TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ZAMAN SERİSİ ANALİZ ve TAHMİNİ: DERİN ÖĞRENME YAKLAŞIMI

YÜKSEK LİSANS TEZİ

Mehmet Uğur GÜDELEK

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Doç. Dr. Ahmet Murat ÖZBAYOĞLU

NİSAN 2019

Fen Bilimleri Enstitüsü Onayı

.....
Prof. Dr. Osman EROĞUL
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

.....
Prof. Dr. Oğuz ERGİN
Anabilimdalı Başkanı

TOBB ETÜ, Fen Bilimleri Enstitüsü'nün 161111113 numaralı Yüksek Lisans Öğrencisi **Mehmet Uğur GÜDELEK** 'in ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"ZAMAN SERİSİ ANALİZ ve TAHMİNİ: DERİN ÖĞRENME YAKLAŞIMI"** başlıklı tezi **11.04.2019** tarihinde aşağıda imzaları olan jüri tarafından kabul edilmiştir.

Tez Danışmanı:	**Doç. Dr. Ahmet Murat ÖZBAYOĞLU** TOBB Ekonomi ve Teknoloji Üniversitesi	.....
Jüri Üyeleri:	**Prof. Dr. Tolga CAN (Başkan)** Orta Doğu Teknik Üniversitesi	.....
	Doç. Dr. Osman ABUL TOBB Ekonomi ve Teknoloji Üniversitesi	.....

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

Mehmet Uğur GÜDELEK

ÖZET

Yüksek Lisans Tezi

ZAMAN SERİSİ ANALİZ ve TAHMİNİ: DERİN ÖĞRENME YAKLAŞIMI

Mehmet Uğur GÜDELEK

TOBB Ekonomi ve Teknoloji Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Doç. Dr. Ahmet Murat ÖZBAYOĞLU

Tarih: Nisan 2019

Günlük hayatta, oldukça fazla problem, zaman serisi verileri içermektedir. Zaman serisi verilerinin analizini veya gelecek değer tahminlerini iyi bir şekilde yapabilmek, bu problemlerin çözümü için çok önemlidir. Çeşitli istatistiksel analiz, matematiksel analiz, sinyal işleme, makine öğrenmesi ve onun alt alanı olan derin öğrenme yöntemleri, zaman serisi verilerini analiz etmek ve gelecek tahmini yapabilmek için kullanılmaktadırlar. Özellikle, son yıllarda popülaritesi giderek artan derin öğrenme yöntemleri, karmaşık zaman serisi problemlerinin çözümünde, geleneksel yöntemlere göre daha başarılı olmuş ve kullanımları hızla artmıştır. Ancak, bahsedilen analiz ve tahminlere, baştan sona nasıl yaklaşılacağını, hangi modellerin kullanılması gerektiğini, seçilen modelin nasıl kullanılacağını ve veri setinin nasıl hazırlanması gerektiğini, bütün bir şekilde ele alan çalışmalara literatürde pek rastlanmamıştır. Önerilen tez ile, çeşitli zaman serisi problemleri incelenmiş ve yaklaşımlar anlatılmıştır. Durum denetlemeli ve denetlemesiz LSTM karşılaştırması yapılmış, basit problemler üzerinde analizleri yapılmış, iç yapıları incelenmiş, bir hanenin elektrik üretim ve tüketim miktarları tahmin edilerek, hanedeki bataryanın optimizasyonu yapılmıştır. Batarya optimizasyonu yapıldığında, optimum sonuca %99 oranında yakınsanmış ve 3-zamanlı elektrik fiyat tarifesi kullanımı ile yüksek oranda kar sağlanmıştır. Bunun dışında, CNN ile finans verisi üzerinden hesaplanan teknik analiz özniteliklerinin yardımı ile gelecek değer tahmini yapılmıştır. Bu tahmin yapılırken, CNN modelinin girdileri olacak olan teknik indikatörlerin 2D resim şeklinde

dönüştürülmesi çalışılmamış bir konudur. 2D resim oluşturulurken kullanılan dendrogram kümeleme algoritması bir eksende korelasyonu sağlamış, diğer eksende zaman serisinin otokorelasyonundan faydalanılmıştır. Geliştirilen model, zaman serisi verisine uygun dönüşümler uygulandığında, eğitilebilmiş ve başarılı sonuçlar çıkarmıştır. Finans verisine uygulanan dönüşümler, farklı alanlardaki zaman serisi verilerine de uygulanabilir olduğu için, farklı mimarideki modeller de kullanışlı duruma geçmişlerdir.

Anahtar Kelimeler: Konvolüsyonel sinir ağı, LSTM, Makine öğrenmesi, Derin öğrenme, Enerji tahmini, Finansal veri analizi, Teknik analiz.

ABSTRACT

Master of Science

TIME SERIES ANALYSIS and FORECASTING: DEEP LEARNING APPROACH

Mehmet Uğur GÜDELEK

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Assoc. Prof. Ahmet Murat ÖZBAYOĞLU

Date: April 2019

In daily life, quite a lot of problems include time series data. Making good time series data analysis or future value estimations is very important to solve these problems. Various statistical analysis, mathematical analysis, signal processing, machine learning and deep learning methods that are subfields of machine learning are used to analyze time series data and to make a future forecast. In particular, deep learning methods, which have become increasingly popular in recent years, have been more successful than traditional methods in solving some complex time series problems and their usage has spread rapidly. However, in the literature, there is not a lot of research which deals with the analysis and estimations, how to approach the end-to-end solutions, which models should be used, how to use the selected model and how the dataset should be prepared. In the proposed thesis, various time series problems are examined and approaches are explained. Stateful LSTM and the stateless LSTM were compared, analyzed on simple problems, internal structures of them were examined. After that, electricity production and consumption amounts of a household were estimated and the battery of household were optimized. When battery optimizations were made, the optimum result was converged at a rate of 99% and a high rate of profit was achieved by the use of 3-rate electricity tariffs. In addition, the technical analysis features calculated over the financial data and used for the estimation of the future value of the financial data with the help of CNN, another deep learning model. In this estimation, the conversion of technical

indicators which will be the inputs of CNN model as 2D images is an uninvestigated issue. The dendrogram clustering algorithm used in 2D image rendering provided one-axis correlation, while the other axis was used for autocorrelation of the time series. When the results were examined and transformations were applied, developed method could be trained and gave successful results. Since the transformations applied to finance data are also applicable to time series data in different fields, models which have different architecture can be useful.

Keywords: Convolutional neural network, LSTM, Machine learning, Deep learning, Energy estimation, Financial data analysis, Technical analysis.

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Doç. Dr. Ahmet Murat Özbayoğlu'na, kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgiayar Mühendisliği Bölümü öğretim üyelerine, eğitimim boyunca bana burs veren TOBB Ekonomi ve Teknoloji Üniversitesi'ne ve destekleriyle her zaman yanımda olan aileme ve arkadaşlarıma, süreç boyunca yardımlarını eksik etmeyen Erdem Sezgin ve Ceren Özveri'ye çok teşekkür ederim.

İÇİNDEKİLER

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

KISALTMALAR

ADADELTA : Adaptif Öğrenme Faktörü Yöntemi (Adaptive Learning Rate Method)
AdaGrad : Adaptif Gradyan Yöntemi (Adaptive Gradient Method)
ADAM : Adaptif Moment Tahmin Yöntemi (Adaptive Moment Estimation Method)
AE : Oto-Enkoder (Auto-Encoder)
ARIMA : Özbağlanımsal Tümleşik Hareketli Ortalama (Autoregressive Integrated Moving Average)
AUC : ROC eğrisi altında kalan alan (Area under the ROC curve)
BCE : İkili Çapraz-Entropi (Binary Cross-Entropy)
CE : Çapraz-Entropi (Cross-Entropy)
CNN : Konvolüsyonel Sinir Ağları (Convolutional Neural Networks)
DBN : Derin-İnanç Ağları (Deep-Belief Networks)
DMLP : Derin Çok Katmanlı Perceptron (Deep Multilayer Perceptron)
DT : Karar Ağaçları (Decision Trees)
EMA : Üssel Hareketli Ortalama (The Exponential Moving Average)
ENTSOE : Avrupa Elektrik İletim Sistemi İşleticileri Birliği (European Network of Transmission System Operators for Electricity)
FCNN : Tamamıyle Bağlı Sinir Ağları (Fully-connected Neural Networks)
FN : Yanlış negatif (False negative)
FNR : Yanlış negatif oranı (False negatif rate)
FP : Yanlış pozitif (False positive)
FPR : Yanlış pozitif oranı (False positive rate)
GRU : Gated Recurrent Unit
LSTM : Long Short-term Memory
MACD : MACD Göstergesi (The Moving Average Convergence/Divergence Oscillator)
MAE : Ortalama mutlak hata (Mean absolute error)
MFI : Para Akış indeksi (The Money Flow Index)
MILP : Karışık-tamsayılı Lineer Programlama (Mixed-integer Linear Programming)
MLP : Çok Katmanlı Perceptron (Multilayer Perceptron)
MSE : Ortalama kare hatası (Mean squared error)
NAG : Nesterov Hızlandırılmış Momentum (Nesterov Accelerated Momentum)
NLP : Doğal Dil İşleme (Natural Language Processing)
RBM : Sınırlı Boltzmann Makinesi (Restricted Boltzmann Machine)

ReLU : Doğrusal Doğrultmaç Unitesi (Rectified Linear Unit)
RL : Pekiştirmeli Öğrenme (Reinforcement Learning)
RMSProp : Adaptif bir eniyileme yöntemi (An adaptive optimizer method)
RNN : Özyinelemeli Sinir Ağları (Recurrent Neural Networks)
ROC : Alıcı İşletim Karakteristiği (Receiver operating characteristic)
RSI : Göreli Güç İndeksi (The Relative Strenght Index)
SMA : Basit Hareketli Ortalama (The Simple Moving Average)
SO : Stokastik Osilatör (The Stochastic Oscillator)
SPY : SPDR S&P 500 Borsa Yatırım Fonu (SPDR S&P 500 Trust ETF)
SVM : Karar-Destek Makinası (Support Vector Machine)
SVR : Karar-Destek Regressörü (Support Vector Regressor)
TN : Doğru negatif (True negative)
TNR : Doğru negatif oranı (True negatif rate)
TP : Doğru pozitif (True positive)
TPR : Doğru pozitif oranı (True positive rate)
UO : Nihai Osilatör (The Ultimate Oscillator)
WR : Williams Yüzdesi (Williams Percent Range)

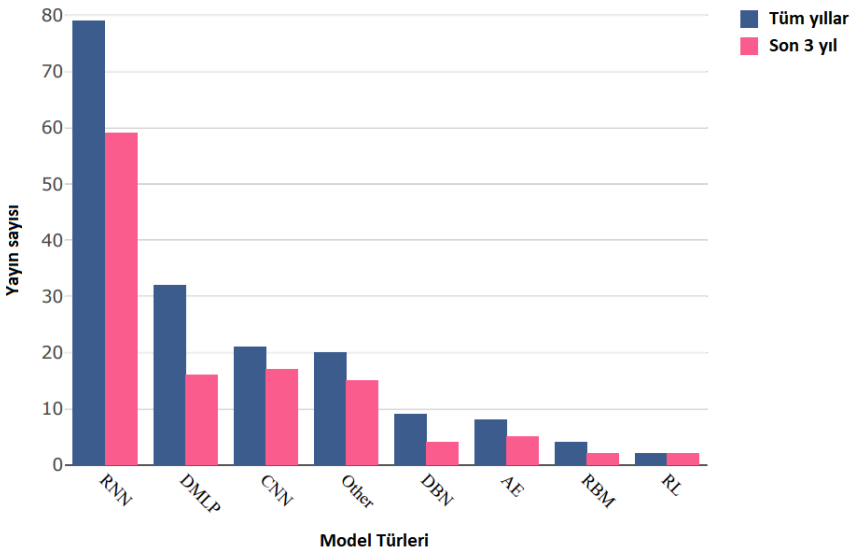
1. GİRİŞ

Günümüz teknolojisinin gelişmesi ile birlikte çok sayıda elektronik cihazdan veri akışı sağlanmaktadır. Sensörler, kameralar, sosyal medya, finans gibi çeşitli veri kaynakları, her yönüyle analiz edilemeyecek derecede fazla veri sağlamaktadır. İşte bunlar içerisinde zamana bağlı olarak değişim gösterenlere, zaman serisi verileri denmektedir.

Meteoroloji, finans, kontrol, video işleme gibi alanlarda zaman serisi analizlerinin önemi büyüktür. Yapılacak olan analiz ve buna bağlı olarak geliştirilecek olan tahminler, bahsedilen alanlar ve bahsedilmeyen fakat zaman serisi verisi ile çalışan alanlar için büyük önem taşımaktadır. Bundan dolayı, zaman serilerinden anlamlı bilgiler çıkarmak için çok sayıda çalışma yapılmaktadır ([1], [2], [3], [4], [5]).

Daha bir çok alanda olduğu gibi finans alanında da zaman serisi analiz ve tahmin yöntemleri kullanılmaktadır. Daha önceki zamanlarda, analiz ve tahminler yapılırken, Özbağlanımsal Tümleşik Hareketli Ortalama (ARIMA) gibi geleneksel yöntemler kullanılmaktaydı. Daha sonraları makine öğrenmesi modelleri de devreye girerek, Çok Katmanlı Perceptron (MLP), Karar-Destek Makinası (SVM), Karar Ağaçları (DT) gibi çeşitli mimariler ile çalışmalara devam edildi ([6], [7], [8], [9]). Fakat, son yıllarda popülaritesi devamlı artan derin öğrenme modelleri gelişerek, geleneksel yöntemlerin ve makine öğrenmesi yöntemlerinin önüne geçti. Önceki yıllara göre hesaplama kaynaklarına erişim kolaylaştığından dolayı, daha büyük ve derin modeller yapılabilir hale geldi. Bu derin modeller, öznitelik çıkarmaya daha az ihtiyaç duyan, sürekli akan veri akışı ile kendini eğiten derin öğrenme modelleri oldu. Bu nedenle yeni çözümler arayan araştırmacılar, bu derin öğrenme modellerine ağırlık vererek, son yıllarda bir çok çalışma yaptılar. Örneğin, derin öğrenme deyince ilk akla gelen modellerden olan Tamamıyle Bağlı Sinir Ağları (FCNN) ile [10], [11], [2], [12], [13], [9], [14], [15], [16], [17] gibi çalışmalar yapılmıştır. Çalışmalara bakıldığında son yılların ağırlıkta olduğu görülmektedir. Bunun dışında, zaman serisi ile çok iyi çalışmadığı düşünülse de çeşitli dönüşümlerle kullanılabilen ve görüntü tanıma, görüntü ayıklama gibi problemlerde oldukça üstün olan Konvolüsyonel Sinir Ağları (CNN) da alandaki yerini aldı ([3], [18], [19]). FCNN her ne kadar çok çalışılıyor gibi dursa da, zaman serisi verisi deyince öne çıkan ve doğası gereği bu tip verilere iyi uyum sağlayan Özyinelemeli Sinir Ağları (RNN) kadar çalışılmamaktadırlar ([10], [2], [20], [18], [16],

[21], [5], [22], [23], [24], [25], [26], [4], [27]). Bu durum sadece finans alanında hangi modellerin ağırlıklı olarak kullanıldığını gösteren grafikte görülebilir (Şekil 1.1).
Finans alanı dışında, enerji alanında da zaman serisi verileri kullanılmaktadır. Enerji alanı deyince, solar enerji, hane elektrik üretim ve tüketimi gibi konular akla gelmektedir. Bu alanlarda da çeşitli makine öğrenmesi ve derin öğrenme modelleri kullanılmıştır. Örneğin, [28] ve [29]'da, kısa zamanlı solar enerji tahmini, Karar-Destek Regressörü (SVR) ile yapılmıştır. [30]'de SVM, [31]'de FCNN yardımı ile yük tahmini çalışması yapılmıştır.



Şekil 1.1: Finans alanında kullanılan modellerin zamana göre değişimi

1.1 Problem ve Motivasyon

Son zamanlarda, teknolojinin gelişmesiyle, çok fazla alanda derin öğrenme modelleri kullanılmaya başlandı.Veri miktarının artması ve hesaplama masraflarının azalmasıyla birlikte geliştirilen derin modeller, zaman serisi verileri ile doğası gereği çok kullanılan RNN gibi modelleri de kapsamaktadır. Zaman serisi özelinde geliştirilmemiş fakat bu verilere çeşitli dönüşümler yaptırılarak kullanılan FCNN, CNN gibi modeller de çıkmıştır.
Zaman serisinden dolayı, finans ve enerji alanlarında da derin öğrenme modelleri çalışılmaya başlandı. Finansal eğilim tahmini, gelecek fiyat tahmini, elektrik üretim ve tüketim tahmini gibi problemleri ele alan çalışmalar literatürde yer almaktadır ([2], [31]). Ancak, literatürdeki eksiklikleri şu şekilde sıralayabiliriz:

- Kullanılan modellerin iç yapısının ve kapasitesinin analizinin yeterince yapılmaması,

- Çalışılan problemlerle ilgili bir sonuca ulaşılması fakat bu sonucun kullanışlı bir yere götürülmemesi,
- Veri seti ile yapılması gereken ön-işlemlere yeterince ağırlık verilmemesi,
- Alanın yeni yaklaşımlara açık olması,

Özellikle finans ve enerji alanında, derin öğrenme modelleri yeterince çalışılmamıştır. Bundan dolayı, bu alanlarda tahmin yapabilen, optimizasyon yapabilen sistemler amaçlanmıştır. Bu sistemler çalışılırken, ön-işlem aşamalarına özen göstermek ve modellerin iç yapılarını anlamaya çalışmak hedeflenmiştir. Bu amaçlardan dolayı, bahsedilen alanlarda çalışılmış CNN ve Long Short-term Memory (LSTM) modeli geliştirilmiştir.

1.2 Tezin Katkıları

Problem ve Motivasyon bölümünde bahsedilen eksiklikler göz önüne alındığında, finans ve enerji alanında tahmin yapabilen, optimizasyon yapabilen, uygun ön-işlem aşamalarından geçirilmiş sistemlerin tasarlanması amaçlanmıştır. Tez kapsamında bu konulara ağırlık veren çalışmalara yer verilmiştir.
Tez kapsamında, çok kullanılan fakat birden çeşitli kullanım şekli olduğu için karmaşıklaşan LSTM modelinin iç yapısının anlaşılması hedeflenmektedir. Hanenin elektrik üretim-tüketim tahmini sayesinde yapılan batarya optimizasyonu ile tahminlerin bir uygulamaya dönüştürülmesi hedeflenmektedir. Finans zaman serisi verilerine yapılan CNN ile yaklaşımın yenilikçi olacağı öngörülmektedir. Bahsedilen ve incelenecek olan yöntem başlıklarını şu şekilde sıralayabiliriz:

- Durum denetlemeli ve denetlemesiz LSTM karşılaştırması
- Sıradan problemler ile LSTM modelinin iç yapısı
- Hane içi batarya optimizasyonu için elektrik üretim ve tüketim tahminleri yapan LSTM
- Teknik indikatörler ile zenginleştirilen finans verisine dönüşümler uygulanarak elde edilen veri seti ile eğitilen CNN

Önerilen yöntemler ile, modellerin iç yapısına bakıldı. Durum bilgisini tutmanın LSTM'i nasıl değiştirdiği incelendi. Derin öğrenme modelleri sayesinde yapılan batarya optimizasyonu incelendi. Veri seti üzerinde yapılan dönüşümler ile farklı modellerin de kullanılabileceği gösterildi.

1.3 Önerilen Çözüm

Tez kapsamında, Problem ve Motivasyon bölümünde bahsedilen literatürdeki eksiklikler ve Tezin Katkıları bölümünde bahsedilen katkılar göz önüne alınarak, önce model çalışmaları yapılmış, sonrasında tahmin yapan bazı yöntemler geliştirilmiştir.
İlk çalışma olarak, durum denetlemeli ve denetlemesiz LSTM karşılaştırması yapılmıştır. Problemlere nasıl uygulanmaları gerektiği üzerinde durulmuştur. Veri setindeki dönüşümlerin önemi vurgulanmıştır.
İkinci çalışma olarak, incelenen LSTM modelinin iç yapısına girebilmek için 2 basit problem çözülmüştür. Bu problemlerden ilkiyle uzun vadeli hafızası incelenmiş, ikincisiyle periyodik bir fonksiyon yakınsanmıştır.
Üçüncü çalışma olarak, hane içi batarya optimizasyonu yapabilmek için solar enerji üretiminin ve hane elektrik tüketiminin tahminini yapan LSTM çalışması incelenmiştir. Bu tahminler sonrası, batarya optimizasyonu için bir optimizasyon sistemi geliştirilmiştir.
Dördüncü çalışma olarak, finans zaman serisi verilerine çeşitli dönüşümler uygulanarak eğitilen CNN incelendi. Bu çalışmada, farklı modellerin, gerekli veri seti dönüşümleri yapılırsa, uygun araç oldukları vurgulanmıştır.

1.4 Tez Taslağı

Tez kapsamında yapılan çalışmalar şu şekilde sıralanmıştır: Bölüm 2'de zaman serisi verileri analizinde kullanılan yöntemler temelden anlatılmaktadır. Ön-işlem aşamalarına, kullanılan derin öğrenme modellerine yer verilmiştir. Bölüm 3'de, çalışılacak olan durum denetlemeli ve denetlemesiz LSTM modellerini anlatılmaktadır. Ayrıca, basit ama temelden zor problemlerle modeller incelenecektir. Sonrasında, hane içi batarya optimizasyonu problemi ve kullanılan LSTM modeli incelenecektir. Son olarak, dönüşümlerden geçirilmiş finans veri seti ile CNN modeli anlatılacaktır. Bölüm 4'de, bölüm 3'de incelenen yöntemlerin sonuçlarına yer verilecektir. Bölüm 5'te, yapılan çalışmalar özetlenmekte ve gelecekte yapılabilecek çalışmalar hakkında fikirler verilmektedir.

2. ÖN BİLGİ

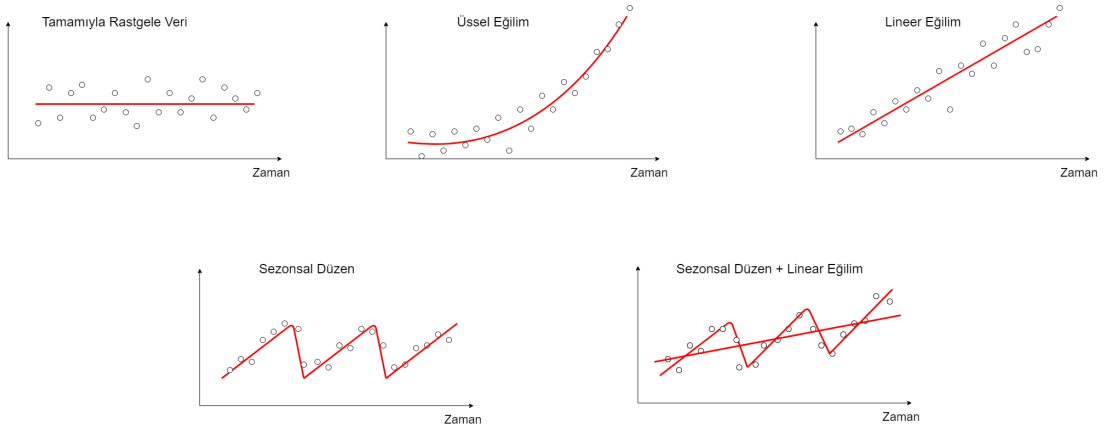
2.1 Zaman Serisi Verileri

Zaman serisi şeklindeki veriler, her alanda karşımıza çıkmaktadır. Finans verisi, sensör verileri, titreşim verileri, hava durumu verileri gibi farklı problemlem verileri zamana bağlı olarak değişim göstermektedirler. Zamana bağlı değişim gösteren verilere, zaman serisi verileri denilmektedir (Şekil 2.1).

Şekil 2.1: Zaman serisi örneği

Zaman serisi verileri, Şekil 2.2'da görüldüğü gibi, farklı biçimlerde olabilirler. Soldan sağa:

1. hiç bir düzenin görülmediği tamamiyle rastsal düzene sahip,
2. zamanla üstsel bir şekilde artan düzene sahip,
3. zamanla lineer bir şekilde artan düzene sahip,
4. sezonsallık düzenine sahip,
5. linear ve sezonsallık karışımı olan düzene sahip zaman serileri şeklinde sıralanabilirler.



Şekil 2.2: Genel zaman serisi düzenleri

Bir zaman serisi, kendisini oluşturan eğilim, sezonsallık, periyodiklik ve artık olarak 4 parçada ifade edilebilir. Bu parçalar birleşerek zaman serisini oluştururlar (2.1).

$$U_t = T_t + S_t + C_t + R_t \tag{2.1}$$

burada;

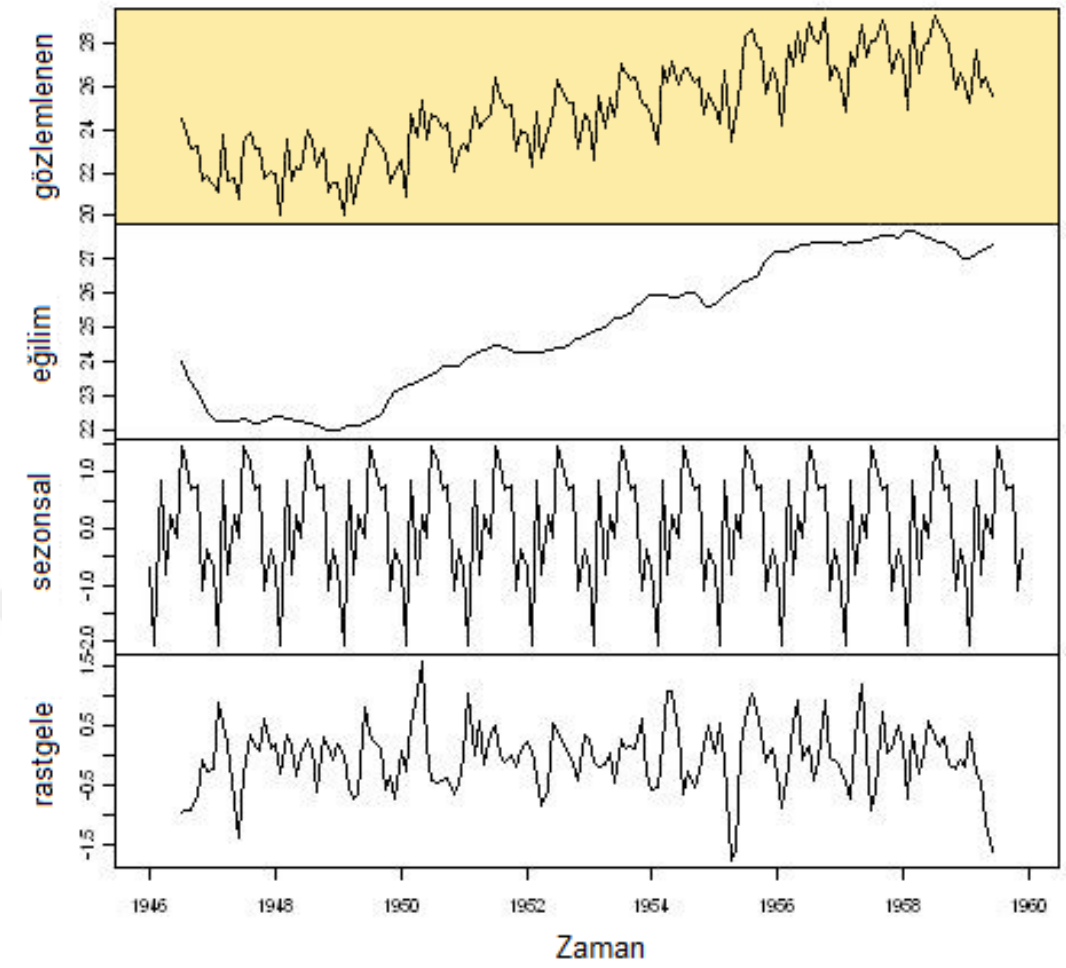
T_t: eğilim,

S_t: sezonsallık,

C_t: periyodik,

R_t: artık parçalarını temsil etmektedir (Şekil 2.3).

Eğilim bileşeni, uzun vadedeki yükseliş veya alçalış eğimini ifade eder. Sezonsallık bileşeni, belirli zamanda sürekli tekrar eden kısımı ifade eder. Hava durumu verisindeki, kışın soğuk, yazın sıcak olması durumu, bu bileşene örnek verilebilir. Periyodiklik bileşeni, sezonsallık bileşenine göre daha uzun vadedeki periyodik kısımla ilgilenir. Artık bileşen ise, diğer bileşenler sinyalden çıkarıldığında geriye kalan artık kısmı oluşturur. Rastsal gürültü olarak ifade edilir ve herhangi bir şekilde tahmin edilemez.



Şekil 2.3: Zaman serisi bileşenleri

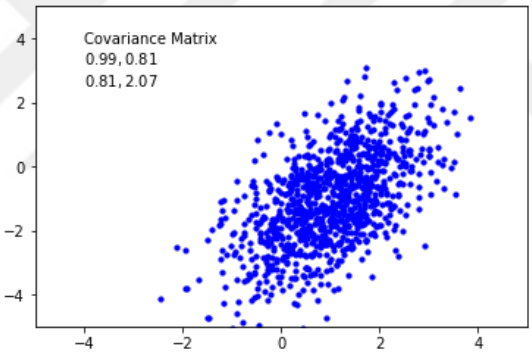
2.2 Ön-işleme

Günümüzde kullanılan makine öğrenmesi ya da derin öğrenme modelleri her ne kadar çok sayıda problemde insan seviyesini aşmış olsalar da, ön işlemden geçirilmemiş olan veriler eğitimi düsensiz hale getirip çözülmeye çalışılan optimizasyon probleminin lokal-minimumda kalmasına sebep olabilirler [32] [33]. Veri setindeki özniteliklerin, istatistiksel olarak bağımsız olmaları, (öznitelik uzayının eksenleri bağımsız öznitelikler ile ifade edilebileceği için) modellerin daha iyi bir öğrenme süreci geçirmesini sağlar. Ayrıca, bağımsız öznitelikler, olasılıksal modellerde, bileşik olasılıkların hesaplanmasını sadece çarpma işlemine indirgediği için karmaşık bir birleşik olasılık hesabına ihtiyaç duyulmamasını da sağlar. Bu bölümde, basitten karmaşığa doğru bazı ön-işleme yöntemleri anlatılacaktır.

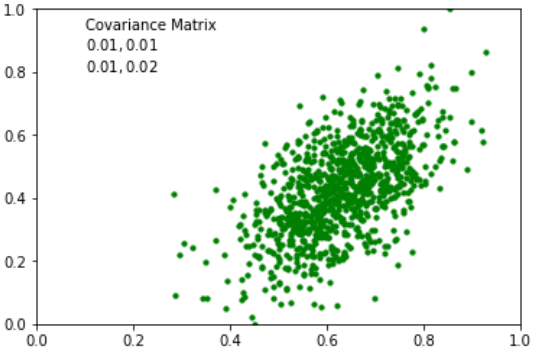
2.2.1 Öznitelik ölçeklendirme

Oznitelik ölçeklendirme, en basit ön-işleme yöntemlerinden biridir. Yöntem sayesinde, öznitelik uzayındaki her bir boyut $0-1$ arasına sıkıştırılır. Bunu, denklem (2.2)'deki işlemi yaparak, yani her bir boyuttan minimumunu çıkarıp, maksimumu ile minimumu arasındaki farka bölerek hesaplar. Örneğin; tek boyutlu $X = [1,2,3]$ şeklinde bir öznitelik olsun. X verisi üzerisinde ölçeklendirme yapıldığında, yeni $\hat{X} = [0, 0.5, 1]$ vektörü elde edilir. Yeni vektörde görüldüğü gibi veri $0-1$ arasına sıkışmış durumdadır ve tam olarak, $\hat{X}_{\text{min}} = 0$ ve $\hat{X}_{\text{max}} = 1$ olacak şekilde yeni bir vektör elde edilmiştir. Bu durum 1000 elemanlı rastgele üretilen başka bir vektör üzerinden grafiksel olarak gösterilecek olursa (Şekil 2.4), işlem sonrasında, Şekil 2.5'de gösterilen dönüştürülmüş $\hat{X}$ uzayı bulunur. Fakat öznitelik ölçeklendirmesiyle yapılan ön-işleme, eğer veride uç değerler var ise, uç değerler dışındaki değerleri fazla küçültüleceği için probleme sebepe olur.

$$\frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.2}$$



Şekil 2.4: Veriler

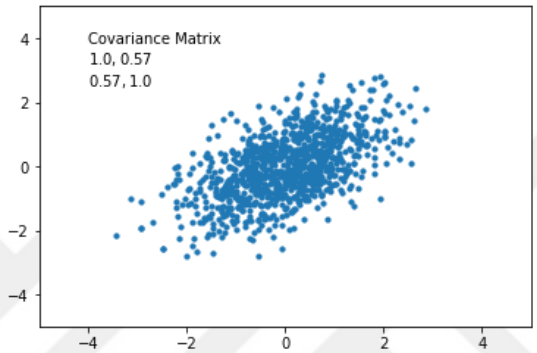


Şekil 2.5: Öznitelik ölçeklendirme işlemi

2.2.2 Standardizasyon

Standardizasyon, öznitelik uzayındaki her bir boyutu $\mu = 0$ ve $\sigma = 1$ olacak şekilde sıkıştırır (2.3). Öznitelik ölçeklendirme olduğu gibi belirli bir aralık arasına sıkıştırma söz konusu değildir. Örnek olarak, 1000 elemanlı rastgele oluşturulan X matrisi, standardizasyon işleminden geçirildikten sonra Şekil 2.6'de görüldüğü hale gelir.

$$\frac{X - \mu}{\sigma} \tag{2.3}$$



Şekil 2.6: Standardizasyon işlemi

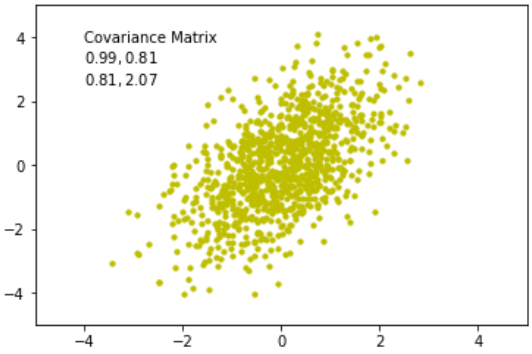
2.2.3 Veri ağartma

Veri ağartma, daha önce bahsedilen ön-işlem yöntemlerine göre biraz daha kompleks bir yapıda olup, dönüştürülen veri setinin birim kovaryans matrisine sahip olmasını sağlar. Birim kovaryans matrisi, tüm boyutlarda istatistiksel bağımsızlığı, yani tüm boyutlardaki varyansın 1'e eşit olmasını ifade eder. Bu bağımsızlık daha önce bahsedilen şekilde bileşik olasılık hesabını kolaylaştırır. Ayrıca, dönüştürülmüş olan veri setindeki her bir özniteliğe de eşit önem verilmiş olur. Bu yöntem için sırasıyla:

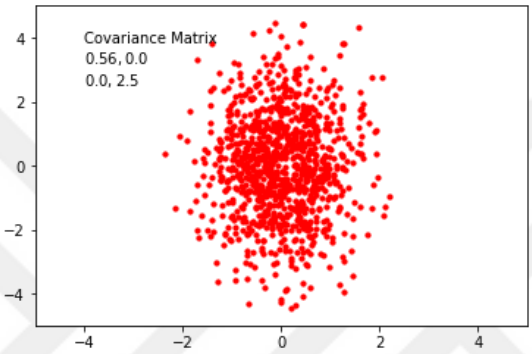
1. $\mu = 0$ işlemi (Şekil 2.7),
2. Dekorelasyon işlemi (Şekil 2.8),
3. Yeniden ölçeklendirme işlemi (Şekil 2.9),

uygulanır.

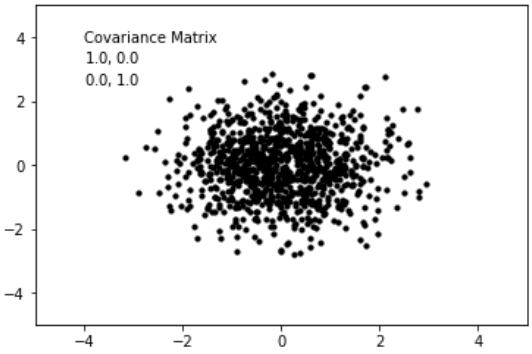
$\mu = 0$ işlemi veriyi merkezde toplar. Dekorelasyon işlemi diyagonal bir kovaryans matrisi oluşturduğu için boyutlar arasındaki korelasyondan kurtulmamızı sağlayan bir dönüşüm yapar. Yeniden ölçeklendirme işlemi ise, kısa olan boyutu uzaltarak, uzun olan boyutu kısaltarak, diyagonal kovaryans matrisinin 1 olmasını sağlar.



Şekil 2.7: Merkeze çekme işlemi



Şekil 2.8: Dekorelasyon işlemi



Şekil 2.9: Ağartma işlemi

2.2.4 Durağanlık

Durağanlık, zaman serinin istatistiksel özelliklerinin, yani ortalama, varyans gibi özelliklerinin zaman bağlı olarak değişmemesi durumudur. Kullanılacak tahmin modelleri, bu istatistiksel özelliklere göre kendilerini adapte edecekleri için, *durağan* olmayan zaman serileri için çeşitli dönüşümler yapılmak zorundadır. Bu dönüşümler yapıldıktan sonraki veri, eğitim esnasında kullanılabilecek ve yeni yapılacak olan tahminler de

uygulanan dönüşümlerin tersi ile gerçek tahminlere çevrilebilecek bir sistem gerekliliği bulunmaktadır. Zamanla değişen istatistiksel özellikleri sabitlemek için, verideki **eğilim** hesaplanıp, verinin kendisinden çıkarılabilir ya da $x^{t+1} - x^t$ şeklinde t ekseni boyunca **çıkarma** işlemi yapılabilir. Örnek olarak çıkarma işlemi yapılmış olsun: yeni oluşacak olan zaman serisi tamamiyle rastgele bir şekilde artış azalış yapıyorsa bu seriye **rastgele yürüyüş** denir ve tahmini imkansızdır. Bu durumdaki seri ile eğitilen modelin verebileceği en iyi çıktı bu serinin ortalamasıdır. Fakat, bu şekilde rastgele olmayan bir seri elde edildiğinde ve istatistiksel özelliklerinin de sabitlendiği düşünüldüğünde, tahmin modelleri ile eğitime uygun bir veri elde edilmiş olur.

2.3 Metrikler

2.3.1 Ortalama kare hatası

Regresyon problemlerinde, hata fonksiyonu olarak kullanılmasının yanı sıra metrik olarak da kullanılan ortalama kare hatası (MSE) fonksiyonu (2.4), artıkların karelerinin toplamının ortalaması ile ifade edilir. Diğer bir deyişle, modelin tahmini ile gerçek değer arasındaki öklid mesafesinin karesidir. Aynı zamanda, $L^2 - norm$ adıyla da tanınır.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.4}$$

2.3.2 Ortalama mutlak hata

Regresyon problemlerinde, ortalama mutlak hata (MAE)'de gibi hem hata fonksiyonu hem de metrik olarak kullanılabilen MAE (2.5), artıkların mutlak değerlerinin ortalaması ile ifade edilir. Manhattan mesafesi ya da $L^1 - norm$ olarak da tanınır.

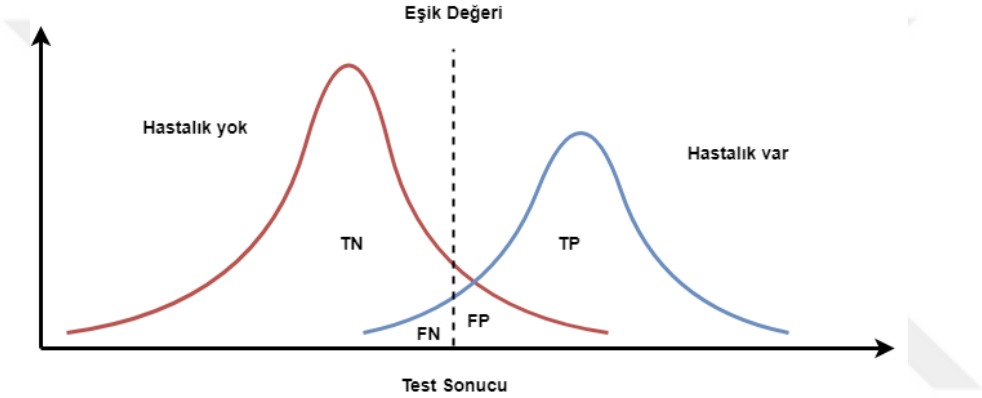
$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.5}$$

2.3.3 Karmaşıklık matrisi

Karmaşıklık matrisi, satırları tahmin sınıflarından ($\hat{y}$), sütunları gerçek sınıflardan (y) oluşan bir matristir (Şekil 2.10). Bu çizelge üzerinden, doğruluk, doğru pozitif oranı (TPR), yanlış pozitif oranı (FPR), yanlış negatif oranı (FNR), doğru negatif oranı (TNR) gibi değerler hesaplanabilir. Sınıf dağılımının eşit olmadığı durumlarda büyük resmi görmek için kullanılır.

n=85	Tahmin: HAYIR	Tahmin: EVET	
Gerçek: HAYIR	TN=30	FP=10	40
Gerçek: EVET	FN=5	TP=40	45
	35	50	

Şekil 2.10: Karmaşıklık matrisi



Şekil 2.11: Sınıf dağılımlarına göre metrik ifadeleri

burada:

- $(y = 1)\,\&\,(tahmin = 1)$ olduğu durum doğru pozitif (TP),
- $(y = 0)\,\&\,(tahmin = 1)$ olduğu durum yanlış pozitif (FP),
- $(y = 1)\,\&\,(tahmin = 0)$ olduğu durum yanlış negatif (FN),
- $(y = 0)\,\&\,(tahmin = 0)$ olduğu durum doğru negatif (TN) temsil etmektedir.

2.3.4 Doğruluk

Doğruluk, sınıflandırma problemlerinde oldukça kullanılan bir mektriktir. Modelin ne oranla doğru tahmin yaptığını ifade eder. Fakat bu oranı ifade ederken, sınıf dağılımındaki dengesizliği dikkate almadığı için, bu durumlarda kullanılması uygun değildir. Örnek olarak, hırsız alarmının başarısını ölçmek için doğruluk metriği kullanılsın ve %99 oranında hırsızın olmadığı varsayılsın: Model tüm durumlar için "alarm çalmasın" der ise %99 başarı elde etmiş olacaktır. Ama bahsedilen olayda asıl değerli olan, hırsız geldiğinde alarmı çaldırmaktır. Bu durumların hepsini kaçıracağı için aslında

oldukça başarısız bir modeldir denilebilir. Bu tip bir sınıf dağılımının dengesiz olduğu durumlarda, modelin başarını daha iyi ölçmek için, kesinlik (precision), hassasiyet (recall), f1-skoru gibi metrikler kullanılabilir. Bunlar bölümün ilerleyen kısımlarında incelenecektir.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.6}$$

2.3.5 Kesinlik

Kesinlik (2.7), sınıflandırma problemlerinde, doğruluğun veri setindeki sınıf düzensizliğinden dolayı kullanılamayacağı durumlar için kullanılan, pozitif sınıf olarak tahmin edilen örneklerin kaç tanesinin gerçekten pozitif olduğunu anlamaya yarayan bir metriktir. FP'lerin masrafının yüksek olduğu durumlarda kullanılır. Örneğin, hastanın hastaneden taburcu edilip edilmeyeceğine karar verecek bir model olsun. Modelin, yanlışlıkla hasta taburcu olabilir raporu vermesi, hastanın hayatını ciddi riske sokabilir. Bu örnek kesinlik metriğinin önemini göstermektedir.

$$Kesinlik = \frac{TP}{TP+FP} \tag{2.7}$$

2.3.6 Hassasiyet

Hassasiyet, kesinlik metriğinde olduğu gibi, sınıflandırma problemlerinde veri setinde sınıf düzensizliği olduğu durumlarda kullanılır. Farkı ise, FP'lerin yerine FN'lerin masrafının yüksek olduğu durumlarda kullanılmasıdır. Hastanın kansere yakalanıp yakalanmadığını tahmin eden bir model örnek olarak ele alınsın: *kanser olan bir hastaya kanser değil demek, geri dönülemeyecek şekilde hayati riskler taşır*. Bu problem için hassasiyet metriğinin kullanılması bir çok hayatı kurtaracak güce sahiptir.

$$Hassasiyet = \frac{TP}{TP+FN} \tag{2.8}$$

2.3.7 F1-Skoru

F1-skoru, hassasiyet ve kesinliğin karışımından oluşan, FP ve FN oranlarının ikisinin birden masraflı olduğu problemlerde kullanılacak bir mektriktir. Denklem (2.9)'te hassasiyet ve kesinliğin harmonik ortalamasından oluştuğu görülmektedir. $F1 = 1$ durumu mükemmel, $F1 = 0$ durumu tamamiyle kusurlu anlamını taşır.

$$F1 - skoru = 2 \times \frac{kesinlik \times hassasiyet}{kesinlik + hassasiyet} \tag{2.9}$$

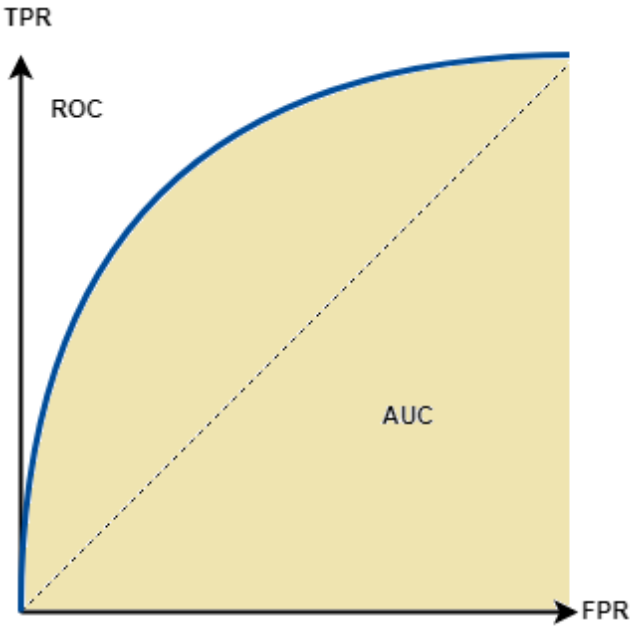
2.3.8 AUC-ROC eğrisi

Alıcı İşletim Karakteristiği (ROC) eğrisi, x ekseni FPR(2.11), y ekseni TPR(2.10) olan ve sınıf ayrımında kullanılan farklı eşik değerleri için hesaplanmış bir eğridir. ROC eğrisi altında kalan alan (AUC) ise, sınıfların birbirinden ne denli ayrıldığını göstermeye yarayan bir metriktir(Şekil 2.12). Sınıflandırma problemlerinde kullanılırlar.

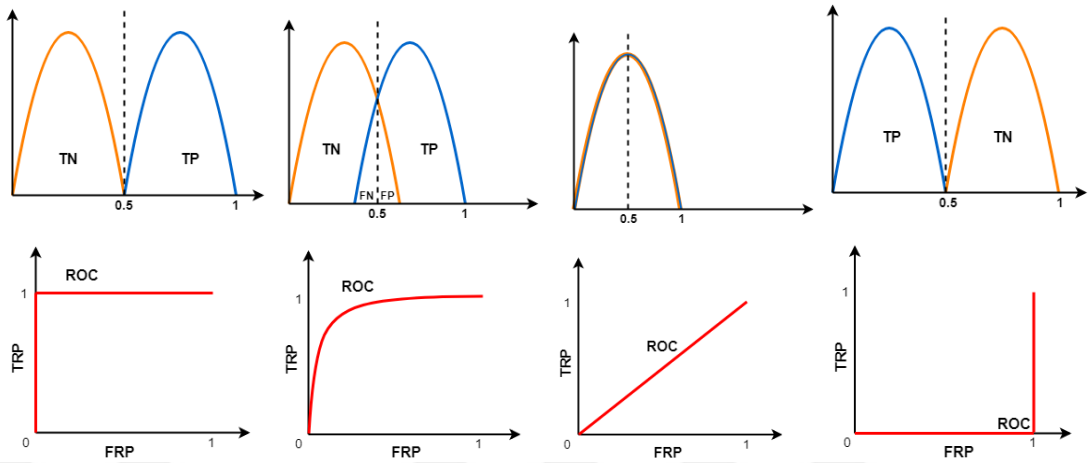
AUC skorunun 1 olması, sınıfların tamamiyle ayrıştırıldığını, yani tüm 1'lerin 1, tüm 0'ların 0 olarak tahmin edildiğini gösterir. 0 olması, aynı şekilde sınıfların tamamiyle ayrıştırıldığını fakat tüm 1lerin 0, tüm 0ların 1 olarak tahmin edildiğinin göstergesidir. 0.5 olması ise en kötü olan durumdur ve modelin hiç bir sınıf ayrımı yapamadığının göstergesidir (Şekil 2.13) .

$$TPR = \frac{TP}{TP+FN} = hassasiyet \tag{2.10}$$

$$FPR = 1 - \frac{TN}{TN+FP} \tag{2.11}$$



Şekil 2.12: ROC eğrisi ve AUC



(a) AUC=1 (b) 0.5<AUC<1 (c) AUC=0.5 (d) AUC=0

Şekil 2.13: Dağılımların ayrıştırılmasına göre AUC skorları

2.4 Hata Fonksiyonları

Hata fonksiyonları yapay sinir ağlarınında çok önemli bir role rahiptir. Modellerin verdiği çıktılar ile gerçek değerleri kıyaslayarak, gerçek doğruya ne kadar yakın olunduğunu ifade etmekte kullanılırlar. Negatif olmayan değerlere sahip olan hata fonksiyonları, sıfıra yaklaştıkça modelin tahmin kapasitesi artmaktadır. Hata fonksiyonları, risk kısmı ve regularizasyon kısmı olarak iki parçada yazılabilirler. Denklem (2.12)'de iki parçadan oluşan hata fonksiyonu, genel haliyle görülmektedir.

$$\begin{aligned} W^* &= \underset{W}{\operatorname{argmin}} \mathscr{L}(W) + \lambda \cdot \Phi(W) \\ &= \underset{W}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L\left(y^i, f(W^T x^i)\right) + \lambda \cdot \Phi(W) \end{aligned} \tag{2.12}$$

burada $\Phi(W)$ regularizasyon kısmını, W modelin ağırlıklarını, $f(\cdot)$ aktivasyon fonksiyonunu, $\mathbf{x}^i = \{x_1^i, x_2^i, \ldots, x_m^i\} \in \mathbb{R}^m$ ise eğitim esnasında kullanılan örnekleri temsil etmektedir. Bu kısımda regularizasyon kısmına girilmeyeceği için denklem sadece hata fonksiyonuna indirgenerek anlatılacaktır:

$$net^i = W^T x^i \tag{2.13}$$

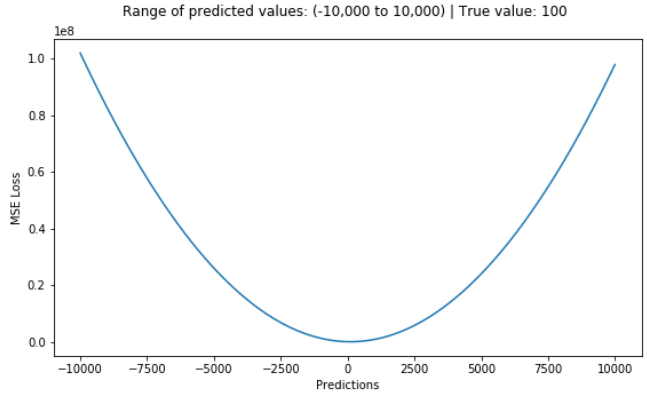
$$\mathscr{L}(W) = \frac{1}{n} \sum_{i=1}^{n} L\left(y^i, f(net^i)\right) \tag{2.14}$$

2.4.1 Ortalama kare hatası

MSE, regresyon problemlerinde en yaygın kullanılan hata fonksiyonudur. Kulanılan örneklerin, çizilen regresyon çizgisine uzaklıklarının karelerinin toplamıyla hesaplanır. En basit haliyle denklem (2.15)'da olduğu gibidir. Şekil 2.14'de MSE hata fonksiyonun $f(x) = x^2$ grafiği oluşturduğunu, yani tahmin gerçek değerden uzaklaştıkça, hatanın *artık* kısmın karesiyle orantılı bir şekilde arttığı görülmektedir.

$$\mathscr{L} = \frac{1}{n}\sum_{i=1}^{n}\left(y^i - f(net^i)\right)^2 \tag{2.15}$$

burada $(y^i - f(net^i))$ artık (residual) olarak adlandırılır ve bu kısmın karesi minimize edilmeye çalışılır.



Şekil 2.14: MSE fonksiyonunun oluşturduğu hata yüzeyi

Yapay sinir ağları W ağırlıklarını güncelleyip, hatayı minimize etmeye çalışırken gradyan iniş algoritmasını kullandıklarından, MSE'nin türevinin alınması gerekir. Denklem (2.16)'de MSE hata fonksiyonun, modelin ağırlıklarına (W) göre türevinin nasıl çıkarıldığı gösterilmiştir. Son adımdan anlaşılacağı üzere, bu türevin, $(artik) \times \frac{\partial(aktivasyonfonksiyonu)}{\partial W} \times (ornek)$'in ortalaması şeklinde olduğu görülür ve W ağırlıkları güncellenirken kullanılır.

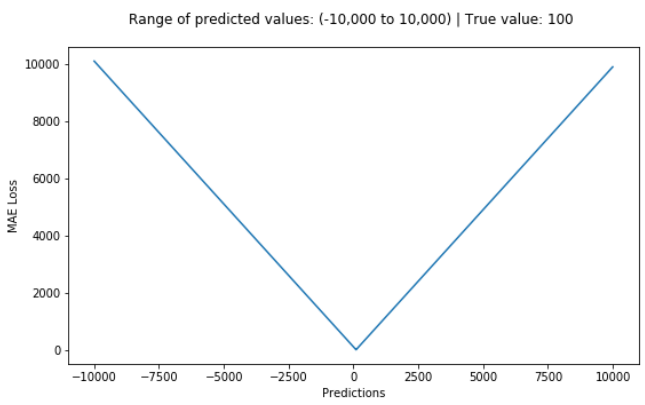
$$\begin{aligned}
\frac{\partial \mathscr{L}}{\partial w} &= \frac{2}{n}\sum_{i=1}^{n}\left(y^i - f(net^i)\right)\left(-\frac{\partial f(net^i)}{\partial w}\right) \\
&= \frac{2}{n}\sum_{i=1}^{n}\left(y^i - f(net^i)\right)\left(-\frac{\partial f(net^i)}{\partial net^i}\cdot\frac{\partial net^i}{\partial w}\right) \\
&= \frac{2}{n}\sum_{i=1}^{n}\left(y^i - f(net^i)\right)\left(-\frac{\partial f(net^i)}{\partial net^i}x^i\right)
\end{aligned} \tag{2.16}$$

MSE yapısı gereği uç değerlere karşı hassastır. Uç değerlerden hesaplanan artığın karesi çok büyük olacağından toplamı fazlasıyla etkilemektedir. Diğer bir deyişle,

kullanılan eğitim örneklerindeki varyans yüksekse MSE fonksiyonu bu varyanstan kötü etkilenmektedir. Sezgisel olarak MSE hata fonksiyonu örnek ile ele alınıyor olsun ve tüm gerçek değerler için yalnızca tek bir sayı tahmini yapmamız gereksin: MSE ile minimum hata sonucunu verecek sayı, tüm gerçek değerlerin ortalaması olacaktır. Bu hesaplanan ortalama, uç değerlerden fazlasıyla etkilenir. Bu nedenle bu gibi durumlarda daha az etkilen, bir başka regresyon hata fonksiyonu olan MAE kullanılabilir. Bir sonraki kısımda MAE hata fonksiyonu anlatılacaktır.

2.4.2 Ortalama mutlak hata

MAE'de MSE'de olduğu gibi regresyon problemlerinde çokça kullanılan diğer bir hata fonksiyonudur. MAE, tahmin ile gerçek değerin mutlak farkını hesaplayarak oluşturulan bir hata fonksiyonudur. MAE fonksiyonunun oluşturduğu hata grafiği Şekil 2.15'de, denklemi ise (2.17)'de görülmektedir.



Şekil 2.15: MAE fonksiyonunun oluşturduğu hata yüzeyi

$$\mathscr{L} = \frac{1}{n}\sum_{i=1}^{n}\left|y^i - f(net^i)\right| \tag{2.17}$$

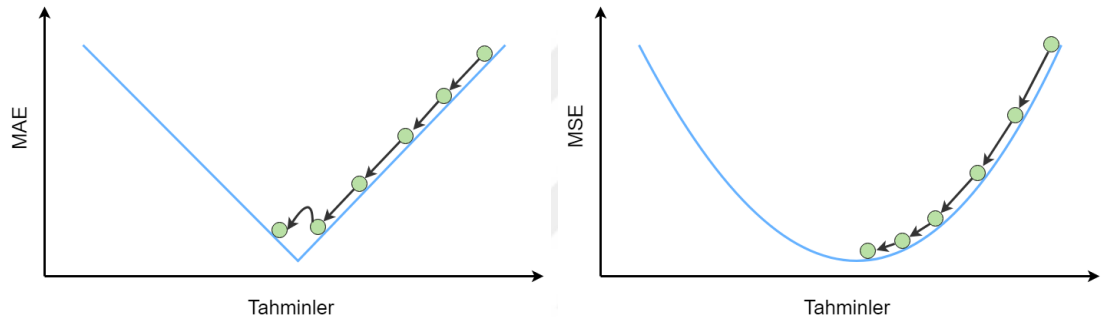
W ağırlık güncellemelerinde kullanmak için türevi denklem 2.18'de adım adım görülmektedir.

$$\begin{aligned}\frac{\partial \mathscr{L}}{\partial w} &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2\sqrt{(y^i - f(net^i))^2}}\left(-\frac{\partial f(net^i)}{\partial w}\right)\\ &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2\sqrt{(y^i - f(net^i))^2}}\left(-\frac{\partial f(net^i)}{\partial net^i}\cdot\frac{\partial net^i}{\partial w}\right)\\ &= \frac{1}{2n}\sum_{i=1}^{n}\frac{1}{|y^i - f(net^i)|}\left(-\frac{\partial f(net^i)}{\partial net^i}x^i\right)\end{aligned} \tag{2.18}$$

MAE fonksiyonu uç değerlere karşı MSE'nin olduğu kadar hassas değildir. Yani eğitim

verisindeki örneklerin varyansının fazla olduğu durumlarda MSE'ye göre daha iyi bir seçim olacaktır. MSE'nin ortalama değer çıktısı oluşturduğu örnek ele alınacak olursa, MAE'nin bu örneğe çıkaracağı sonuç medyandır. Medyan, ortalama ile kıyaslandığında, uç değerlere daha az hassastır. Bundan dolayı, uç değerlere sahip eğitim verisi varsa, MAE, MSE'den daha gürbüzdür.

MAE'nin MSE'ye göre uç değerlerde olan gürbüzlüğü büyük bir avantajdır fakat sürekli MSE yerine kullanılamamasının arkasında temel bir problem vardır: optimum noktaya yaklaşırken dahi gradyanın sabit kalması. Bu dezavantaj, MAE'nin türevinin alınması gereken her durumda ortaya çıkar. Optimum noktaya yaklaşırken gradyanın sabit kalması, optimum noktanın kaçırılmasına sebebiyet verebilir. Bunu önlemek için tabiki öğrenme faktörü zamanla düşürülebilir fakat MSE, böyle bir işleme gerek duymadan gradyanları küçülterek daha gürbüz (sabit gradyan özelinde) bir öğrenme durumu oluşturabilir. Bu dezavantaj, görsel olarak Şekil 2.16'de gösterilmektedir.



(a) MAE (b) MSE

Şekil 2.16: MAE-MSE gradyan karşılaştırması

2.4.3 Cross-Entropy hatası

Capraz-entropi (CE) hata fonksiyonu, sınıflandırma problemlerinde en çok kullanılan hata fonksiyonudur. Gerçek değerlerin oluşturduğu olasılıksal dağılım ile tahminlerin oluşturduğu olasılıksal dağılımın arasındaki uzaklığı hesaplamak için kullanılır. CE hatası büyükse, iki dağılım birbirine uzak, küçükse yakındır.

Gerçek sınıf ve model çıktı vektörü sırasıyla $y = [0,1,0]$ ve $net^{(i)} = W^T x^{(i)} = [1,4,2]$ şeklinde olsun. Burada 2 numaralı sınıf doğru sınıf değeriyken, modelin çıktısı da aynı şekilde 2 numaralı sınıf çıktısında 4 değerini taşıyarak sınıfı doğru tahmin etmiş görünmektedir. Tahmin sınıfını çıkarmak için yapılacak en basit işlem $\text{argmax}(net^{(i)})$ işlemi yaparak tahmin vektöründen bu 4 değerinin indeksini çekmektir. Fakat argmax fonksiyonu türevlenebilir olmadığı için sinir ağlarında kullanılan gradyan iniş optimizasyon algoritması ile kullanılamamaktadır. Bu nedenle hem tahmin vektöründen sınıfın indeksini hesaplayacak hem de $\text{argmax}(net^{(i)})$'ye olabildiğince yakınsayacak bir fonksiyona

ihtiyaç vardır. Basitçe şu şekilde bir fonksiyon seçilsin: $\frac{\psi^i}{\sum_j \psi^j}$. Tanımlanana yeni fonksiyon ile çıktı vektörü hesaplanacak olursa $net^{(i)} = [0.125, 0.5, 0.375]$ olarak hesaplanır. Bu yeni vektör, doğru sınıfı önceki halindeki gibi işaret ediyor olsa da, doğru sınıfa verdiği 0.5 değeri hedef vektörü olan $y_2 = 1$'e oldukça uzaktır. Bu nedenlerden dolayı, sınıflandırma problemlerinde kullanılan ve argmax'a oldukça yakınsayan bir fonksiyon olan softmax 2.19 fonksiyonu ve türevi 2.20 kullanılmaktadır. softmax ile hesaplanan çıktı vektörü $net^{(i)} = [0.04, 0.70, 0.26]$ olarak hesaplanır. softmax'ta kullanılan üstel terim sayesinde, bu yeni vektör, argmax'a daha çok (ortalama fonksiyonuna kıyasla) yakınsamıştır. Bu üstel terim negatif değerlerin de olasılık dağılımına katılmasını sağlayarak daha yumuşak bir öğrenme ortamı oluşturur. Ek olarak, e sabitinden daha büyük bir üstel terim de seçilebilir fakat hem türevi daha zor alınan bir fonksiyona sebebiyet verir hem de nümerik stabiliteyi zora sokar.

$$S^i = \frac{e^{net^i}}{\sum_k e^{net^k}} \tag{2.19}$$

$$\begin{aligned}
\frac{\partial S^i}{\partial net^j} &= \frac{\partial \left(\frac{e^{net^i}}{\sum_k e^{net^k}} \right)}{\partial net^j} \\
&= \frac{\frac{\partial e^{net^i}}{\partial net^j} \cdot \sum_k e^{net^k} - \frac{\partial \sum_k e^{net^k}}{\partial net^j} \cdot e^{net^i}}{\left(\sum_k e^{net^k} \right)^2}
\end{aligned}$$

if $i == j$: (2.20)

$$\begin{aligned}
\frac{\partial e^{net^i}}{\partial net^j} &= e^{net^i} \\
\frac{\partial S^i}{\partial net^j} &= \frac{e^{net^i} \cdot \sum_k e^{net^k} - e^{net^j} \cdot e^{net^i}}{\left(\sum_k e^{net^k} \right)^2} \\
&= \frac{e^{net^i} \left(\sum_k e^{net^k} - e^{net^j} \right)}{\left(\sum_k e^{net^k} \right) \left(\sum_k e^{net^k} \right)} \\
&= \frac{e^{net^i}}{\left(\sum_k e^{net^k} \right)} \left(1 - \frac{e^{net^j}}{\left(\sum_k e^{net^k} \right)} \right)
\end{aligned}$$

$$= S^i(1 - S^j)$$

if $i \neq j$:

$$\frac{\partial e^{net^i}}{\partial net^j} = 0$$

$$\frac{\partial S^i}{\partial net^j} = -\frac{e^{net^j} \cdot e^{net^i}}{\left(\sum_k e^{net^k}\right)^2}$$

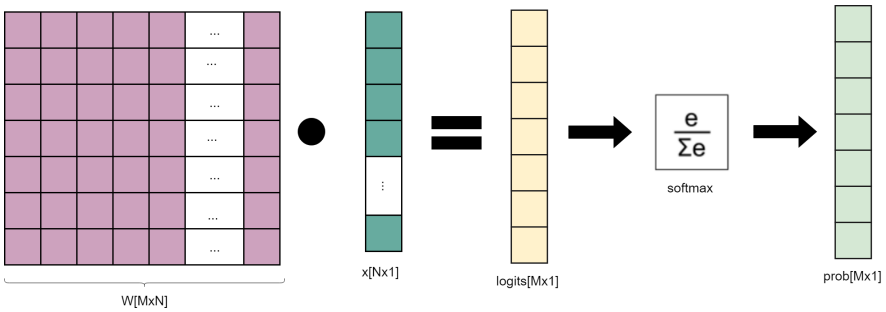
$$= -\left(\frac{e^{net^j}}{\sum_k e^{net^k}}\right)\left(\frac{e^{net^i}}{\sum_k e^{net^k}}\right)$$

$$= -S^j S^i$$

softmax ile hesaplanan çıktının olasılıksal dağılımı ve gerçek değerlerin olasılıksal dağılımı, CE hata fonksiyonu ile (2.21) ile hesaplanarak W ağırlıklarının güncellenmesinde kullanılırlar (Şekil 2.17). Bu güncellenme esnasındaki zincir kuralıyla alınan türev hesabı ise (2.22) denkleminde gösterildiği gibidir.

$$\mathscr{L} = -\sum_i y^i \log\left(S^i\right) \tag{2.21}$$

$$\begin{aligned}
\frac{\partial \mathscr{L}}{\partial net^i} &= -\sum_k y^k \frac{\partial \log\left(S^k\right)}{\partial net^i} \\
&= -\sum_k y^k \frac{\partial \log\left(S^k\right)}{\partial S^k} \frac{\partial S^k}{\partial net^i} \\
&= -\sum_k y^k \frac{1}{S^k} \frac{\partial S^k}{\partial net^i} \\
&= -y^i \frac{1}{S^i}\left(S^i(1 - S^i)\right) - \sum_{k \neq i} y^k \frac{1}{S^k}(-S^k S^i) \\
&= -y^i(1 - S^i) + \sum_{k \neq i} y^k S^i \\
&= -y^i + y^i S^i + \sum_{k \neq i} y^k S^i \\
&= S^i \underbrace{\left(y^i + \sum_{k \neq i} y^k\right)}_{=1} - y^i \\
&= S^i - y^i
\end{aligned} \tag{2.22}$$



Şekil 2.17: CE hata fonksiyonu adımları

2.5 Aktivasyon Fonksiyonları

Sinir ağlarının herhangi bir katmanındaki herhangi bir nöron $\sum_i wx^i + b$ şeklinde kendisine gelen girdileri toplar, *bias* terimi ekler ve bir çıktı verir. Aktivasyon fonksiyonu $f(\cdot)$, çıktı veren nöronun aktive olup olmayacağını belirleyen ve nöron denklemini $f(\sum_i wx^i + b)$ haline getiren bir fonksiyondur.

Aktivasyon fonksiyonu olarak step fonksiyonu (2.23) ele alınırsa; 1 çıktısı verdiğinde aktive olmuş nöron, 0 çıktısı verdiğinde deaktive olmuş nöron temsil edilebilir. Fakat çok sınıflı problemlerde birden fazla nöron aktive olduğunda karmaşıklığa sebebiyet vereceğinden kullanılmamaktadır.

$$u(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{2.23}$$

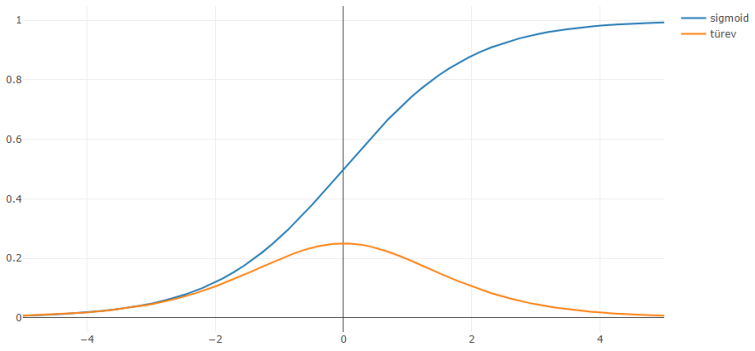
Aktivasyon fonksiyonu olarak m eğimine sahip doğru fonksiyonu (2.24) ele alınırsa; step fonksiyonun birden fazla aktive nöronda ortayan çıkan sorunu çözülmüş olur. Fakat fonksiyonun gradyan iniş algoritması ile türevi alındığı sırada, her zaman sabit olan m türev değeri, x vektöründen bağımsızdır. Bu sebeple, hep sabit olan bir gradyan sisteme dahil edilecek ve tahmindeki hataya göre hesaplanan adaptif gradyan kullanılmaz olacaktır. Ayrıca, lineer bir fonksiyon olan doğru fonksiyonu, çok katmanlı sinir ağı ile yığınlanarak kullanıldığında, yığınlanan lineer fonksiyonların birleşimi yeni bir lineer fonksiyon oluşturacaktır. Bu yeni lineer fonksiyon, sinir ağında tek bir katman ile ifade edilebilir. Bu durum çok katman kavramını ortadan kaldıracağı için çok katmanlı sinir ağlarının getirdiği tüm faydaları silip atacaktır. Ayrıca, lineer doğru denklemi $(-\infty, \infty)$ aralığında bir sonuç vereceği için sonraki katmanlar fazla büyük sayıları girdi olarak almak zorunda kalır. Bu tip nedenlerden dolayı lineer-olmayan aktivasyon fonksiyonları kullanılmamalıdır.

$$f(x) = mx \tag{2.24}$$

2.5.1 Sigmoid

Sigmoid fonksiyonu (2.25) denklemi ile ifade edilen, kendisi ve türevi Şekil 2.18'de gösterilmiş olan bir fonksiyondur. Lineer bir fonksiyon olmadığı için kombinasyonları da lineer olmayan bir fonksiyon oluşturur. Bu sayede çok katmanlı yapılar oluşturulabilir. Ayrıca, her noktasında, türevi x vektörüne bağımlı olduğundan, gradyan iniş algoritması ile kullanılmaya, yani yapay sinir ağlarının eğitimi için kullanılmaya uygun bir yapıdadır. Ayrıca çıktı sonucunu $(0,1)$ aralığına sıkıştıracağı için gradyan patlamasının önüne geçer.
Avantajlarının yanı sıra, şekil 2.18'den görüleceği gibi, $x = 0$ noktasından uzaklaştıkça, 0 ya da 1 e gittikçe daha az eğimle yakınsayan bir fonksiyondur. En büyük değeri $\frac{\partial \sigma(x)}{\partial x}|_{x=0} = 0.25$ olan, gaussian fonksiyonuna yakın bir türev fonksiyonuna sahip olmasından dolayı, 0'dan uzaklaştıkça gradyan küçülür ve geri besleme evresinde bu gradyanlar birbirleriyle çarpılacağı için sonuç olarak daha küçük gradyanlara sebep olurlar. Bu durum *"gradyanların yok olması (vanishing gradients)"* adı verilen ve öğrenme işleminin yavaşlamasıyla, hatta çok büyük ağlarda gradyan bilgisinin tamamiyle kaybolup öğrenmenin durmasıyla tanınan soruna yol açar.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.25}$$

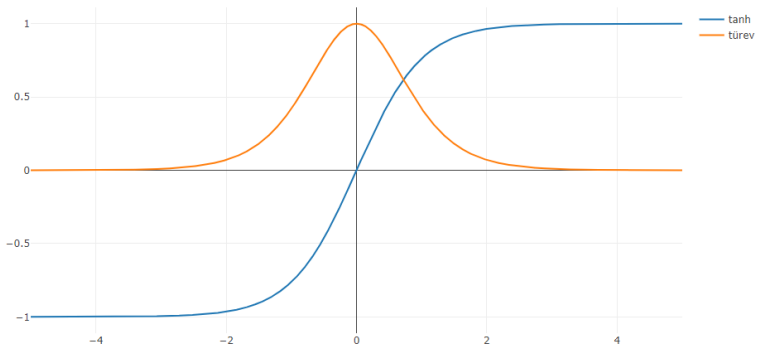


Şekil 2.18: Sigmoid

2.5.2 Tanh

tanh$(\cdot)$ fonksiyonu denklem (2.25) görüleceği gibi $\sigma(\cdot)$ fonksiyonunun ölçeklendirilip kaydırılmasıyla elde edilmiştir.Bu nedenle $\sigma(\cdot)$'in tüm avantaj ve dezavantajlarına sahiptir. Yalnızca, $\sigma(\cdot)$'de göre daha büyük bir gradyan değerine sahip olduğu için "gradyanların yok olması" problemini daha az ortaya çıkarır. Kendisi ve türevi Şekil 2.19'de gösterildiği gibidir.

$$\begin{aligned}\tanh(x) &= 2\sigma(2x) - 1 \\ &= \frac{2}{1+e^{-2x}} - 1\end{aligned} \tag{2.26}$$

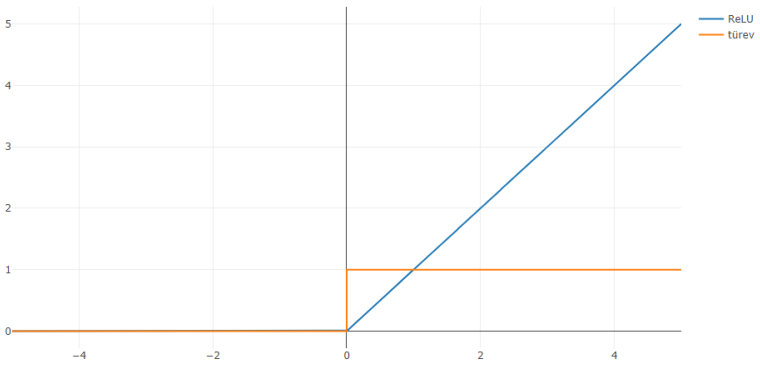


Şekil 2.19: Tanh

2.5.3 ReLU

Bir başka aktivasyon fonksiyonu olan Doğrusal Doğrultmaç Unitesi, Rectified Linear Unit (ReLU), önceden bahsedilen lineer olmayan doğru denklemine çok benzemektedir. $[0,\infty)$ aralığında bir sonuç verdiği için sorun çıkarabilen çok büyük pozitif sayılar üretebilir. Negatif sayıların hepsini 0 noktasında sıkıştırması, bazı nöronların aktif olmayıp gradyan hesabına herhangi bir katkıda bulunamamalarına ve bundan dolayı bu nöronların öğrenmeyi tamamen durdurup ölmelerine sebep olabilir. Fakat aynı zamanda yaptığı 0'a sıkıştırma işlemi, seyrek bir çıktı matrisi üretmesine sebebiyet verir. Yoğun matrisle kıyaslandığında, seyrek matrisin hesap masrafı daha düşük olduğundan daha verimli sinir ağları oluşturur. Ayrıca parçalı-lineer olan bu fonksiyon, çok iyi bir yakınsayıcıdır ve çeşitli kombinasyonlar ile her türlü fonksiyona yakınsayabilir. Kendisi ve türevi Şekil 2.20'de gösterildiği gibidir. Pozitif sayılar için türevi hep 1 olduğu için, gradyan patlamasına ya da yok olmasına sebep olmaz.

$$ReLU(x) = \max(0,x) \tag{2.27}$$

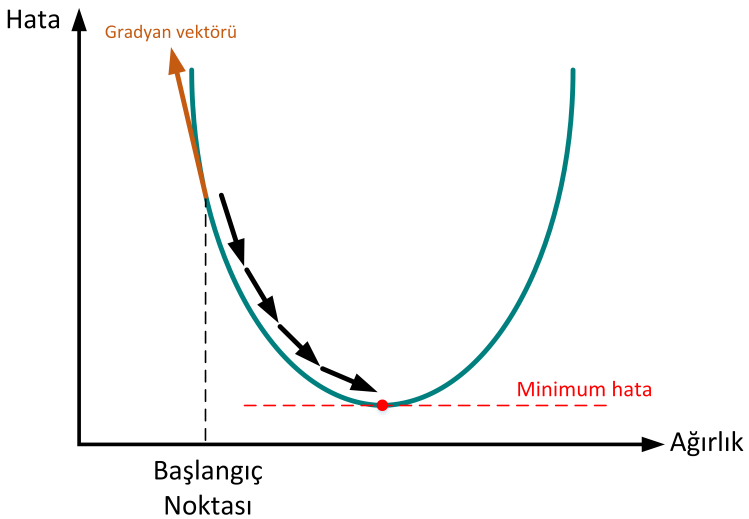


Şekil 2.20: ReLU

2.6 Gradyan İniş

Hedef fonksiyonun, model parametrelerinin W ile ifade edildiği, $L(W)$ olduğunu varsayarsak, $\nabla_W L(W)$, bu fonksiyonun gradyanı olarak ifade edilir ve denklem (2.28)'te görüldüğü gibi d boyutunda bir türev vektörüdür. Şekil 2.21'deki fonksiyonun x noktasındaki gradyanı yeşil ok ile gösterilmiş olup, m ile gösterilen optimum noktaya ulaşmak için gradyan vektörünün ters yönünde ilerlemek gerekmektedir. Gradyan vektörünün tersi yönünde ilerleyerek, W parametrelerini güncelleyen ve sonunda m noktasına ulaşan algoritmanın adı *gradyan iniş (gradient descent)* algoritmasıdır.

$$\nabla_W L(W) = \left\langle \frac{\partial L(W)}{\partial W_1}, \frac{\partial L(W)}{\partial W_2}, \ldots, \frac{\partial L(W)}{\partial W_{d-1}}, \frac{\partial L(W)}{\partial W_d} \right\rangle \tag{2.28}$$



Şekil 2.21: Gradyan iniş anlatımı

Gradyan iniş algoritmasının temelde 3 farklı çeşidi vardır:

1. **Yığın gradyan iniş**:

   En temel gradyan iniş varyasyonudur. Her W ağırlık güncellemesinde N boyutlu örnek uzayının hepsini gezer. Bu tek ve büyük güncelleme, hem çok fazla hafıza gerektirir hem de oldukça yavaştır. Ayrıca *online* öğrenmeye de olanak tanımaz. Örnek uzayında birbirine benzeyen örnekler için gereksiz bir hesaplama yapması da diğer bir dezavantajıdır. Fakat ortalamaya dahil olan örnek sayısı fazla olduğu için düşük varyansa sahip olur ve daha yumuşak bir öğrenme eğrisine sahiptir.

$$W = W - \eta \cdot \nabla_W L(W) \tag{2.29}$$

2. **Stokastik gradyan iniş**:

   Yığın gradyan inişin aksine örnek uzayındaki her örnek için tek tek gradyan hesaplar ve hemen ardından güncelleme yapar. Bundan dolayı *online* öğrenmeye olanak tanır ve yığın gradyan inişin hafıza ile ilgili yaşadığı sorunlarını ortadan kaldırır. Fakat her örnek için güncelleme yapmak varyansı oldukça arttıracağından, fazla gürültülü bir öğrenme eğrisine sahiptir. Aynı sebepte, uç değerlere karşı oldukça hassastır. Gürültülü öğrenme eğrisini çözebilmek için, öğrenme faktörünü ilerleyen iterasyonlarda düşürebilir ve yığın gradyan inişe yakınsayan bir öğrenme eğrisi elde edilebilir.

$$W = W - \eta \cdot \nabla_W L(W, X^{(i)}, y^{(i)}) \tag{2.30}$$

3. **Mini-yığın gradyan iniş**[34]:

   Yığın gradyan iniş ile stokastik gradyan iniş arasında olan bu gradyan varyasyonu, iki tarafın hem avantaj hem de dezavantajlarına sahiptir. Öğrenme faktörü ve yığın sayısı ayarlanarak her iki tarafa da çekilebilir. Günümüzde stokastik gradyan iniş (SGD) terimi bu varyasyon için kullanılmaktadır (Bu tezde de bu aşamadan sonra SGD, mini-yığın gradyan iniş algoritmasını temsil edecektir). Her ne kadar SGD diğer varyasyonlardan daha esnek bir yapıya sahip olsa da *gradyan iniş* özelinde bazı dezavantajları vardır. 1. Uygun öğrenme faktörü seçmek zor olabilir. 2. öğrenme faktörünün sürekli aynı kalması ilerleyen iterasyonlarda hata küçüldükçe probleme yol açabilir (*learning rate annealing* kullanılabilir [35], [36]). 3. Tüm boyutlarda ortak bir öğrenme faktörü uygun olmayabilir (Adaptif optimizasyon algoritmaları kullanılabilir 2.7.1, 2.7.2, 2.7.3, 2.7.4). 4. Öğrenme platolara çakılıp kalabilir (Momentum kullanılabilir 2.6.1).

$$W = W - \eta \cdot \frac{1}{m} \nabla_W \sum_{i=k}^{k+m-1} L(W, X^{(i)}, y^{(i)}) \tag{2.31}$$

2.6.1 Momentum

Optimizasyon algoritması içerisinde momentum [37] kullanmak, gradyanın sıfırlandığı durumlarda öğrenmenin devam etmesini sağlar. Gradyanın, bir boyutta yükselen diğer bir boyutta ise alçalan eğimin olduğu durumlarda sıklıkla ortaya çıkan *eyer noktalarında (saddle points)* sıfırlandığı görülür. Bu gibi plato oluşturan durumlarda t zamanındaki hesaplanan gradyanın etkisi kullanılarak, $t+1$ zamanında hesaplanan gradyan sıfıra eşit dahi olsa, öğrenme sağlanır. $\mu \Delta W_t$ momentum ifadesi, (2.30) denklemindeki gradyan güncellenme adımına eklenerek denklem (2.32) elde edilir. t ve $t+1$ zamanındaki gradyan yönleri aynı ise, o yöndeki güncellemeyi arttırır. Fakat, gradyanlar ters yönlü ise güncellemeyi azaltarak daha az salınım oluşmasını sağlar.

$$\begin{aligned} \Delta W_{t+1} &= \mu \Delta W_t - \eta \nabla_W L(W, X, y) \\ W_{t+1} &= Wt + \Delta W_{t+1} \end{aligned} \tag{2.32}$$

- $\mu \in [0,1]$: momentum faktörü
- $\eta > 0$: öğrenme faktörü
- $\nabla L(W_t)$: W_t'deki gradyan
- $L(x)$: optimize edilmesi gereken hata fonksiyonu

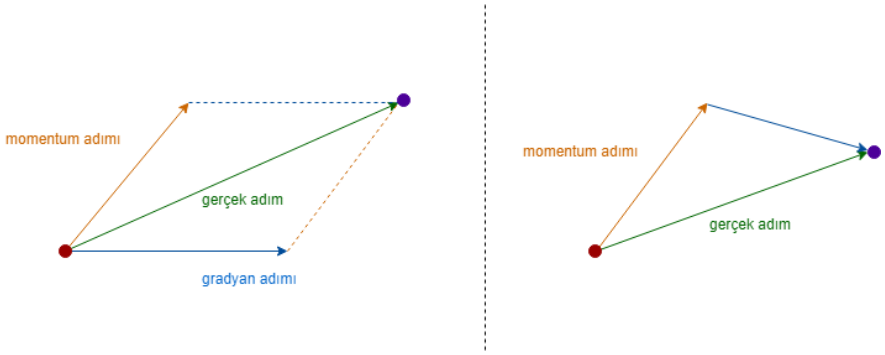
2.6.2 Nesterov hızlandırılmış momentum

Momentum güncellemesinin biraz farklı bir versiyonu olan Nesterov Hızlandırılmış Momentum (NAG) [38] popülaritesi günden güne artmaktadır. Klasik momentumdan farklı olarak, gradyanın hesaplandığı W_t noktası farklıdır. W_t noktasında gradyan almak yerine, momentumun uygulanmış hali olan $W_t + \mu \Delta W_t$ noktasında gradyan hesaplanır. Bu şekilde bir adım ilerideki konuma göre gradyan alarak, pratikte klasik momentumdan biraz daha iyi çalışır. Şekil 2.22'de klasik momentum ile nesterov hızlandırılmış momentumu arasındaki fark görülebilir.

$$\begin{aligned} \Delta W_{t+1} &= \mu \Delta W_t - \eta \nabla_W L(W_t + \mu \Delta W_t, X, y) \\ W_{t+1} &= Wt + \Delta W_{t+1} \end{aligned} \tag{2.33}$$

- $\mu \in [0,1]$: momentum faktörü
- $\eta > 0$: öğrenme faktörü

- $\nabla L(W_t + \mu \Delta W_t)$: momentum eklenmiş ağırlıkların hata fonksiyonundan geçirildikten sonraki W_t'deki gradyan
- $L(x)$: optimize edilmesi gereken hata fonksiyonu



Şekil 2.22: Nesterov Hızlandırılmış Momentum

2.7 Eniyileştiriciler

2.7.1 AdaGrad

Gradyan tabanlı bir başka algoritma olan Adaptif Gradyan Yöntemi (AdaGrad) [39], d boyutlu W için farklı bir öğrenme faktörü hesaplar ve uygular. Denkem (2.34)'te Adagrad algoritmasının matematiksel ifadesi verilmiştir. Bu denklemde, pay evrensel öğrenme faktörü nü temsil ederken, payda daha önceki gradyanların L^2 *norm*undan oluşur. Evrensel öğrenme faktörünün, eski gradyanların ortalamasına bölünmesi, her boyut için ayrı olan adaptif öğrenme faktörünü ortaya çıkarmış olur. Ayrıca büyük olan gradyanlı boyutların gradyan ortalaması büyük olacağı için daha küçük bir öğrenme faktörüne sahip olurlar, küçük gradyanlı boyutlar için de tam tersi bir durum söz konusudur. Bu yapı, katmanlarında farklı öğrenme faktörüne ihtiyaç duyan sinir ağları için öğrenmeyi hem iyileştirir hem de gürbüzleştirir. Evrensel öğrenme faktörünün seçimi hala gerekli olsa da, görece büyük bir değer seçilerek, küçültme işlemi algoritmanın kendisine bırakılabilir. Ayrıca, iterasyon ilerledikçe kumulatif L^2 *norm* artacağından *learning rate annealing* yöntemi otomatikman uygulanmış olur. Fakat paydadaki bu sürekli artış bir yerden sonra sonra güncelleme için kullanılacak olan ΔW_t'yi oldukça küçültür ve öğrenme durur. Iterasyon ilerledikçe kötüye giden bu durumu düzeltmek için bölüm 2.7.2 ve 2.7.3 da anlatılacak olan optimizasyon algoritmaları geliştirilmiştir.

$$\begin{aligned} g_t &= \nabla_W L(W_t, X, y) \\ \Delta W_t &= -\frac{\eta}{\sqrt{\sum_{t=1}^{\tau} g_\tau^2}} \\ W_{t+1} &= W_t + \Delta W_t \end{aligned} \tag{2.34}$$

2.7.2 ADADELTA: Adaptif öğrenme faktörü

Adaptif Öğrenme Faktörü Yöntemi (ADADELTA) [40], AdaGrad yönteminin dezavantajları olan; öğrenme boyunca öğrenme faktörünün düşmesi ve evrensel öğrenme faktörünün seçilme ihtiyacını ortadan kaldırmak için oluşturulmuş olan bir optimizasyon algoritmasıdır. AdaGrad'daki en başından beri hesaplanan kumulatif gradyan $l2norm$'u yerine, *low pass*ten geçirilmiş gradyan $l2norm$'unu kullanır. *Low pass filtresi* doğası gereği son iterasyonlara ağırlık verdiğinden ilk iterasyonlardaki gradyanların etkisi giderek azalır. Bu son kısma ağırlık veren filtre sayesinde sürekli büyüyen paydadan ve sonucunda öğrenmenin durma noktasına gelmesi probleminden kurtulunmuş olur. Ayrıca, hem evrensel öğrenme faktörü seçmeyi kaldırmak hem de SGD, momentum ve AdaGrad'daki pay payda birim uyuşmazlığını gidermek adına, paya ΔW^2 teriminin *low pass filtresi* ile yumuşatılmış hali koyulur. Bu pay, eski güncelleme miktarları büyük ise büyük bir *öğrenme faktörü*, küçükse küçük bir *öğrenme faktörü* ortaya çıkarır. Böylelikle minimuma yaklaşılan durumlarda *öğrenme faktörünü* küçültmesinden dolayı *learning rate annealing* etkisi yaratır.

$$\begin{aligned} g_t &= \nabla_W L(W_t, X, y) \\ E[\psi^2]_t &= \rho E[\psi^2]_{t-1} + (1-\rho)\psi_t^2 \\ \Delta W_t &= -\frac{\sqrt{E[\Delta W^2]_{t-1} + \varepsilon}}{\sqrt{E[g^2]_t + \varepsilon}} g_t \\ W_{t+1} &= W_t + \Delta W_t \end{aligned} \tag{2.35}$$

2.7.3 RMSProp

Yayınlanmayan ve ADADELTA ile aynı zamanlarda ortaya çıkan (RMSProp), ilk defa Hinton [41]'ın dersinde ortaya çıktı. İlk olarak da Graves [42] tarafından bir yayında kullanıldı. AdaGrad'ın öğrenmenin durmasına sebebiyet veren *yok olan öğrenme faktörü* sorununu çözmek için ADADELTA'nın kullandığı aynı yöntemi kullanmıştır (*low pass filter*). Fakat ADADELTA'nın aksine, evrensel öğrenme faktörü seçme ihtiyacını ortadan kaldırmamıştır.

$$\begin{aligned} g_t &= \nabla_W L(W_t, X, y) \\ E[\psi^2]_t &= \rho E[\psi^2]_{t-1} + (1-\rho)\psi_t^2 \\ \Delta W_t &= -\frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} g_t \\ W_{t+1} &= W_t + \Delta W_t \end{aligned} \tag{2.36}$$

2.7.4 ADAM

Adaptif Moment Tahmin Yöntemi (ADAM) [43]; AdaGrad, ADADELTA, RMSProp gibi adaptif öğrenme faktörü sahip, yayınlandığı günden itibaren popularitesi giderek artan, RMSProp ve momentumlu SGD karışımı olan bir optimizasyon algoritmasıdır. Güncelleme denklemindeki payda, ADADELTA algoritmasındaki ΔW^2 yerine gradyanın birinci momentini kullanır. Payda olarak ise ADADELTA, AdaGrad ve RMSProp gibi gradyanın ikinci momentini kullanır. Ayrıca, ilk iterasyonlarda sorun oluşturan 1. ve 2. momentteki bias terimini ortadan kaldırmak için 2.37'daki bias denklemlerini sisteme eklemişlerdir.

$$g_t = \nabla_W L(W_t, X, y) \tag{2.37}$$

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \tag{2.38}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \tag{2.39}$$

$$\hat{m}_t = \frac{m_t}{(1-\beta_1^t)} \tag{2.40}$$

$$\hat{v}_t = \frac{v_t}{(1-\beta_2^t)} \tag{2.41}$$

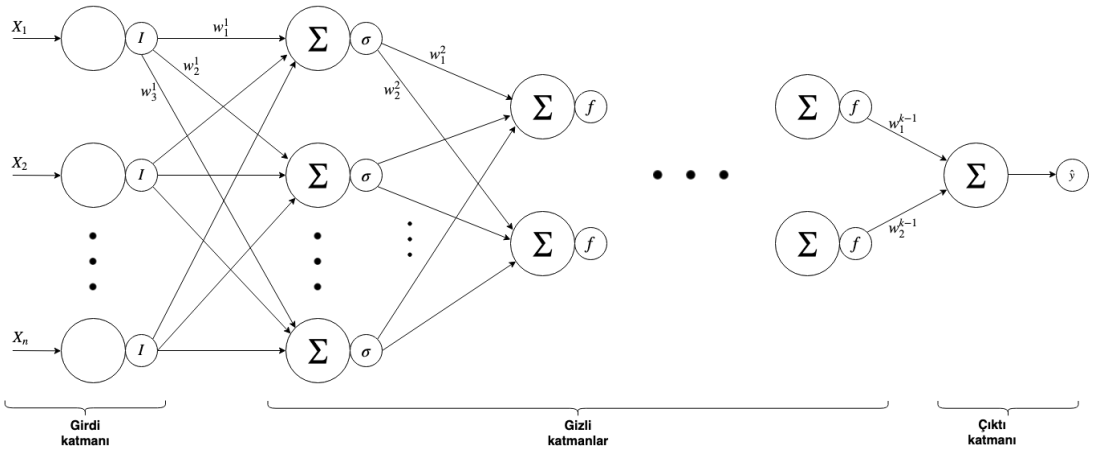
$$\Delta W_t = -\eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{2.42}$$

$$W_{t+1} = W_t + \Delta W_t \tag{2.43}$$

$$\beta_1 : 0.9$$
$$\beta_2 : 0.999$$
$$\varepsilon : 10^{-8}$$

Çok fazla avantaja sahip olduğu halde halen, çözüm uzayını momentumlu SGD kadar yumuşak şekilde tarayamaz. Bundan dolayı zaman sorunu yaşanmadığı durumlarda öğrenme faktörü annealing kullanan momentumlu SGD, diğer durumlarda ADAM algoritmasına başvurulmaktadır.

2.8 Tamamen Bağlı Sinir Ağları



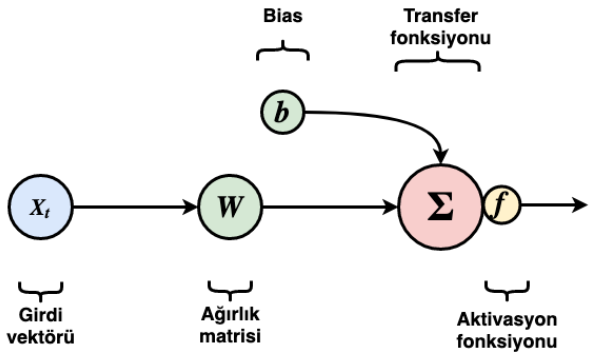
Şekil 2.23: Tamamen Bağlı Sinir Ağları örneği

Derin öğrenme alanındaki ilk model olarak sayılan FCNN ile ilgili çalışmalar Warren McCulloch ve Walter Pitts [44], [45] tarafından 1942-43 yılları arasında yapılmıştır. Yaptıkları çalışmada, ikili-eşik aktivasyon fonksiyonuna sahip bir nöronun birinci dereceden mantık kapılarını modelleyebileceğini göstermişlerdir. XOR, XNOR kapıları dışında kalan AND, OR, NOT, NAND, NOR kapılarının modellenebileğini gösterip, XOR ve XNOR kapıları için birden fazla katmana ve nörona ihtiyaç duyulduğunu söylemişlerdir. 1961 yılında Frank Rosenblatt yazdığı kitabında [46], Donald Hebb'in buldukları [47] ile McCulloch-Pitts nöronunu birleştirerek nöronlara giren girdiler için ağırlıklar tanımlamıştır. Bu ilk hali verilen perceptronlar yıllar içerisinde hem katman sayısı olarak hem de katmanlar içindeki nöron sayısı olarak genişleyerek, derin öğrenmenin ilk modellerinden olan FCNN'i ortaya çıkarmıştır.
FCNN'ler vektör olan bir girdi alırlar ve bu vektörü çeşitli sayıda olabilen ve çok sayıda gizli nörona sahip katmanlardan geçirerek, değişime uğratırlar. Herhangi bir k katmanı içerisindeki gizli nöronlar, kendi aralarında bir bağlantıya sahip değillerdir. Fakat, $k-1$ katmanının tüm nöronları, k katmanının tüm nöronları ile tamamiyle bağlı durumdadır. Aynı şekilde k katmanındaki tüm nöronlar da, $k+1$ katmanındaki tüm nöronlar ile bağlı olmak durumundadır. İlk katman girdi katmanı, son katman çıktı katmanı, aradaki katmanlara ise gizli katmanlar adı verilir (Şekil 2.23).

İleri besleme

Girdi katmanına gelen girdiler, birim aktivasyon fonksiyonundan geçerek ilk gizli katmana ulaşırlar. Gizli katmana gelen girdiler, gizli nöronların içindeki transfer fonksiyonundan ve daha sonra da aktivasyon fonksiyonundan (2.5) geçirilerek bir sonraki gizli katmana

aktarılırlar. Son gizli katmandan da çıkan çıktılar, çıktı katmanına iletilerek probleme uygun bir şekilde tahmin yapmak için kullanılırlar (2.44). Çıktı katmanı, eğer problem sınıflandırma problemi ise sınıfı, regresyon problemi ise bir sayıyı çıktı olarak verir. Bu yapının içerisindeki nöron yapısı Şekil 2.24'de görülmektedir.



Şekil 2.24: Tamamen Bağlı Sinir Ağları nöron yapısı

$$\begin{aligned} h_0^{out} &= x \\ h_i^{in} &= h_{i-1}^{out} * W_i + b_i \\ h_i^{out} &= f_i(h_i^{in}) \\ \hat{y} &= h_n^{out} \\ \mathscr{L}(W) &= L(y, \hat{y}) \end{aligned} \tag{2.44}$$

- i: katman numarası,
- f(.): aktivasyon fonksiyonu,
- X:girdi vektörü,
- n:katman sayısı,
- $\hat{y}$:tahmin
- $L(.)$: hata fonksiyonu
- $\mathscr{L}(W)$:minimize edilmesi gereken nihai fonksiyon

Geri besleme

Son katmandan sonucu olan çıktı hata fonksiyonuna girerek bir masraf çıkartılır. Bu masraf, son katmandan ilk katmana doğru zincir türev kuralı uygulanarak W ağırlık güncellemesi için kullanılır. Denklem (2.45)'de geri besleme adımında kullanılacak matematiksel ifadenin çıkarımı görülmektedir.

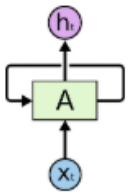
$$\begin{aligned}\frac{\partial \mathscr{L}(W)}{\partial W} &= \frac{\partial L}{\partial f}\frac{\partial f}{\partial (W^T x^i)}\frac{\partial (W^T x^i)}{\partial (W)} \\ &= \frac{\partial L}{\partial f}\frac{\partial f}{\partial (W^T x^i)}x^i\end{aligned} \tag{2.45}$$

2.9 Özyinelemeli Sinir Ağları

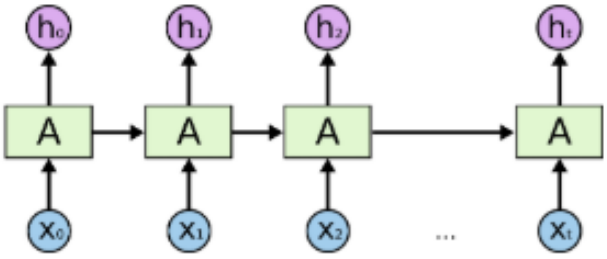
İnsanlar bilgiyi işlerken, eskiden öğrendikleri bilgileri, tecrübe ettikleri olayları, yeni durum ile harmanlayıp öyle tepki verirler. Fakat geleneksel modeller olaylar arasındaki bu bağları koruyamazlar. Olayların birbirinden tamamiyle bağımsız olduğunu varsayıp, ona göre bir çıktı verirler. Oysaki, veri setindeki her örneğinin birbirinden bağımsız olmadığı bir çok problem mevcuttur. Örnek verecek olursak; hava durumu tahmininde t zamanındaki veri, $t-1$ zamanındaki veri ile oldukça yüksek bir korelasyona sahiptir. Farklı bir örnek olarak; doğal dil işleme problemlerinde sözcükler cümlenin gidişatına göre farklı anlamlara gelebilirler. Bu tip problemleri çözebilmek için mutlaka eski bilgilerin saklanabileceği bir sisteme ihtiyaç vardır.

1986 yılında Rumelhart et al. [48] *Learning representations by backpropogating errors* adıyla bir çalışma yapmıştır. Bu çalışmada RNN mimarisinin temelleri atılmış, o günden bu zamana kadar çok çeşitli alanlarda kullanılmış ve çeşitli varyasyonları türetilmiştir (LSTM, Gated Recurrent Unit (GRU) vb.).

FCNN'in aksine, girdi vektörü tek bir ileri besleme geçişi ile çıktı vermez. Bunun yerine girdi vertörü zaman serisi boyunca RNN'in gizli katmanlarında özyinelemeli şekilde döndürülür ve sonrasında çıktıya dönüştürülür. Özyinelemeli ismini de buradan alır. Sahip olduğu bu doğal yapı sayesinde, zaman serisi ya da Doğal Dil İşleme (NLP) gibi problemlerde ön plana çıkmaktadır. Şekil 2.25'de basit RNN yapısını görülmektedir. Buradaki x_t girdi vektörünü, h_t ise çıktı vektörünü temsil etmektedir. A olarak tanımlanan kısım ise RNN modelinin iç yapısını temsil etmektedir. Bu yapının özyinelemeli yapısı kırılıp açıldığında, Şekil 2.26'deki yapı ortaya çıkar. $x_0..x_t$ şeklindeki t uzunluğundaki yapı RNN içerisinde şekildeki gibi dağılır ve her x_i için bir h_i çıktısı elde edilir.



Şekil 2.25: RNN yapısı [49]

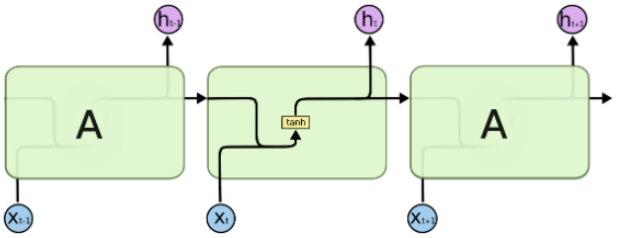


Şekil 2.26: RNN yapısı (açık) [49]

Denklem (2.46)'de basit RNN yapısının *ileri besleme* işlemi görülebilir. Bu denklemden anlaşılacağı üzere, aktivasyon fonksiyonu tanh olan basit bir sinir ağı kullanılmıştır. Bir önceki zaman adımının *gizli hal bilgisi* aktarılarak daha sonraki zaman adımındaki *gizli hal bilgisi* hesaplanabilir.

$$h_t = \tanh(W_x.[x_t, h_{t-1}]) \tag{2.46}$$

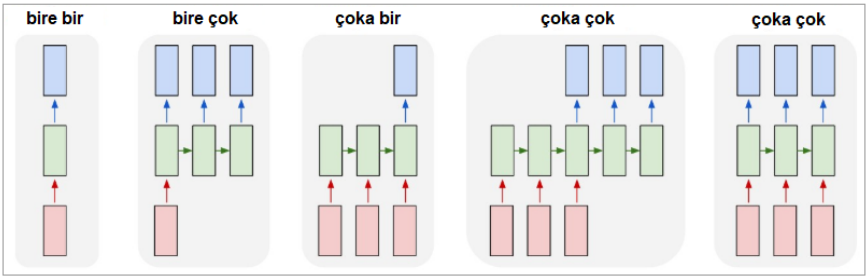
$$\left(\frac{\partial L}{\partial h_t}\right)\left(\frac{\partial h_t}{\partial h_{t-1}}\right)\left(\frac{\partial h_{t-1}}{\partial h_{t-2}}\right)\ldots\left(\frac{\partial h_2}{\partial h_1}\right) \tag{2.47}$$



Şekil 2.27: RNN'in basit iç yapısı [49]

Problemlere göre RNN'in vereceği çıktıları çeşitli şekillerde kullanabiliriz. Şekil 2.28'de kırmızı nöronlar girdi vektörünü, mavi olanlar çıktı vektörünü, yeşil olanlar ise RNN'in gizli nöronlarını temsil etmektedir. Soldan sağa doğru açıklanacak olursa: (1)'deki kullanım RNN'in özyinelemeli özelliğini kullanmak yerine onu standart bir MLP kullandığımızda ortaya çıkar. (2)'deki durum ise normal girdi alıp dizi olarak çıktı vermek istediğimiz durumlarda kullanılır (örneğin, verilen resme göre cümle çıktısı oluşturmak istenilirse). (3)'deki durum dizi şeklinde girdi verip tek bir çıktı istediğimizde kullanılır (örneğin, cümle olarak girdi verilip cümlenin olumlu mu yoksa olumsuz mu olduğuna karar verilmek istendiğinde). (4)'deki durum dizi olarak girdi verilip dizi olarak çıktı istendiğinde kullanılır (örneğin, türkçe bir cümle verilip ingilizce bir cümle çevirisi istendiğinde). (5)'deki durum senkronlu dizi şeklinde girdi

ve çıktı istendiğinde kullanılır (örneğin, bir videodaki her bir çerçevenin etiketlenmesi istendiğinde) [49].

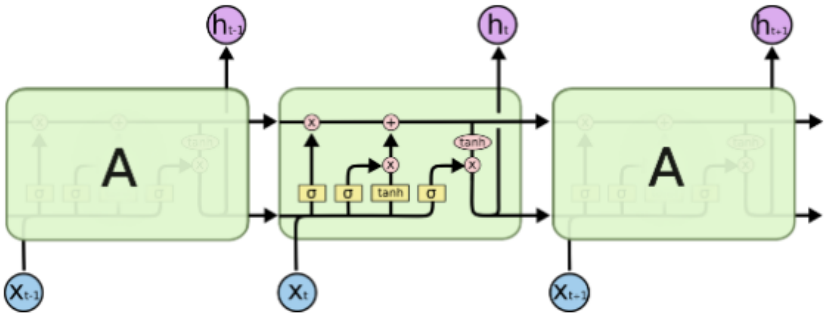


Şekil 2.28: Problem tiplerine göre RNN [49]

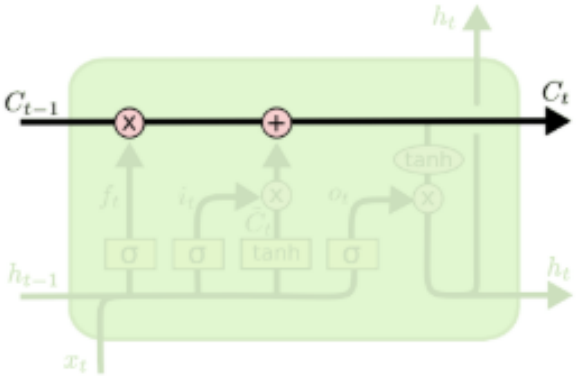
Fakat bu yapıdaki RNN modelinin bazı sorunları vardır. En önde gelen problem ise *vanishing gradient problem*dir. Bu problem, model karmaşıklaştıkça (arka arkaya eklenen RNN katmanlarının sayısı arttıkça) kendisini daha fazla gösterir. *İleri besleme* işlemi sonuçlandıktan sonra, hata fonksiyonuyla hesaplanan modelin tahmin hatasını, modelin iç katmanlarındaki nöronlara dağıtmak gerekir. Bu dağıtılacak hata daha önceden bahsedilen şekilde 2.6 *zincir-kuralı* uygulanarak modelin ağırlıklarının *hesaplanan gradyanlarına* göre dağıtılır. Bu işleme *geri besleme* adı verilir. Burada bahsedilen *zincir-kuralı* uygulanırken rnn modeli önce açılır, açıldığında kullanılan zaman serisi büyüklüğü alınacak olan *gradyanı* ya çok büyütür ya da çok küçültür. Bu yok olan ya da patlayan *gradyan* etkisi dizi boyunca uzun bağımlılık barındıran problemleri çözmeyi imkansız kılar. Bu sorunu çözmek için Hochreiter et al. [50] LSTM modelini geliştirmişlerdir. Bu model ve iç yapısı 2.9.1 bölümünde daha detaylı bir şekilde incelenecektir.

2.9.1 LSTM

Daha önceden bahsedildiği gibi Hochreiter et al. [50] LSTM modelini geliştirmişlerdir. Çalışma mantığı olarak *Vanilla RNN*den farklı olmasa da çok daha uzun dizilerin çalışmasına olanak vermektedir. *Vanilla RNN*e göre *yok-olan ya da patlayan gradyan sorununu* büyük ölçüde çözmektedir. Bu model vanilla RNN'de olan tek sinir ağını 4'e çıkarmış ve bu 4 sinir ağını birbirine özel bir şekilde bağlanmıştır. Şekil 2.29'de, LSTM modelinin iç yapısı ve bu 4 sinir ağının nasıl bağlandığı görülmektedir.



Şekil 2.29: LSTM modelinin iç yapısı [49]



Şekil 2.30: LSTM hücre bilgisi aktarımı [49]

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{2.48}$$

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{2.49}$$

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{2.50}$$

$$\hat{C}_t = \tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{2.51}$$

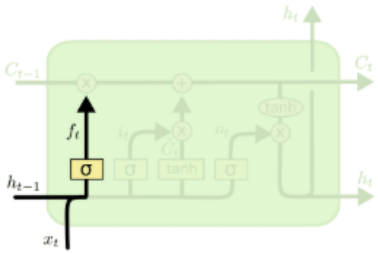
$$C_t = f_t * C_{t-1} + i_t.\hat{C}_t \tag{2.52}$$

$$h_t = o_t * \tanh(C_t) \tag{2.53}$$

Sigmoid fonksiyonu (daha önceden bahsedilen 2.18 grafiğindeki sigmoid eğrisine de bakılabilir) 0 ile 1 arasında bir çıktı vermektedir. 0 çıktısı "hiç bir şeyi geçirme", 1 çıktısını ise "her şeyi geçir" demektir. LSTM'in sahip olduğu bu şekildeki 3 kapı LSTM'in hücre halini korumasını, saklamasını ya da unutmasını kontrol eder.

Şekil 2.31'de gösterilen aktivasyon fonksiyonu sigmoid olan sinir ağı, *t anındaki girdi* ve $t-1$ *anındaki gizli hali* girdi olarak alır ve hücre halinin ne kadar bilgiyi unutup unutmayacağına karar verir. Buradaki f_t, 1 e yaklaştıkça $t-1$ anındaki hücre hali t

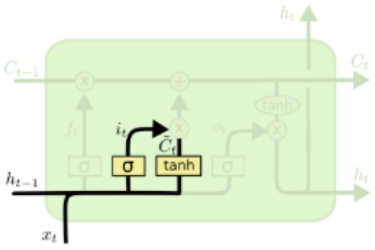
anına aktarılır. Aynı şekilde 0'a yaklaştıkça $t-1$ anındaki hücre hali unutularak sonraki sinir ağlarından gelen bilgiler bir sonraki t anına aktarılır. Denklem (2.48)'de t anındaki *unut kapısı*nın nasıl hesaplandığı görülebilir.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Şekil 2.31: LSTM unut kapısı [49]

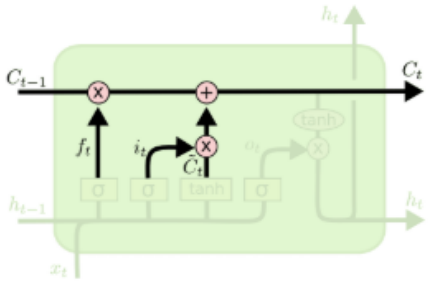
Şekil 2.32'da gösterilen aktivasyon fonksiyonu sigmoid olan sinir ağı, *t anındaki girdi* ve $t-1$ *anındaki gizli hali* girdi olarak alır ve çıktı olarak verdiği i_t de -1 ile 1 arasına sıkıştırılmış (tanh fonksiyonu verilen girdiyi -1 ile 1 arasına sıkıştırır) olan *kısmi yeni hücre bilgisi*nin ne kadarının eski hücre bilgisine ekleneceğine karar verir. Denklem (2.49)'de t anındaki *girdi kapısı*nın, denklem (2.51)'de t anındaki *kısmi yeni hücre bilgisinin* nasıl hesaplandığı görülebilir.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Şekil 2.32: LSTM girdi kapısı [49]

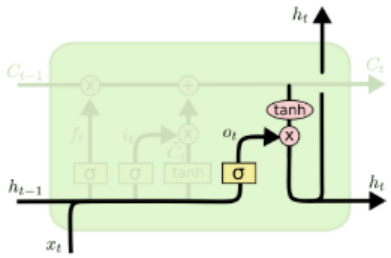
Şekil 2.33'de unut kapısının verdiği karar $t-1$ anındaki hücre bilgisiyle çarpılıp, girdi kapısının ne kadar yeni hücre bilgisinin sisteme dahil edileceğine karar verdiği vektör ile toplanarak t anındaki -bir sonraki nörona aktarılacak ya da özyinelemeli şekilde kendisine tekrar girecek olan- hücre bilgisi elde edilmiş olur. Denklem (2.52)'de t anındaki *hücre bilgisi*nin nasıl hesaplandığı görülebilir.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Şekil 2.33: LSTM hücre bilgisi güncelleme [49]

Şekil 2.34'de -1 ile 1 arasına sıkıştırılmış (tanh) t anındaki hücre bilgisi, $t-1$ anından gelen gizli hal bilgisi ile çarpılarak t anındaki -bir sonraki nörona aktarılacak ya da özyinelemeli şekilde kendisine tekrar girecek olan olan- gizli hal bilgisi elde edilmiş olur. Denklem (2.50)'de t anındaki *çıktı kapısı*nın, denklem (2.53)'da t anındaki *gizli hal bilgisi*nin nasıl hesaplandığı görülebilir.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Şekil 2.34: LSTM gizli durum bilgisi güncelleme [49]

LSTM, her ne kadar Hochreiter et al. [50] tarafından bulunarak *gradyanların yok olması ya da patlaması* sorunu büyük ölçüde çözülmüş olsa da nasıl ve ne şekilde kullanılması gerektiği problemin ve kullanılan zaman serisinin yapısına bağlı olduğu için uygulanması karışık olan bir modeldir. Bölüm 3.1.1 ve 3.1.2'de LSTM modeli, zaman serisinin türüne göre kullanılması gereken durum-denetlemeli LSTM ve durum-denetlemesiz LSTM olarak iki farklı başlıkça incelenecektir.

2.10 Konvolüsyonel Sinir Ağları

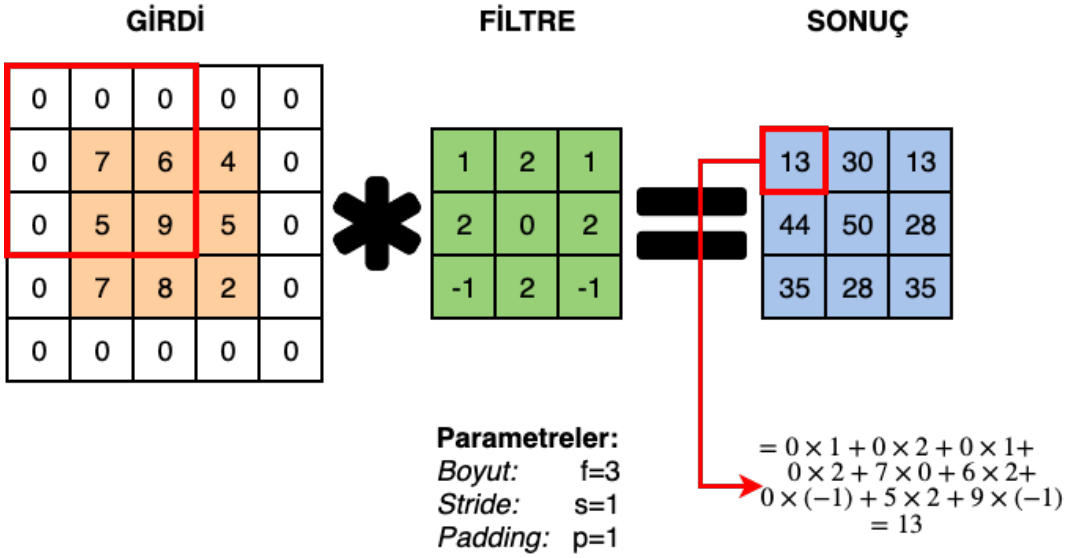
2012 yılındaki ImageNet yarışmasındaki üstün başarısından [51] sonra hemen hemen her yerde kullanılan CNN ilk olarak [52], [53] yayınlarında çalışılmış ve modern anlamda bildiğimiz yapıya Lecun et al. ın yaptığı çalışmada kavuşmuştur [54].
FCNNnda geçerli tüm kurallar, CNNna da aynı şekilde uygulanabilir. İleri besleme adımında katman katman ağırlık matrisleri çarpılarak bir sınıf ya da bir sayı hesaplanır,

daha sonra gerçek değerle kıyaslanan bu çıktı üzerinden gradyan hesaplanarak geri besleme adımında ağırlıklar güncellenir.

FCNNndan farklı olarak, belirli işlemlerde özelleşmiş katmanlara sahiptir: konvolüsyonel katman, havuz katmanı, tamamıyle bağlı katman. Bu katmanlar arka arkaya bağlanarak, asıl sinir ağı elde edilir. Bahsedilen katmanlar sırasıyla incelenecektir.

2.10.1 Konvolüsyonel katman

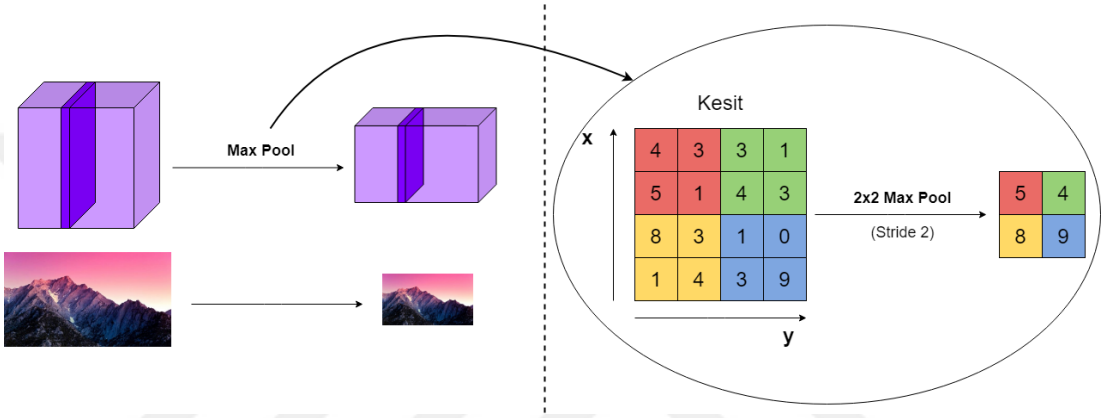
CNN adını veren bu katman, bu sinir ağının en önemli katmanıdır. Öğrenilebilen parametrelere sahip olan filtrelerden oluşur. Filtreler, girdi olarak verilen resimlerin (resim olmak zorunda değildir ama anlaşılması açısından resim örneğiyle devam edilecektir) üzerlerinde gezerek konvolüsyon operatörünü uygularlar. Stride miktarı, bir sonraki konvolüsyon penceresinin kaç adım atlayacağına, padding ise, filtreden çıkan tensörün boyutunu eşitlemek için tensörün çevresine 0'ların eklenip eklenmeyeceğine karar verir. Örneğin; 6x6x1 boyutlarındaki bir resim (6:genişlik, 6:yükseklik, 1:renk kanal sayısı) ele alınmış olsun: resim üzerinde gezdirilen 3x3x1'lik filtre, stride ve padding Şekil 2.35'de görülmektedir. Ek olarak, resim gibi komşu piksellerle bağlantı içerisinde bulunan verilerle oldukça iyi çalışmalarının sebebi, filtrelerin gezerken, boyutları kadar bir alan içerisindeki tüm pikselleri konvolüsyon hesabına katmalarındandır. Her katmanda resme farklı filtreler uygulayarak basit özniteliklerden, daha kompleks özniteliklere doğru bir geçiş sağlanır.



Şekil 2.35: Konvolüsyon operatörü

2.10.2 Havuz katmanı

Havuz katmanı, genelde iki konvolüsyonel katman arasına konulur. Boyutu azalttığı için, öğrenilmesi gereken parametre sayısını da azaltmış olur. En çok kullanılanı maks-havuzdur. Şekil 2.36'de 2x2'lik filtreye ve 2 birim stride'a sahip maks-havuz operasyonu görülmektedir. Daha önceleri ortalama-havuz ve l2-norm havuz versiyonları da kullanılmaktaydı fakat son yıllarda maks-havuzun gölgesinde kalmışlardır. Etkili bir şekilde kullanılmasına rağmen, havuz katmanı, [55] et al.ın yaptığı çalışmaya göre önemi yitirmeli ve yerini daha büyük strideların kullanıldığı tamamiyle konvolüsyonel katmanlara bırakmalıdır.



Şekil 2.36: Maks-havuz

2.10.3 Tamamen bağlı katman

Bölüm 2.8'de anlatılan FCNN ile birebir aynıdır. Bu katmanların başlamasından hemen önceki katmandan sonra kare ya da dikdörtgen şekilde gelen filtre çıktıları uzun bir vektör haline getirilip klasik şekilde çalıştırılmaya devam edilir. FCNN gibi bir sınıf ya da sayı tahmini bu katmanların çıktısıdır.

3. YÖNTEM

3.1 Durum Denetlemeli ve Denetlemesiz LSTM Karşılaştırması

3.1.1 Durum-denetlemeli LSTM

Sözü edilen LSTM modeli, [48], [50] makalelerinde anlatılan sırasıyla RNN ve LSTM modelleri teorik olarak durum-denetlemeli LSTM kapsamına girmektedir. Hücre hali ve gizli halnin, bir önceki durumdan bir sonraki duruma sürekli aktarılarak, özyinelemeli bir şekilde güncellenmeye devam edildiği tipteki LSTM, durum-denetlemeli LSTM olarak adlandırılmaktadır. Fakat yıllar içerisinde oluşturulan derin öğrenme kütüphanelerinde bu modeller tanımlanırken, kullanılacak olan zaman serisinin başlangıcında hücre hali ve gizli hal 0'larla ya da 0'a yakın küçük rastsal sayılarla doldurulur. Ileri besleme evresinden geçirilen zaman serisinden sonra hesaplanan hata, geri besleme evresiyle matris ağırlıklarını kendilerine ait *gradyan*lar oranında güncellerken, sürekli geriye doğru türev almak zorundadır. Örneğin, ilk ileri-geri besleme evresi her bir ağırlık matrisi üzerinde 1 kere türev alır (3.1), fakat herhangi bir ekstra işlem yapmadan tekrar yapılan bir ileri-geri besleme evresi esnasında hesaplanan türev, her bir ağırlık üzerinden 2 kere geçmek zorundadır (3.2). Bu durum sürekli tekrarlandığında, hesaplama çizgesi üzerinde defalarca türev hesaplamak gerekir (3.3). Fakat, oluşturulan derin öğrenme kütüphaneleri (tensorflow, pytorch vb.) aynı çizge üzerinde birden fazla gradyan hesabına izin vermezler. Bunun nedeni, bu türev hesabının oldukça maliyetli olması ve başka şekillerde yaklaşılırsa kolayca çözülebilecek bir durum olmasındandır. Durum-denetlemeli LSTM'in bu sorunu her bir ileri besleme evresinden sonra hücre hali ve gizli halnin bu hesaplama çizgesinden koparılmasıyla (*detach*) çözülebilir. Koparılan bu *haller*, geri besleme evresinde birden fazla geriye gitmeyi engelleyerek, ikinci bir çizge türevinin önüne geçer. Farklı bir çözüm ise, problemi durum-denetlemesiz LSTM ile çözülebilecek hale getirmektir. Bu durum 3.1.2 bölümünde daha detaylı bir şekilde anlatılacaktır.

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_t^1} \cdot \frac{\partial h_t^1}{\partial h_{t-1}^1} .. \frac{\partial h_2^1}{\partial h_1^1} \tag{3.1}$$

$$\frac{\partial L}{\partial h_t} = (\frac{\partial L}{\partial h_t^2} \cdot \frac{\partial h_t^2}{\partial h_{t-1}^2} .. \frac{\partial h_2^2}{\partial h_1^2} \cdot)(\frac{\partial h_t^1}{\partial h_{t-1}^1} .. \frac{\partial h_2^1}{\partial h_1^1}) \tag{3.2}$$

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_t^m} \cdot \prod_{i=1}^{m} (\frac{\partial h_t^i}{\partial h_{t-1}^i} .. \frac{\partial h_2^i}{\partial h_1^i}) \tag{3.3}$$

Durum-denetlemeli LSTM modelinin bir başka problemi ise yığın içerisinde sadece 1 adet örneğin olmasına izin vermesidir. Bu durum, *yakınsama* sürecinin oldukça uzamasına sebebiyet verir. Bunun nedeni, aktarılan *hal bilgilerinin*, modele girdi olarak verilen peşpeşe dizilerin birbirine *bağımlı* olduğunun varsayılmasındandır. Paralel bir şekilde beslenen modelde diziler arasındaki *bağımlılık* kaybedileceği için yığın tek bir örnek içermelidir. Bu durum Şekil 3.1'de daha açık bir şekilde görülebilir. Sistemin yakınsaması her ne kadar uzasa da verilen diziler arasındaki bağ korunduğu için çoğu zaman serisi problemine daha iyi uyum sağlar.(Ek bir not olarak: bu durum veri düzgün bir şekilde ayarlandığında belirli bir seviyeye kadar yığın sayısının arttırılmasına olanak tanır, fakat bu verinin uygun ve detaylı bir şekilde incelenmesinden sonra karar verilecek bir durum olduğundan model kullanımını daha da zorlaştıran bir durumdur. Bu şekilde incelemeye harcanacak zaman ile modelin tek yığın sayısı ile çalışırkenki harcayacağı ekstra zaman, bunları kullanacak olan kişinin vermesi gereken bir karardır. Ama yine de, her ne kadar iyi inceleme yapılıp düzgün parametreler ayarlanmış olsa da, teorik olarak yığın sayısını 1 ayarlamaktan daha kötü bir *yakınsama*ya ulaşılabilecektir.)

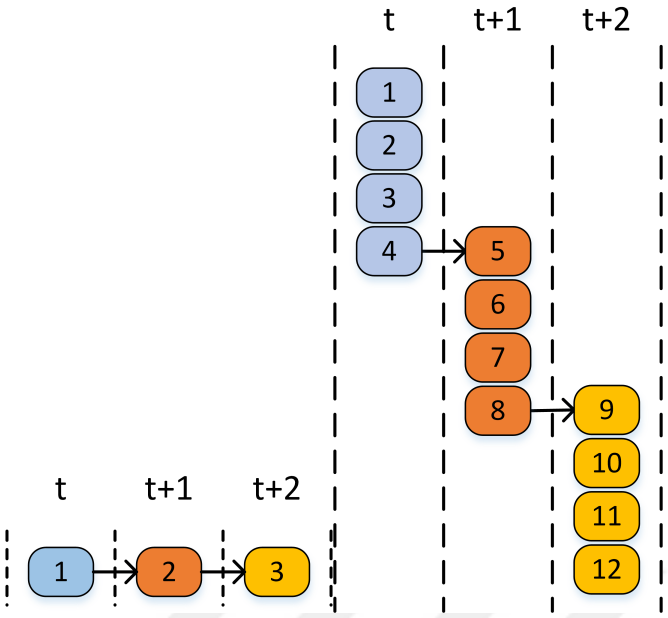
Yukarıda bahsedilen durumlar:

1. Her bir ileri besleme evresinden sonra gizli halnin koparılması ve bir sonraki ileri besleme evresinin başında tekrardan çizgeye bağlanması,
2. yığın sayısının 1 olarak ayarlanması

şeklinde özetlenebilir.

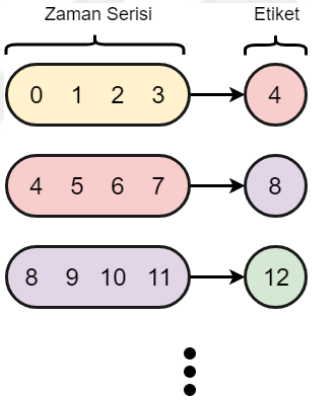
Bu ayarların yapılması ile kullanılabilinecek olan durum-denetlemeli LSTM modeli, manuel bir şekilde hücre halinin ve gizli halnin sıfırlanmasına olanak tanır. Problemde kullanılan zaman serileri periyodik ise (hava durumu, enerji yük tahmini gibi) onların periyodiklik durumlarına göre veya periyodiklik yoksa (finans tahmini gibi) hiç bir zaman sıfırlanmayarak diziler arasındaki *bağımlı* durum korunabilir.

Bu model kullanılırken verisetinin de uygun bir şekilde verilmesi gerekir. Önceden bahsedildiği gibi diziler arasındaki *bağımlı* durum korunmalı ve birbirini takip edecek şekilde herhangi bir rastsal fonksiyondan geçirilmeden kullanılmalıdır (Şekil 3.2).



(a) Yığın sayısı: 1 (b) Yığın sayısı: 4 veya daha fazla

Şekil 3.1: Farklı yığın sayılarındaki veri seti karşılaştırması

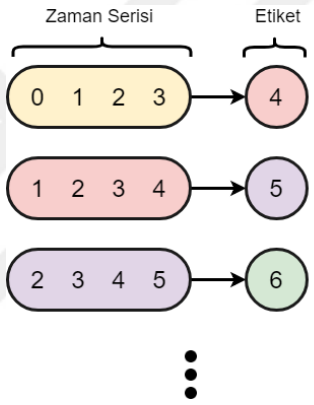


Şekil 3.2: Durum-denetlemeli Dataset

3.1.2 Durum-denetlemesiz LSTM

Diğer bir LSTM kullanma yöntemi ise durum-denetlemesiz LSTMdir. Yukarıda bahsedilen yöntemin aksine, kullanılan veri istenilen rastsal fonksiyondan geçirilebilir ve yığın sayısı istenilen sayıya ayarlanabilir. Bundan dolayı daha hızlı bir şekilde *yakınsar* ve verinin gidişatından kaynaklanan (sürekli artan ya da sürekli azalan diziler gibi) problemlere, bu yöntemde rastlanılmaz. Fakat uygulanabilecek problemler daha kısıtlıdır ya da daha fazla ön çalışma ve problem analizi gerektirir. Her ileri besleme evresinin başında hücre hali ve gizli hal sıfır ya da sıfıra yakın bir bilgiyle sıfırlanacağı için

koparılma işlemi uygulanmadan defalarca ileri-geri besleme evrelerinden geçirilebilir. Ancak her ileri besleme işlemi başında yapılan *koparılma* işlemi, diziler arasındaki bağı koparacağı için, diziler arasında herhangi bir *bağımlılık* olmadığı varsayılarak kullanılmalıdır. Yine aynı sebepten ötürü bu LSTM yöntemi, yalnızca verilen bir dizi içerisindeki bilgiyi yorumlayabilir. Önceki dizilerden gelecek herhangi bir bilgi unutulur. Bundan dolayı finans gibi belirli bir periyodikliğe sahip olmayan problemlerde kullanılması uygun değildir (aylık ya da haftalık periyodiklik olduğu varsayılarak kullanılabilir fakat bu varsayımın dışında kalan bilginin saklanamayacağı unutulmamalıdır). Bu model kullanılırken verisetinin Şekil 3.3'de görüldüğü gibi olması gerekmektedir. Zaman serisi üzerinde *kayan pencere* yöntemi kullanılarak oluşturulan bu verisetinden görüleceği üzere, $0,1,2,3$ girdisinden sonra $1,2,3,4$ girdisi verilebilmektedir. Diziler arasında *bağımlılık* aranmadığı için ilk dizide verilen 3'ten sonra ikinci dizeye 1 ile başlanabilir.



Şekil 3.3: Durum-denetlemesiz veri seti

3.2 LSTM Çalışması I: Uzun Vadeli Hafıza Problemi

3.2.1 Motivasyon

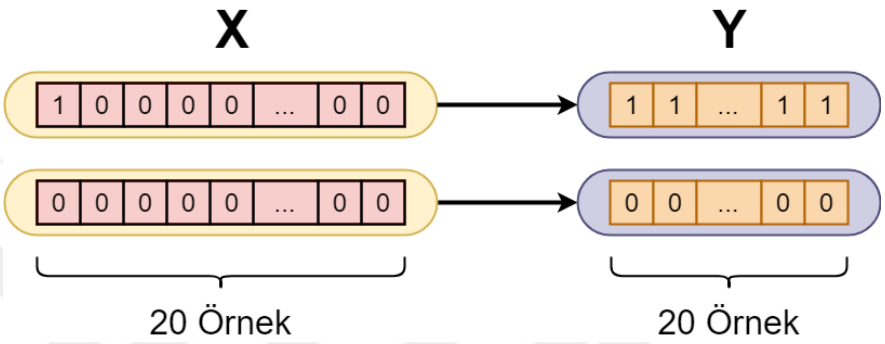
LSTM modeli ile ilgili gerçek verilerle yapılan yöntemlere geçmeden önce, LSTM'in gücünü anlamak adına, sentetik olarak hazırlanmış verilerin kullanıldığı ve LSTM'in uzun vadeli hafızasının ölçüleceği, basit bir çalışma yapılmıştır. FCNN, CNN ya da hafızaya sahip olmayan diğer derin öğrenme modelleri ile sonuç alınamayacak bir problem tasarlanmıştır.

3.2.2 Veri seti

Tasarlanan problemin veri seti örnekleri Şekil 3.4'de görülmektedir. Şekilden anlaşılacağı üzere, 20 zaman adımında 1 *high* ya da *low* olan, geri kalan 19 zaman adımı boyunca *low* kalan bir eğitim verisi oluşturulmuştur. Bu verisetinin her bir örneği, denklem (3.4)'deki

ifade ile elde edilmiştir. 1 ve 0'lardan oluşan veri seti 20 adımlık parçalara bölünürse, 0.5 olasılıkla ilk bit ya 0 ya da 1'lerden, geri kalan 19 bit 0'lardan oluşmaktadır. Etiketler ise ilk bitin değerine göre oluşturulmuştur. Kısaca, ilk bit değeri 1 ise 20 adım boyunca 1, 0 ise 20 adım boyunca 0 etiketi atanmıştır.

$$\begin{gathered} \delta(t \mod 20) \cdot I(X > 0.5) \qquad for\ t \in [0..\infty] \\ \text{where } \delta(t) \text{ is an impulse function} \\ \text{where } I(x) \text{ is indicator function and } X \sim U(0,1) \end{gathered} \tag{3.4}$$



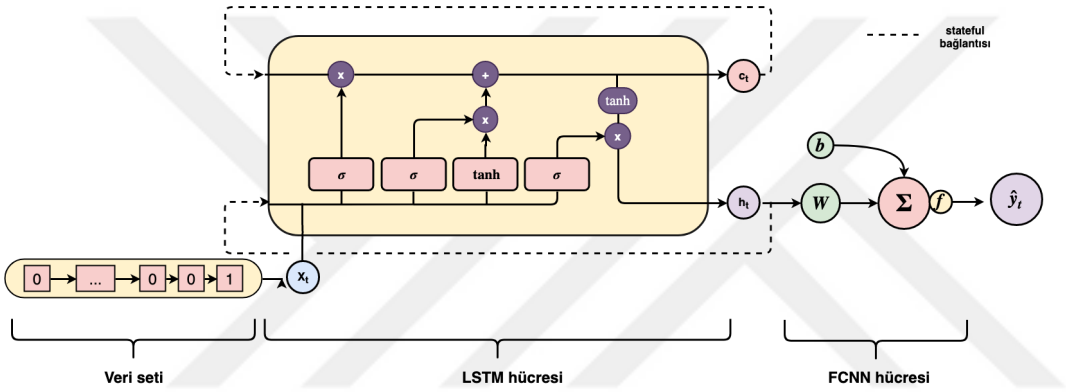
Şekil 3.4: Veri seti

3.2.3 Model

Çalışmada kullanılan model Şekil 3.5'de gösterilmiştir. Model, sadece bir adet durum-denetlemeli LSTM nöronu ve bir adet FCNN nöronuna sahiptir. Durum-denetlemeli LSTM nöronundan gelen gizli hal ile beslenen FCNN nöronu iki sınıflı sınıflandırma problemi çözülmeye çalışıldığı için, iki elemanlı bir çıktı vektörü vermektedir.
Modelde sadece bir adet durum-denetlemeli LSTM nöronunun kullanılması, LSTM'in yalnızca hafıza kapasitesini göstermek için hazırlanan bir çalışma yapılmaya çalışılmasındandır. Daha fazla nöron eklendiği zaman modelin iç yapısını göstermek zorlaşacak ve tam olarak gizli hal ve hücre halinin nasıl aktarıldığını açıklamak zorlaşacaktır. Bunlara ek olarak, en az karmaşık yapı ile problemin çözülebildiğini göstermek de hedeflenen diğer amaçtır.
Veri setindeki her bir bit değeri modele tek tek verilerek, modelin bu girdilerden tahmin yapması beklenmiştir. Bu şekilde tek tek verilen veriler ile beslenen FCNN ya da CNN gibi modeller, problem çözümünde yetersiz kalmaktadırlar. Bunun nedeni, bu modellerde, her hangi bir aşamada LSTM'deki hücre hali gibi bir yer tutucunun kullanılmamasındandır. Durum-denetlemesiz LSTM, bahsedilen yer tutucuya sahip olduğu halde, bu yer tutucu her adımda sıfırlanacağı için öğrenme açısından diğer modellerden daha fazla başarı kazanacak yetisi kalmamıştır. Bu nedenle kullanıma

uygun değildir.

Problemin çözülebilmesi için, yalnızca, herhangi bir t zamanı için, $[t-19,t]$ aralığındaki herhangi bir yerdeki *impulse* bilgisinin saklanabilmesi gerekir. Bu sebepten ötürü, durum-denetlemesiz LSTM yerine kullanımı daha zor olan durum-denetlemeli LSTM kullanılmıştır. Durum-denetlemeli LSTM sayesinde, herhangi t zamanındaki gizli hal ve hücre hali $t+1$ girdisi için kullanılmakta ve bu bilgi daha sonraki zaman adımlarına aktarılabilmektedir. 20 zaman adımında bir gizli hal ve hücre hali sıfırlanarak, 20 adım öncesindeki geçmişe dönelik özyinelemeli gradyan hesabının alınması engellenmiştir. Zaten önemli olan bilgi $t \in [1,20]$ içerisinde olacağı için sıfırlanma periyodunun 20 olarak atanması uygun görülmüştür. Sıfırlama periyodu daha küçük bir sayı seçildiğinde bilgiye ulaşamadan sıfırlanma söz konusu olabilmekte, daha büyük bir sayı seçildiğinde ise gereksiz gradyan hesabı yapılarak hesaplama masrafı arttırılmaktadır.



Şekil 3.5: Model

Bu tip bir problem için SGD, ADADELTA, AdaGrad ya da ADAM gibi optimizasyon algoritmaları kullanılabilir. Optimizasyon açısından çözülmesi gereken karmaşık bir problem bulunmadığı için çok kritik bir kara vermeden, en yaygın olarak kullanılan ve adaptif olarak çalışan ADAM optimizasyon algoritması kullanılmıştır. Sınıflandırma problemi olmasından dolayı, optimize edilmesi istenilen hata fonksiyonu CE (daha spesifik olmak gerekirse, ikili çapraz-entropi (BCE)) olarak belirlenmiştir. Modelde kullanılan tüm sabitler aşağıda listelenmiştir:

- optimizer = ADAM
- öğrenme faktörü= 0.001
- hata fonksiyonu = CE
- girdi boyutu = 1
- çıktı boyutu = 1
- LSTM gizli nöron sayısı = 1
- FCNN gizli nöron sayısı = 1
- yığın boyutu = 1

- katman sayısı = 1
- durum-denetlemeli = doğru
- durum sıfırlama periyodu = 20

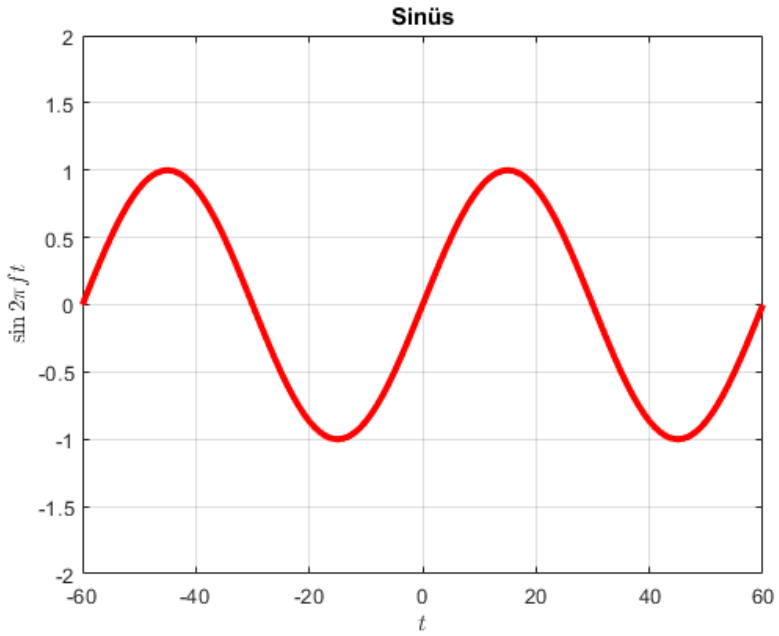
3.3 LSTM Çalışması II: Sinüs Dalgası

3.3.1 Motivasyon

Bölüm 3.2'de olduğu gibi, bu kısımda da gerçek verilere geçmeden önce periyodik bir sinyalin nasıl öğrenildiğini açıklayacak deney düzeneği hazırlanmıştır. LSTM modelinin, zaman serisi şeklinde verilen girdiler ile öğrenme işlemini gerçekleştirebilmekte ve doğru tahminler yapabilmekte olduğu gösterilecektir.

3.3.2 Veri seti

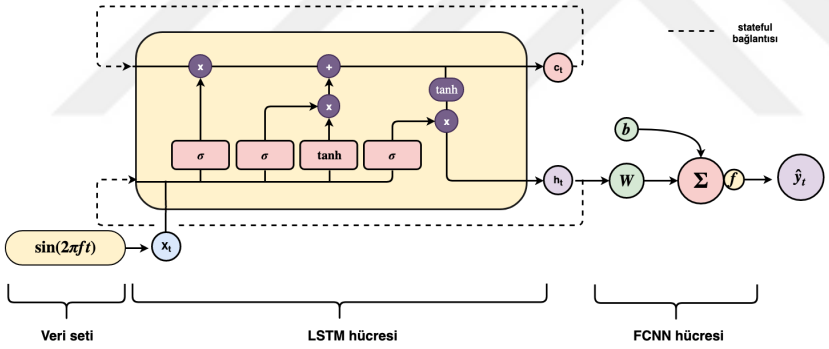
Tasarlanan sistemin veri setini oluşturmak için en basit periyodik sinyallerden biri olan $x(t) = sin(\omega t)$ kullanılmıştır. Daha spesifik olmak gerekirse, $x(t) = sin(2\pi\frac{1}{60}t)$ fonksiyonu kullanılmıştır. Denklemden anlaşılacağı üzere, frekans $1/60$ yani diğer bir deyişle periyot 60 olacak şekilde bir sinüs dalgası kullanılmıştır. Şekil 3.6'de oluşan veri setinin 60 birimlik zaman serisi yani bir periyodu görülmektedir.



Şekil 3.6: Veri seti

3.3.3 Model

Çalışmada kullanılan model Şekil 3.7'de görülmektedir. Önceki çalışmada olduğu gibi bu modelde de 1 adet durum-denetlemeli LSTM nöronu ve 1 adet FCNN nöronu kullanılmıştır. LSTM nöronundan gelen gizli haller ile beslenen FCNN nöronu sinüs fonksiyonunun bir sonraki zaman adımını tahmin etmeye çalışmaktadır. Modelde bu denli az sayıda nöron kullanılmasının sebebi, problemi az sayıda nöronla çözüp iç katmanların çıktılarını incelemek içindir.
Bir önceki çalışmada olduğu gibi veriler, modelin uzun vadeli hafızasını ölçmek adına, tek tek verilmektedir. Bu şekilde veriye ait başka bilginin verilmediği sistemlerde RNN yöntemlerinden başka bir modelin bunu çözebilmesi mümkün değildir. Bu nedenle kullanılan durum-denetlemeli LSTM, sinüs verisinin önceki değerlerini hafızada tutarak, yeni değer de geldikten sonra, tüm bilgileri harmanyalayıp bir tahmin yapmaktadır. Kullanılan durum-denetlemeli LSTM, hesaplamalı çizgede çok fazla geriye dönük gradyan hesabı yapılmaması için, belirli periyotlarla sıfırlanmalıdır. Sinüs dalgasının periyodu 60 olarak belirlendiğinden dolayı, sıfırlama periyodu da 60 olarak belirlenmiştir. Böylelikle model, $t \in [1, 60]$ aralığındaki herhangi bir noktadaki bilgiyi saklayabilecek ve sonraki adımların tahmininde bu bilgi kullanabilecek hale gelmiştir.



Şekil 3.7: Model

Öğrenme esnasında, hatada bazen dalgalanmalar olduğu tespit edildiği ve daha yumuşak bir öğrenme süreci istendiği için ADAM yerine nesterov momentuma sahip SGD kullanılmıştır. Regresyon problemi olmasından dolayı, masraf fonksiyonu olarak ise MSE fonksiyonu kullanılmıştır. Modelde kullanılan tüm sabitler aşağıda listelenmiştir:

- optimizer = Nesterov momentumlu SGD
- hata fonksiyonu = MSE
- öğrenme faktörü = 0.0001
- momentum = 0.99
- girdi boyutu = 1
- çıktı boyutu = 1

- dizi uzunluğu = 1
- LSTM katman sayısı = 1
- LSTM gizli nöron sayısı = 1
- FCNN gizli nöron sayısı = 1
- yığın boyutu = 1
- durum-denetlemeli LSTM = doğru
- durum sıfırlama periyodu = 60
- tur sayısı = 45

3.4 Enerji Üretim ve Tüketim Tahmini Çalışması

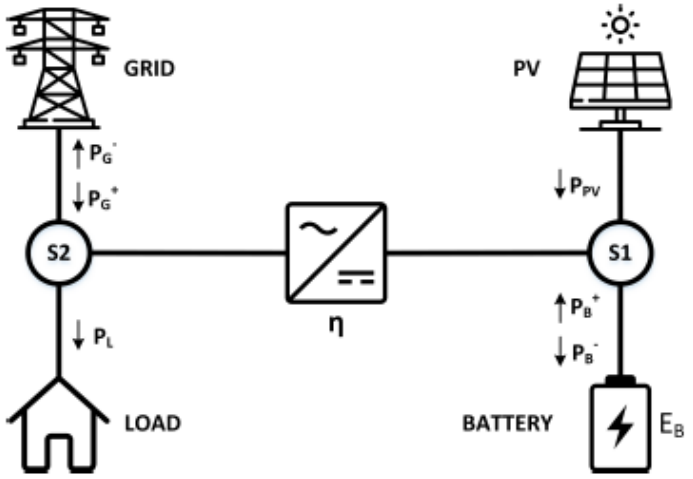
3.4.1 Motivasyon

Literatürde enerji üretim ve tüketim verilerini kullanan çok sayıda çalışma bulunmaktadır ([28], [29], [30], [31]). Fakat bu çalışmalar, genellikle geleneksel tahmin yöntemlerini kullanmaktadırlar ve çoğunlukla konuyu sadece belirli bir yönünden (sadece tüketim tahmini, sadece üretim tahmini ya da sadece optimizasyon) ele almaktadırlar. Bu çalışmada, [56] da belirtilen problemler birleştirilerek, baştan sonra bir optimizasyon problemi çözülmeye çalışılmıştır. Çalışma kapsamında, enerji tüketim ve üretim tahminleri yapılmış ve bu tahminleri kullanarak Karışık-tamsayılı Lineer Programlama, Mixed-integer Linear Programming (MILP) optimizasyon algoritması desteğiyle, batarya optimizasyonu yapılmıştır.

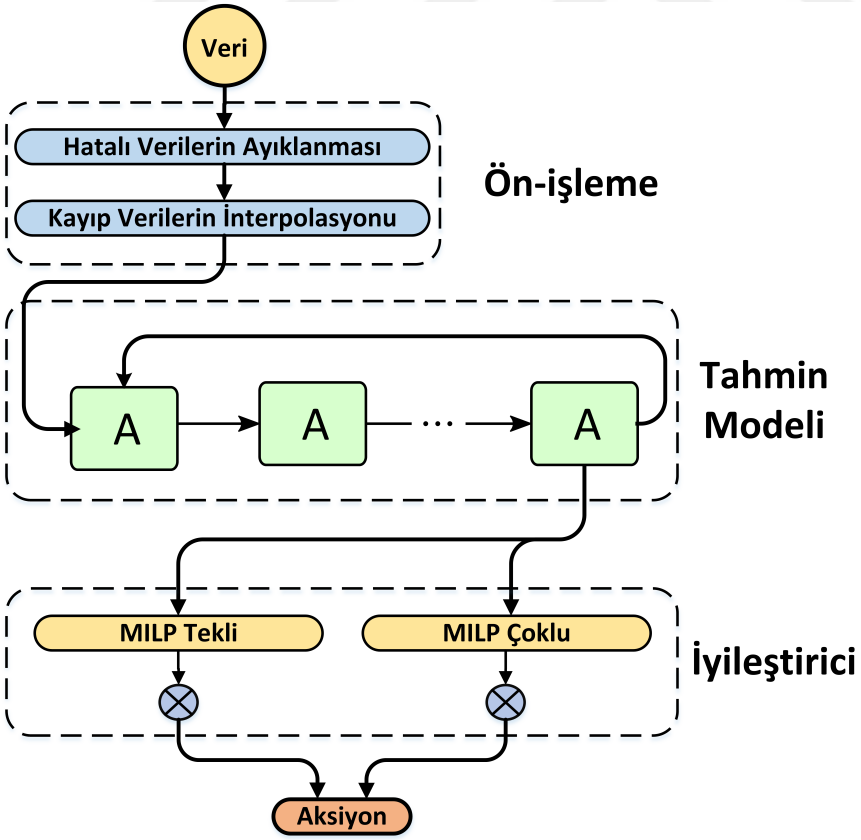
Bu çalışmada yapılan tahmin, batarya optimizasyonu için kullanılacak olsa da, günlük hayatta, üretim santraline sahip olan şirketin gün öncesi piyasada daha doğru bir teklif vermesini sağlamak için kullanılır. Yapılan teklifin doğruluğu yüksek olmazsa, tahmin ile gerçek değer arasındaki fark fazla çıkar ve devlet bu üretim şirketlerinden cezalı olarak tahsilat yapar. EnerjiSA gibi dağıtım şirketleri ise alacağı miktarı belirlemek için tahmin yaparlar. Eğer yanlış tahmin yaparlarsa, örneğin eksik tüketim tahmini yaptıklarsa, eksik kalan miktarı gün-içi piyasadan almak zorunda kalırlar ve gün öncesi piyasaya göre daha pahalıdan almış olurlar, fazla tüketim tahmini yaptıklarında ise santrallerden vaad edilen gücü çekemedikleri için sistemin operasyonel dengesini bozarlar(elektrik sisteminin frekansı yükselir) ve aynı şekilde cezasını ödemek zorunda kalırlar.

Taşarlanmış olan nanogrid evreni Şekil 3.8'de görülmektedir. Evren, şebeke, tüketim yapan hane, üretim yapan güneş enerjisi sistemi ve batarya elemanları ile modellenmiştir. Sistemdeki tüketim ve üretim fiyatlandırması saate göre değişiklik göstermektedir (ToU tarifesi). Şekildeki $S1$ ve $S2$ anahtarları tüketim ve üretimin nereden geleceğine karar veren anahtarlardır. Bu anahtarlar açılıp kapatılarak güç akışının yönü değiştirilebilir. Şebekedeki, güneş panelindeki, hanedeki ve bataryadaki enerji akışları sırasıyla P_G,

P_{PV}, P_L, P_B ile gösterilmiştir. Bu çalışma, sistemdeki güç açışını kontrol ederek hanenin ödemesi gereken masrafı azaltmayı amaçlamaktadır. Fakat, güç akışını kontrol etmek için güneş panelinin üretiminin ve hanenin tüketiminin tahmin edilmesi gerekmektedir. Bu tahminleri yapabilmek ve kullanılan veriler de zaman serisi olduğu için LSTM modeli kullanılmıştır.



Şekil 3.8: Sistem tasarımı



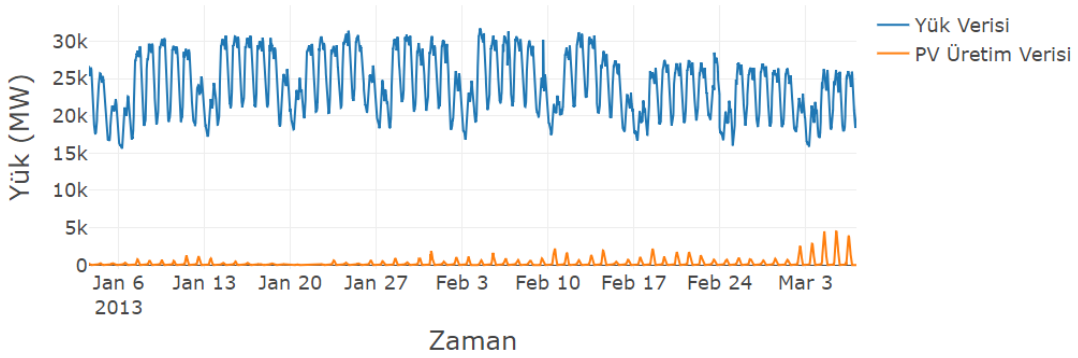
Şekil 3.9: Model

3.4.2 Veri seti

Tasarlanan sistemin akış diyagramı Şekil 3.9'de verilmiştir. Sistem verileri için Avrupa Elektrik İletim Sistemi İşleticileri Birliği, European Network of Transmission System Operators for Electricity (ENTSOE) dahil olan ve Almanyada yer alan Amprion GmbH elektrik dağıtım şirketinin verileri kullanılmıştır. Kuzey Rhine-Westphalia, Rhineland-Palatinate and Saarland bölgelerinin elektrik dağıtımını yapan bu şirketin 01.04.2011 ve 31.12.2017 tarihleri arasındaki ve günlük 96 adet zaman diliminden oluşan yani 15 dakikalık frekansa sahip verileri kullanılmıştır.

Şekil 3.9'de görüldüğü gibi veri seti öncelikle ön-işlemden geçirilmiştir. Bu ön-işlem sırasında, ilk olarak eksik olan ve negatif değere sahip olan veriler temizlenmiş, sonrasında, tanımsız veri kalmaması için, yük verileri için bir hafta sonraki veriler, güneş enerjisi için ise en fazla korelasyona sahip olduğu düşünülen gelecek günlerdeki veriler kullanılmıştır. En son olarak, ölçüm sensörlerinden kaynaklanan sorunlardan dolayı oluşan eksik verilere *lineer-interpolasyon* uygulanmıştır. Ayrıca, LSTM modelinin gradyan patlaması yaşamaması için anlatılan bu işlemlerden sonra, öznitelik ölçeklendirilmesi (2.2) uygulanmıştır. Temizleme işlemi sırasında uç değer kalmadığı için öznitelik ölçeklendirmesi yöntemi bir sorun teşkil etmemiştir. Ek olarak, LSTM modelini daha yüksek yığın sayısı ile kullanabilmek adına, saat, gün, hafta, ay ve yıl gibi öznitelikler veriye eklenmiştir. Bu yeni zamansal öznitelikler ile, kesin olarak tanımlanabilecek sinyal periyodu olmadığı için uygun olmayan yerlerde sıfırlanabilecek hücre ve gizli hal bilgilerinin, sisteme olacak olan kötü etkisinin düşürülmesi amaçlanmıştır.

Eğitim için 2922 günlük verinin 2700 günü, test kısmı için ise son 222 günlük veri kullanılmıştır.

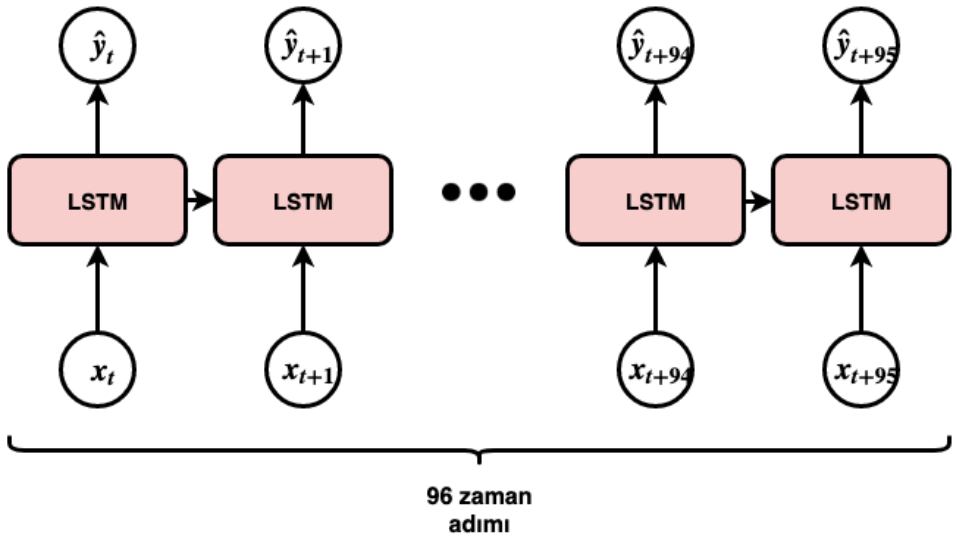


Şekil 3.10: Örnek veri

3.4.3 Model

PV üretimi ve hane enerji tüketimi tahmin problemleri veri seti bakımından oldukça yakın problemlerdir. Bu sebeple, tek bir mimari seçilmiş fakat farklı iki model eğitilmiştir. Bir tam gün içerisinde her iki veride de 96 zaman dilimi (saatte 4 veri örneği) olduğundan dolayı ve her bir zaman diliminin belirli bir hücre içerisinde tutulmasını kolaylaştırmak adına 96 adet LSTM hücresi kullanılmıştır. Böylelikle model 1 günlük verinin her parçasını farklı hücrelerde tutabilecektir. Ek olarak, veriyi 96 zaman adımlık parçalar halinde bölmek, verilen zaman dizileri arasındaki bağımlılığı kaldırdığı için önceden bahsedilen (3.1.2) durum-denetlemesiz LSTM kullanımının önünü açmıştır ve bununla birlikte yığın sayısı diğer parametrelerden bağımsız olarak ayarlanabilir hale gelmiştir. Kullanılan modelin mimarisi Şekil 3.11'de gösterilmektedir. Kullanılan hiper-parametreler ise aşağıda görülebilir. Modelin ağırlıklarını optimize etmek için ADADELTA optimizasyon algoritması kullanılmıştır. Bu algoritmanın, öğrenme faktörü kısmının da adaptif olması sayesinde yeni bir hiper-parametre oluşturmaya gerek kalmamıştır.

- optimizer = ADADELTA
- hata fonksiyonu = MSE
- girdi boyutu = 5
- çıktı boyutu = 1
- dizi uzunluğu = 96
- LSTM katman sayısı:1
- LSTM gizli nöron sayısı = 96
- yığın boyutu = 500
- durum-denetlemeli = yanlış
- tur sayısı = 400



Şekil 3.11: Model

3.4.4 Optimizasyon

Tahminler ne kadar iyi olursa olsun, optimizasyon algoritması seçimi, iyi sonuçlar elde etmede oldukça önemli bir rol oynamaktadır. Batarya optimizasyonu probleminin doğrusal olmayan bir problem olması, yapısını oldukça karmaşıklaştırmaktadır. Bu nedenle, problemi doğrusal-olmayan optimizasyon problemi yapısına sokan parçalar incelenmiş ve yeniden düzenlenmişlerdir. Örneğin, kullanılan bataryanın yaşlanması doğrusal olmayan bir eğriye sahiptir. Bataryanın uzun vadeli etkileri çalışma kapsamında incelenmeyeceği için, bu parça problemden çıkarılmış ve geri kalan kısım parçalı doğrusal bir problem olarak modellenebilmiştir. Parçalı doğrusal yapıya sahip olmasının nedeni ise, sistemdeki masraf fonksiyonunun ve çeviricilerin verimliliğinin parçalı doğrusal yapıya sahip olmasındandır. Neyseki bu tip parçalı doğrusal fonksiyonlar, doğrusal kısıtlar ve bazı mantık kapılarıyla, doğrusal hale getirilebilmektedir. Denklem 3.4.4'de gerekli kısıtların ve mantık kapılarının uygulanmış olduğu, bir *tamsayı programlama optimizasyon* algoritması olan MILP ile optimize edilmeye hazır yapı oluşturulmuştur. Optimizasyon modeli için kullanılan parametreler ve anlamları Çizelge 3.1'te verilmiştir. Bu çizelgedeki fiyat tarifesi (p_n), Çizelge 3.2'te ayrıca gösterilmiştir. Sistemi parçalı-doğrusal hale getiren işaretsiz P_{G_n} parametresi $P_{G_n}^{+}$ ve $P_{G_n}^{-}$ şeklinde iki işaretli parametre ile ifade edilmiş ve gerekli kısıtlar denklem 3.4.4'daki gibi eklenmiştir. Bu parametreye uygulanan işlem, aynı şekilde diğer güç akışını temsil eden ve sistemi parçalı-doğrusal hale sokan parametrelere de uygulanmış ve bu sayede sistemin doğrusallığı sağlanmıştır. Denklem 3.6-3.12 arasında kısıtlar, denklem 3.5'de ise minimize edilmeye çalışılan hedef fonksiyonu tanımlanmıştır.

Çizelge 3.1: Parametreler

Parametre	**Tanım**	**Değer**
N	Optimizasyon sürecindeki zaman dilimi sayısı	96
m	Fiyat çarpanı	0.9
η	İnvertör verimi	0.9
T	Zaman diliminin süresi (saat)	0.25
E_{max}	Batarya enerji depolama kapasitesi (kWh)	6
B_{max}	Batarya giriş-çıkış gücü (kW)	3
I_{max}	İnvertör nominal gücü (kW)	3
p_n	Zaman dilimi-n içerisindeki enerji fiyatı (fiyat/kWh)	
P_{L_n}	Zaman dilimi-n içerisindeki yük (kW)	
P_{PV_n}	Zaman dilimi-n içerisindeki PV üretimi (kW)	

Çizelge 3.2: Almanya ve Türkiye enerji fiyat tarifeleri

Ülke	**Saat Dilimi**	**fiyat/kWh**
DE	08.00-20.00	27.52 ct.
	20.00-08.00	24.31 ct.
TR	06.00-17.00	9.18 ct.
	17.00-22.00	14.07 ct.
	22.00-06.00	5.71 ct.

Minimize et

$$\sum_{n=1}^{N} p_n \cdot (P_{G_n}^{+} - m \cdot P_{G_n}^{-}) \cdot T \tag{3.5}$$

Kısıtlamalar

$$\sigma_n^{+} - \sigma_n^{-} = P_{PV_n} + P_{B_n}^{+} - P_{B_n}^{-} \tag{3.6a}$$
$$\sigma_n^{+} \leq I_{max} \cdot z_{1,n} \tag{3.6b}$$
$$\sigma_n^{-} \leq I_{max} \cdot (1 - z_{1,n}) \tag{3.6c}$$
$$P_{G_n}^{+} - P_{G_n}^{-} = P_{L_n} - \eta \cdot \sigma_n^{+} + \frac{1}{\eta} \cdot \sigma_n^{-} \tag{3.6d}$$

$$P_{G_n}^{+} \leq (P_{L_n} + \frac{1}{\eta} \cdot I_{max}) \cdot z_{2,n} \tag{3.7a}$$
$$P_{G_n}^{-} \leq I_{max} \cdot (1 - z_{2,n}) \tag{3.7b}$$

$$E_n = E_{n-1} - (P_{B_n}^{+} - P_{B_n}^{-}) \cdot T \tag{3.8}$$
$$E_n \leq E_{max} \tag{3.9}$$

$$P_{B_n}^{+} \leq B_{max} \cdot z_{3,n} \tag{3.10a}$$
$$P_{B_n}^{-} \leq B_{max} \cdot (1 - z_{3,n}) \tag{3.10b}$$

$$P_{G_n}^{+}, P_{G_n}^{-}, P_{B_n}^{+}, P_{B_n}^{-}, E_n \geq 0 \tag{3.11}$$
$$z_{1,n}, z_{2,n}, z_{3,n} \in \{0, 1\} \tag{3.12}$$

Sisteme eklenen ilk kısıt, güç akışının yönüne göre değişiklik gösteren, çeviricilerin verimliliğidir. Bu lineer-olmayan durum, denklem 3'teki ifadeler ve yeni tanımlanan z_1 ikili değişkeni ile çözülmüştür. z_1 değişkeni, çeviriciler üzerindeki akış yönünü temsil etmektedir. Satış fiyatı, gerçek enerji fiyatı ile Çizelge 3.1'daki m değişkeni çarpılarak elde edilmiştir. z_2 ikili değişkeni kullanılarak şebeke ve yük arasındaki akış yönü tanımlanmış ve denklem 3.7 kısıtı ile probleme eklenmiştir. Batarya dolup boşalırken, E_n değişkeni sürekli değişim göstermektedir. Bu etki, denklem 3.10 ile sisteme eklenmiştir. Denklem 3.8 ile bataryanın fiziksel kapasite limitleri, denklem 3.9

ile fiziksel güç limitleri z_3 ikili değişkeni yardımı ile düzenlenmiştir. Denklem 3.11, kullanılan yönlü güç akışlarının negatif olmayan bir yapıda olmasını sağlarken, denklem 3.12 ise yeni z ikili değişkenlerini tanımlamaktadır.

Optimizasyon stratejisi olarak, çok-zamanlı ve tek-zamanlı olmak üzere iki farklı strateji izlenmiştir. Bu stratejilerden çok-zamanlı olanı, optimizasyon uygulanacak günün başlangıcında bir kere çalıştırılır ve gün boyunca yapılacak işlemlerin kararı tek seferde alınır. Tek-zamanlı olanda ise probleme daha dinamik bir çözüm sunulmuş ve bunu gerçekleştirebilmek için 15 dakikalık periyotlarla optimizasyon çalıştırılmış, çıkan 96 zaman dilimli sonucun sadece ilki kullanılmış ve gün boyu bu işlem tekrarlanmıştır. Çözülmeye çalışılan bu problem gerçek hayata uygulanmaya çalışıldığında, her ne kadar tahmin verileri ile gerçek veriler arasındaki uyuşmazlık az olsa da tüketimin ya da üretimin tahminden farklı olacağı durumlar oluşmaktadır. Bu gibi durumlarda, üretim tahmin edilenden fazla olmuşsa fazla üretilen enerji şebekeye satılmış, tüketim fazla olmuşsa batarya yetmeyeceği için güç şebekeden çekilmiştir.

3.5 CNN ve Finans Verileri Çalışması

3.5.1 Veri seti

Öncelikle, fiyat ya da trend tahmini yapmak için verilerin olması gerekir. Bu verileri elde etmek için, Google Finance API kullanılmıştır. Google Finance'in sağladığı veriler günlük olup; tarih, açılış, kapanış, en düşük, en yüksek, işlem hacmi ve düzenleşmiş kapanış değerlerini barındırır. Düzenlenmiş kapanış değerleri, kar payı, sermaye değişmeden hisse sayısının değiştirilmesi ya da bedelli sermaye arttırımı gibi sebeplerden bozulan kapanış değerleri verisinin düzenlenmesi ile oluşturulmuştur.

Finans verileri doğası gereği yüksek değişkenliğe sahiptirler. Tek bir şirkete ait hisselerin değişkenliği, bunların birleşiminden oluşan borsa yatırım fonlarından daha da fazladır. Bu değişkenliği azaltmak adına, çalışmada borsa yatırım fonlarından olan SPDR S&P 500 Borsa Yatırım Fonu (SPY) kullanılmıştır.

Finansal veriyi analiz etmek için tüm yöntemler iki ana çatı altında birleşirler: temel analiz, teknik analiz. Temel analiz, şirketlerin söylemleri, yıllık kazançları gibi değişkenleri inceler. Bu tip değişkenlerden öznitelik çıkarmak için doğal dil işleme gibi farklı makine öğrenmesi alanlarına girilmelidir. Teknik analiz ise, finans verisindeki sayısal serilerle ilgilenir ve daha az alana özel bilgi gerektirir. Teknik analiz için incelenecek yöntemler, finansın zaman serisi verilerine sahip olmasından kaynaklı olarak, zaman serisi analizi altında toplanabilir, LSTM, CNN (belirli önişlemler yapılırsa) gibi mimarilerin kullanımına olanak tanır.

Çalışmadaki verisetini güçlendirmek için, Göreli Güç İndeksi (RSI), Basit Hareketli Ortalama (SMA), MACD Göstergesi, The Moving Average Convergence/Divergence

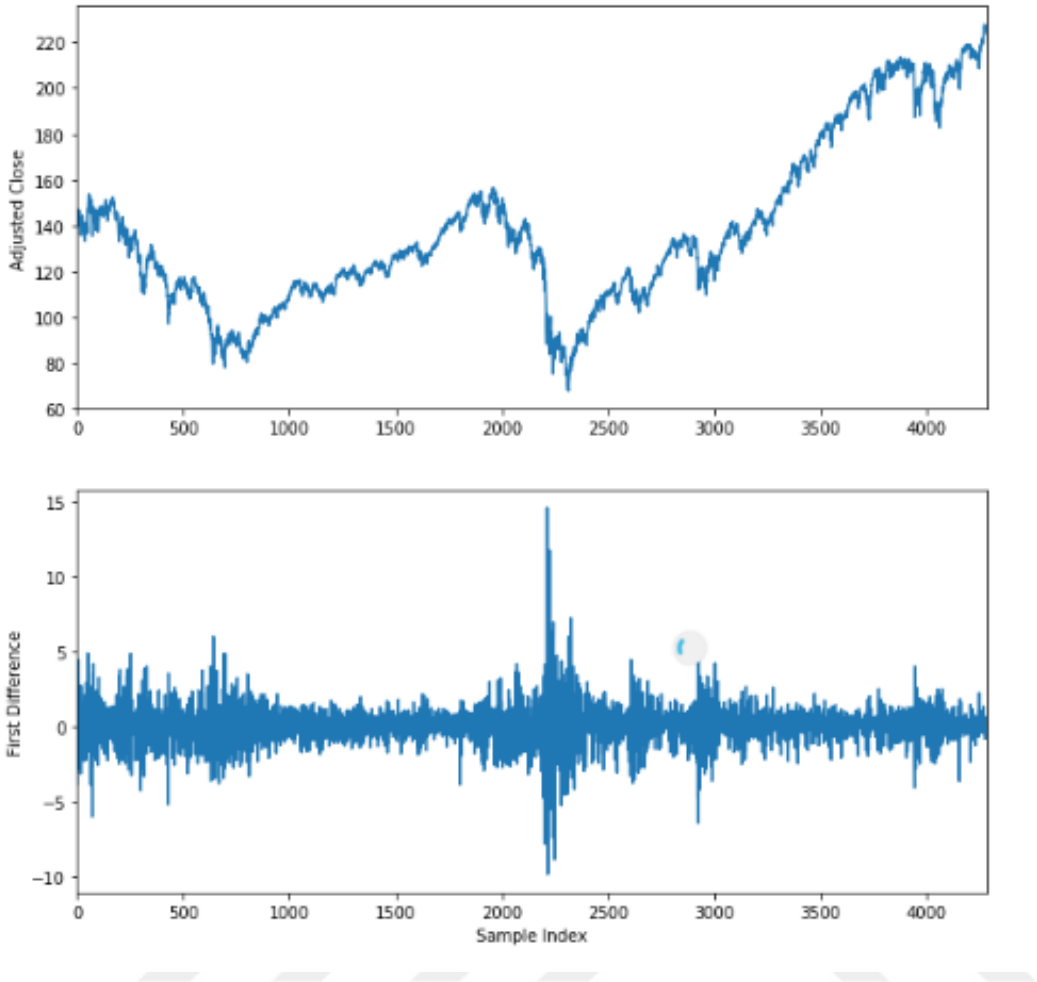
Çizelge 3.3: Öznitelikler

Öznitelik Adı	Parametreler	Miktar
tanh((stationary(close)))	-	1
hacim	-	1
RSI	15-20-25-30	4
SMA	15-20-25-30	4
MACD	26,12 – 28,14- 30,16	3
MACD T	9,26,12- 10,28,14- 11,30,16	3
Williams %R	14-18-22	3
SO	14-18-22	3
UO	7,14,28-8,16,22-9,18,36	3
MFI	14-18-22	3

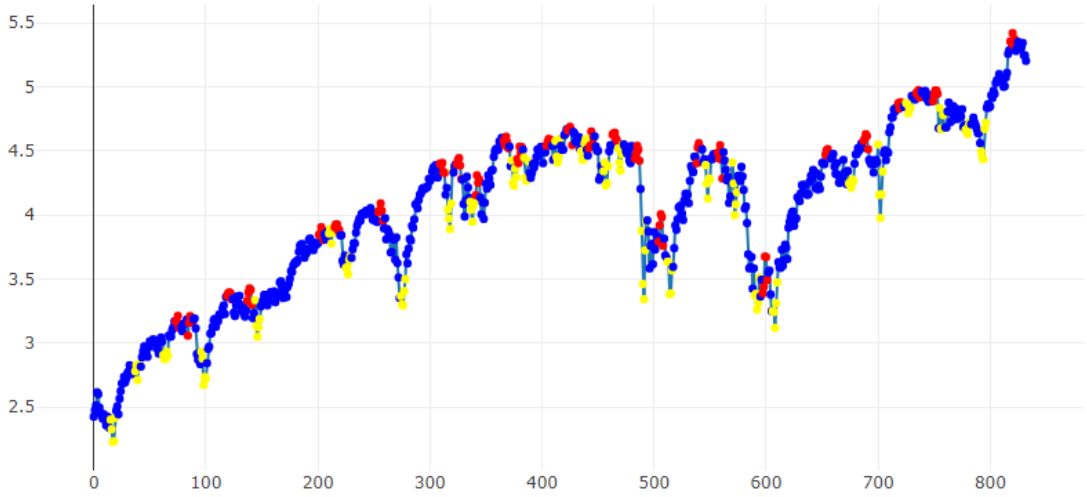
Oscillator (MACD), Williams Yüzdesi (Williams %R), Stokastik Osilatör (SO), Nihai Osilatör, The Ultimate Oscillator (UO), Para Akış indeksi (MFI) gibi çeşitli teknik analiz yöntemleri kullanılmıştır. Bu indikatörlerin ne ifade ettikleri ve çıkarmak için kullanılam matematiksel denklemler Çizelge 3.4'de görülmektedir.
Genellikle, finans verilerinde kullanılan fiyat verileri, marketin değeri ve hacmi arttığından, yükseliş gösterirler. Bu nedenledir ki, al-tut stratejisi hem temel ölçüt olarak kullanılır hem de geçmesi oldukça zordur. Sürekli yükseliş gösteren bir akıma sahip olmak, rastsal seçilen bir pencere aralığındaki fiyatların istatistiksel özelliklerinin, verinin her yerinde aynı olmamasına sebebiyet verir. İstatistiksel özelliklerin değişiklik göstermesi, bu değişkenlerin uygun bir şekilde tahmin modellerinde kullanılmasını engeller. İstatistiksel değişkenleri de sistemlere entegre edebilmek adına, ortalama, standard sapma gibi momentlerin sabitlenmesi gerekir. Bu sabitleme işlemine, *durağanlık* adı verilir. *Durağanlık* işlemleri oldukça kapsamlı bir şekilde literatürde incelenmiştir. Bu işlemlerden, en hızlı sonuç verebilecek ve uygulanması kolay olan *ilk-fark yöntemi*, verilere uygulanmıştır. Yani, fiyat verileri olduğu gibi kullanılmayıp, aralarındaki fiyat farkları veri setine işlendi. Şekil 3.12'de *durağanlık işlemi* uygulanmış ve uygulanmamış veri görülmektedir. Ek olarak, şekildeki -10 ile 15 arasında olan veriler, modelleri eğitirken gradyan patlaması yaşatmasın diye veri *tanh* fonskiyonundan geçirildi ve böylelikle $[-1, 1]$ aralığında bir veriseti oluşturuldu.

Çizelge 3.4: Teknik indikatörler, açıklamaları ve denklemleri

Öznitelik Adı	Açıklama	Formül
RSI	Güncel zaman dilimi için borsanın kuvvetini ve zayıflığını verir.	$RSI = 100 - \frac{100}{1+\frac{averagegain}{averageloss}}$
SMA	Verilen bir zaman dilimindeki ortalama kapanış fiyatıdır. Zaman içinde kayan pencelerin ortalamasıdır.	$SMA(M,n) = \sum_{k=a+1}^{a+n} \frac{M(k)}{n}$
EMA	SMA'ya benzer fakat son veriye daha fazla ağırlık verir.	$EMA(M,t,\tau) = (M(t) - EMA(M,t-1,\tau)).\frac{2}{\tau+1} + EMA(M,t-1,\tau)$
MACD	Seriler arasındaki ilişkiyi gösteren momentum göstergesidir.	$MACD(t) = (EMA(M,t,\tau_1) - EMA(M,t,\tau_2))$
MACD T	MACD ile başka tetikleme sinyali arasındaki ilişkiye bakar ve al-sat sinyali verir.	$MACDT(t) = MACD(t) - EMA(MACD,t,\tau_3)$
Williams %R	Borsadaki aşırı yükseliş ve düşüşleri ölçer.	$R = \frac{max(high) - close}{max(high) - min(low)} * -100$
SO	William %R gibi uç hatları belirler.	$K = \frac{close(t) - min(low(t))}{max(high(t)) - min(low(t))} * 100$ $D = SMA(K,3)$ $SO = K - D$
UO	Volatilite ve yanlış işlemleri azaltmak için kullanılır.	$BP = close(t) - min(low(t), close(t-1))$ $TR = max(high(t), close(t-1)) - min(low(t), close(t-1))$ $UO = 100\left(\frac{(\frac{\tau_3}{\tau_1}\sum_{i-\tau_1}^{i}\frac{BP}{TR}) + (\frac{\tau_2}{\tau_1}\sum_{i-\tau_2}^{i}\frac{BP}{TR}) + (\frac{\tau_1}{\tau_1}\sum_{i-\tau_3}^{i}\frac{BP}{TR})}{(\tau_1+\tau_2+\tau_3)}\right)$
MFI	Hacim verisini kullanan bir momentum göstergesidir. herhangi bir zaman dilimindeki para giriş ve çıkışlarını ölçer.	$TypicalPrice(t) = \frac{high(t)+low(t)+close(t)}{3}$ $RawMoneyFlow = TypicalPrice(t).Volume(t)$ $MFR(i) = \frac{(\sum_{i-\tau}^{i} PositiveMoneyFlow)}{(\sum_{i-\tau}^{i} NegativeMoneyFlow)}$ $MFI(i) = 100 - 100/(1+MFR(i))$



Şekil 3.12: İlk-fark alınarak yapılan durağanlaştırma işlemi



Şekil 3.13: Veri seti ve etiketleri

Yükşeliş ve düşüş etiketleri, *durağan* olmayan veri üzerinde 20'lik pencere (haftasonları veri içerisinde bulunmadığından 1 aylık veriyi içeren pencere) kaydıralarak belirlendi. Bu penceredeki en yüksek değere *sat* etiketi, en düşük değere *al* etiketi, diğer değerlere *tut* etiketi uygulandı. Daha sonra $x = [1,1,1]$ vektörü tüm veri üzerinde kaydırılırken konvolusyon operatörü uygulandı. Bu sayede, örneğin, *sat* etiketine sahip bir veri 5 birim yayılmış oldu. Bu işlem sayesinde, karmaşık yapıya sahip finans verisinden, gürültü bir miktar uzaklaştırılmış oldu (Şekil 3.13). Her ne kadar, konvolusyon işlemi, sınıflar arasındaki dengesizliği azaltmış olsa da, hala veri içerisinde sınıf düzensizliği bulunmaktadır. Eğitim sırasında bu durumun problem oluşturmaması için, modeli eğitmek için kullanılan veriler, özel bir örnekleyicinden geçirildiler. Bu örnekleyici, az bulunan sınıf etiketlerine ağırlık vererek, modele her seferinde eşit sayıda sınıftan örnek yollamaktadır.

3.5.2 Kümeleme

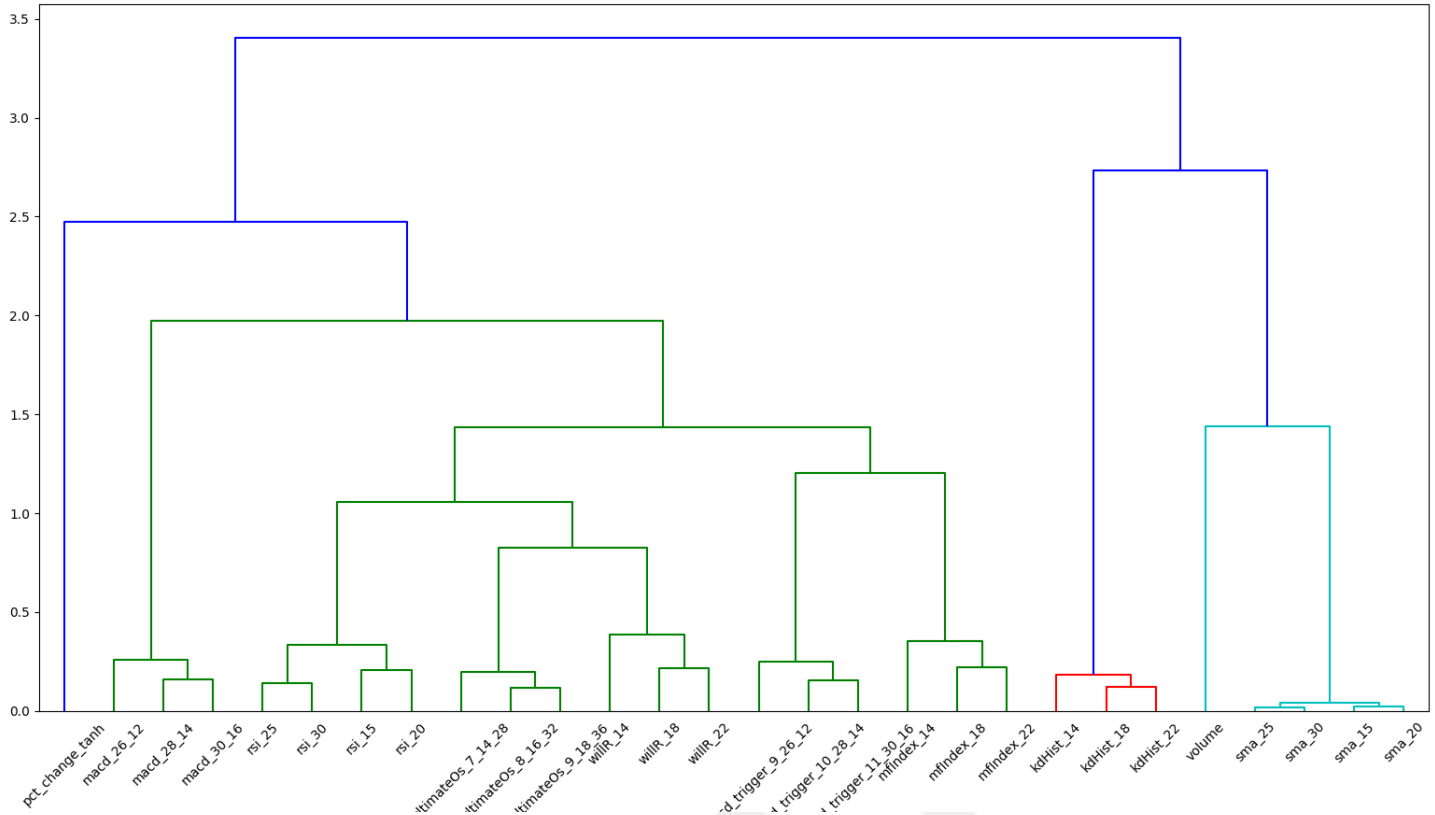
CNN modelinin resimlerle çok iyi olduğu bilinmektedir. Bunun sebebi daha önceden bahsedildiği gibi (2.10), CNN'in sahip olduğu konvolusyon operatöründen kaynaklıdır. Konvolusyon operatörü sayesinde resimlerdeki her bir pikselin komşu piksellerle ilişkisi sisteme dahil edilebilmektedir. Bu nedenle, CNN gibi model ile eğitim yapılacaksa, aynı resimlerde olduğu gibi, tüm boyutlarda korelasyon sağlanmalıdır. Zaman serisinin doğası gereği t zaman ekseni boyunca korelasyon zaten bulunmaktadır.Fakat, diğer boyutlar için ön-işlem yapılması gerekmektedir. Kullanılan özniteliklerin korelasyonlarına göre sıralanarak bu sorunun önüne geçilmeye çalışılmıştır. Öznitelik sıralaması için, bir kümeleme algoritmalarından hiyerarşili-kümeleme algoritmaları üyesi olan agglomerative kümeleme algoritmasından faydalanılmıştır. Bu algoritmanın çıkardığı *dendrogram*ın yapraklarına bakıldığında (3.14), öznitelikler yakınlıklarına göre sıralanmaktadırlar. Bahsedilen yakınlıktan kasıt, Pearson korelasyon sabiti (3.13) ve Öklid uzaklığı (3.14) kullanılarak hesaplanmış uzaklık matrisidir.

$$\rho_{x,y} = \frac{cov(X,Y)}{\sigma_x \sigma_x} \tag{3.13}$$

burada;

- cov: kovaryans
- σ_x: X'in standart sapması
- σ_y: Y'nin standart sapmasıdır

$$||a-b||_2 = \sqrt{\sum_a^b (a_i - b_i)^2} \tag{3.14}$$



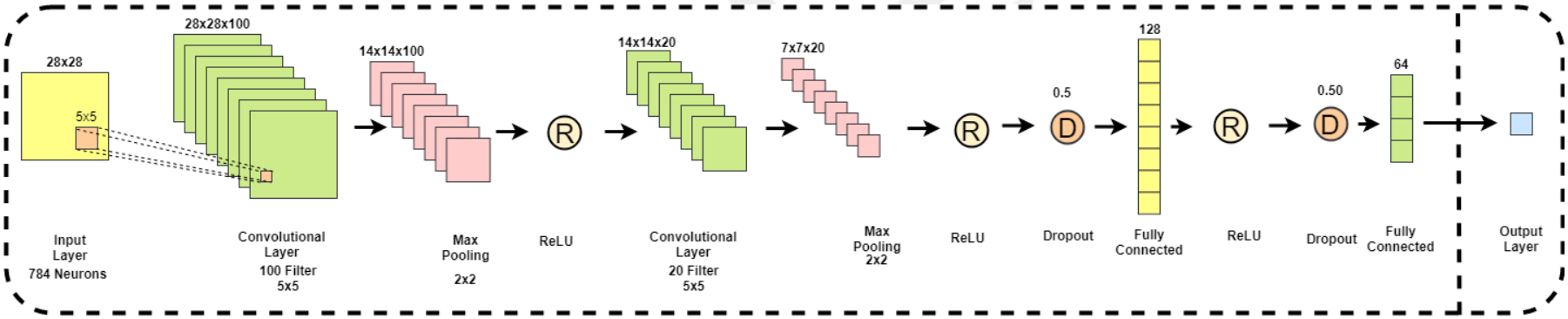
Şekil 3.14: Dendrogram

CNN modeli için, resim gibi 2D girdi matrisleri gerekmektedir. Çizelge 3.3'deki 28 adet öznitelik seçilip veri seti oluşturulduğu için ve kullanılan kütüphaneler kare matris istediği için 28'lik kayan pencere yaklaşımıyla veri 28×28 lik dendrogram ile sıralanmış matrislere bölünmüştür. Bu matrisler oluşturulurken herhangi stride parametresi kullanılmamıştır. Diğer bir deyişle, pencereler birbirlerin $[(i) : (i)+28]$, $[(i+1) : (i+1)+28]$ şeklinde takip etmektedirler.

3.5.3 Model

Çalışmada zaman serisi verileri CNN modeliyle öğrenilmeye çalışılmıştır. Şekil 3.15'de kullanılan model mimarisi görülmektedir. 28×28 boyutlarındaki girdi matrisi sırasıyla:

- $conv(5 \times 5)$, $maxpooling(2 \times 2)$, $ReLU$,
- $conv(5 \times 5)$, $maxpooling(2 \times 2)$, $ReLU$, $dropout(0.5)$,
- fc, $ReLU$, $dropout(0.50)$, fc katmanlarından geçerek 3-sınıf çıktısı vermektedir.
- optimizer = ADAM
- öğrenme faktörü = 0.001
- hata fonksiyonu = CE
- girdi boyutu = 1 resim
- çıktı boyutu = 3
- resim boyutu = 28x28x1
- yığın boyutu = 100
- tur sayısı = 1000



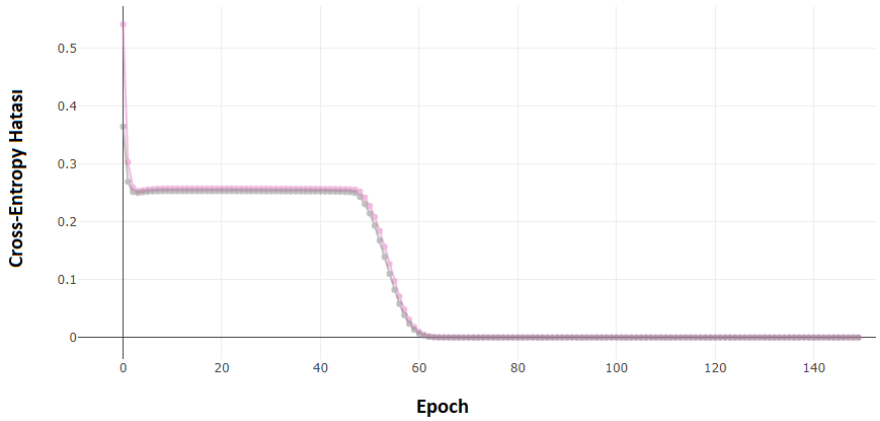
Şekil 3.15: Model

4. SONUÇLAR

4.1 LSTM Çalışması I: Uzun Vadeli Hafıza Problemi

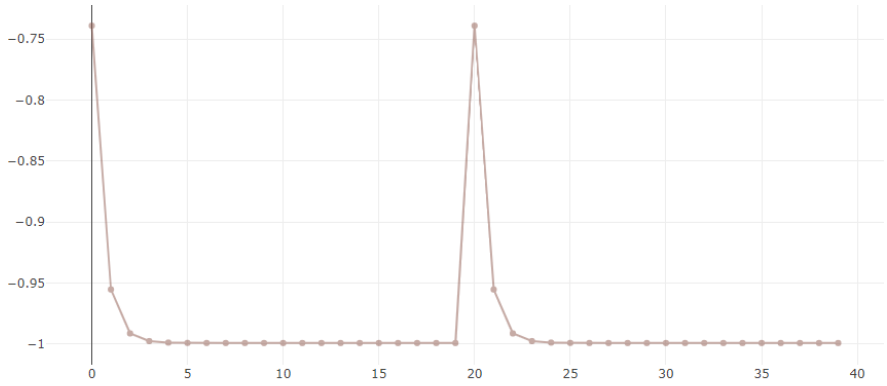
Bu kısımda uzun vadeli hafıza problemi çalışması ele alınacak, sistemin iç yapısı, tahmin ve optimizasyon sonuçları incelenecektir.

Şekil 4.1'de uzun vadeli hafıza probleminin öğrenme eğrisi görülmektedir. İlk bir kaç tur ile düşen hata, modelin girdi olarak ne gelirse gelsin çıktı olarak aynı sonucu vermesinden dolayı ortaya çıkmıştır. Model probleme en basit cevabı ($f(x) = x$) vererek 0.25 civarında bir hata çıkarmıştır. 50. tura gelene kadar hatada en ufak bir değişim gözlenmemesi modelin bu aşamalarda 20 adımlık bilgiyi yakalayamamasındandır. 50. turdan itibaren 0'a doğru azalış gösterdiği yer ise modelin 20 adımda bir verilen impulse'ları yakayabildiğinin göstergesidir. Hata 60. turdan sonra 0 olmuş ve problemin her örneğine doğru cevap verilmiştir.



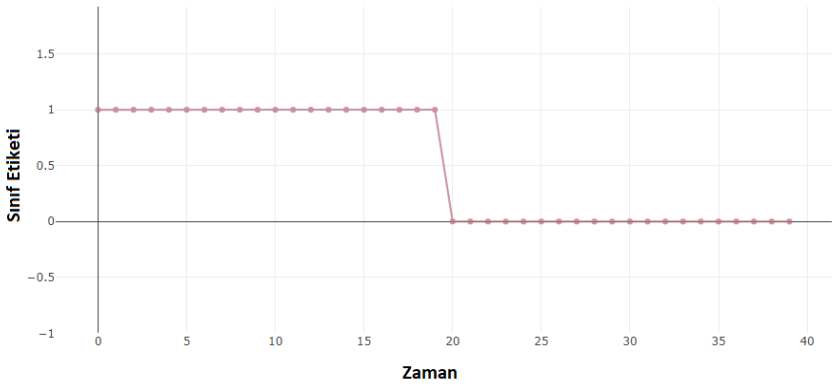
Şekil 4.1: Öğrenme eğrisi

Şekil 4.2'de kullanılan tek nöronlu LSTM mimarisinde gizli hali görülmektedir. Modelin $t \mod 20 == 1$ denklemi gibi çalışıp her 20 zaman adımında bir impulse verdiği görülmüştür. Bu impulse, incelenen basit problemin tek bir nöronla çözüldüğünün göstergesidir. Bu bilgi tek nöronluk FCNN'e aktarılarak, ağırlık(w) ile ölçeklenip, *bias*(b) ile kaydırılmış ve $0-1$ çıktısı elde edilmiştir. FCNN deki w ve b değeri ise sırasıyla -1.33 ve 0'dır. Kısaca, $w \cdot h_t + b$ işlemi yapılarak $(-1.33) \cdot (-0.75) + 0 = 1$ sonucu elde edilmiştir.



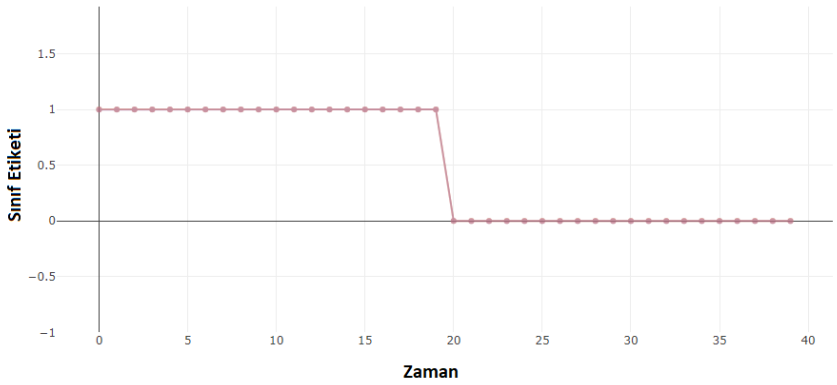
Şekil 4.2: LSTM hücresinin tepkisi

Şekil 4.3'de kullanılan modelin test veri setindeki (aslında böyle bir veri seti için eğitim, validasyon ya da test ayrımından söz edilemez) bir örneğe verdiği tahminler görülmektedir. $t = 0$ anında verilmiş olan impulse 20 zaman adımı boyunca çıktının 1 olmasına sebebiyet vermiştir. Aynı şekilde $t = 20$ anında impulse verilmediği için sonraki 20 zaman adımı boyunca çıktı 0 olmuştur.



Şekil 4.3: Tahmin eğrisi

Tahminlerin doğru olmasının dışında modeller için biraz daha problemli olan gelecek tahmini yaptırılmıştır. Şekil 4.4'da görüleceği gibi, impulse vererek başlayan sistem 20 adım boyunca 1 çıktısını verebilmiştir. Bu çalışmadaki gelecek tahmini her ne kadar doğru da olsa, problem fazla basit olduğundan büyük bir çıkarım yapmamızı engellemektedir. Bu durum 4.2 bölümünde daha detaylı anlatılacaktır.

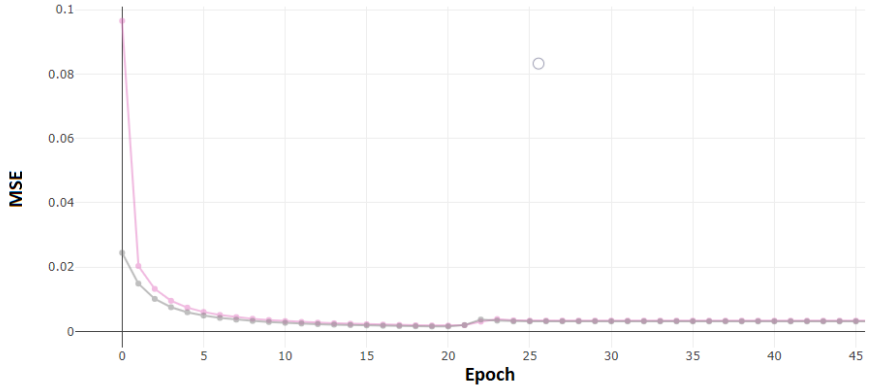


Şekil 4.4: Gelecek tahmini

Aynı problem stateless LSTM kullanılarak modellendiğinde, çıktı olarak 0 ya da 1 değerlerini rastgele verdiğinden dolayı, hata 0.5 değerinden daha iyi bir noktaya gelememektedir.

4.2 LSTM Çalışması II: Sinüs Dalgası

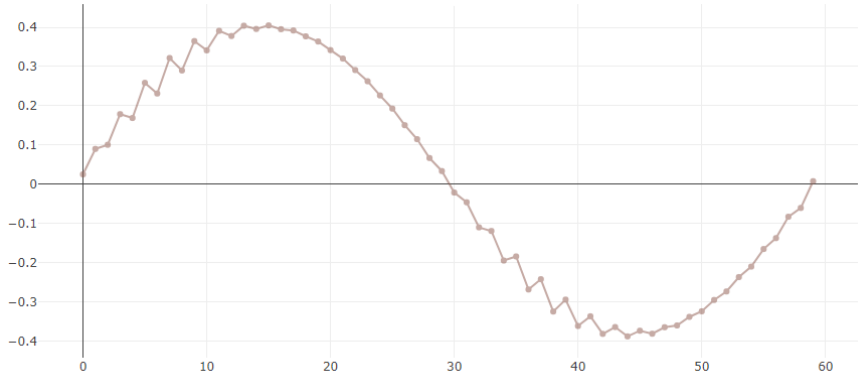
Bu kısımda sinüs fonksiyonunu öğrenmeye çalışan LSTM modelinin sonuçları ve iç yapısı incelenecektir.
Şekil 4.5'de sinüs probleminin öğrenme eğrisi MSE 2.3.1 cinsinden görülmektedir. Bu şekilde, modelin 1-2 *epoch* içerisinde sinüsü tamamiyle öğrendiği görülmektedir.



Şekil 4.5: Öğrenme eğrisi

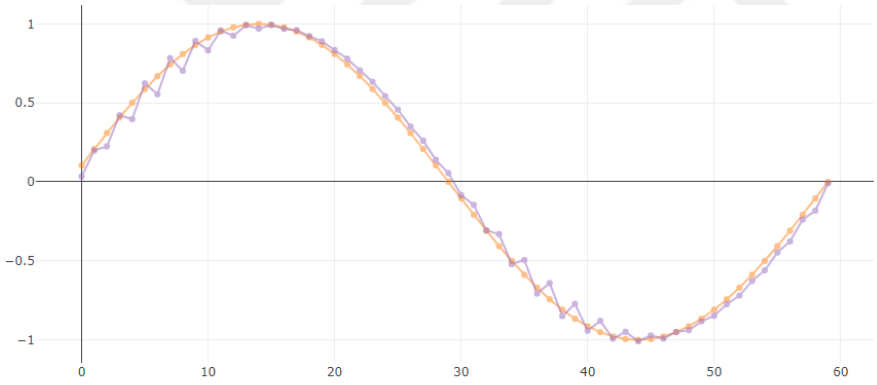
Şekil 4.6'de, kurulan modeldeki LSTM nöronunun, verilen girdilere tepkisi görülmektedir. Kendi başına sinüs yapısını tam anlamıyla çıkardığı görülmektedir (belirli yerlerdeki dalgalanmalar hariç). Bu nöronun çıktısı FCNN nöronuna verilip belirli ağırlıklar(w) ile ölçeklenip, *bias*(b) ile kaydırılarak bir sonraki adımdaki sinüs değeri tahmin edilmiştir. FCNN'deki w ve b değerleri ise sırasıyla 2.52 ve 0.0'dır. Sinüs fonksiyonunun 15.

zaman adımı için bir hesap yapılacak olursa, FCNN nöronunda $w \cdot h_t + b$ işlemi yapılmış $(2.52) \cdot (0.4) + 0$ ve 1 sonucu elde edilmiştir.



Şekil 4.6: LSTM iç durumları

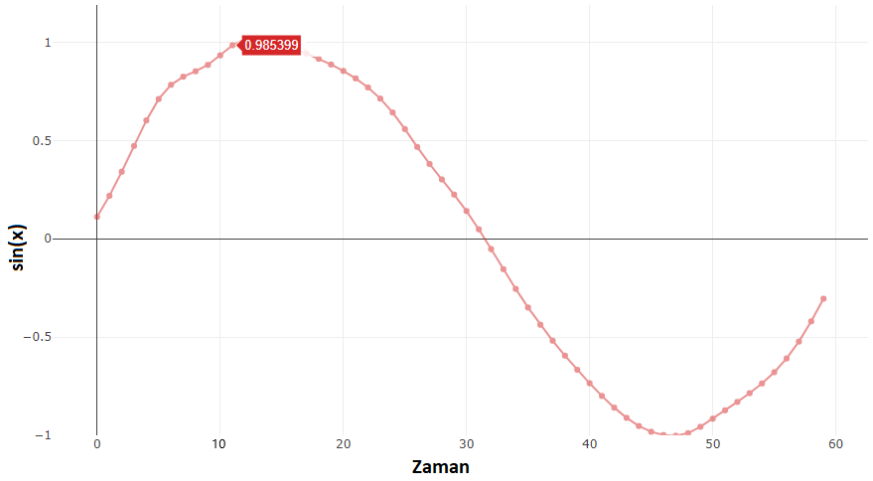
Şekil 4.7'de LSTM modelinin veri setinden alınan bir periyotluk örneğin tahmini görülmektedir. Tahmin küçük salınımlar (kullanılan LSTM nöronundan kaynaklı salınımlar) yapsa da sinüsün periyodunu ve büyüklüğünü birebir öğrenmiştir.



Şekil 4.7: Tahmin eğrisi

Modellerin, gelecek tahmini yapması daha zordur. Veri setinden her bir zaman dilimi için yalnızca tek bir değer çekildiği için, modelin, girdiler arasındaki farka bakması gibi bir durum da söz konusu olmamaktadır. Tek öğrenme yolu kendi içerisindeki gizli hal ve hücre hali ile yeni gelen girdiyi harmanlamaktır. Fakat gelecek tahmini yaparken peş peşe gelen girdiler sonucu açığa çıkan küçük hatalar, kümülatif olarak birikir ve sonraki adımların tahminini imkansız kılabilir. Şekil 4.8'de görüldüğü gibi, model genel olarak 60 zaman adımında periyodu tamamlayabilmiş ve sinüse benzer bir yapı ortaya çıkarmış olsa da, tepe ve dip noktalarındaki geçişleri normalden daha sert yapmıştır. Modelden daha uzun periyotlarda tahmin yapması istenirse, hatalar çok daha fazla büyüyerek, osilasyona girmiş ve sönümlenmiş bir sinyal oluşturacak ve bu sinyalde de

bilginin kaybolduğu gözlemlenebilecektir. Fakat, sıfırlama periyodu olarak belirlenen 60 sayısı, kümülatif biriken hatayı da belirli noktada sıfırlayarak istenilen uzunlukta sinüs oluşumuna izin vermektedir.



Şekil 4.8: Gelecek tahmini

Aynı problem durum-denetlemesiz LSTM ile modellendiğinde, model, tıpkı FCNN ya da CNN gibi modellerde olacağı gibi, geçmiş bilgisi model içerisinde saklanamayacağı için, 0 ile 1 arasında rastsal bir sayı çıktısı verecektir. Bundan dolayı, hata hiç bir zaman düşmeyecek ve öğrenme gerçekleşmeyecektir.

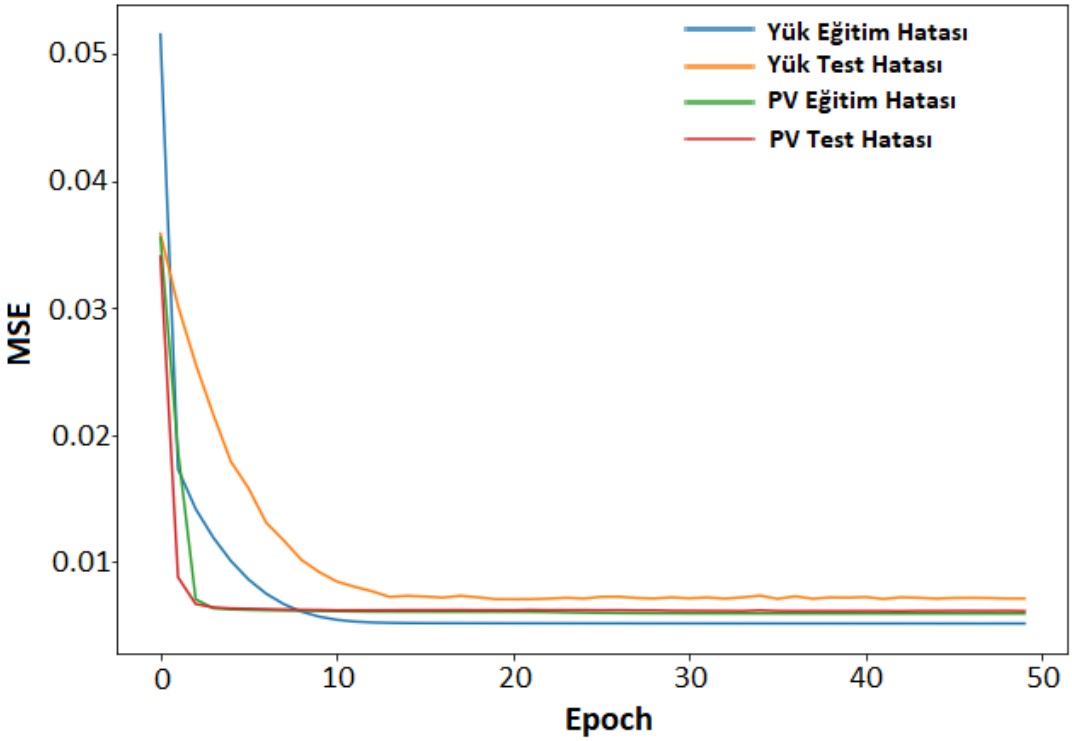
4.3 Enerji Üretim ve Tüketim Tahmini Çalışması

Bu kısımda, Enerji Üretim ve Tüketim Tahmini çalışmasının tahmin ve optimizasyon sonuçları ele alınacaktır.

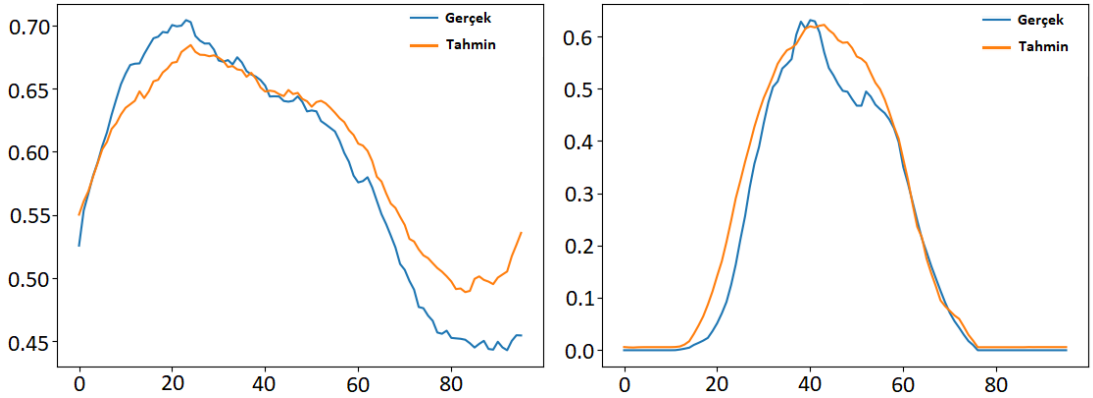
Şekil 4.9'de hane tüketim ve solar enerji üretim tahminin eğitim süresi boyunca MSE görülmektedir. Model 400 epoch çalıştırılmış olsa da 50 epochtan sonra hata yok sayılacak derecede az düştüğünden dolayı, şekilde sadece ilk 50 epoch sonucu gösterilmiştir. Eğitim süreci sona erdikten sonraki, eğitim ve validasyon sonuçları, Çizelge 4.1'de gösterilmiş ve buna göre $7.19e-3$ tüketim MSE'si, $6.03e-3$ üretim MSE'si elde edilmiştir. Şekil 4.10'de eğitim aşamasında test verisinden örneklenmiş birer tahmin örneği gösterilmiştir. Bu grafiğe göre, tahmin modellerinin gerçek değerlere çok yakınsadığı fakat her değişimi yakalayamadığı görülmüştür.

Çizelge 4.1: Eğitim ve test hataları

Hata	Yük Eğitim	Yük Test	PV Eğitim	PV Test
MSE	5.0877e-3	7.1906e-3	5.8255e-3	6.0386e-3
MAE	7.1328e-2	8.47e-2	7.63e-2	7.77e-2



Şekil 4.9: Öğrenme eğrisi



(a) Yük örneği
Eğitim hatası:0.00509
Test hatası:0.00718
Epoch:303

(b) PV örneği
Eğitim hatası:0.00583
Test hatası:0.00607
Epoch:301

Şekil 4.10: Yük ve PV tahmin örnekleri

MILP optimizasyon işleminden sonraki sonuçlar, Çizelge 4.2'de gösterilmiş ve bu tablonun ne ifade ettiğinin açıklaması aşağıda verilmiştir:

- **Tarife:** Ülke fiyat tarifesi
- **Karar Stratejisi:** Kullanılan optimizasyon stratejisi
- **Sistem:** nano-şebeke sistem türü
- **Tahmin Edilen Maliyet:** tahmin edilen değer masrafı
- **Gerçek Maliyet:** gerçek hayata uygulanan sistemin tahmin masrafı
- **Optimum Maliyet:** %100 doğruluk ile tahmin yapıldığındaki masraf
- **Gerçek Kazanç:** tahmin edilen değerler ile yapılan kazanç
- **Kazanç Oranı:** $\frac{\text{Gerçek Kazanç}}{\text{Optimum Masraf}}$
- **Optimum Kazanç:** %100 doğruluk ile tahmin yapıldığındaki kazanç
- **Optimum Oranı:** $\frac{\text{Gerçek Kazanç}}{\text{Optimum Kazanç}}$

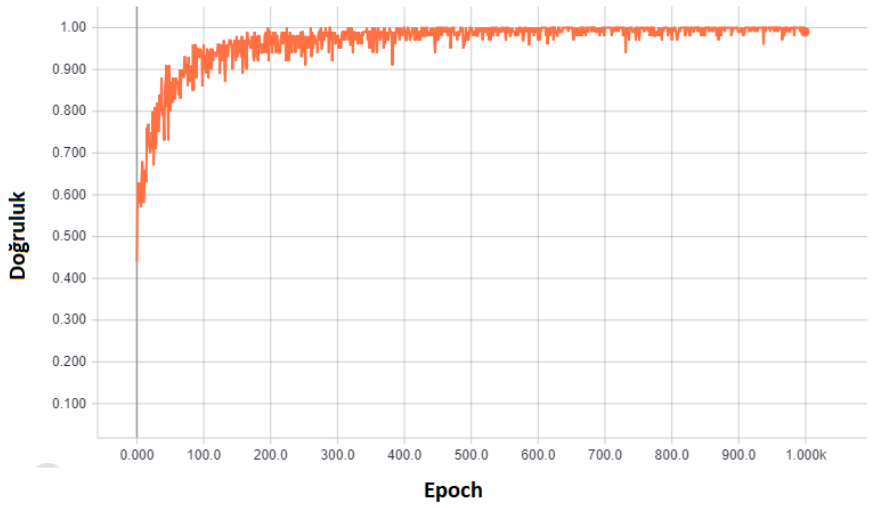
Optimizasyon sonuçları (Çizelge 4.2), PV ve batarya sistemi kullanmanın, haneye ekonomik açıdan büyük destek olacağını göstermektedir. Mutlak değerler ile yapılan optimizasyon işlemi %100 doğru olarak alındığında, tahmin değerleriyle yapılan optimizasyonun bu sonuçlara $>$ %99 oranında yakınsadığı görülmektedir. Çizelgeden çıkarılacak bir diğer sonuç ise, tek-zamanlı optimizasyon çok-zamanlı optimizasyona göre %0.12 oranında daha iyi sonuç vermiş olsa da, tek-zamanlı optimizasyonun 96 kat daha fazla çalıştırıldığı düşünüldüğünde, çok-zamanlı optimizasyonun daha uygun bir seçim olacağı görülmektedir.

Çizelge 4.2: Optimizasyon sonuçları

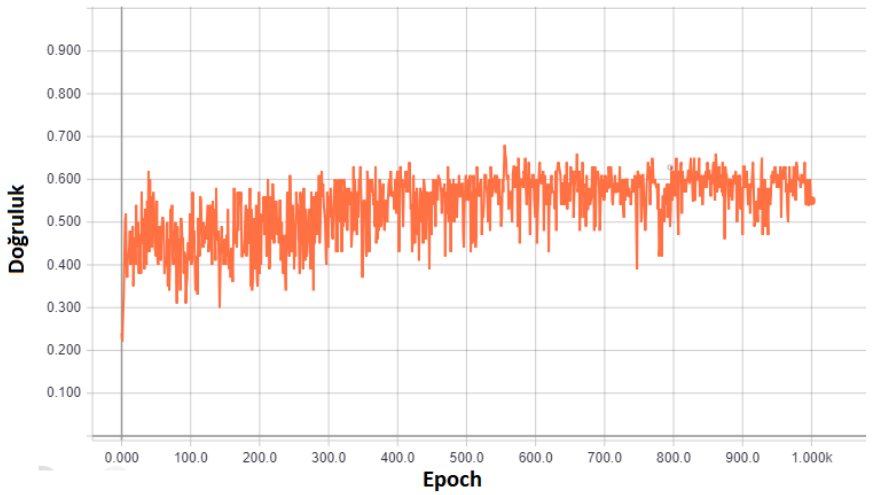
Tarife	**Karar Str.**	**Sistem**	**Tahmin Edilen Maliyet**	**Gerçek Maliyet**	**Opt. Maliyet**	**Gerçek Kazanç**	**Kazanç Oranı (%)**	**Opt. Kazanç**	**Opt. Oranı (%)**
DE	Çok Zamanlı	Normal	912.73	912.73	912.73	-	-	-	-
DE	Çok Zamanlı	Batarya	865.25	912.73	912.73	0.00	0.00	0.00	-
DE	Çok Zamanlı	Batarya ve PV	241.42	297.83	294.48	614.90	67.37	618.25	99.46
DE	Tek Zamanlı	Normal	912.73	912.73	912.73	-	-	-	-
DE	Tek Zamanlı	Batarya	868.71	912.73	912.73	0.00	0.00	0.00	-
DE	Tek Zamanlı	Batarya ve PV	243.84	297.08	294.48	615.65	67.45	618.25	99.58
TR	Çok Zamanlı	Normal	324.88	324.88	324.88	-	-	-	-
TR	Çok Zamanlı	Batarya	186.83	205.60	204.77	119.27	36.71	120.10	99.31
TR	Çok Zamanlı	Batarya ve PV	-31.57	-8.59	-10.10	333.47	102.65	334.98	99.55
TR	Tek Zamanlı	Normal	324.88	324.88	324.88	-	-	-	-
TR	Tek Zamanlı	Batarya	188.38	205.74	204.77	119.14	36.67	120.10	99.20
TR	Tek Zamanlı	Batarya ve PV	-30.23	-8.78	-10.10	333.66	102.70	334.98	99.61

4.4 CNN ve Finans Verileri Çalışması

Bu kısımda, CNN ve finans verileri çalışmasının tahmin sonuçları ele alınacaktır. Şekil 4.11'de finans tahmini probleminin, eğitim süresi boyunca doğruluk grafikleri görülmektedir. Eğitim veri setinde doğruluk hata yapılmayan bir seviyeye kadar yükselmiş olsa da, test veri setinde durum biraz daha farklıdır. Buradaki doğruluk değeri, çok fazla salınım yaparak gürültülü bir grafik ortaya çıkarmıştır. Ama üzerindeki eğilime bakılacak olursa, 0.3 değerlerinde başlayan süreç, 0.55 seviyelerinde son bulmuştur. Hatta bazen, 0.6 eşiğinin üstüne çıktığı görülmüştür. 3 sınıflı bir problem incelendiği için, rastgele tahminler yapıldığında ortaya çıkacak olan 0.33 değerli doğruluk oranı şekilde ayrıca gösterilmiştir.



Şekil 4.11: Eğitim doğruluk grafiği



Şekil 4.12: Test doğruluk grafiği

1000 tur sonrasında test veri seti üzerinden Çizelge 4.3'de görülen karmaşıklık matrisi çıkarılmıştır. Bu karmaşıklık matrisinin diyagonal kısmına bakılırsa, doğru tahminlerin daha fazla olduğu görülmektedir. *al* ya da *sat* etiketi yerine *tut* etiketine daha fazla ağırlık vermiştir fakat, bu etiketler yerine *tut* etiketinin tahmin edilmesi finans problemi açısından büyük bir sorun teşkil etmemektedir. *al* etiketi için $8 > 3$ ve *sat* etiketi için $8 > 4$ sonuçları bu çalışma için yeterlidir.

Çizelge 4.3: Karmaşıklık matrisi

		gerçek		
		al	**tut**	**sat**
	al	8	7	4
tahmin	**tut**	22	36	9
	sat	3	3	8

5. DEĞERLENDİRME

Bu tez kapsamında, zaman serisi verileri için analiz yöntemleri ve derin öğrenme modelleri önerilmiştir. Finans ve enerji verileriyle yapılan çalışmalar ile farklı modellerin çeşitli dönüşümler yapıldığında kullanılabileceği gösterilmiştir.

Tez kapsamında, ilk olarak, LSTM'in uzun vadeli hafıza kapasitesini gösteren bir deney düzeneği kurulmuştur. Bu deney içerisinde gizli bilgileri sürekli sıfırlanmayan yani durum-denetlemeli LSTM gibi hafızaya sahip modellerin çözebileceği bir veri seti oluşturulmuştur. Bu veri seti ve durum-denetlemeli LSTM kullanımı ile modelin iç yapısı incelenmiş, sorunsuz bir şekilde öğrenme gerçekleştirilmiştir. İkinci deney düzeneği yine aynı mimari üzerine kurulmuş fakat veri seti olarak 60 birim periyotluk bir sinüs dalgası kullanılmıştır. Modelin tek bir LSTM hücresi ile problemi çözerkenki iç yapısı incelenmiştir. Bu oluşturulan model ile sinüs dalgasının başlangıç girdisi verildiğinde tüm sinüsün oluşturabildiği görülmüştür.

İkinci olarak, durum-denetlemesiz LSTM mimarisi ile birlikte çalışılacak olan enerji veri setine yer verilmiştir. Bu problemde, bir hanenin solar enerji üretimi ve elektrik tüketimi tahmini yapılarak, evde bulunan bataryanın optimizasyonu önerilmiştir. Tahmin yapıldığından MSE ya da MAE gibi hataları açıklamanın ötesinde, optimizasyon problemi çözülerek, gerçek değerlere ne kadar yaklaşıldığı incelenmiştir. Sonuçlardan görüleceği gibi $> \%99$ oranında gerçek sonuçlara yaklaşılarak hem tahminlerin hem de optimizasyon çözümünün başarımı gösterilmiştir.

Üçüncü olarak, CNN modeli ile zaman serisi tahmini önerilmiştir. LSTM deneylerinde olduğu gibi veri için sadece normalizasyon ya da standardizasyon gibi bir ön işlem yeterli olmadığından, çeşitli dönüşümler incelenmiştir. Zaman serisi şeklinde olan finans verisi ve teknik indikatörler, aralarındaki korelasyona göre dendrogram yardımı ile sıralanmışlardır. Bu sayede resimlerde olduğu gibi her iki eksendeki korelasyon sağlanmış ve CNN modeli eğitilebilmiştir. İncelenen sonuçlar ve karmaşıklık matrisi ışığında, CNN modelinin, veri seti üzerinde gerekli dönüşümler yapıldığında zaman serisi verileri için de kullanılabileceğini göstermiştir.

Sonuç olarak, tez kapsamında 3 farklı deney düzeneği kurulmuş ve çeşitli mimariler incelenmiştir. Yapılan çalışma ve öneriler, yenilikçi ve farklı yaklaşımlara sahiptir. Finans verilerine uygulanan dönüşümler, farklı alanlardaki zaman serilerine de uygulanarak,

LSTM dışındaki derin öğrenme modellerinin önünü açabilir. LSTM modeli ve çeşitleri de, çözülmeye çalışılan problem incelenip anlaşıldıktan sonra kullanılarak daha iyi tahmin sonuçları elde edilebilir.

5.1 Gelecekteki Çalışmalar

Tez kapsamında önerilen çalışmaların performansları gelecek çalışmalar ile daha da iyileştirilebilir. Bu çalışmaları şu şekilde sıralayabiliriz:

- Önerilen modeller kendileriyle ya da başka modellerle birleştirilerek performans artışına gidilebilir.
- Finans verisi ile kullanılan CNN için alım-satım modeli geliştirilerek, gerçek başarısı ölçülebilir.
- Derin öğrenme modellerinin iç yapıları, verilen girdiye göre aktive olan nöronların takibi yapılarak daha derin bir çıkarım yapılabilir.

KAYNAKLAR

[1] **Wei**, **W. W.** (2006). Time series analysis. In: *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2*.

[2] **Samarawickrama**, **A.** and **Fernando**, **T.** (2017). A recurrent neural network approach in predicting daily stock prices an application to the Sri Lankan stock market. In: *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*. IEEE.

[3] **Sezer**, **O. B.** and **Ozbayoglu**, **A. M.** (2018). Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. In: *Applied Soft Computing* 70, pp. 525–538.

[4] **Mourelatos**, **M.** et al. (2018). Financial Indices Modelling and Trading utilizing Deep Learning Techniques: The ATHENS SE FTSE/ASE Large Cap Use Case. In: *2018 Innovations in Intelligent Systems and Applications (INISTA)*. IEEE.

[5] **Shen**, **G.** et al. (2018). Deep Learning with Gated Recurrent Unit Networks for Financial Sequence Predictions. In: *Procedia Computer Science* 131, pp. 895–903.

[6] **Hosaka**, **T.** (2018). Bankruptcy prediction using imaged financial ratios and convolutional neural networks. In: *Expert Systems with Applications*.

[7] **Wang**, **Q.**, **Xu**, **W.**, and **Zheng**, **H.** (2018). Combining the wisdom of crowds and technical analysis for financial market prediction using deep random subspace ensembles. In: *Neurocomputing* 299, pp. 51–61.

[8] **Dang**, **L. M.** et al. (2018). Deep Learning Approach for Short-Term Stock Trends Prediction based on Two-stream Gated Recurrent Unit Network. In: *IEEE Access* 6, pp. 1–1.

[9] **Abe**, **M.** and **Nakayama**, **H.** (2018). Deep Learning for Forecasting Stock Returns in the Cross-Section. In: *Advances in Knowledge Discovery and Data Mining*. Springer International Publishing, pp. 273–284.

[10] **Yümlü**, **S.**, **Gürgen**, **F. S.**, and **Okay**, **N.** (2005). A comparison of global, recurrent and smoothed-piecewise neural models for Istanbul stock exchange (ISE) prediction. In: *Pattern Recognition Letters* 26.13, pp. 2093–2103.

[11] **Lachiheb**, **O.** and **Gouider**, **M. S.** (2018). A hierarchical Deep neural network design for stock returns prediction. In: *Procedia Computer Science* 126, pp. 264–272.

[12] **Yong**, **B. X.**, **Rahim**, **M. R. A.**, and **Abdullah**, **A. S.** (2017). A Stock Market Trading System Using Deep Neural Network. In: *Communications in Computer and Information Science*. Springer Singapore, pp. 356–364.

[13] **Das**, **S.**, **Mokashi**, **K.**, and **Culkin**, **R.** (2018). Are Markets Truly Efficient? Experiments Using Deep Learning Algorithms for Market Movement Prediction. In: *Algorithms* 11.9, p. 138.

[14] **Feng**, **G.**, **He**, **J.**, and **Polson**, **N. G.** (2018). Deep Learning for Predicting Asset Returns. eprint: `arXiv:1804.09314`.

[15] **Möws**, **B.** (n.d.). Deep Learning for Stock Market Prediction: Exploiting Time-Shifted Correlations of Stock Price Gradients. In:

[16] **Chandra**, **R.** and **Chand**, **S.** (2016). Evaluation of co-evolutionary neural network architectures for time series prediction with mobile application in finance. In: *Applied Soft Computing* 49, pp. 462–473.

[17] **Navon**, **A.** and **Keller**, **Y.** (2017). Financial Time Series Prediction Using Deep Learning. eprint: `arXiv:1711.04174`.

[18] **Vargas**, **M. R.**, **Lima**, **B. S. L. P. de**, and **Evsukoff**, **A. G.** (2017). Deep learning for stock market prediction from financial news articles. In: *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*. IEEE.

[19] **Gudelek**, **M. U.**, **Boluk**, **S. A.**, and **Ozbayoglu**, **A. M.** (2017). A deep learning based stock trading model with 2-D CNN trend detection. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE.

[20] **Kraus**, **M.** and **Feuerriegel**, **S.** (2017). Decision support from financial disclosures with deep neural networks and transfer learning. In: *Decision Support Systems* 104, pp. 38–48.

[21] **Huynh**, **H. D.**, **Dang**, **L. M.**, and **Duong**, **D.** (2017). A New Model for Stock Price Movements Prediction Using Deep Neural Network. In: *Proceedings of the Eighth International Symposium on Information and Communication Technology - SoICT 2017*. ACM Press.

[22] **Terna**, **P.**, **D'Acunto**, **G.**, and **Caselle**, **M.** (n.d.). A Deep Learning Model to Forecast Financial Time-Series. In:

[23] **Li**, **Z.** and **Tam**, **V.** (2017). Combining the real-time wavelet denoising and long-short-term-memory neural network for predicting stock indexes. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE.

[24] **Fischer**, **T.** and **Krauss**, **C.** (2018). Deep learning with long short-term memory networks for financial market predictions. In: *European Journal of Operational Research* 270.2, pp. 654–669.

[25] **Verma**, **I.**, **Dey**, **L.**, and **Meisheri**, **H.** (2017). Detecting, quantifying and accessing impact of news events on Indian stock indices. In: *Proceedings of the International Conference on Web Intelligence - WI 17*. ACM Press.

[26] **Althelaya**, **K. A.**, **El-Alfy**, **E.-S. M.**, and **Mohammed**, **S.** (2018). Evaluation of bidirectional LSTM for short-and long-term stock market prediction. In: *2018 9^{th} International Conference on Information and Communication Systems (ICICS)*. IEEE.

[27] **Yan**, **H.** and **Ouyang**, **H.** (2017). Financial Time Series Prediction Based on Deep Learning. In: *Wireless Personal Communications* 102.2, pp. 683–700.

[28] **Fentis**, **A.** et al. (2016). Short-term PV power forecasting using Support Vector Regression and local monitoring data. In: *2016 International Renewable and Sustainable Energy Conference (IRSEC)*. IEEE, pp. 1092–1097.

[29] **Alfadda**, **A.** et al. (2017). Hour-ahead solar PV power forecasting using SVR based approach. In: *2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, pp. 1–5.

[30] **Yang**, **M.** et al. (2016). Parameters Optimization Improvement of SVM on Load Forecasting. In: *2016 8^{th} International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*. IEEE, pp. 257–260.

[31] **Ahmad**, **A.** et al. (2017). An Accurate and Fast Converging Short-Term Load Forecasting Model for Industrial Applications in a Smart Grid. In: *IEEE Transactions on Industrial Informatics* 13.5, pp. 2587–2596.

[32] **Pal**, **K. K.** and **Sudeep**, **K. S.** (2016). Preprocessing for image classification by convolutional neural networks. In: *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pp. 1778–1781.

[33] **LeCun**, **Y. A.** et al. (2012). Efficient BackProp. Ed. by **Montavon**, **G.**, **Orr**, **G. B.**, and **Müller**, **K.-R.** Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 9–48.

[34] **Robbins**, **H.** and **Monro**, **S.** (1951). A Stochastic Approximation Method. In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407.

[35] **Darken**, **C.**, **Chang**, **J.**, and **Moody**, **J.** (1992). Learning rate schedules for faster stochastic gradient search. In: *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop*. IEEE, pp. 3–12.

[36] **Pan**, **H.** and **Jiang**, **H.** (2015). Annealed Gradient Descent for Deep Learning. In: *Proceedings of the Thirty-First Conference on Uncertainty in*

Artificial Intelligence. UAI'15. Amsterdam, Netherlands: AUAI Press, pp. 652–661.

[37] **Polyak**, **B.** (1964). Some methods of speeding up the convergence of iteration methods. In: *USSR Computational Mathematics and Mathematical Physics* 4.5, pp. 1–17.

[38] **Nesterov**, **Y. E.** (1983). A method for solving the convex programming problem with convergence rate $O(1/k^2)$). In: *Dokl. akad. nauk Sssr*. Vol. 269, pp. 543–547.

[39] **Duchi**, **J.**, **Hazan**, **E.**, and **Singer**, **Y.** (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. In: *J. Mach. Learn. Res.* 12, pp. 2121–2159.

[40] **Zeiler**, **M. D.** (2012). ADADELTA: An Adaptive Learning Rate Method. In: *CoRR* abs/1212.5701. arXiv: 1212.5701.

[41] **Tieleman**, **T.** and **Hinton**, **G.** (2012). rmsprop: Divide the gradient by a running average of its recent magnitude. In: *Lecture 6.5*.

[42] **Graves**, **A.** (2013). Generating Sequences With Recurrent Neural Networks. In: *CoRR* abs/1308.0850. arXiv: 1308.0850.

[43] **Kingma**, **D. P.** and **Ba**, **J.** (2014). Adam: A Method for Stochastic Optimization. In: *CoRR* abs/1412.6980. arXiv: 1412.6980.

[44] **Pitts**, **W.** (1942). Some observations on the simple neuron circuit. In: *The bulletin of mathematical biophysics* 4.3, pp. 121–129.

[45] **McCulloch**, **W. S.** and **Pitts**, **W.** (1943). A logical calculus of the ideas immanent in nervous activity. In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

[46] **Rosenblatt**, **F.** (1961). *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Tech. rep. CORNELL AERONAUTICAL LAB INC BUFFALO NY.

[47] **Hebb**, **D. O.** (1949). The organization of behavior: A neuropsychological theory. Wiley, New York.

[48] **Rumelhart**, **D. E.**, **Hinton**, **G. E.**, and **Williams**, **R. J.** (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1. In: ed. by **Rumelhart**, **D. E.**, **McClelland**, **J. L.**, and **PDP Research Group**, **C.** Cambridge, MA, USA: MIT Press. Chap. Learning Internal Representations by Error Propagation, pp. 318–362.

[49] **Olah**, **C.** (2015). Understanding LSTM Networks.

[50] **Hochreiter**, **S.** and **Schmidhuber**, **J.** (1997). Long Short-Term Memory. In: *Neural Comput.* 9.8, pp. 1735–1780.

[51] **Krizhevsky**, **A.**, **Sutskever**, **I.**, and **Hinton**, **G. E.** (2012). ImageNet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems 25*. Ed. by **Pereira**, **F.** et al. Curran Associates, Inc., pp. 1097–1105.

[52] **Fukushima**, **K.** (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. In: *Biological Cybernetics* 36.4, pp. 193–202.

[53] **Waibel**, **A.** et al. (1989). Phoneme recognition using time-delay neural networks. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3, pp. 328–339.

[54] **LeCun**, **Y.** et al. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. In: *Neural Computation* 1.4, pp. 541–551.

[55] **Springenberg**, **J. T.** et al. (2014). Striving for Simplicity: The All Convolutional Net. In: *CoRR* abs/1412.6806. arXiv: 1412.6806.

[56] **Gudelek**, **M. U.** et al. (2018). Load and PV Generation Forecast Based Cost Optimization for Nanogrids with PV and Battery. In: *2018 53rd International Universities Power Engineering Conference (UPEC)*, pp. 1–6.

ÖZGEÇMİŞ

Ad-Soyad : Mehmet Uğur Güdelek

Uyruğu : T.C.

Doğum Tarihi ve Yeri : 06.06.1991 Antakya

E-posta : ugurgudelek@gmail.com

ÖĞRENİM DURUMU:

- **Yüksek Lisans** : 2019, TOBB ETÜ, Bilgisayar Müh.
- **Lisans** : 2015, ODTÜ, Elektrik-Elektronik Müh.

MESLEKİ DENEYİM VE ÖDÜLLER:

Yıl	**Yer**	**Görev**
2016 - Halen	TOBB ETÜ	Özel Başarı Burslu Yüksek Lisans Öğrencisi
2015 - 2016	REOTEK	Elektronik & Bilgisayar Mühendisi

YABANCI DİL: İngilizce

TEZDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

- **Gudelek, M. U.,**, Boluk,S. A., Ozbayoglu, A. M., A deep learning based stock trading model with 2-D CNN trend detection, *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, Honolulu, HI, 2017, pp. 1-8. doi: 10.1109/SSCI.2017.8285188
- **Gudelek, M. U.**, Cirak, C. R., Arin, E., Sezgin, M. E., Ozbayoglu, A. M., & Gol, M. (2018, September). Load and PV Generation Forecast Based Cost Optimization for Nanogrids with PV and Battery. In *2018 53rd International Universities Power Engineering Conference (UPEC)* (pp. 1-6). IEEE.

DİĞER YAYINLAR, SUNUMLAR VE PATENTLER:

- Ceylan, D., **Gudelek, M. U.,**, Keysan, O., Armature Shape Optimization of an Electromagnetic Launcher Including Contact Resistance, in *IEEE Transactions on Plasma Science*, vol. 46, no. 10, pp. 3619-3627, Oct. 2018. doi: 10.1109/TPS.2018.2845948

- Ceylan, D., **Gudelek, M. U.,**, Keysan, O., Armature shape optimization of an electromagnetic launcher using genetic algorithm, *2017 IEEE 21st International Conference on Pulsed Power (PPC)*, Brighton, 2017, pp. 1-6. doi: 10.1109/PPC.2017.8291202

- Serin, G., **Gudelek, M. U.,** Ozbayoglu, A. M., Unver, H. O., Estimation of parameters for the free-form machining with deep neural network, *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA, 2017, pp. 2102-2111. doi: 10.1109/BigData.2017.8258158