,best}[t]}$ ve $\boldsymbol{x_{gbest}[t]}$ pozisyonlarına doğru yakınlaştırmaktadır. Olasılıksal hızlanmayı ifade etmektedir. Bu parametrelerin düşük değerler ile denenmesi parçacıkların hedef bölgeye gitmesinden önce, bu bölgeden uzak yerlerde dolaşmalarına olanak sağlar. Fakat bu durum hedefe ulaşma süresi uzatmasına sebep olur. Yüksek değerler ile denenmesi ise, hedefe ulaşmayı hızlandırırken, beklenmeyen durumların meydana gelmesine ve hedef bölgesine uğramama gibi durumları ortaya çıkarmasına sebebiyet verir. Bu yakınsama problemlerini çözebilmek için eylemsizlik ağırlığı ($\boldsymbol{w}$) parametresi eklenmiştir. Aynı zamanda $\boldsymbol{c_1}$, $\boldsymbol{c_2}$ ve $\boldsymbol{w}$ katsayılarının değerleri doğrusal olarak iterasyonlar da değiştirilmektedir (Yılmaz, 2021).

İKİNCİ BÖLÜM

2. PSO-LSTM MELEZ ÇERÇEVE

Melez PSO-LSTM çerçevesi, DÖ ve meta-sezgisel optimizasyon yöntemlerinden biri olan PSO yönteminin birleştirildiği yenilikçi bir yaklaşımdır. Bu melez çerçeve, zaman serileri ve sıralı verilerin analizinde kullanılmak üzere tasarlanmıştır ve gelecek tahminleri ve paternlerin analizi gibi birçok uygulama alanında etkili sonuçlar elde etmek için kullanılmaktadır. Aşağıda, LSTM ve PSO'nun hiper-parametrelerinin ayarlanmasının (optimize edilmesinin) önemi ve çalışmaya katkıları yer almaktadır.

YSA ile çeşitli meta-sezgisel algoritmaların kullanıldığı farklı melez çerçeveler bulunmaktadır. PSO-Çok Katmanlı Algılayıcı, Genetik Algoritma-YSA, Karınca Kolonisi Optimizasyonu-YSA, PSO-RNN, Genetik Algoritma-RNN, Yapay Arı Kolonisi Algoritması-RNN bazı yaygın melez çerçeveler ve kullanılan meta-sezgisel algoritmalardır. PSO-LSTM melez çerçevesi gibi, diğer melez çerçeveler de özellikle tahmin ve optimizasyon alanlarında kullanılmaktadır. Bu çalışmada PSO-LSTM melez çerçevesinin performansı ve etkinliği detaylı bir şekilde incelenmiştir.

2.1. LSTM HİPER-PARAMETRELERİNİN ÖNEMİ

LSTM'nin başarısı birçok faktöre bağlıdır. Bu faktörlerden bazıları, modelin mimarisi, veri setinin özellikleri ve kullanılan hiper-parametrelerdir. Çalışmamızda LSTM'nin hiper-parametrelerini optimizasyonunu sağlayarak en iyi sonuca varma hedeflenmiştir.

Veri LSTM ile modellenmeden önce ölçeklendirme yapılmaktadır. Verilerin ölçeklendirilmesi, bir MÖ veya veri analitiği modeli eğitirken önemli bir adımdır. Verilerin ölçeklendirilmesi, verilerin farklı özelliklerinin farklı ölçeklere sahip olabileceği durumlarda önemlidir. Verinin ölçeklendirilmesinin temel sebepleri aşağıda belirtilmiştir.

Eşitlik, veri özellikleri farklı ölçeklerde olabilir. Örneğin, bir özellik 0 ile 1000 arasında değerler alırken, başka bir özellik -1 ile 1 arasında değerler alabilir. Bu durumda, farklı ölçeklerdeki özelliklerin bir araya getirilmesi, modelin dengeli bir şekilde çalışmasını engelleyebilir. Verilerin ölçeklendirilmesi, tüm özelliklerin benzer bir ölçek üzerinde olmasını sağlayarak modelin daha tutarlı ve kararlı olmasını sağlar.

Hızlı Yakınsama, verilerin ölçeklendirilmesi, modelin daha dengeli yakınsamasını sağlar. Özelliklerin farklı ölçeklere sahip olması durumunda, modelin daha büyük ölçekli özelliklere daha fazla odaklanma eğilimi vardır. Bu, daha küçük ölçekli özelliklerin model tarafından göz ardı edilmesine veya yetersiz temsil edilmesine neden olabilir.

Gradyan Hesaplamaları, birçok optimizasyon algoritması, modelin parametrelerini güncellemek için gradyan hesaplamaları kullanır. Verilerin ölçeklendirilmemesi durumunda, gradyan hesaplamaları farklı ölçeklerdeki özelliklerin etkisi altında gerçekleşir. Bu, gradyanları dengesiz hale getirir ve modelin doğru bir şekilde güncellenmesini zorlaştırır. Verilerin ölçeklendirilmesi, gradyan hesaplamalarını daha dengeli hale getirir ve modelin daha etkili bir şekilde güncellenmesine yardımcı olur.

Verilerin ölçeklendirilmesi, özellikler arasındaki farklı ölçekleri dengelemek, modelin daha hızlı yakınsamasını sağlamak, gradyan hesaplamalarını dengelemek ve bazı modellerin gereksinimlerini karşılamak için önemlidir. Veri ölçeklendirme, genellikle önceki ve mevcut değer aralıklarını koruyarak veya özellikleri normalleştirerek gerçekleştirilebilir. Verinin ölçeklendirme işlemi sonrasında LSTM'nin ***birim, eniyileyici, aktivasyon, iterasyon sayısı, yığın boyutu*** parametrelerinin ayarlanması (opjtimize edilmesi) üzerinde çalışılmıştır.

Birimler (ing: Units), LSTM'nin temel yapı birimi hücrelerden oluşur. Birimler, bellek hücrelerinin ve girişlerin taşınması, depolanması ve kullanılmasında sorumludur. Birimlerin sayısı, LSTM'nin öğrenme kapasitesini ve karmaşıklığını belirler. Daha fazla birim, daha fazla bilgiyi kodlama potansiyeline sahip olabilir, ancak aşırı birim kullanımı aşırı öğrenmeye yol açabilir. Birim sayısı, modelin genel

performansını etkileyen önemli bir parametredir. Genellikle, model karmaşıklığı arttıkça daha fazla birim kullanılır.

Eniyileyici (ing: Optimizer), bir DÖ modelinin öğrenme sürecini yönetir ve modelin kayıp fonksiyonunu minimize etmek için ağırlıkları günceller. ***SGD, RMSprop ve Adam*** gibi yaygın eniyileyiciler vardır. Hangi eniyileyicilerin kullanılacağı, modelin eğitim hızı, kayıp fonksiyonunun konverjansı ve aşırı öğrenme gibi faktörleri etkiler.

Aktivasyon (ing: Activation), LSTM hücresinin çıktılarını hesaplarken kullanılır. ***Sigmoid, Tanh ve ReLU*** gibi yaygın aktivasyon fonksiyonları vardır. Aktivasyon fonksiyonu, modelin kapasitesini artırabilir, gradyan kaybolma sorununu gidermeye yardımcı olabilir veya çıktıları sınırlayabilir. Aktivasyon fonksiyonunun doğru seçimi, modelin öğrenme yeteneği ve genel performansı üzerinde büyük bir etkiye sahip olabilir.

Gradyan kaybolması, DÖ eğitimi sırasında gradyan değerlerinin çok küçük hale gelerek geriye doğru yayılma sırasında ağırlıkların güncellenmesini zorlaştırmasıdır (Yao ve ark., 2019).

İterasyon Sayısı (ing: Epoch), eğitim veri setinin tamamen modele ne kadar kez sunulacağını belirtir. Bir iterasyon sayısı, tüm eğitim verilerinin bir kez ileri ve geri geçirilmesi anlamına gelir. Daha fazla iterasyon sayısı, modelin daha fazla eğitim verisine maruz kalmasını sağlar ve genel olarak daha iyi sonuçlara yol açabilir. Ancak, aşırı öğrenmeye neden olabileceği için dikkatli olunmalıdır. Optimal iterasyon sayısı, modelin karmaşıklığına, veri setinin boyutuna ve hiper-parametrelerin diğer değerlerine bağlı olarak değişebilir.

Yığın Boyutu (ing: Batch Size), eğitim verilerinin kaç örneğinin bir seferde modele sunulacağını belirtir. Büyük bir yığın boyutu, paralel hesaplama ve bellek verimliliği avantajlarına sahip olabilir, ancak modelin genel performansını etkileyebilir. Küçük bir yığın boyutu, modelin daha hızlı öğrenmesini sağlayabilir, ancak dalgalanmalı bir gradyan tahmini yapabilir. Yığın boyutu seçimi, donanım kaynaklarına, veri setinin boyutuna ve diğer hiper-parametrelerin değerlerine bağlı olarak yapılmalıdır.

Bırakma/Atma (ing: Dropout), DÖ modellerinde aşırı öğrenmeyi önlemek için kullanılan bir yöntemdir. Bırakma/atma yöntemi, eğitim sırasında rastgele seçilen nöronların çıkartılması veya "bırakılması" anlamına gelir. Bu çıkarma işlemi, her eğitim örneği için farklı nöronlar seçilerek yapılır. Bu sayede, modeldeki nöronlar arasında bağımlılık azalır ve nöronların daha genel ve temsilci özellikler öğrenmesi teşvik edilir.

LSTM modelinin performansını etkileyen *birimler, eniyileyici, aktivasyon fonksiyonu, iterasyon sayısı ve yığın boyutu* gibi hiper-parametrelerin doğru bir şekilde ayarlanması önemlidir. Bu parametrelerin optimal değerlerini bulmak genellikle deneme yanılma yoluyla elde edilir ve modelin belirli bir görevde en iyi sonuçları vermesini sağlar. Deneyler MSE değerlerine göre sonuçları yorumlanmıştır. En küçük MSE değerini veren eniyileyici sonucuna göre PSO parametreleri ile eniyileyici öğrenme oranı optimizasyonu uygulanmıştır.

2.2. PSO HİPER-PARAMETRE VE ÖĞRENME ORANININ ÖNEMİ

Eniyileyicilerin öğrenme oranı (ing: learning rate), DÖ modellerinde önemli bir hiper-parametredir. Öğrenme oranı, modelin ağırlıklarını güncellerken adım büyüklüğünü belirler. Öğrenme oranının doğru bir şekilde ayarlanması, modelin etkin bir şekilde öğrenmesini ve istikrarlı bir şekilde kayıp fonksiyonunu minimize etmesini sağlar. Öğrenme oranı, modelin öğrenme hızı, konverjansı, aşırı öğrenme ve gradyan sorunları gibi faktörleri etkiler. Öğrenme oranının etkilediği durumlar aşağıda belirtilmiştir.

Hızlı Öğrenme, yüksek değere sahip bir öğrenme oranı, modelin hızlı bir şekilde öğrenmesini sağlar. Büyük adımlarla güncelleme yaparak, model daha hızlı bir şekilde yakınsamasını sağlayabilir ve eğitim süresini kısaltabilir. Ancak, çok yüksek bir öğrenme oranı aşırı öğrenmeye veya dalgalanmalı bir yakınsamaya yol açabilir. Global optimum değerini ıskalayabilir. Bu durumda, model eğitim setinde yüksek performans gösterirken, yeni verilere karşı zayıf bir performans sergileyebilir.

İstikrarlı Öğrenme, düşük bir öğrenme oranı, modelin daha istikrarlı bir şekilde öğrenmesini sağlar. Küçük adımlarla güncelleme yaparak, model daha sağlam bir yakınsama sağlayabilir ve genelde daha iyi bir performans elde edebilir. Düşük bir

öğrenme oranı, yerel minimumlara takılı kalma olasılığını azaltabilir, ancak eğitim süresini uzatabilir.

Gradyan Kaybolması/Patlaması, yanlış bir şekilde ayarlanmış bir öğrenme oranı, gradyan kaybolması veya patlaması sorunlarına yol açabilir. Gradyan kaybolması, gradyanların zamanla küçülen ve kaybolan değerler alması durumudur. Bu, daha derin katmanlarda gradyanların düzgün bir şekilde geri yayılamamasına ve öğrenmenin etkin olmamasına neden olabilir. Öte yandan, gradyan patlaması, gradyanların hızla büyümesi ve ağırlıkların hızla sıçramasıyla sonuçlanabilir (Yao ve ark., 2019). Bu da eğitim sürecinin istikrarsızlaşmasına ve aşırı öğrenmeye neden olabilir. Doğru bir öğrenme oranı, bu sorunları minimize etmek için önemlidir.

Hiper-parametre Arama, öğrenme oranı, hiper-parametre arama sürecinde önemli bir rol oynar. Farklı öğrenme oranlarını denemek, modelin performansını değerlendirmek ve en iyi sonuçları elde etmek için gereklidir.

Bu sebeple LSTM modelinde eniyileyicilerin varsayılan öğrenme oranları iyileştirilerek daha iyi MSE sonuçları elde edilmesi için PSO algoritması kullanılmıştır. PSO için pyswarm kütüphanesi içerisinde yer alan Globalbestpso algoritması kullanılmıştır.

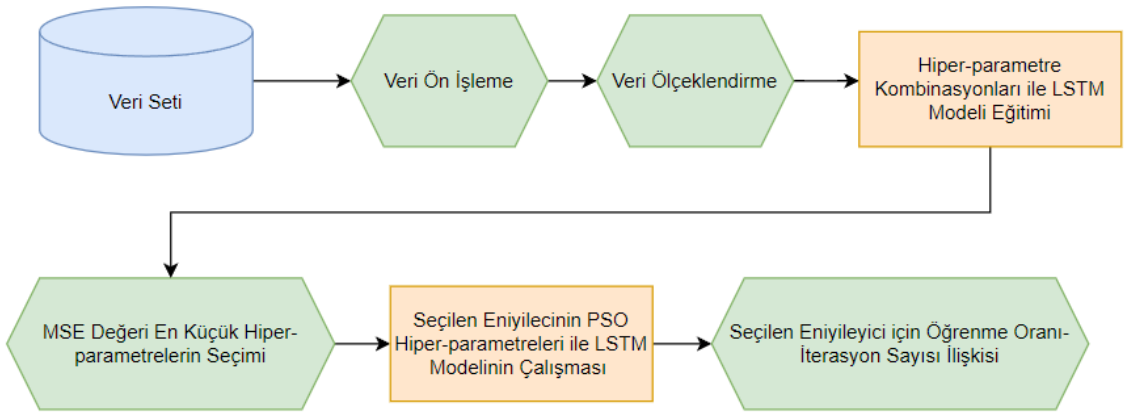
Globalbestpso (ing: Global Best Particle Swarm Optimization), bir optimizasyon algoritmasıdır ve çeşitli problemlerde kullanılır. Globalbestpso'nun başarı oranı, kullanılan hiper-parametre değerlerine bağlıdır. Globalbestpso'da kullanılan c1, c2, w değerlerinin önemi, aşağıda izah edilmektedir:

c1 (kendilik faktörü), her bir parçacığın kendi en iyi konumunu güncellemedeki etkisini kontrol eden bir faktördür. Bu faktör, bir parçacığın mevcut konumunu ve hedefe olan uzaklığını değerlendirirken kullanılır. c1, bir parçacığın kendi en iyi konumuna doğru hareket etme isteğini kontrol eder. Daha yüksek c1 değerleri, parçacıkların daha hızlı kendi en iyi konumlarına doğru yaklaşmasını sağlar, ancak aşırı öğrenme riskini artırır. Daha düşük c1 değerleri ise parçacıkların daha yavaş bir şekilde konumlarını güncellemesine ve genelde daha iyi bir keşif yapmasına yol açar.

c2 (sosyal faktör), parçacıkların en iyi konuma doğru hareket etmek için diğer parçacıkları kullanma isteğini kontrol eden bir faktördür. c2, parçacıkların en iyi konumların birlikte keşfedilmesi için sosyal etkileşimi sağlar. Daha yüksek c2 değerleri, parçacıkların diğer en iyi konumlara daha fazla dikkat etmesini ve global optimuma doğru daha hızlı yaklaşmasını sağlar. Ancak, aşırı öğrenme riski de artabilir. Daha düşük c2 değerleri, parçacıkların daha bağımsız bir şekilde hareket etmesine ve yerel optimumdan kaçınmaya yardımcı olur.

w (sürü ağırlık faktörü), bir parçacığın mevcut hızını ve geçmiş hızlarını nasıl dikkate aldığını kontrol eden bir faktördür. w, parçacıkların hareket hızının denge ve istikrarını sağlar. Daha yüksek w değerleri, parçacıkların hızını daha fazla korumasını ve daha fazla keşif yapmasını sağlar. Ancak, aşırı hareket riskini ve yakınsama hızını azaltabilir. Daha düşük w değerleri, parçacıkların daha hızlı bir şekilde yakınsamasını sağlar, ancak yerel optimuma hapsolma riskini artırabilir.

Globalbestpso'da bu hiper-parametrelerin doğru bir şekilde ayarlanması, optimizasyon algoritmasının performansını ve konverjansını etkiler. Optimal hiper-parametre değerlerini bulmak için genellikle *deneme yanılma yöntemi* kullanılır. Probleme, veri setine ve algoritmanın gereksinimlerine bağlı olarak bu değerler değişebilmektedir. Tüm çalışma sürecine ilişkin iş akış diyagramı Şekil 2.1.'de gösterilmiştir.



Şekil 2.1. Çalışmamızın iş akışı

ÜÇÜNCÜ BÖLÜM

3. LİTERATÜR ÇALIŞMALARI

3.1. LSTM MİMARİSİ İLE İLGİLİ YAPILAN ÇALIŞMALAR

LSTM mimarisinin en optimal olduğunun kanıtı veya daha iyi mimarilerin olup olmadığını tespit etmek için on binin üzerinde farklı RNN mimarisi değerlendirilmiştir. Benzer RNN yapılarının olduğu ortaya çıkarılmıştır (Jozefowicz ve ark., 2015). Varyantların hiçbirinin standart LSTM mimarisini dikkat çeken bir ölçüde iyileştirmediğine ve unutma geçidinin, çıktı etkinleştirme işlevinin önemli değişkenler olduğuna beş binin üzerinde yapılan deneylerin sonuçları ile varılmıştır (Greff ve ark., 2016).

LSTM DÖ yaklaşımının kullanmış çeşitli çalışmalar bulunur. Ulusal Menkul Kıymetler Borsa'sının farklı sektörlerden şirketlerin hisse senedi tahmininde LSTM ile gerçekleştirilmiştir. Bu çalışmada iki farklı tür LSTM kullanılmıştır. Sayısal veri setlerinde, düşük standart sapma ve yayılım değerlerine sahip olduğu durumlarda durum bilgisi tutulmayan LSTM tercih edilmiş, metin içeren modeller oluşturulduğunda ise durum bilgisi tutulan LSTM tercih edilmiştir. Ayrıca, LSTM modelini optimize etmek amacıyla katman sayısı hiper-parametresi üzerinde değişiklikler yapılmıştır. Yapılan çalışmada, tek katmanlı LSTM modelinin en iyi sonuçları verdiği gözlemlenmiştir. Elde edilen sonuçların doğruluğunu test etmek için tek yönlü ANOVA testi kullanılarak aynı sonuçlara ulaşılmış ve modelin doğruluğu kanıtlanmıştır (Yadav ve ark., 2020). Başka bir çalışmada, LSTM, MLP kullanarak 1992-2015 yılları arasındaki S&P 500 endeksinin trend yönünü tahmin edilmiştir (Fischer ve Krauss, 2018). LSTM ve GRU teknikleri kullanılarak kısa vadeli trafik akışı tahminlemesi yapısı öneri olarak sunulmuştur (Fu ve ark., 2016). Başka bir çalışmada, LSTM kullanılarak günlük vaka sayısı, ölüm sayısı ve iyileşen hasta sayısı dikkate alınarak Kanada da salgın hastalığın gelecek zamandaki durumu tahmini yapılmıştır (Chimmula ve Zhang, 2020). Başka bir çalışmada stok fiyatlarını tahmin edilmesi için LSTM ve RNN kullanılmıştır (Sreelekshmy ve ark., 2017). Başka bir çalışmada ise 1980-2020 yılları arasında Konya ilinin aylık yağış, nem ve sıcaklık verileri ile bu verilere dayalı buğday üretim miktarı ve buğday verimlilik verileri

kullanıldı. Bu veriler kullanılarak, GRU ve LSTM yöntemleri gibi RNN tabanlı algoritmalarla buğday verimliliği tahmini gerçekleştirildi. LSTM yöntemi GRU yöntemine göre daha iyi sonuçlar vermiştir. Fakat LSTM yönteminin eğitim modelleme süresi GRU yöntemine göre daha uzun sürmüştür (Çetiner ve Kara, 2022).

3.2. MELEZ LSTM YAKLAŞIMLARI İLE YAPILAN ÇALIŞMALAR

DÖ ve özellikle LSTM gibi YSA yöntemleri, son yıllarda çeşitli alanlarda önemli başarılar elde etmiştir. Bu başarılar, özellikle zaman serisi tahminleri ve modelleme alanında dikkat çekicidir. Ancak, bazı durumlarda, yalnızca tek bir DÖ modeli kullanmak yerine, farklı yöntemleri bir araya getiren melez yaklaşımların daha iyi performans sergileyebileceği gözlenmiştir. Özellikle LSTM ile diğer yöntemlerin kombinasyonunun, tahmin doğruluğunu artırmada ve modelin kararlılığını sağlamada olumlu sonuçlar verdiği belirtilmiştir. Xu ve arkadaşlarının çalışmasında, sel tahminlerinin doğruluğunu ve öncü süresini artırmak için YSA ve PSO yöntemlerini temel alan bir DÖ sinir ağı modeli geliştirmeyi amaçlamaktadır. Geleneksel yöntemlerin sınırlamalarını aşmak için LSTM ağları kullanılarak yapılan bu çalışma, havza bazında yağış ve akış verilerini kullanarak sel süreçlerini tahmin etmeyi hedeflemektedir. Performans değerlendirmesi için kullanılan Nash Sutcliffe verimlilik katsayısı, kök ortalama kare hata ve önyargı, PSO-LSTM modelinin diğer yöntemlere kıyasla daha iyi performans gösterdiğini göstermiştir. Özellikle 6 saatten daha uzun süreleri kapsayan farklı öncü sürelerde, PSO-LSTM modeli sel tahmini doğruluğunu artırmış ve daha yüksek tahmin doğruluğuna ve kararlılığa sahip olmuştur (Xu ve ark., 2022).

Lv ve arkadaşlarının çalışmasında, hisse senetlerinin kapanış fiyatlarının tahmininde kullanılan geliştirilmiş bir LSTM YSA algoritması olan PSO ile ilgili bir incelemedir. PSO, LSTM ağı için optimal ağırlıkları belirlemek için kullanılmakta ve tahmin hatasını azaltmaktadır. Çalışmada, hisse senetlerinin özelliklerini içeren geçmiş verilerle eğitilen LSTM modeli, hisse senedinin son iki yılının kapanış fiyatını tahmin etmek için kullanılmaktadır. Karşılaştırma yapılan diğer algoritmalarla kıyaslandığında, LSTM'nin daha iyi güvenilirlik ve uyumluluk performansı gösterdiği ve iyileştirilmiş PSO-LSTM algoritmasının daha yüksek doğruluk sağladığı belirtilmektedir (Lv ve ark., 2018). Sun ve arkadaşlarının çalışmasında, rüzgar enerjisi

tahmininde kullanılmak üzere geliştirilmiş bir LSTM ağına dayalı iki aşamalı bir dikkat mekanizması sunmaktadır. Dikkat mekanizması, rüzgar enerjisinin giriş verisi özelliklerini ve eğilim özelliklerini ölçmek ve veri hazırlama sürecini iyileştirmek için kullanılmıştır. Ayrıca, PSO, LSTM ağının hiper-parametrelerini optimize etmek amacıyla kullanılmıştır. Bu iyileştirmelerle tahmin doğruluğu önemli ölçüde artırılmış ve modelin yakınsama hızı iyileştirilmiştir (Sun ve ark., 2022).

Yao ve arkadaşlarının çalışmasında, gemi hareketinin doğrusal olmayan (ing: non-lineer) özelliklerinden kaynaklanan tahmin zorluklarına odaklanmaktadır. Gemi hareketinin duruşu, bağlantılılık, belirsiz periyot, gürültü sinyalleri ve kaotik faktörler nedeniyle gelecekteki gemi hareketini doğru bir şekilde tahmin etmek zorlaşır. Bu nedenle, makalede LSTM ve Yankı Durumu Ağı (ing: Echo State Network-[ESN]) gibi sinir ağı modelleri gemi hareketi tahmininde kullanılmıştır. Ancak, YSA algoritmalarının parametre ayarlaması zorlukları olduğu belirtilmiştir. Bu makalede, gemi hareketi tahmininde daha iyi performans sağlamak için PSO optimizasyonuyla LSTM modeli birleştirilmiştir. Test simülasyon sonuçları, bu birleşimin gemi hareketi tahmininin doğruluğunu artırdığını göstermektedir. PSO'nun optimize edilmiş LSTM modeli ile birlikte Adam kullanılması, daha iyi çözümlerin global çözüm alanında bulunma olasılığını artırabilir. Ancak, orijinal verilerdeki gürültü modelin tahmin yeteneğini azaltmaktadır. Bu nedenle, daha fazla iyileştirme için Kalman filtresinin kullanılması önerilmektedir. PSO'nun başlangıç parçacıklarının, yakınsamayı sağlamak için ADAM'a dayandığı belirtilmektedir (Yao ve ark., 2018).

Kumar ve arkadaşlarının çalışmasında, üç önemli hisse senedi endeksi (Sensex, S&P 500 ve Nifty 50) için kısa ve uzun vadeli fiyat tahminleri için yeni bir yaklaşım sunulmaktadır. Makalede LSTM ağı ve PSO tabanlı bir hibrit derin öğrenme modeli tanıtılmıştır. LSTM, belirsiz, sıralı ve doğrusal olmayan verileri işleyebilmesi nedeniyle tercih edilen bir yöntemdir, ancak ağırlıkların ve sapmaların optimize edilmesi en büyük zorluklardan biridir. Bu nedenle, makalede PSO tabanlı hibrit DÖ modeli önerilmiştir. PSO, LSTM'nin başlangıç ağırlıklarını ve tam bağlantılı katmanın (ing: Fully Connected Layer-[FCL]) ağırlıklarını evrimleştirmek için kullanılmıştır. Ayrıca, PSO'nun atalet katsayısını iyileştirmek için parçacıkların hızını kullanarak adaptif bir yaklaşım da tanıtılmıştır. Önerilen yöntem, adaptif PSO ve Adam

eniyileyici birleştirerek LSTM'nin eğitimini gerçekleştiren bir yöntemdir. Adaptif PSO, LSTM ağının farklı katmanlarının başlangıç ağırlıklarını ve FCL'nin evrimleşmesini amaçlamaktadır. Yapılan deneysel bulgular, önerilen modelin LSTM ve FC katmanlarının optimum başlangıç ağırlıklarını ve sapmalarını elde etmede başarılı olduğunu ve üstün tahmin doğruluğuna sahip olduğunu göstermektedir. Çalışmanın sonucunda, hisse senedi fiyatları zaman serilerinin tahmininde kullanılmak üzere adaptif PSO-LSTM adlı doğal ilham alınan optimizasyon tabanlı hibrit DÖ yaklaşımının umut verici sonuçlar sunduğu belirtilmiştir. Özellikle, S&P 500, Sensex ve Nifty 50 gibi önemli endeksler üzerinde kısa ve uzun vadeli tahminlerde önerilen modelin daha iyi performans gösterdiği tespit edilmiştir. YSA ile PSO'nun birlikte melez çalışmaları mevcuttur. Bir çalışmada, zaman serilerindeki hareketliliğin tahminleri için PSO ile eğitilmiş Quantile Regression Neural Network adında yeni bir yöntem sunulmuştur. Modelin iyi sonuçlar verdiği gözlemlenmiştir (Pradeepkumar ve Ravi, 2017). Başka bir çalışmada ise PSO ile konvolüsyonel sinir ağı hiper-parametre ayarlaması amacı için kullanılmıştır. Farklı hiper-parametre seçim algoritmaları doğruluk değerleri ile hesaplama süreleri kıyas edilmiştir. PSO'nun doğruluk bakımından diğer optimizasyon algoritmalarına göre daha kötü bir sonuç çıkarmıştır. Fakat hesaplama süre kıyasında PSO'nun daha iyi olduğu görülmüştür (Lorenzo ve ark., 2017).

DÖRDÜNCÜ BÖLÜM

4. DENEYSEL ÇALIŞMALAR

Bu bölüm altında tez çalışmasında kullanılan LSTM mimarileri ve PSO-LSTM melez çerçevesi gerçeklenirken, öğrenme ve test aşamalarında kullanılmasına karar verilen deneysel tasarım kararları ve veri setleri hakkında genel bilgilere yer verilmiştir.

4.1. DENEYSEL TASARIM

Çalışmada geliştirilen ve test edilen modeller 8.00GB RAM, AMD Ryzen5-4600H CPU işlemci ve Windows 10 64 bit işletim sistemi mimarisine sahip bir bilgisayarda yapılmıştır.

Yaklaşımlar, Python 3.10 versiyonu programlama dili kullanılarak geliştirilmiştir. Veri setinin okunması, ön işleme adımları için Numpy, Pandas, Math, görselleştirme için Matplotlib, MÖ algoritmaları ve hiper-parametre optimizasyonu için scikit-learn, derin öğrenme mimarilerinin geliştirilmesi için Keras, Tenserflow ve son olarak optimizasyon için pyswarms içerisindeki GlobalBestPSO kütüphanelerinden yararlanılmıştır. Veriler excelde tutulmaktadır. Çalışmada kullanılan excel formatı csv türündedir.

Öncelikle veri seti gerekli ön işleme adımlarından geçirilmiş ve LSTM ve PSO-LSTM modelleri çalıştırılmadan veriler rassal olarak %20 oranında test ve %80 oranında eğitim setleri olarak ayrılmıştır.

Ayrıca daha özelleşmiş tarasrım kararları ve seçilen parametre değerleri “Bulgular ve Tartışmalar” bölümü altında ilgili altbaşlıklarda açıklanacaktır.

4.2. VERİ SETİ

Çalışmada, Wikipedia'nın farklı dillerdeki makalelerinin trafik verilerini içeren veri seti kullanılmıştır. Veriler https://www.kaggle.com/competitions/web-traffic-time-series-forecasting/data internet adresinden alınmıştır. Bu veri setinde, Wikipedia'nın farklı dillerdeki makalelerinin web trafik bilgileri yer almaktadır. Bu zaman serilesilerini içeren veriler, 1 Temmuz 2015’ten 31 Aralık 2016 tarihleri arasına

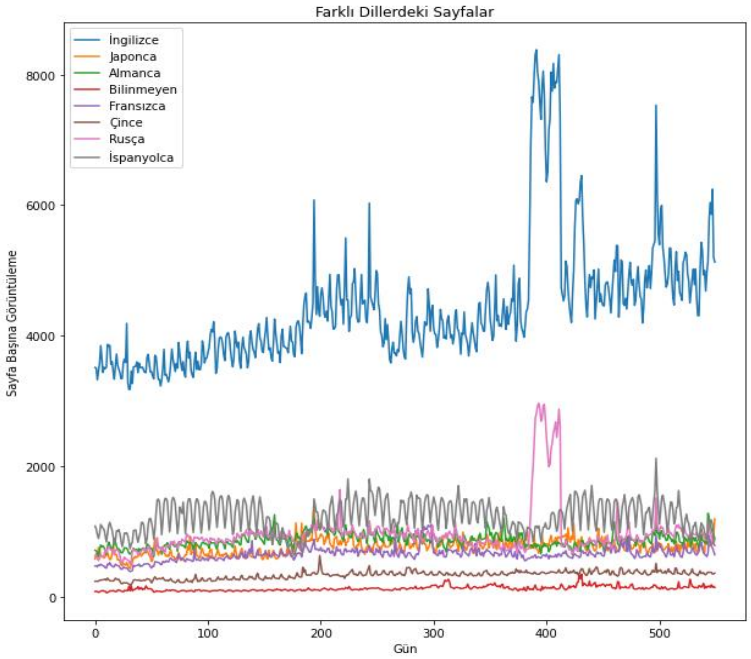
kadar olan günlük görüntülenme sayılarıdır ve her bir zamana karşılık bir ziyaretçi sayısı ile ilişkilidir. Çalışma yapılan veri seti 145.063 adet zaman serisinden meydana gelmektedir. Veri seti, web sitesine ilişkin tek özellik ve 550 adet tarih bilgisinden oluşmaktadır. Wikipedia makale sitesine ait özellik, makale adı veya başlığı, Wikipedia sayfanın dili, erişim türü ve araç bilgilerinden oluşur. Wikipedia sayfanın dili, İngilizce (en), Japonca (ja), Almanca (de), Fransızca (fr), Çince (zh), Rusça (ru), İspanyolca (es) ve bilinmeyen diller (na) olabilmektedir.

Wikipedia sayfanın dillerine göre 1 Temmuz 2015'ten 31 Aralık 2016'ya kadar toplam görüntülenme sayıları Tablo 4.1.'de verilmiştir.

Tablo 4.1. Sayfa diline göre makale sayıları

Wikipedia Sayfasının Dili	**Gözlemlenen Makale Sayısı**
İngilizce (en)	**24108**
Japonca (ja)	**20431**
Almanca (de)	**18547**
Bilinmeyen Diller (na)	17855
Fransızca (fr)	17802
Çince (zh)	17229
Rusça (ru)	15022
İspanyolca (es)	14069

İngilizce uzantılı sitelerdeki makalelerin çok fazla görüntülenme aldığı, en az görüntülenme ise İspanyolca da olduğu ve diğer dillerinde günlük görüntülenme sayıları Şekil 4.1.'de verilmiştir.



Şekil 4.1. Farklı dillerde sayfaların günlük görüntülenme sayıları

Tez kapsamında, LSTM, BiLSTM ve çok katmalı LSTM mimarileri ve ayrıca PSO-LSTM çerçevesine ait hiper-parametre analizleri gerçekleştirilen İngilizce uzantılı makalelerin yer aldığı veri seti kullanılmış. Ardından PSO-LSTM melez çerçevesine ait başarımı Japonca ve Almanca uzantılı makale sitelerine ait veri setleri üzerinde de başarım testleri gerçekleştirilmiştir.

4.3. DEĞERLENDİRME METRİKLERİ

Tez kapsamında LSTM mimarileri ve PSO-LSTM çerçevesine ait tüm eğitim ve test çalıştırmalarında en çok tercih edilen performans metriklerinden biri olan MSE kullanılmıştır.

4.4. VERİ ÖN İŞLEME

Çalışmamızda her bir zamana karşılık gelen değerlerin tümü boş ise bu veriler silinmiştir. Kalan boş değerler ise öncesinde var olan değerlerden biri ile yok ise sonraki bulduğu değerlerden biri ile doldurulur. Çalışmanın devamında Wikipedia İngilizce uzantılı makale sitelerinin toplam görüntülenme sayıları ile tahminleme işlemi yapılmıştır. İngilizce makale sayısı 24108, Japonca makale sayısı 20431 ve Almanca makale sayısı 18457 adettir. Günlük olarak İngilizce, Japonca ve Almanca uzantılı Wikipedia sayfalarının görüntülenmeleri incelenir ve makale sitelerine günlük

olarak toplam kaç görüntülenme olduğu hesaplanır. Veri seti tarih ve görüntüleme sayısı olacak şekilde ön işleme adımı tamamlanır.

Sonrasında her bir veri setinin, Standard, Robust ve MinMax ölçeklendirme ile ön testleri yapılmıştır. Bu ön testlere ilişkin MSE değerleri Tablo 4.2.'de verilmiştir. Elde edilen MSE değerlerine göre MinMax ölçeklendirmenin daha iyi olduğu görüldüğünden, bundan sonraki adımlarda yapılacak olan tüm deneysel çalışmalarda bu ölçeklendirme ile devam edilmiştir.

Tablo 4.2. Ölçeklendirmeler göre LSTM'nin en iyi MSE değerleri

Ölçeklendirme	**MSE**
MinMax	**0,005727**
Standard	0,181609
Robust	0,283783

LSTM modelinin hazırlık aşamasındaki çerçeveleme işlemi, sıralı verileri uygun bir formatta modele verebilmek için yapılan ön işlemlerden biridir. Her bir giriş örneği, belirli bir zaman aralığı içindeki veri noktalarını içerir ve bir hedef değeriyle eşleştirilir. Model, her bir çerçevedeki verilere dayalı olarak, bir sonraki zaman adımındaki değeri tahminlemeye çalışır. Çerçeveleme değeri kadar önceki verilere bakar ve bu bilgilere dayalı olarak gelecekteki değeri tahmin eder. Burada çerçeveleme işlemi (ing: look back veya ing: window size) için 1 alınmıştır

4.5. HİPER-PARAMETRE DEĞERLERİ

İkinci bölümde anlatıldığı üzere DÖ ve MÖ gibi yaklaşımlara ait ayarlanması (optimize edilmesi) gereken ve başarımı oldukça etkileyen bir dizi hiper-parametreler mevcuttur. Ayrıca, kullanılan veri setine bağlı olarak bahsi geçen parametrelerin bazılarının başarıma etkileri diğerlerinden daha fazla olmaktadır ve hangilerinin daha etkin olduklarına dair analizlerin yapılması ve ilgili değerlerinin belirlenmesi önem arz etmektedir.

Zaman serileri analizinde LSTM birimlerinde **tanh** aktivasyon fonksiyonu tercih edilmektedir (Yao ve ark., 2019). Bu fonksiyonun tercih edilmesinin en önemli sebebi girdi değerlerini [-1,1] arasında dönüştürerek örneğin relu ve sigmoid aktivasyon fonksiyonlarından daha geniş bir aralıkta değer üretmesidir. Bu sayede *"gradyan kaybolması"* problemi ortadan kalkmış olur. Aynı zamanda girdi verilerini 0 etrafında merkezlendiğinden modelin daha hızlı eğitilmesine yardımcı olur. Bu avantajları göz önüne alınarak yapılan ön testlerde de **tanh** aktivasyon fonksiyonunun diğer fonksiyonlara kıyasla daha başarılı olduğu gözlendiğinden tez kapsamında yapılan tüm deneysel çalışmalara bu fonksiyon ile devam edilmiştir.

DÖ mimarilerinin başarımını etkileyen belki de en önemli faktör, kullanılacak olan eniyileyici metodudur. Bu eniyileyici metotlarının başarımını neredeyse tek etkileyen hiper-parametre ise öğrenme oranıdır. Arama uzayının örüntülerine göre farklı değerler alabilen öğrenme oranlarının kullanılması başarım için oldukça önemli olduğundan, DÖ mimarilerinde adaptif öğrenme oranı kullanan ADAM ve RMSPROP gibi eniyileyiciler geliştirilmiştir. Bu eniyileyiciler, adaptif olmayan eniyileyicilere kıyasla başarımı çok daha iyidir. SGD ise DÖ kütüphanesinde kullanılan tek adaptif olmayan eniyileyici metodudur. SGD eniyileyicisi için kütüphanenin kullandığı varsayılan öğrenme oranını 0,01 olarak alınabildiği gibi parametre olarak farklı öğrenme oranlarının kullanılmasına da olanak sağlamaktadır. Bu çalışmada öncelikli olarak LSTM mimarilerine ait en iyi hiper-parametre değerleri bulunurken SGD eniyileyicisinin varsayılan öğrenme oranı kullanılmış ve ardından kullanılan PSO-LSTM melez çerçevesi ile en iyi öğrenme oranı belirlenerek başarımları karşılaştırılmıştır.

Tablo 4.3.'de klasik LSTM ve BiLSTM modellerinde kullanılan eniyileyiciler, bu mimarilere ait hiper-parametreler ve denenmesine karar verilen ilgili hiper-parametre değerleri yer almaktadır. Bu değerler, yapılan ön testlerden elde edilen sonuçlara göre sezgisel olarak karar verilmiştir. Bu hiper-parametre kombinasyonları toplam 192 adettir. Örneğin, *"birim sayısı"* 8, *"eniyileyici"* ADAM, *"iterasyon sayısı"* 100 ve *"yığın boyutu*" 16 olası denenecek kombinasyonlardan biridir.

LSTM mimarilerinde yer alan optimum hiper-parametre değerleri Python scikit-learn kütüphanesinin GridSearch metodu kullanılarak belirlenmiştir.

Tablo 4.3. LSTM ve BiLSTM mimarileri için hiper-parametre değerleri

Hiper-Parametreler	**Değerler**
Birimler (Units)	8, 16, 32, 64
Eniyileyici (Optimizer)	ADAM, RMSPROP, SGD
İterasyon Sayısı (Epoch)	100, 200, 300, 450
Yığın Boyutu (Batch Size)	16, 32, 64, 128

LSTM mimarilerine ilişkin yapılan detaylı hiper-parametre analizleri neticesinde karar verilen hiper-parametre değerleri kullanılarak PSO-LSTM melez çerçevesinde kullanılacak olan PSO algoritmasına ait hiper-parametrelerin başlangıç değerleri belirlenmiş ve Tablo 4.4.'de bu parametreler ve denenecek olan değerler yer almaktadır. PSO algoritması çalıştırılırken parçacık sayısı 50 olarak seçilmiş ve algoritma 50 adım çalıştırılmıştır.

Tablo 4.4. Çalışmamızda kullanılan PSO hiper-parametreleri

PSO Hiper-parametreleri	**Değerler**
c1	0.2, 0.5, 0.8
c2	0.1, 0.3, 0.6
w	0.6, 0.9, 1.2

Tez çalışması kapsamında, elde ettiğimiz PSO hiper-parametrelerin değerlerine ek olarak, literatürde PSO hiper-parametrelerinin başlangıç değerlerine ilişkin genel kabul görmüş değerleri (Sun ve ark., 2022) de kullanılarak PSO-LSTM melez çerçevesinin başarım testleri yapılmış ve sonuçlar kıyaslanmıştır.

BEŞİNCİ BÖLÜM

5. BULGULAR VE TARTIŞMALAR

Bu bölümde, tez çalışmasında bir önceki bölümde anlatılmış olan deneysel tasarımlar baz alınarak veri setleri üzerinde yapılmış olan deneysel bulgular ve ilgili değerlendirmeler yer almaktadır.

Tez kapsamında tasarlanan deneysel çalışmalar iki aşamadan oluşmaktadır. İlk aşamada İngilizce uzantılı sitelerdeki makalelerin yer aldığı veri seti kullanılarak LSTM, çok katmanlı LSTM ve BiLSTM mimarilerine ilişkin hiper-parametreler analiz edilmiş ve değerlendirmeler sonucunda ilgili hiper-parametre değerleri belirlenmiştir. Bu mimarilerin başarımları kıyaslanmış ve seçilen mimari ile sonraki aşamaya devam edilmiştir. Ardından, birinci aşamada elde edilen en iyi LSTM hiper-parametreleri baz alınarak PSO'nun parametreleri ile birlikte LSTM'nin adaptif olmayan eniyileyici metodu SGD algoritmasının *"öğrenme oranı"* PSO meta-sezgiseli kullanılarak adaptif olarak belirlenmiştir. Herbir aşamada yaklaşımlara ait *"sapma-değişkenlik dengelemesi"* (ing: bias-variance trade-off) analizleri yapılmıştır. Son olarak da, PSO-LSTM melez çerçevesinin başarımları, diğer Japonca ve Almanca uzantılı sitelerdeki makalelerin veri setleri ile test edilmiştir.

5.1. LSTM ve BiLSTM MİMARİLERİ

Bu bölümde, LSTM, çok katmanlı LSTM ve BiLSTM mimarilerine ilişkin en etkin eniyileyici, başarımı en çok etkileyen hiper-parametrelerin neler olduğu detaylı analiz edilmiştir. Ardından her bir eniyileyiciye ait en iyi hiper-parametre kombinasyonları belirlenerek *"sapma-değişkenlik dengelemesi"* analizi yapılmıştır.

5.1.1. LSTM Mimarileri İçin Hiper-Parametre Ayarı ve Analizleri

Bu çalışmada ilk olarak tek yönlü (ing: Unidirectional) LSTM -bu mimari tablolarda **LSTM** olarak yer almaktadır- ve çift yönlü (ing: Bidirectional) LSTM -bu mimari tablolarda BiLSTM olarak geçmektedir- mimarileri ele alınmıştır.

LSTM (tek yönlü) mimarisinde, sıralı veriler yalnızca ileriye doğru tek yönlü işlenmektedir. Yani her sıralı veri, geçmişten geleceğe doğru işlenmektedir. BiLSTM mimarisinde ise sıralı veriler hem ileri hem de geri yönde yani her zaman adımında hem geçmişten geleceğe hem de gelecekten geçmişe doğru işlenir.

Bu başlık altındaki testler sadece İngilizce uzantılı makale sitelerinin web trafik verisi ile gerçeklenmiştir. Diğer dillere ait veri setleri de benzer sonuçlara sahip olduğundan burada sonuçlar yer almamaktadır.

Modellerin tasarımında bir adet giriş, bir adet gizli ve bir adet çıkış katmanı kabulü ile başlanmıştır. BiLSTM mimarisi sadece gizli katmanda kullanılmıştır. Çıkış katmanında ise ***tanh*** aktivasyon fonksiyonu ile sonlandırılmıştır.

Bu kapsamdaki analizler LSTM mimarilerine ilişkin hiper-parametre değerleri Tablo 5.3.'de belirtilen değerler için hesaplanmış.

Tablo 5.1.'de tek yönlü LSTM mimarisi için hesaplanmış en düşük test MSE değerine göre bulunan en iyi hiper-parametre kombinasyonlarına ait değerler verilmiştir. Tabloda koyu olarak belirtilen değerler en düşük test MSE değerine ait hiper-parametre değerlerini ifade etmektedir. Tablo incelendiğinde, en iyi başarım varsayılan 0,01 öğrenme oranı ile SGD metodu vermiştir. Bilindiği gibi SGD metodu adaptif olmayan bir yaklaşımdır ve varsayılan düşük öğrenme oranı ile adaptif olan ADAM ve RMSPROP eniyileyicilerine kıyasla ancak çok daha uzun 450 iterasyon adımı ile en düşük MSE değerini elde edebilmiştir. Bu da zaman karmaşıklığının çok daha yüksek olması anlamına gelmektedir. Diğer taraftan ADAM ve RMSPROP gibi adaptif öğrenme oranına sahip eniyileyiciler, eğitim sırasında model parametrelerini daha hızlı ve etkili bir şekilde güncellediklerinden 100 gibi küçük iterasyon adımlarında en düşük MSE değerlerine ulaşabilmektedir.

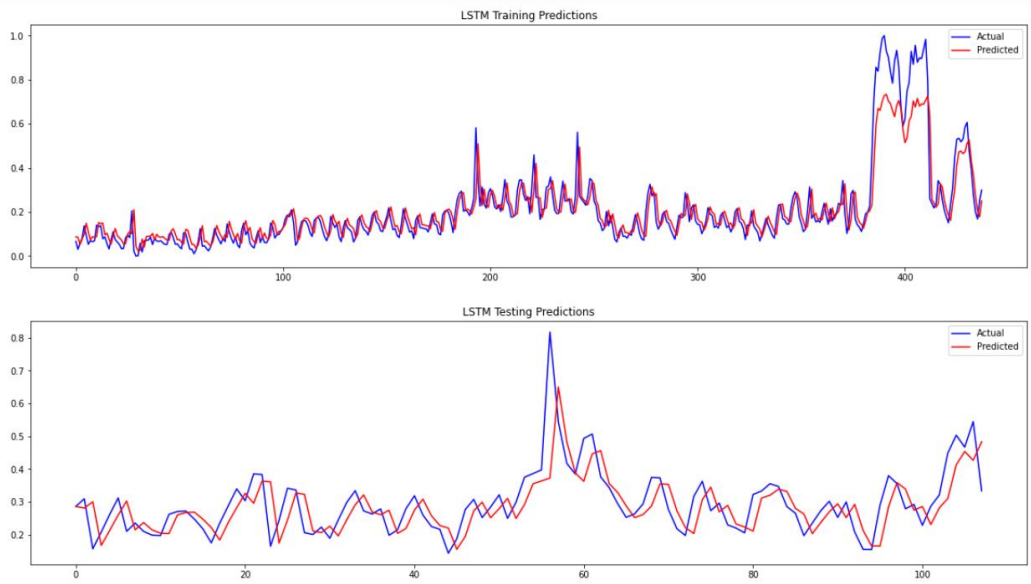
Tablo 5.1. LSTM mimarisinde farklı eniyileyici metotlarına göre başarımı en yüksek hiper-parametre değerleri

Eniyileyici	**Öğrenme Oranı**	**Birimler**	**İterasyon Sayısı**	**Yığın Boyutu**	**Eğitim MSE**	**Test MSE**
ADAM	0.001	8	100	64	0,005223	0,005965
RMSPROP	0.001	8	100	128	0,007520	0,006241
SGD	0.01	64	100	16	0,017416	0,009322
SGD	**0.01**	**64**	**450**	**16**	**0,006082**	**0,005727**
SGD	0.01	8	450	16	0,005742	0,005820

Tablo 5.1.'de ayrıca SGD eniyileyicisine ait 100 iterasyon adımı boyun elde edilmiş MSE değerlerine yer verilmiştir. Test MSE değerleri incelendiğinde, SGD eniyileyicisinin 100 iterasyon adımında düşük uyum (ing: underfitting) durumunda kaldığı ve hata oranının diğer eneyicilere kıyasla fazla olduğu görülmektedir.

Tabloda görüldüğü üzere, en düşük test MSE değeri elde edilen SGD eniyileyicisine ait *"birim sayısı"* 64, ADAM ve RMSPROP eniyileyicilerinde ise 8 olarak bulunmuştur. Bu farkı göstermek için SGD'ye ait *"birim sayısı"* 8 olan test sonuçları da yer almaktadır. LSTM'in başarımını etkileyen diğer hiper-parametrelerin etkilerine dair detaylı analizlere daha sonra yer verilecektir, ancak şu aşamada *"birim sayısı"*'nın etkisi diğer hiper-parametrelere kıyasla çok daha az olduğunu söylemekle yetineceğiz.

Şekil 5.1. en yüksek başarım değerleri elde edilmiş SGM eniyileyicisine ait eğitim ve test süreçlerine ait gerçek ve tahmin edilen değerlerine ilişkin grafiği göstermektedir. Test sürecine ait grafik incelendiğinde SGD eniyileyicisinin tahmin başarısının oldukça tatmin edici olduğu görülmektedir. Fakat zaman karmaşıklığı en büyük dezavantajıdır. Tezin ana amaçlarından biri, SGD eniyileyicisinin başarımına en yakın, zaman karmaşıklığı RMSPROP ve ADAM kadar düşük olabilecek bir melez çerçevenin başarımını göstermektir.



Şekil 5.1. SGD eniyileyicisinin test ve eğitim süreçlerine ait gerçek ve tahmin edilen değerleri

Benzer şekilde LSTM için yapılan hiper-parametre değerlerinin hesaplanması BiLSTM mimarisi için de yapılmıştır. Tablo 5.2.'de BiLSTM mimarisi için hesaplanmış en düşük test MSE değerine göre bulunan en iyi hiper-parametre değerleri hesaplanmıştır. Tabloda koyu olarak belirtilen değerler en düşük test MSE değerine ait hiper-parametre ve ilgili değerlerini ifade etmektedir. Tablo incelendiğinde, en iyi başarımı, varsayılan 0,01 öğrenme oranı ile SGD metodu benzer şekilde 450 iterasyon adımında vermiştir. Tablo 5.1.'de gözlemlediğimiz benzer sonuçlar burada da elde edilmiştir.

Tablo 5.2. BiLSTM mimarisinde farklı eniyileyici metotlarına göre en iyi sonuç veren hiper-parametre kombinasyonu

Eniyileyici	**Öğrenme Oranı**	**Birimler**	**İterasyon Sayısı**	**Yığın Boyutu**	**Eğitim MSE**	**Test MSE**
ADAM	0.001	16	100	128	0,005174	0,005928
RMSPROP	0.001	32	100	64	0,005079	0,006373
SGD	**0.01**	**8**	**450**	**16**	**0,006082**	**0,005733**

Zaman serilerinde LSTM *"birim sayısı"*'nın modelin karmaşıklığına ve dolayısı ile başarıma önemli etkileri vardır. Bu değeri, olması gerektiğinden küçük alırsak model veri setindeki örüntüleri gerektiği kadar öğrenemez ve düşük-uyum (ing: underfitting) dediğimiz durum oluşur. Diğer taraftan, gerektiğinden yüksek alınması ise veri setindeki örüntülerin aşırı öğrenilmesi (ezberlenmesi) aşırı-uyum (ing: overfitting) durumu meydana gelir. Bu nedenle veri setindeki örüntülerin karmaşıklığına göre karar verilmesi gerekmektedir.

Gerek LSTM gerekse BiLSTM mimarilerine ait toplamda elde edilmiş 384 adet hiper-parametre kombinasyonlarına ilişkin MSE değerleri incelendiğinde *"birim sayısı"*'nın başarıma etkisinin olmadığı gözlemlenmiştir. Bu etkiyi göstermek için SGD eniyileyicisi için 450 iterasyon sayısı sabit tutularak en düşük (16) ve en yüksek (128) *"yığın boyutu"* değerlerine karşılık, farklı *"birim sayısı"* (8, 16, 32, 64) değerlerinde eğitim ve test MSE değerleri hesaplanmış ve ilgili değerler Tablo 5.3.'de verilmiştir.

Tablo 5.3. SGD eniyileyicisi için birim sayısı ve yığın boyutuna göre MSE değerleri

Birimler	**Yığın Boyutu**	**Eğitim MSE**	**Test MSE**	**Yığın Boyutu**	**Eğitim MSE**	**Test MSE**
8	16	0,005742	**0,005820**	128	0,028002	0,013457
16	16	0,005837	**0,005789**	128	0,030821	0,014524
32	16	0,005688	**0,005777**	128	0,026241	0,012748
64	16	0,006082	**0,005727**	128	0,025544	0,012615

Tablodaki MSE değerleri (eğitim ve test) incelendiğinde en düşük (16) *"yığın boyutu"*'na karşılık tüm *"birim sayısı"* için MSE değerlerinde kayda değer bir farklılık gözlemlenmemiştir. Benzer şekilde en büyük (128) *"yığın boyutu"*'na karşılık tüm *"birim sayısı"* için MSE değerlerinde kayda değer bir farklılık gözlemlenmemiştir. Fakat, *"yığın boyutu"* yüksek (128) olduğu durumdaki MSE değerinin, en düşük (16) olduğundaki duruma kıyasla daha yüksek olduğu görülmektedir. Kullandığımız veri seti özelinde, *"yığın boyutu"*'nun değeri gerek başarımı gerekse zaman karmaşıklığını

etkilese de *"birim sayısı"* başarıma etkisi ihmal edilebilecek kadar az olduğundan bu değer 8 olarak alınmıştır.

Diğer eniyileyiciler için de*"yığın boyutu"*'nun zamana ve başarıma olan etkilerinin analiz edilebilmesi amacı ile, her bir eniyileyici için *"birim sayısı"* 8, *"iterasyon sayısı"* 100 olarak sabitlendikten sonra eğitim ve test MSE değerleri ile birlikte çalıştırma zamanları ölçülmüştür. İlgili değerler Tablo 5.4.'de verilmiştir.

Tablo 5.4. Birim sayısı 8, iterasyon sayısı 100 olması durumunda farklı yığın boyutuna göre ADAM ve RMSPROP'un MSE değerleri

Eniyileyici	**Yığın Boyutu**	**Eğitim MSE**	**Test MSE**	**Çalıştırma Zamanı (s)**
RMSPROP	16	0,004669	0,006550	11,2985
	32	0,004849	0,006577	8,2478
	64	0,004950	0,006453	7,0279
	128	0,007520	0,006241	6,5616
ADAM	16	0,004694	0,006529	10,2812
	32	0,004740	0,006525	8,2354
	64	0,005223	0,005965	6,6878
	128	0,008743	0,006275	6,0453
SGD	16	0,017416	0,009322	10,5323
	32	0,020914	0,010843	7,8853
	64	0,034239	0,015671	6,7471
	128	0,030571	0,014206	6,3365

Verilen tabloda MSE değerleri ve çalıştırma zamanları incelendiğinde, *"yığın boyutu"*'nun büyük (128) alınması durumunda çalıştırma zamanları daha kısa olmaktadır. Çünkü eğitim adımında aynı anda 128 örnek kullanılacağından bu süreyi büyük oranda düşürecektir ve bellek kullanım açısından daha verimli olabilir fakat daha fazla bellek alanına ihtiyaç duymaktadır. Diğer taraftan, *"yığın boyutu"*'nun küçük (16) alınması durumunda çalıştırma zamanları uzamaktadır. Eğitim adımında aynı anda sadece 16 örnek kullanılacağından bu süreyi büyük oranda arttırır ve bellek kullanım verimini azaltırken, daha az bellek alanına ihtiyaç duyacağından beklenen bir sonuçtur. Çalıştırma zaman ölçü birimi ***sn*** cinsinden olduğu için ve tezin asıl amacının başarımı arttırmak olduğundan *"yığın boyutu"*'nu 16 olmasına karar verilmiştir.

Literatürde, daha karmaşık örüntülere sahip veri setlerinde, LSTM'nin çok katmanlı mimari uygulamalarının daha başarılı olduğu görülmektedir. Çok katmanlı LSTM, birden fazla LSTM katmanının ardışık olarak birleştirilmesiyle oluşturulur. Her katman, daha yüksek seviyeli özellikler ve bağlamsal ilişkiler öğrenmeye odaklanır. Alt katmanlar, veri içindeki düşük seviyeli özellikleri yakalayabilirken, üst katmanlar daha yüksek seviyeli ve soyut özellikleri temsil eder. Bu sayede, model, zaman içindeki uzun süreli bağımlılıkları daha etkin bir şekilde öğrenerek verilerin daha iyi temsil edilmesini ve daha iyi tahminler yapılmasını sağlar. Ancak, çok katmanlı LSTM'nin daha karmaşık yapıları, eğitim sürecini daha zorlu hale getirebilir ve aşırı öğrenme (ing: overfitting) riskini artırabilir. Bu nedenle, uygun düzeyde model karmaşıklığı ve düzenleme teknikleri kullanılarak, çok katmanlı LSTM'nin başarılı bir şekilde eğitilmesi ve verimli bir şekilde kullanılması önemlidir. Çok katmanlı LSTM'de bırakma/atma (ing: Dropout) kullanımı, modelin daha iyi performans ve daha iyi genelleştirme yeteneği elde etmesine katkıda bulunurken, eğitim sürecini daha istikrarlı hale getirir.

Veri setimize çok katmanlı LSTM uygulanırken *"birim değerleri"* 8, *"yığın boyutu"* 16 ve *"bırakma/atma"* değeri 0,2 alınarak deneyler yapılmıştır. SGD'nin iyi sonuç verdiği *"iterasyon sayısı"*, 450 olarak alınmıştır. Tablo 5.5.'de katman sayısının arttırılması ve katmanlara bırakma/atma eklenmesi modeli tek katmanlı ve bırakma/atma eklenmemiş LSTM modeline göre daha yüksek MSE değerleri aldığı görülmüştür.

Tablo 5.5. Çok katmanlı LSTM SGD eniyileyicisi için eğitim ve test MSE değerleri

Katman Sayısı	**Katmanlara Eklenen Bırakma/Atma Sayısı**	**Eğitim MSE**	**Test MSE**
1	**0**	**0,005742**	**0,005820**
1	1	0,011633	0,007473
2	0	0,024377	0,013208
2	2	0,024442	0,013049
3	0	0,033760	0,016445

Benzer deneyi ADAM ve RMSPROP'un iyi sonuç verdiği *"iterasyon sayısı"* 100, *"birim sayısı"* 8, ve *"yığın boyutu"* 16 alınarak deneyler yapılmıştır. Tablo 5.6.'da tek katmanlı LSTM ve çok katmanlı LSTM'nin MSE değerleri benzer çıkmıştır. Ancak, diğer hiper-parametrelerden birim sayısının veya yığın boyutunun değiştirerek daha iyi sonuçlar elde edilebileceği düşünülmektedir.

Tablo 5.6. Çok katmanlı LSTM ADAM ve RMSPROP eniyileyicisi için eğitim ve test MSE değerleri

Eniyileyici	**Katman Sayısı**	**Eğitim MSE**	**Test MSE**
RMSPROP	**1**	**0,004669**	**0,006550**
RMSPROP	2	0,004955	0,005276
ADAM	**1**	**0,004694**	**0,006529**
ADAM	2	0,004837	0,005362

Hiper-parametre analizleri ile birlikte değerlendirildiğinde, BiLSTM ve çok katmanlı LSTM, klasik LSTM'e (tek yönlü ve tek katmanlı) kıyasla hesap karmaşıklığı daha fazla olan bir mimariye sahip olduğundan bundan sonraki deneysel çalışmalara klasik LSTM ile devam edilecektir.

5.1.2. LSTM Ait Sapma-Değişkenlik Dengelemesi Analizi

MÖ'de geliştirilen modellerin karmaşıklığı ile performansı arasındaki dengenin *"sapma-değişkenlik dengelemesi"* (ing: bias-variance trade-off) sağlanması oldukça önemli bir kavramdır. Beklenen, oluşturulacak modelin, ne düşük uyum içinde -ki bu yüksek sapma (bias) anlamına gelir-, ne de aşırı uyum içinde -ki bu yüksek değişkenlik (ing: variance) anlamına gelir- olması tercih edilmez. Bu nedenle *"sapma-değişkenlik"* arasında bir dengenin (ing: trade-off) sağlanması gerekir.

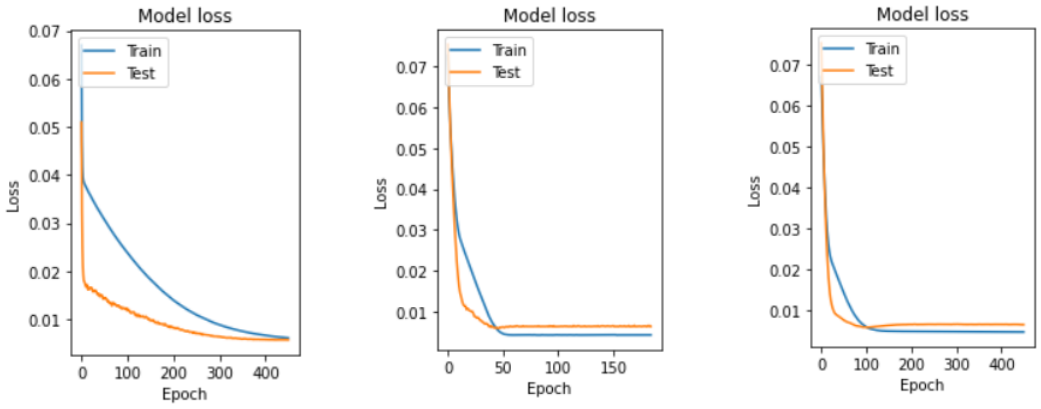
Bu çerçevede, hiper-parametrelerin etkileri analiz edildiğinde *"sapma-değişkenlik dengelemesi"*'ni etkileyen en belirgin hiper-parametrenin *"iterasyon sayısı"* olduğu gözlemlenmiştir. Bu amaçla, tüm eniyileyiciler için, *"iterasyon sayısı"* 450 olarak sabit tutulmuş, elde edilen en düşük test MSE değerleri ölçülmüştür, Tablo 5.7.

Tablo 5.7. LSTM'nin iterasyon sayısı 450 olması durumunda farklı eniyileyicilere göre en düşük test MSE değerlerine göre bulunan hiper-parametrelere değerleri

Eniyileyici	**Birimler**	**İterasyon Sayısı**	**Yığın Boyutu**	**Eğitim MSE**	**Test MSE**
ADAM	64	450	128	0,004592	0,006519
RMSPROP	32	450	64	0,004669	0,006471
SGD	**64**	**450**	**16**	**0,006082**	**0,005727**

Tablo 5.7. incelendiğinde ADAM ve RMSPROP eniyileyicilerin eğitim MSE değerlerinin test MSE değerlerinden daha küçük olduğu görülmektedir. Her bir eniyileme yaklaşımına ait eğitim ve test süreçlerine ait kayıp (ing: loss) değerlerinin grafiği çizilmiştir.

Şekil 5.2. grafikleri incelendiğinde 450 iterasyon adımı boyunca RMSPROP yaklaşık 50 iterasyon adımından sonra, ADAM ise 100 iterasyon adımından sonra aşırı öğrenmeye başlıyor. Buna karşın SGD eniyileyicisi ancak 450 iterasyon adımında sapma-değişkenlik dengesine kavuşmuştur. 450 iterasyon adımından sonra SGD metodunun da aşırı uyum davranışı sergileyeceği aşikardır.



(a) SGD (b) RMSPROP (c) ADAM

Şekil 5.2. Farklı eniyileyicilere ait iterasyon adımına karşı kayıp (loss) değerleri

Aynı zamanda, grafiklerin eğitim süreleri incelendiğinde ADAM ve RMSPROP gibi adaptif öğrenme oranına sahip eniyileyiciler, SGD eniyileyicisine kıyasla çok daha hızlı bir öğrenme süreci geçirdikleri görülmektedir.

5.2. PSO-LSTM MELEZ ÇERÇEVE

Bu bölümde, PSO-LSTM melez çerçevesinde kullanılan PSO algoritmasına ait en iyi hiper-parametre değerleri bulunurken, SGD eniyileyicisine parametre olarak aktarılacak en optimum *"öğrenme oranı"* belirlenmiştir. Bu sayede SGD eniyileyicisi adaptif olarak LSTM mimarilerinde kullanılmaya başlanmıştır. Ardından LSTM ve PSO-LSTM yaklaşımlarının başarımı ve *"sapma-değişkenlik dengelemesi"* analizi yapılmıştır. Son olarak PSO-LSTM melez çerçevenin etkinliği diğer dillere ait veri setleri üzerinden tartışılmıştır.

5.2.1. PSO-LSTM İçin Hiper-Parametre Ayarı

Tezde asıl amaçlanan zaman serilerinde PSO-LSTM melez çerçevesinin etkinliğinin gösterilmesi olduğundan, öncelikle bu çerçeveye ait hiper-parametrelerin ayarlanması gerekmektedir. Bu deneyleri yaparken, bir önceki bölümde detaylı analizi yapılarak karar verilen hiper-parametre değerleri alınarak devam edilmiştir. Belirlenen hiper-parametre değerleri: *"birim sayısı"* 8, *"iterasyon sayısı"* 100 ve *"yığın boyu"* 16. Bu hiper-parametre değerlerine ilişkin farklı eniyileyicilere ait test MSE değerleri Tablo 5.8.'de verilmiştir.

Tablo 5.8. Birim sayısı 8, iterasyon sayısı 100, yığın boyutu 16 olması durumunda eniyileyicilerin MSE değerleri

Eniyileyici	**Test MSE**
ADAM	0,006529
RMSPROP	0,006550
SGD	0,009322

Burada SGD için iterasyon sayısını 100 almamızın nedeni, yukarıda da bahsedildiği üzere, *"sapma-değişkenlik dengelemesi"* göz önünde bulundurarak, zaman karmaşıklığı en az olacak şekilde en düşük MSE değerine sahip alternatif PSO-LSTM melez çerçeve modeli kurmaktır.

Bu amaçla, Tablo 5.9.'da PSO algoritmasına ait hiper-parametreler (c1, c2, w) için en uygun (optimum) değerleri hesaplanmıştır. Aynı zamanda PSO algoritmasına, *"öğrenme oranı"* parametre olarak verilmiş ve [0.001, 0.9] arasında rastgele değer seçerek çalışma yapılmıştır. Ayrıca, literatürde benzer çalışmalarda kabul gören (Sun ve ark., 2022). **c1** ve **c2** hiper-parametrelerinin değerleri 1.5 ve **w** hiper-parametresinin değeri 0.9 alınarak elde edilen bulgular karşılaştırılmıştır.

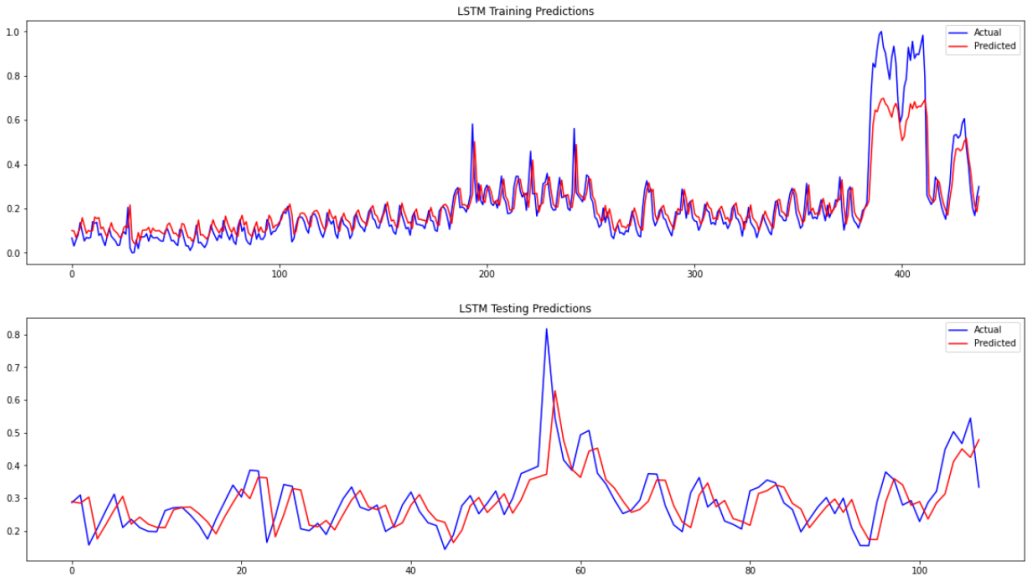
Tablo 5.9. PSO-LSTM'nin PSO'ya ait optimum hiper-parametre ve literatür kabul değerleri

c1	**c2**	**w**	**Test MSE**	**Öğrenme Oranı**
0,2	**0,6**	**1,2**	**0,005577**	**0,031645**
1,5	1,5	0,9	0,006401	0,083986

Tablo 5.9'da bahsedilen karşılaştırmaya ait test MSE değerleri ve bulunan en uygun öğrenme oranları değerleri görülmektedir. İlk satır bizim bulduğumuz hiper-parametre değerleri ile elde edilmiş öğrenme oranı ile çalıştırılan LSTM mimarisinin sonuçları, 2. Satırda literatürde kabul gören hiper-parametre değerleri ile elde edilen öğrenme oranı ve sonuçta elde edilen MSE değeri görülmektedir. Bulgular bizim

bulduğumuz en iyi hiper-parametre değerleri ile daha yüksek başarım elde edilmiş olduğunu göstermiştir.

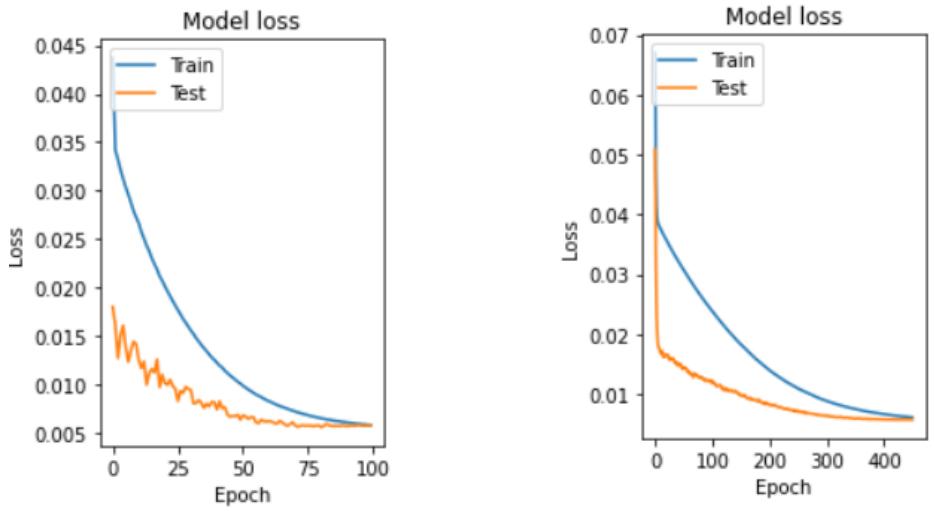
Diğer taraftan Tablo 5.8.'teki sonuçlar ile karşılaştırıldığında SGD eniyileyicisinin varsayılan *"öğrenme oranı"* ile 100 iterasyonda elde ettiği başarımdan çok daha iyi olduğu, hatta ADAM ve RMSPROP eniyileyicilerine kıyasla çok daha düşük MSE değerlerinin elde edildiği görülmektedir. Diğer bir önemli başarım ise, Tablo 5.1.'de gözlemlediğimiz gibi SGD eniyileyicisinin 450 iterasyon sayısı ile elde edilen MSE değeri **0,005727** iken, bulduğumuz hiper-parametre değerleri ile elde edilen MSE değeri **0,005577** olarak bulunmuştur ki bu da PSO-LSMT melez çerçevenin başarımının daha iyi olduğunu göstermektedir. Şekil 5.3.'de İngilizce web trafik verileri için yapılan deneyde gerçek ve tahmin değerlerinin yakın seyrettiği görülür.



Şekil 5.3. İngilizce web trafik için PSO-LSTM eğitim ve test tahminleri

5.2.2. PSO-LSTM Sapma-Değişkenlik Dengelemesi Analizi

PSO-LSTM melez çerçevenin *"sapma-değişkenlik dengelemesi"* analizini yapmak amacı ile oluşturulan Şekil 5.4 incelendiğinde, PSO-LSTM tabanlı mimari (SGD), klasik LSTM'deki SGD'ye kıyasla çok daha düşük iterasyon sayısında (~100) *"sapma-değişkenlik"* dengesi sağlanmıştır.



(a) PSO-LSTM (b) LSTM

Şekil 5.4. SGD için PSO-LSTM ve LSTM iterasyon adımına karşı kayıp (loss) değerleri

SGD, düşük iterasyonlarda test ve eğitim kayıp değerlerinin yeterince yakınsamasını sağlayamayan bir optimizasyon yöntemidir. Ancak, melez bir yaklaşım olan PSO-LSTM yöntemi sayesinde bu durum önemli ölçüde iyileştirilmiştir. PSO, LSTM modelinin eğitiminde kullanılarak, zaman serilerinin analizinde daha iyi performans elde edilmiştir. PSO'nun adaptif öğrenme oranı ve global arama yeteneği sayesinde, SGD eniyileyicisinin düşük iterasyonlardaki yetersizliği giderilmiştir.

5.2.3. PSO-LSTM'nin Diğer Veri Setleri Üzerinde Başarımı

Yöntemin genel geçerliliğini değerlendirmek amacıyla, diğer iki büyük görüntülenme alan olan Japonca ve Almanca veri setleri üzerinde de deneyler yapılmıştır ve başarılı sonuçlar elde edilmiştir. Melez yönteme ait deneyler sonucu elde edilen PSO hiper-parametreleri ve öğrenme oranı Tablo 5.10.'da verilmiştir. İngilizce web trafiği için elde edilen PSO hiper-parametre değerleri baz alınarak japonca ve almanca web trafiği veri seti üzerindeki etkisi incelendiğinde öğrenme oranının, en iyi PSO hiper-parametre değerlerindeki öğrenme oranları ile benzer çıktığı görülmektedir.

Tablo 5.10. PSO-LSTM'nin farklı veri setleri için PSO'ya ait optimum hiper-parametre değerleri

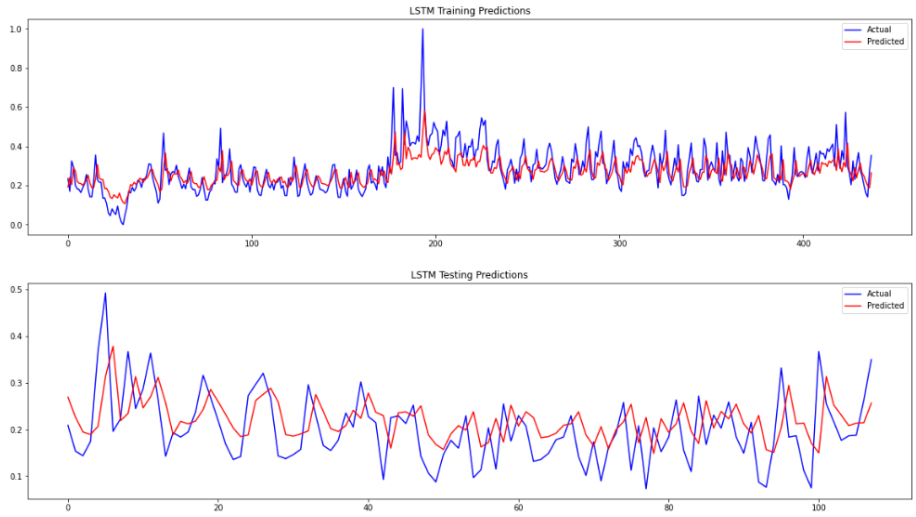
Veri Seti	**c1**	**c2**	**w**	**Öğrenme Oranı**
İngilizce Web Trafiği	0,2	0,6	1,2	0,031645
Japonca Web Trafiği	0,2	0,6	1,2	0,207277
Almanca Web Trafiği	0,2	0,6	1,2	0,056092
Japonca Web Trafiği	0,5	0,1	0,9	0,195492
Almanca Web Trafiği	0,5	0,1	0,6	0,050301

Bu sonuçlar, yöntemin dil ve kültürel farklılıklara bağlı olmaksızın farklı web trafiği zaman serilerini analiz edebileceğini ve güçlü bir genelleme yeteneğine sahip olduğunu Tablo 5.11.'de gösterilmektedir.

Tablo 5.11. LSTM ve PSO-LSTM sonuçları

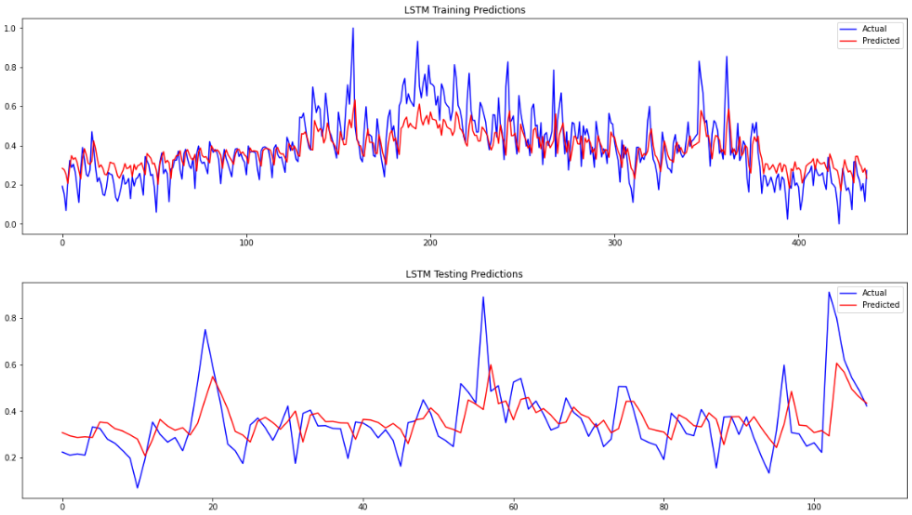
Yöntem	**Eğitim MSE**	**Test MSE**
Web Trafik İngilizce		
LSTM	0,017416	0,009322
PSO-LSTM	**0,006840**	**0,005577**
Web Trafik Japonca		
LSTM	0,010975	0,010986
PSO-LSTM	0,007085	0,005807
Web Trafik Almanca		
LSTM	0,024958	0,020203
PSO-LSTM	0,012787	0,015034

Japonca web trafiği için PSO-LSTM melez yönteminin eğitim ve test grafikleri için gerçek ve tahmin değerleri birbirine yakın olduğu Şekil 5.5.'de gözlemlenmiştir.



Şekil 5.5. Japonca web trafik veri seti için PSO-LSTM eğitim ve test tahminleri

Almanca web trafiği için PSO-LSTM melez yönteminin eğitim ve test grafikleri için gerçek ve tahmin değerleri birbirine yakın olduğu Şekil 5.6.'de gözlemlenmiştir.



Şekil 5.6. Almanca web trafik veri seti için PSO-LSTM eğitim ve test tahminleri

Bu çalışma PSO tabanlı adaptif öğrenme oranı kullanılarak düşük iterasyonlarda SGD'nin iyi sonuçlar verdiğini göstermiştir. LSTM modelinin performansını artırmak için SGD'nin adaptif bir şekilde optimize edilmesinin önemini vurgulamaktadır.

SONUÇ VE GELECEK ÇALIŞMA

Bu çalışmada, Wikipedia makaleleri web sitesi görüntülemeleri veri seti üzerinde LSTM ve PSO algoritması kullanılarak zaman serisi tahmini gerçekleştirildi.

Tek yönlü, çift yönlü, katmanlı LSTM modelleri İngilizce Wikipedia web trafiği zaman serisi verileri üzerinde uygulandı. LSTM'nin derin öğrenme yetenekleri sayesinde, zaman serisi verilerindeki gizli yapıları tanımlayabildi ve gelecekteki değerleri tahmin etmek için öğrendi. Bu çalışma kapsamında farklı birim sayısı, iterasyon sayısı, yığın boyutu ve eniyileyicinin değerleri ile LSTM modeli eğitilmiş ve sonuçlar incelenmiştir. Diğer yöntemlere göre en düşük MSE değerini tek yönlü LSTM modeli vermiştir. Elde edilen sonuçlar incelendiğinde, SGD optimizasyonu yüksek iterasyon sayısında iyi sonuçlar elde edebilmektedir. Ancak, düşük iterasyonlarda SGD'nin performansı yetersiz kalmıştır. Bu nedenle, SGD optimizasyonunun düşük iterasyonlarda da iyi sonuçlar vermesini sağlamak için adaptif öğrenme oranı ile çalışan PSO yöntemi kullanılmıştır. PSO, birçok olası parametre kombinasyonu arasında gezinerek en iyi kombinasyonu bulmaya çalışır. Bu çalışmada, PSO parametreleri (kendilik faktörü, sosyal faktör, sürü ağırlık faktörü) sistemli bir şekilde değiştirilerek, en iyi öğrenme oranı ve en düşük MSE değerleri elde edildi. PSO ile LSTM modelinin öğrenme oranı düşük iterasyonlarda optimize edilerek, SGD'nin düşük iterasyonlardaki performansı önemli ölçüde artırılmıştır.

Bu çalışmada, web trafik zaman serilerinin analizi için kullanılan melez PSO-LSTM yöntemi, ingilizce veri seti üzerinde başarılı bir şekilde uygulanmış ve gelecek tahminleri için daha doğru sonuçlar elde edilmiştir. Aynı deneyleri almanca ve japonca web trafik verileri için de denenmiş ve LSTM yöntemine oranla melez PSO-LSTM yönteminin daha düşük MSE değeri verdiği görülmüştür.

Gelecekte, farklı veri setleri ve LSTM yapıları üzerinde daha geniş kapsamlı deneyler yapılabilir. Ayrıca, farklı optimizasyon yöntemleri ve hiperparametre kombinasyonları da değerlendirilebilir. Bu çalışmanın sonuçları, LSTM modelinin optimizasyonunda yeni yaklaşımların geliştirilmesine katkı sağlayabilir ve daha iyi performans elde etmeyi mümkün kılabilir. Kendi çalışmamızın verilerine, sonuçlarına ve amacına göre uyarlamanız önemlidir.

KAYNAKÇA

Adhikari, R., & Agrawal, R.K. (2013). An Introductory Study on Time Series Modeling and Forecasting. Lambert Academic Publishing.

Agarap, A. F. (2018). Deep Learning using Rectified Linear Units (ReLU). Issue 1, pp. 2–8. Retrieved from http://arxiv.org/abs/1803.08375.

Anderson, D., & McNeill, G. (1992). Artificial neural networks technology. Kaman Science Corporation, New York, USA.

Awad, M., & Khanna, R. (2015). Efficient Learning Machines. Apress Berkeley, CA.

Bayramoğlu, M. F. (2007). Finansal endekslerin öngörüsünde yapay sinir ağı modellerinin kullanılması: İMKB ulusal 100 endeksinin gün içi en yüksek ve en düşük değerlerinin öngörüsü üzerine bir uygulama. Yüksek Lisans Tezi, Karaelmas Üniversitesi, Sosyal Bilimler Enstitüsü, Zonguldak.

Bengio, Y. (2009). Learning deep architectures for AI. Foundations and trends in Machine Learning, 2(1), 1-127.

Bengio, Y., Simard, P., & Franscon, P. (2015). Learning Long-Term Dependencies with Gradient Descent is Difficult. IEEE Transactions on Neural Networks, 5(2), 157-166.

Bishop, C. (1994). Novelty detection and neural network validation. Proceedings of the IEEE Conference on Vision, Image and Signal Processing, 141, IET, 217–222.

Chatfield, C. (1989). The Analysis of Time Series: An Introduction. Chapman and Hall.

Chatfield, C. (1995). The Analysis of Time Series: An Introduction (5th ed.). New York: Chapman and Hall.

Chatfield, C. (2000). Time-Series Forecasting. Chapman & Hall/CRC.

Chatfield, C. (2016). The analysis of time series: An introduction. CRC Press.

Cheng, B., & Titterington, D. M. (1994). Neural networks: A review from a statistical perspective. Statistical Science, 9(1), 2-30.

Chimmula, V. K. R., & Zhang, L. (2020). Time series forecasting of COVID-19 transmission in Canada using LSTM networks. Chaos, Solitons & Fractals, 135.

Cho, K., Merriënboer, B., Schwenk, H., Bougares, F., Bengio, Y., & Bahdanau, D. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. ArXiv:1406.1078v3 [cs.CL].

Çaparoğlu, Ö. F. (2021). Yapay Sinir Ağları İle Pandemi Sürecinde Kısıtlama Stratejilerinin Etkinliğinin Belirlenmesi İçin Bir Model Önerisi. Atatürk Üniversitesi, Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Bölümü.

Çetiner, H., & Kara, B. (2022). Recurrent Neural Network Based Model Development for Wheat Yield Forecasting. 9(16), 204-218. https://doi.org/10.54365/adyumbd.1075265.

Demuth, H., Beale, M., & Hagan, M. (2009). User Guide, Neural Networks ToolboxTM 6. The MathWorks, Inc., ISBN:0-9717321-0-8, 2-25, 3-5, 8-6.

Deng, L., & Yu, D. (2014). Deep learning: methods and applications. Foundations and Trends in Signal Processing, 7(3-4), 197-387.

Diamantaras, K. I., & Kung, S. Y. (1996). Principal component neural networks: theory and applications. John Wiley and Sons Inc., New York, USA.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. Micro Machine and Human Science, Proceedings of the Sixth International Symposium, 39-43.

Elmas, Ç. (2003). Yapay sinir ağları. Seçkin Yayıncılık, Ankara.

Estabrooks, A., Jo, T., & Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced data sets. Computational Intelligence, 20(1), 18-36. doi: 10.1111/j.0824-7935.2004.t01-1-00228.x.

Fausett, L. (1994). Fundamentals of neural networks. Prentice Hall, USA.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669.

Folk, M. (2012). A First Course on Time Series Analysis — Examples with SAS. Chair of Statistics, University of Würzburg.

Franses, P.H. & Van Dijk, D. (2000). Non-linear time series models in empirical finance. Cambridge University Press.

Fu, R., Zhang, Z., & Li, L. (2016). Using LSTM and GRU neural network methods for traffic flow prediction. Proceedings of the Youth Academic Annual Conference of Chinese Association of Automation (YAC), 324-328.

Garcia, S., Luengo, J., & Herrera, F. (2015). Data Preprocessing in Data Mining. In Intelligent Systems Reference Library, 72.

Gilik, A., Ogrenci, A. S., & Ozmen, A. (2022). Air quality prediction using CNN+LSTM-based hybrid deep learning architecture. Environmental Science and Pollution Research, 29, 11920-11938.

Granger, C.W.J. (1981). Some properties of time series data and their use in econometric model specification. In Journal of econometrics, 16(1), 121–130.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A Search Space Odyssey. IEEE Transactions on Neural Networks and Learning Systems, 28(10), 2222-2232.

Guyton, A. C., & Hall, J. E. (2006). Textbook of medical physiology. Elsevier Saunders, Pennsylvania, USA.

Hagan, M. T., Demuth, H. B., & Beale, M. (1996). Neural Network Design. Boston, PWS Publishing Co., 10-04.

Hamilton, J.D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. In Econometrica: Journal of the Econometric Society, 357–384.

Haykin, S. (1999). Neural networks: a comprehensive foundation. Pearson Prentice Hall.

Haykin, S. (2009). Neural Networks and Learning Machines (3d Edition). Prentice Hall, ISBN-13:978-0-13-147139-9, 1-68.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.

Houssein, E. H., Gad, A. G., Hussain, K., & Suganthan, P. N. (2021). Major Advances in Particle Swarm Optimization: Theory, Analysis, and Application. Swarm and Evolutionary Computation, 63, 100868. doi:10.1016/j.swevo.2021.100868.

Hsu, T. Y. (2021). Machine learning applied to stock index performance enhancement. Journal of Banking and Financial Technology, 5, 21-33.

Jahnke, P. (2015). Machine Learning Approaches for Failure Type Detection and Predictive Maintenance.

Jin, X., Xu, C., Feng, J., Wei, Y., Xiong, J., & Yan, S. (2015). Deep learning with s-shaped rectified linear activation units. Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16).

Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). An Empirical Exploration of Recurrent Network Architectures. International Conference on Machine Learning. PMLR, 2342-2350.

Kayım Halil. (1985). İstatiksel Ön Tahmin Yöntemleri. H.Ü. İ.İ.B.F. Yayın No.11, Ankara, s:12.

Kennedy, J. (2011). Particle swarm optimization. In Encyclopedia of machine learning (ss. 760-766). Boston: Springer.

Kingma, D. P., & Ba, J. L. (2014). Adam: A method for stochastic optimization. arXiv preprint, arXiv:1412.6980.

Kotsiantis, S., Kanellopoulos, D., & Pintelas, P. E. (2006). Data Preprocessing for Supervised Learning. International Journal of Computer Science, 1, 111-117.

Krenker, A., Bester, J., & Kos, A. (2011). Introduction to Artificial Neural Networks. In Artificial Neural Networks-Methodological Advances and Biomedical Applications (Ed. K. Suzuki), InTech, 3-19.

Kumar, S., & Ningombam, D. (2018). Short-Term Forecasting of Stock Prices Using Long Short Term Memory. 2018 International Conference on Information Technology (ICIT), 182-186.

Kumar, G., Singh, U. P., & Jain, S. (2022). An adaptive particle swarm optimization-based hybrid long short-term memory model for stock price time series forecasting. Soft Computing, 26, 12115-12135. https://doi.org/10.1007/s00500-022-07451-8

Kushwah, J. S., Kumar, A., Patel, S., Soni, R., Gawande, A., & Gopta, S. (2022). Comparative study of regressor and classifier with decision tree using modern tools. Materials Today: Proceedings, 56(6), 3571-3576.

Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., & Ng, A. Y. (2011). On optimization methods for deep learning. Proc. 28th Int. Conf. Machine Learning, 265–272.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

Lewis, N. D. (2017). Neural Networks for Time Series Forecasting With R: Intuitive Step by Step Blueprint for Beginners. South Carolina: CreateSpace Publishing.

Liao, T.W. (2005). Clustering of time series data—a survey. In Pattern Recognition, 38(11), 1857–1874.

Lipton, Z. C., Berkowitz, J., & Elkan, C. A. (2015). Critical review of recurrent neural networks for sequence learning. Retrieved from https://arxiv.org/abs/1506.00019

Lorenzo, P. R., Nalepa, J., Kawulok, M., Ramos, L. S., & Pastor, J. R. (2017). Particle swarm optimization for hyper-parameter selection in deep neural networks. Proceedings of the Genetic and Evolutionary Computation Conference. ACM.

Lv, L., Kong, W., Qi, J., & Zhang, J. (2018). An improved long short-term memory neural network for stock forecast. MATEC Web of Conferences, 232, 01024. https://doi.org/10.1051/matecconf/201823201024

Lydia, A. A., & Francis, F. S. (2019). Adagrad: An Optimizer for Stochastic Gradient Descent. International Journal of Information And Computing Science, 6(5), 599-568.

MacKay, D. J. C. (2003). Information theory, inference and learning algorithms. Cambridge University Press.

Martinez-Álvarez, F., Troncoso, A., Asencio-Cortés, G., & Riquelme, J. C. (2015). A survey on data mining techniques applied to electricity-related time series forecasting. Energies, 8(11), 13162-13193.

McCleay, R., & Hay, R. A. (1983). Applied Time Series Analysis: for the Social Sciences. Sage Publications, s.43, 52, 53.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10) (pp. 807-814).

Özmen, A., Yüzer, A.F., Ağaoğlu, E., Tatlıdil, H., & Şıklar, E. (2006). İstatistik (3rd ed.). Anadolu Üniversitesi. Eskişehir.

Öztemel, E. (2003). Yapay sinir ağları. Papatya Yayıncılık.

Öztemel, E. (2006). Yapay Sinir Ağları. İstanbul: Papatya Yayıncılık, 25.

Öztürk, K., & Şahin, M. E. (2018). Yapay Sinir Ağları ve Yapay Zeka'ya Genel Bir Bakış. Takvim-i Vekayi, 6(2), 25-36.

Patterson, J., & Gibson, A. (2017). Deep Learning: A practitioner's Approach. O'Reilly Press, ISBN:978-1-491-91425-0, 41-114, 125-165.

Pindyck, R. S., & Rubinfield, D. L. (1991). Econometric Models and Economic Forecasts. 3.ed. McGraw-Hill, USA, p.443-444, 473-500.

Pradeepkumar, D. ve Ravi, V. (2017). Forecasting Financial Time Series Volatility Using Particle Swarm Optimization Trained Quantile Regression Neural Network. Applied Soft Computing, 58, pp. 35-52.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

Sagheer, A. ve Kotb, M. (2019). Time series forecasting of petroleum production using deep LSTM recurrent networks. Neurocomputing, 323, 203-213.

Scharf, L.L. & Demeure, C. (1991). Statistical signal processing: detection, estimation, and time series analysis. Vol. 63. Addison-Wesley Reading, MA.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117.

Seddighi, H. R., Lawyer, K. A., & Katos, A. V. (2000). Econometrics: A Practical Approach. Routledge Taylor & Francis Group, London, UK, p.252.

Serper, Ö. (2000). Uygulamalı istatistik. Ezgi Kitabevi, Bursa.

Seyyarer, E., Karci, A., & Ateş, A. (2022). Stokastik ve deterministik hareketlerin optimizasyon süreçlerindeki etkileri. Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 37(2), 949-966.

Shen, G., Tan, Q., Zhang, H., Zeng, P., & Xu, J. (2018). Deep Learning With Gated Recurrent Unit Networks for Financial Sequence Predictions. Procedia Computer Science, 131, 895-903.

Smith, K. A. (2002). Neural Networks for Business: An Introduction. In Neural Networks in Business: Techniques and Applications (Eds.: Smith, K. A. & Gupta, J. N. D.) Idea Group Publishing, USA.

Sreelekshmy, S., Vinayakumar, R., Gopalakrishnan, E. A., Vijay Krishna, M., & Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN sliding window model. ISSN: 2148-2683.

Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM-a tutorial into Long Short-Term Memory Recurrent Neural Networks.

Sun, Y., Wang, X., & Yang, J. (2022). Modified Particle Swarm Optimization with Attention-Based LSTM for Wind Power Prediction. Energies, 15, 4334. https://doi.org/10.3390/en15124334.

Subasi, A. (2020). Machine learning techniques. In: Practical Machine Learning for Data Analysis Using Python, 91-202.

Toprak, Ş. (2023). Makine Öğrenmesi Yöntemlerini Kullanarak Bir Petrokimya Firmasının Hisse Senedi Fiyat Tahmini. Sakarya Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı Yüksek Lisans Tezi.

Tüzen, F.T. (2012). Türkiye Turizm Gelirinin Öngörüsünde Zaman Serilerinin Bileşenlerine Ayrıştırılarak Yapay Sinir Ağları ve Box-Jenkins Yöntemleri İle Karşılaştırmalı Analizi. Kafkas Üniversitesi Sosyal Bilimler Enstitüsü İşletme Anabilim Dalı Yüksek Lisans Tezi.

Wang, J., Gu, Q., Wu, J., Liu, G., & Xiong, Z. (2016). Traffic speed prediction and congestion source exploration: A deep learning method. Data Mining (ICDM), IEEE 16th International Conference, 499–508.

Wei, W. W. S. (2006). Time Series Analysis: Univariate and Multivariate Methods. Pearson Education.

Wei, W.S. (2006). Time series analysis. In The Oxford Handbook of Quantitative Methods in Psychology: Volume 2.

Xu, Y., Hu, C., Wu, Q., Jian, S., Li, Z., Chen, Y., Zhang, G., Zhang, Z., & Wang, S. (2022). Research on particle swarm optimization in LSTM neural networks for rainfall-runoff simulation. Journal of Hydrology, 608, 127553. https://doi.org/10.1016/j.jhydrol.2022.127553

Yadav, A., Jha, C. K., & Sharan, A. (2020). Optimizing LSTM for time series prediction in Indian stock market. Procedia Computer Science, 167, 2091-2100.

Yaffee, R., & McGee, M. (2000). Introduction to Time Series Analysis and Forecasting with Applications of SAS and SPSS. Academic Press, USA, p.5-7, 154-169, 173-174, 266, 269-346.

Yazan, E., & Talu, M. F. (2017). Comparison of the stochastic gradient descent based optimization techniques. International Artificial Intelligence and Data Processing Symposium, Malatya-Türkiye, 1-5.

Yao, Y., Han, L., & Wang, J. (2018). LSTM-PSO: Long Short-Term Memory Ship Motion Prediction Based on Particle Swarm Optimization. 2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC), Xiamen, China, pp. 1-5. doi: 10.1109/GNCC42960.2018.9018688.

Yao, H., Tang, X., Wei, H., Zheng, G., & Li, Z. (2019). Revisiting Spatial-Temporal Similarity: A Deep Learning Framework for Traffic Prediction. Proceedings of the AAAI Conference on Artificial Intelligence, 33(01), 5668-5675.

Yıldız, B. (2009). Finansal Analizde Yapay zekâ. İstanbul: Beta Basım.

Yılmaz, A., (2021). Zaman Serisi Öngörüsünde Dendrit Yapay Sinir Ağları İçin Parçacık Sürü Optimizasyonuna Dayalı Dayanıklı Öğrenme Algoritması (Yüksek Lisans Tezi). İstatistik Anabilim Dalı, Enstitü Anabilim Dalı.

Zeiler, M. D. (2012). Adadelta: An Adaptive Learning Rate Method. arXiv preprint arXiv:1212.5701.

Zhang, P. G. (2003). Business Forecasting with Artificial Neural Networks: An Overview. In Neural Networks in Business Forecasting (Ed.: Zhang, G. P.) Idea Group Publishing.

Url-1 < *https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html*>, erişim tarihi 09.06.2023.

Url-2 < *https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html*>, erişim tarihi 09.06.2023.

Url-3 < *https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html*>, erişim tarihi 09.06.2023.

Url-4 http://karpathy.github.io/2015/05/21/rnn-effectiveness>, erişim tarihi 11.06.2023.

Url-5 < *http://colah.github.io/posts/2015-08-Understanding-LSTMs/*>, erişim tarihi 11.06.2023.