T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ELEKTRİK ENERJİSİ VERİLERİ İÇİN VERİ MADENCİLİĞİ VE ÇOK-ADIMLI İLERİ TAHMİN STRATEJİLERİ

YÜKSEK LİSANS TEZİ

BATUHAN BİLGİ

DENİZLİ, AĞUSTOS - 2021

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ELEKTRİK ENERJİSİ VERİLERİ İÇİN VERİ MADENCİLİĞİ VE ÇOK-ADIMLI İLERİ TAHMİN STRATEJİLERİ

YÜKSEK LİSANS TEZİ

BATUHAN BİLGİ

DENİZLİ, AĞUSTOS - 2021

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

BATUHAN BİLGİ

ÖZET

ELEKTRİK ENERJİSİ VERİLERİ İÇİN VERİ MADENCİLİĞİ VE ÇOK-ADIMLI İLERİ TAHMİN STRATEJİLERİ

YÜKSEK LİSANS TEZİ

BATUHAN BİLGİ

PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
(TEZ DANIŞMANI: PROF. DR. SERDAR İPLİKÇİ)

DENİZLİ, AĞUSTOS - 2021

Elektrik enerjisi bir ülkedeki toplumun iş verimliliğini, yaşam kalitesini ve üretimini doğrudan etkilemektedir. Bu sebeple ülkeler kullanıcılara güvenilir, kaliteli ve sürekli olarak elektrik enerjisini sağlamak zorundadırlar. Bu zorunluluk, elektrik enerjisi tüketiminin sürekli olarak planlanması ve yönetilmesi sonucunu ortaya çıkarmaktadır. Sürekli olarak doğru bir planlama ve yönetim stratejisi oluşturmak ancak elektrik enerjisi tüketiminin en az hatayla tahmin edilebilmesi ile mümkün olabilmektedir. Bu tez çalışmasında, Enerji Piyasaları İşletme A.Ş. (EPİAŞ)'ye ait internet sitesinde bulunan Şeffaflık Platformu'ndan alınan, Türkiye geneline ait gerçekleşen gerçek zamanlı tüketim verileri kullanılarak yapay sinir ağları, karar ağaçları ve destek vektör makineleriyle oluşturulan toplamda 13 farklı çok-adımlı ileri tahmin yöntemi ile 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini gerçekleştirilmiştir. Ayrıca veri madenciliği yöntemleri ile elektrik enerjisi tüketim verisi içerisindeki ilişkiler tespit edilmeye çalışılmıştır.

ANAHTAR KELİMELER: Veri Madenciliği, Çok-Adımlı İleri Tahmin, Elektrik Enerjisi Verileri

ABSTRACT

DATA MINING AND MULTI-STEP AHEAD PREDICTION STRATEGIES FOR ELECTRICAL ENERGY DATA
MSC THESIS
BATUHAN BİLGİ
PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE
ELECTRICAL AND ELECTRONICS ENGINEERING

(SUPERVISOR: PROF. DR. SERDAR İPLİKÇİ)

DENİZLİ, AUGUST 2021

Electrical energy directly affects the work efficiency, quality of life and production of the society in a country. For this reason, countries have to provide reliable, high quality and continuous electrical energy to users. This necessity results in the continuous planning and management of electrical energy consumption. Consistently creating an accurate planning and management strategy is only possible if electrical energy consumption can be estimated with the least error. In this thesis, 13 different multi-step ahead prediction methods along with a 24-hour multi-step electrical energy consumption forecast was created by forming artificial neural networks, decision trees and support vector machines using real time electricity consumption data gathered across Turkey and published on the Transparency Platform belonging to Energy Exchange Istanbul (EXIST)'s website. In addition, data mining methods and the relationships within the electrical energy consumption data were tried to be determined.

KEYWORDS: Data Mining, Multi-step Ahead Prediction, Electrical Energy Data

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

KISALTMALAR LİSTESİ

ARM	**:**	Association Rule Mining
ECLAT	**:**	Equivalence Class Transformation
ELM	**:**	Extreme Learning Machine
FFNN	**:**	Feed Forward Neural Network
FIM	**:**	Frequent Itemset Mining
FP-Growth	**:**	Frequent Pattern Growth
H-Mine	**:**	Hyper Structure Mining
LCM	**:**	Linear time Closed itemset Miner
MAD	**:**	Median Absolute Deviation
MAE	**:**	Mean Absolute Error
MAPE	**:**	Mean Absolute Percentage Error
MIMO	**:**	Multiple Input-Multiple Output
MISO	**:**	Multiple Input-Single Output
MSE	**:**	Mean Squared Error
RMSE	**:**	Root Mean Squared Error
RNN	**:**	Recurrent Neural Network
SISO	**:**	Single Input-Single Output
SVM	**:**	Support Vector Machine
SVR	**:**	Suppor Vector Regression
TID	**:**	Transaction Identifier

ÖNSÖZ

Lisans eğitimim ve yüksek lisans eğitimim boyunca değerli bilgilerini ve tecrübelerini benimle paylaşan, bana destek olan ve yolumu aydınlatan çok değerli danışman hocam Prof. Dr. Serdar İPLİKÇİ'ye ve değerli hocam Dr. Öğr. Üyesi Bedri BAHTİYAR'a sonsuz teşekkür ederim.

Üniversite ve çalışma hayatım boyunca yanımda olan ve desteklerini benden esirgemeyen değerli dostlarım Ahmet YAVUZ'a ve Ali MENEMEN'e teşekkürlerimi bir borç bilirim.

Hayatımın her döneminde bana maddi ve manevi destek olan, hayat tecrübelerinden her daim yararlandığım ve haklarını asla ödeyemeyeceğim anneme, babama ve beni dünyanın en şanslı abisi yapan kardeşime sonsuz teşekkürlerimi sunarım.

1. GİRİŞ

Elektrik enerjisi globalleşen dünyadaki en önemli unsurlardan biridir. Dünyadaki enerji talebi her geçen gün daha da artmaktadır. Artan elektrik enerjisi talebine insan nüfusunun hızlı büyümesi, teknolojinin gelişimiyle beraber artan enerji kullanımı gibi çeşitli faktörler sebep gösterilebilmektedir. Elektrik enerjisi bir ülkedeki toplumun iş verimliliğini, yaşam kalitesini ve üretimini doğrudan etkilemektedir. Bu sebeple ülkeler kullanıcılara güvenilir, kaliteli ve sürekli olarak elektrik enerjisini sağlamak zorundadırlar. Bu zorunluluk, elektrik enerjisi tüketiminin sürekli olarak planlanması ve yönetilmesi sonucunu ortaya çıkarmaktadır. Sürekli olarak doğru bir planlama ve yönetim stratejisi oluşturmak ancak elektrik enerjisi tüketiminin en az hatayla tahmin edilebilmesi ile mümkün olabilmektedir.

Elektrik enerjisi tüketim tahmini kısa, orta ve uzun vadeli olarak yapılmaktadır. Uzun vadeli tahmin yıllık veya daha uzun dönemler şeklinde, orta vadeli tahmin aylık veya üç aylık dönemler şeklinde ve kısa vadeli tahmin ise saatlik, günlük veya haftalık dönemler şeklinde yapılmaktadır.

Kısa vadeli elektrik enerjisi tüketim tahmini önemli bir sorun olarak karşımıza çıkmaktadır. Kısa vadeli elektrik enerjisi tüketimini bilmek güç sistemlerinin optimum çalışma durumlarını belirlemek için önemlidir. Bu nedenle kısa vadeli tahminler güç sistemlerinin gelecekteki tahmin edilen elektrik enerjisi tüketimine göre hazırlanmasında yardımcı olabilmektedir.

En az hata ile yapılan tahminlerin ekonomik etkileri de vardır ve güç sistemlerinin güvenirliğini artırabilmektedir. Bir güç sisteminin güvenirliği, enerji talebindeki ani değişimlerden etkilenmektedir. Elektrik enerjisi tüketimi az tahmin edilirse güç kaynağı sıkıntısı yaşanabilmektedir. Öte yandan elektrik enerjisi tüketimi fazla tahmin edilirse kaynaklar enerji üretiminde israf edilebilmektedir. Bu gibi nedenlerle kısa vadeli elektrik enerjisi tüketim tahmininin doğru bir şekilde yapılması oldukça önemli bir konu haline gelmektedir.

Hamzaçebi ve Kutay (2004) tarafından yapılan çalışmada Türkiye'nin uzun dönem elektrik enerjisi tüketimi tahmini yapılmıştır. Zaman serisi analizi teklikleri, regresyon tekniği ve yapay sinir ağları tahmin için kullanılmış ve bu yöntemler birbirleriyle karşılaştırılmıştır.

Hamzaçebi (2007) tarafından yapılan çalışmada, sektörel bazda Türkiye'nin net elektrik enerjisi tüketimi tahmin edilmiştir. Tahmin aracı olarak yapay sinir ağı seçilmiştir. Bu tercihin yapılmasındaki nedenler, yapay sinir ağının birçok değişkenin gelecekteki değerini aynı anda tahmin edebilmesi ve veri yapısındaki doğrusal olmayan ilişkileri modelleyebilmesidir.

Kavaklıoğlu (2011) tarafından yapılan çalışmada, Türkiye'nin elektrik tüketimini modellemek ve tahmin etmek için Destek Vektör Resresyon (SVR) metodolojisi kullanılmaktadır. Çeşitli SVR yöntemleri arasında eğitim modeli seti nispeten küçük olduğu için ε-SVR yöntemi kullanılmıştır. Elektrik tüketimi; nüfus, gayri safi milli hasıla, ithalat ve ihracat gibi sosyo-ekonomik göstergelerin bir fonksiyonu olarak modellenmiştir. Elektrik tüketiminin gelecekteki tahminlerini kolaylaştırmak için, mevcut ve geçmiş değerleri kullanılarak girdi değişkenlerinin her biri için ayrı bir SVR modeli oluşturulmuştur ve bu modeller, tüketim tahmin değerleri elde etmek için birleştirilmiştir. 1975-2006 verileri kullanılarak Türkiye'nin elektrik tüketimi 2026 yılına kadar tahmin edilmiştir.

Zor ve diğ. (2017) tarafından yapılan çalışmada, kısa vade elektrik yükü tahmini için yapay sinir ağı, destek vektör makineleri ve ANFIS yöntemlerinin kapsamlı bir incelemesi ve karşılaştırılması yapılmıştır.

Başoğlu ve Bulut (2017) tarafından yapılan çalışmada, Türkiye'nin piyasa ve mevsimsel koşulları göz önüne alınarak, yapay sinir ağları ve uzman sistemlerin birlikte kullanıldığı, kısa vadeli elektrik talep tahminlerinde yüksek doğrululuk derecesi sağlayan bir hibrit sistem geliştirilmiştir. EPSİM NN adını verdikleri yeni tahmin sisteminde, günlük ortalama saatlik talep miktarı ve 24 saatlik talep şekli iki farklı yapay sinir ağı kullanılarak belirlenmektedir. Bu ağlardan elde edilen sonuçlar birleştirilerek günlük talep tahmini elde edilmiştir.

Aydın ve Toros (2018) tarafından yapılan çalışmada, Türkiye elektrik tüketiminin sıcaklıkla ilişkisi Ocak 2012 ile Kasım 2016 arasında incelenmiştir. Aylık ve mevsimsel zaman dilimlerinde sıcaklığa bağlı olarak tüketimin nasıl değiştiği ve ne kadar değiştiği araştırılmış ve tüketim tahmin modeline sıcaklık girdisi olarak eklenerek daha tutarlı bir tüketim tahmini yapılması amaçlanmıştır. Çalışmada, MATLAB programlama dili üzerinde Levenberg Marquardt geri yayılım algoritması ile modellenen veri grupları ve Yapay Sinir Ağı (YSA) yöntemi kullanılarak kısa dönemli elektrik tüketim tahminleri yapılmıştır.

Özden ve Öztürk (2018) tarafından yapılan çalışmada, zaman serileri ve yapay sinir ağları olmak üzere iki farklı yaklaşım kullanılarak Türkiye'deki bir endüstri bölgesi için enerji ihtiyaç tahmini üzerinde çalışılmış ve sonuçlar test edilmiştir. Daha önceki çalışmalardan farklı olarak, kısıtlı veri ile kısa dönem tahmini için basit bir model geliştirilmiştir. Model, giriş parametresi olarak geçmiş günlere ait tüketim verileri ve sıcaklığı içermektedir. Sıcaklık verisi, endüstri bölgelerinde ısıtma amaçlı enerji tüketiminde kullanıldığı için anahtar rol oynamaktadır. Zaman serileri yaklaşımında sadece geçmişe ait enerji tüketim verileri kullanılmıştır. Her iki yaklaşım enerji ihtiyaç tahmininde kullanılmış, sonuçlar tartışılmış ve karşılaştırılmıştır.

Özkan ve diğ (2020) tarafından yapılan çalışmada, Ocak 2016-Aralık 2018 arası dönemdeki elektrik enerjisi tüketim verileri üzerinde çalışılmış ve tahminleme başarısı yüksek bir tahmin yöntemi belirlemek üzere analizler yapılmıştır. Tahminleme yaklaşımı olarak En Küçük Kareler Yöntemi ile Fourier Analizi ve Winters' Yöntemi kullanılmıştır. 2019 yılına ait aylık veriler tahmin yöntemi seçimi için analiz edilmiştir. Tahmin analizinde elektrik tüketim değerleri (GWh) 12 ay, 24 ay ve 36 ay olmak üzere dönemsel olarak incelenmiş ve değişkenlik (varyasyon) katsayıları hesaplanmıştır. Veri setinin dönemsel değişkenliği ve tahmin modellerinin başarısı; değişkenlik katsayısı (Cv), MAPE (Ortalama Mutlak Yüzde Hata), MSE (Hataların Kareleri Ortalaması), RMSE (Hataların Kareleri Ortalamasının Karekökü) ve MAD (Ortalama Mutlak Sapma) başarı ölçütleri ile değerlendirilmiştir. Çalışmanın sonraki aşamasında başarı ölçütleri bakımından tatmin edici tahmin yöntemine karar verilmiş ve 2020 yılı için 12 aylık tahmin değerleri elde edilmiştir.

Uzlu ve Dede (2020) tarafından yapılan çalışmada, Jaya algoritması ile yapay sinir ağları kullanılarak Türkiye'nin elektrik enerjisi tüketimi tahmin edilmiştir. Ağın giriş parametreleri olarak gayri safi yurt içi hasıla, nüfus ve ithalat-ihracat değerlerini içeren ve çıkış parametresi olarak da elektrik enerjisi tüketimini içeren üç katmanlı YSA modeli oluşturulmuştur. Ağ eğitimi için Jaya algoritması kullanılmıştır. YSA-Jaya algoritması kullanılarak Türkiye'nin elektrik enerjisi tüketimi tahmini iki farklı senaryoya göre 2023 yılına kadar yapılmıştır.

Zor ve diğ. (2020) tarafından yapılan çalışmada, Doğu Akdeniz'de bulunan büyük bir hastane kompleksine ait tahmin değişkenleri ve hedef değişkenler arasında özgün ve kolay anlaşılır matematiksel modeller oluşturmak ve kısa vadeli elektrik enerjisi tüketimini tahmin etmek için gen ekspresyon programlaması (GEP) ve grup veri işleme ağları (GMDH) yöntemi kullanılmıştır.

Azadeh ve diğ (2008) tarafından yapılan çalışmada, yapay sinir ağı (YSA), bilgisayar simülasyonu ve stokastik prosedürler kullanılarak deney tasarımına dayalı aylık elektrik enerjisi tüketimini tahmin etmek için entegre bir algoritma sunulmuştur. İlk olarak, elektrik tüketimi tahmini için denetimli çok katmanlı algılayıcı (MLP) ağına dayalı bir YSA yaklaşımı gösterilmektedir. Bu nedenle seçilen model, zaman serisi modeli tarafından tahmin edilen modelle karşılaştırılabilir. Aylık elektrik tüketimi için rastgele değişkenler üretmek için bilgisayar simülasyonu geliştirilmiştir. Bu, olasılıklı dağıtımın aylık elektrik tüketimi üzerindeki etkilerini öngörmek için sağlanmıştır. Önerilen algoritmanın uygulanabilirliğini ve üstünlüğünü göstermek için İran'da Mart 1994 ile Şubat 2005 (131 ay) arasındaki aylık elektrik tüketimi kullanılmış ve önerilen algoritmaya uygulanmıştır.

Bianco ve diğ. (2009) tarafından yapılan çalışmada, Ekonomik ve demografik değişkenlerin İtalya'daki yıllık elektrik tüketimi üzerindeki etkisi, uzun vadeli bir tüketim tahmin modeli geliştirmek amacıyla araştırılmıştır. 1970'ten 2007'ye kadar olan veriler kullanılmıştır. Tarihsel elektrik tüketimi, gayri safi yurtiçi hasıla (GSYİH), kişi başına gayri safi yurtiçi hasıla (kişi başına GSYİH) ve nüfus kullanılarak farklı regresyon modelleri geliştirilmiştir.

Suganthi ve Samuel (2012) tarafından yapılan çalışmada, çeşitli enerji talebi tahmin modellerini gözden geçirme girişiminde bulunulmuştur. Talep tarafı yönetimi için zaman serisi, regresyon, ekonometrik, ARIMA gibi geleneksel yöntemler ile bulanık mantık, genetik algoritma ve sinir ağları gibi yumuşak hesaplama tekniklerinin yaygın olarak kullanıldığı aktarılmıştır. Destek vektörü regresyonu, karınca kolonisi ve parçacık sürüsü optimizasyonunun, enerji talebi tahmini için benimsenen yeni teknikler olduğu ifade edilmiştir. MARKAL ve LEAP gibi aşağıdan yukarıya modellerin de enerji talep yönetimi için ulusal ve bölgesel düzeyde kullanıldığından bahsedilmiştir.

Ahmad ve diğ. (2014) tarafından yapılan çalışmada, destek vektör makineleri (SVM) ve yapay sinir ağları (ANN) gibi yapay zeka (AI) yöntemlerini kullanan bina elektrik enerjisi tahmin yöntemi incelenmiştir.

Kaboli ve diğ. (2017) tarafından yapılan çalışmada, iki farklı tarihsel veri türünün ASEAN-5 şehirlerinin elektrik enerjisi tüketimi üzerindeki etkileri formüle edilmiştir. Geçmiş veriler ve elektrik tüketimi arasındaki ilişkileri tam olarak formüle etmek için optimize edilmiş gen ifadesi programlama (GEP) yöntemi uygulanmıştır. Önerilen yöntemin uygulanabilirliğini ve doğruluğunu değerlendirmek için, tahminleri yapay sinir ağları (ANN), destek vektör regresyonu (SVR), uyarlanabilir nöro-bulanık çıkarım sistemi (ANFIS), kural tabanlı veri madenciliği algoritması, GEP, parçacık sürüsü optimizasyonu (PSO), cuckoo arama algoritması (CSA) ve geri izleme arama algoritması (BSA) tarafından optimize edilmiş doğrusal ve ikinci dereceden modellerden elde edilenlerle karşılaştırılmıştır. Simülasyon sonuçları, 1971'den 2011'e yılına kadar gözlemlenen gerçek veri setleriyle doğrulanmıştır.

Wang ve diğ. (2018) tarafından yapılan çalışmada, elektrik tüketim tahmininde karşılaşılan, birden fazla faktöre dayanan kararsız davranışı ve etki mekanizması zorluğunu çözmek için bir sinir ağı topluluğu yaklaşımı tasarlanmıştır. Önerilen yöntemde, genelleme yeteneğini geliştirmek ve topluluk maliyetini azaltmak için topluluk çerçevesi olarak yeni bir seyrek adaboost (adaboostsp) tasarlanmıştır ve elektrik talebi ile çoklu faktörler arasında doğrusal olmayan ilişkileri kurmak için yankı durum ağı (ESN) benimsenmiştir. Giriş değişkenlerinin zaman gecikmesi etkilerini dikkate alarak seçilmesine yardımcı olmak üzere bir meyve sineği optimizasyon algoritması (FOA) kullanılmıştır. Önerilen topluluk

tahmini yaklaşımının etkinliğini doğrulamak için Çin'deki iki endüstriyel elektrik tüketimi (IEC) tahmin uygulaması araştırılmıştır. Önerilen tekniklere dayalı olarak, Hubei Eyaletindeki gelecekteki IEC talebi tahmin edilmiştir. Sayısal sonuçlara göre, FOA'lı adaboostsp-ESN'nin gelecekteki IEC'yi çeşitli kıyaslama yöntemlerinden daha iyi tahmin edebileceği gösterilmiştir.

Divina ve diğ. (2018) tarafından yapılan çalışmada, kısa vadeli yük tahmini probleminin üstesinden gelmek için topluluk öğrenmesine dayalı bir strateji önerilmiştir. Önerilen yöntem, üç temel öğrenme yöntemi tarafından üretilen tahminlerin, nihai tahminleri üretmek için en üst düzey bir yöntemle kullanıldığı bir yığınlama topluluğu öğrenme şemasına dayandırılmıştır. Önerilen yöntem, İspanya'daki enerji tüketimini dokuz yıldan uzun süredir rapor eden bir veri seti üzerinde test edilmiştir. Bu çalışmada, bir topluluk şeması kullanmanın çok doğru tahminler elde edebileceğini ve dolayısıyla bunun kısa vadeli yük tahmini problemini ele almak için uygun bir yaklaşım olduğu gösterilmiştir.

Ruzi ve diğ. (2018) tarafından yapılan çalışmada, kamu binalarında enerji tüketimi tahmini için bir yöntem önerilmiş ve böylece enerji verimliliğini artırmak için konfor ve sağlığı etkilemeden enerji tasarrufu sağlanmıştır. Enerji tüketim tahmini için bir Elman sinir ağı önerilmiş ve modellerin ağırlığını optimize etmek için bir genetik algoritma kullanılmıştır. Önerilen yöntemin enerji tüketimi tahminini optimize ettiği ve önceki çalışmalarda elde edilen sonuçları iyileştirdiği sonucuna varılmıştır.

Johannesen ve diğ. (2019) tarafından yapılan çalışmada, kentsel alan elektrik yükü tahmini için daha büyük veri kümeleri kullanılarak çeşitli regresyon araçları analiz edilmiştir. Çalışmada rastgele orman regresörü, k-en yakın komşu regresörü ve lineer regresör kullanılmıştır. Regresyon analizi, sürekli zaman bazında ve dikey zaman ekseni yaklaşımında yapılmıştır. Dikey zaman yaklaşımı, dört yıllık bir örnek zaman periyodunu (örneğin mevsimsel ve haftalık) dikkate alır ve ardışık yıl için aynı zaman periyodunda test edilmiştir.

Chou ve Truong (2020) tarafından yapılan çalışmada, bölgesel enerji tüketiminin tarihsel modelini belirlemek için doğrusal olmayan makine öğrenme modelleri ile doğrusal zaman serilerini optimize eden yeni bir tahmin sistemi

geliştirmesi amaçlanmıştır. Doğrusal zaman serisi modeli, Mevsimsel Otoregresif Entegre Hareketli Ortalama (SARIMA), doğrusal bileşeni simüle etmek için uygulanırken, zaman serisi enerji verilerinin doğrusal olmayan bileşenini yakalamak, doğrusal ve doğrusal olmayan bileşenleri birleştirmek için en küçük kareler destek vektör regresyon (LSSVR) modeli kullanılmıştır. Önerilen tahmin sisteminde uygulanmak üzere sonuçları karşılaştırmak için yüksek boyutlu matematiksel kıyaslama fonksiyonları kullanılarak çeşitli optimizasyon algoritmaları araştırılmıştır. Sonra olarak da çeşitli tahmin yöntemlerini değerlendirmek için elektriğin toptan satışını koordine eden bölgesel bir iletim organizasyonundan elde edilen üç gerçek enerji zaman serisi veri seti kullanılmıştır.

Bu tez çalışmasında, Enerji Piyasaları İşletme A.Ş. (EPİAŞ)'ye ait internet sitesinde bulunan Şeffaflık Platformu'ndan alınan, Türkiye geneline ait gerçekleşen gerçek zamanlı tüketim verileri kullanılarak yapay sinir ağları, karar ağaçları ve destek vektör makineleriyle oluşturulan toplamda 13 farklı çok-adımlı ileri tahmin yöntemi ile 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini gerçekleştirilmiştir. Ayrıca veri madenciliği yöntemleri ile elektrik enerjisi tüketim verisi içerisindeki ilişkiler tespit edilmeye çalışılmıştır. Tezin ikinci bölümünde kısıtsız optimizasyon, üçüncü bölümünde kısıtlı optimizasyon, dördüncü bölümünde veri madenciliği, beşinci bölümünde modelleme ve zaman serisi tahmini, altıncı bölümünde yapay sinir ağları, yedinci bölümünde karar ağaçları, sekizinci bölümünde destek vektör makineleri anlatılmıştır. Tezin dokuzuncu bölümünde deneysel sonuçlar verilmiştir. Tezin onuncu bölümünde ise sonuçlar ve öneriler aktarılmıştır.

2. KISITSIZ OPTİMİZASYON

Herhangi bir kısıt içermeyen, bir-boyutlu ve çok-boyutlu doğrusal olmayan optimizasyon problemleridir.

2.1 Bir-Boyutlu Doğrusal Olmayan Optimizasyon

Bir-boyutlu doğrusal olmayan optimizasyon problemi, tek değişkenli bir amaç fonksiyonunu ifade etmektedir. Gerçek hayatta, tek değişkenle ilgili bir problem bulmak oldukça zordur. Bununla birlikte, bir-boyutlu optimizasyon yöntemleri, çok değişkenli yöntemler için temel oluşturmaktadırlar. Bu yöntemler, çok değişkenli optimizasyon problemlerinin bir alt problemini oluşturduğundan, literatürde her biri diğerlerine göre farklı avantaj ve dezavantajlara sahip olan çok sayıda yöntem önerilmiştir. Bu yöntemler, gradyan tabanlı ve gradyan tabanlı olmayan yöntemler olarak sınıflandırılmaktadır. Örnek olarak tek değişkenli bir amaç fonksiyonunu aşağıdaki gibi tanımlayalım.

$$f(x) = 2x^2 - 2x + 8 \tag{2.1}$$

Denklem (2.1)'de belirtilen $f(x)$ amaç fonksiyonunu en aza indirgeyecek x değerinin belirlenmesi, bir-boyutlu doğrusal olmayan optimizasyon problemi olarak tanımlanmaktadır. x değerini, $a \leq x \leq b$ içinde kısıtlamamız gerekiyorsa, burada a ve b gerçek sayılarsa, bu bir kısıtlı optimizasyon problemi haline gelir. $f(x)$ fonksiyonu, a ve b noktaları arasında sürekli olarak artar ya da azalırsa, $f(x)$ fonksiyonu monoton fonksiyon olarak adlandırılır. Tek modlu bir fonksiyonda, fonksiyon minimum noktasının (x^*) her iki tarafında da monotondur. Denklem (2.1)'de gösterilen amaç fonksiyonu da tek modlu fonksiyonlardandır. Minimum noktanın her iki tarafında sürekli olarak azaldığı veya arttığı tek modlu fonksiyonun özelliğini kullanarak, tek değişkenli arama algoritmaları, minimumun bulunmadığı fonksiyonun belirli bölgelerini ortadan kaldıracak şekilde tasarlanabilir (İplikçi 2018).

2.1.1 Gradyan Tabanlı Yöntemler

Daha önce belirtildiği gibi, bir-boyutlu optimizasyon problemlerine yönelik çözüm teknikleri, gradyan tabanlı ve gradyan tabanlı olmayan yöntemler olarak sınıflandırılabilir. Adından da anlaşılacağı gibi, gradyan tabanlı yöntemler türev bilgisi gerektirir. Bu yöntemler, türevlerin kolayca hesaplanabileceği uygulamaların çözümlerinde kullanılabilir. Bu yöntemlere ait algoritmaların arama işlemlerinde, fonksiyonun türevi sıfıra yönlendirilir. Amaç fonksiyonun türevinin sıfıra çok yakın olduğunda ve x değeri, fonksiyonun minimum değerinin oluştuğu noktaya ($x^* = x$) karşılık geldiğinde algoritma sonlandırılır.

2.1.1.1 İkiye Bölme Yöntemi

İkiye bölme yönteminde, amaç fonksiyonuna ait minimum noktasına doğrudan ulaşılmaz. Bunun yerine, amaç fonksiyonunun türevinin sıfıra eşit olduğu nokta bulunur. Bulunan bu noktada amaç fonksiyonu minimum ya da maksimum değerine ulaşmaktadır. Burada gradyan fonksiyonu, optimum noktanın yakınında işaret değiştirir. Eğer $f'(x_1)$ ve $f'(x_2)$, x_1 ve x_2 noktalarında hesaplanan fonksiyonların türevleriyse ve Denklem (2.2)'de belirtilen şart sağlanıyorsa, amaç fonksiyonunu minimum değeri x_1 ve x_2 arasında bulunur.

$$f'(x_1)f'(x_2) < 0 \tag{2.2}$$

Denklem (2.2)'deki koşula bağlı olarak, arama alanının belirli bölgeleri ortadan kaldırılabilir. İkiye bölme yöntemine ait algoritma Algoritma 2.1'de gösterilmiştir (Arora 2015).

Algoritma 2.1: İkiye Bölme Yöntemi Algoritması

Adım 1: x_1, x_2, ε ve Δx değerlerini belirle

Adım 2: $\alpha = \frac{x_1+x_2}{2}$, $f'(x_1)$ ve $f'(x_2)$ değerlerini hesapla

Eğer $f'(x_1)f'(\alpha) < 0$

ise $x_2 = \alpha$

değilse $x_1 = \alpha$

Adım 3: Eğer $|x_1 - x_2| > \varepsilon$

ise Adım 2'ye git

değilse Adım 4'e git

Adım 4: $x^* = x_1, f(x^*) = f(x_1)$

Algoritma (2.1)'de x_1 ve x_2 fonksiyonun sınırlarıdır. Δx ise türev hesaplamak için merkezi fark formülünde kullanılır ve ε, $|x_1 - x_2| < \varepsilon$ şartı sağlandığında algoritmayı sonlandırmak için gereken küçük bir sayıdır.

2.1.1.2 Newton-Raphson Yöntemi

Newton-Raphson yöntemi, $f'(x) = 0$ denkleminin kökünün hesaplandığı bir kök bulma yöntemidir. Birinci dereceden Taylor seri açılımı kullanılarak, herhangi bir x_k değeri için aşağıdaki ifadeye yaklaşıklanabilir.

$$f'(x_k) + f''(x_k)\Delta x_k \tag{2.3}$$

Denklem (2.3) sıfıra eşitlendiğinde, bir sonraki yaklaşım noktası aşağıdaki şekilde verilebilir.

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \tag{2.4}$$

Newton-Raphson yöntemine ait algoritma Algoritma 2.2'de açıklanmıştır (Arora 2015).

Algoritma 2.2: Newton-Raphson Yöntemi Algoritması

Adım 1: x_k ve Δx_k değerlerini belirle

Adım 2: $f'(x_k)$ ve $f''(x_k)$ fonksiyonlarını hesapla

Sakla, $x_{k-1} = x_k$ ve güncelle, $x_k = x_{k-1} - \frac{f'(x_k)}{f''(x_k)}$

Adım 3: Eğer $|x_k - x_{k-1}| > \varepsilon$

ise Adım 2'ye gir

değilse Adım 4'e git

Adım 4: $x^* = x_k, f(x^*) = f(x_k)$

Newton-Raphson yöntemi, diğer yöntemlere göre bazı dezavantajlara sahiptir. Bu dezavantajlar aşağıdaki gibi sıralanabilir.

- Yakınsama başlangıç değerine duyarlıdır. Yöntem, farklı başlangıç değerleri için farklı eğilimler gösterebilmektedir.
- Gradyan değeri sıfıra yaklaştığında, yakınsama yavaşlamaktadır.
- Fonksiyonun ikinci türevi sıfırdan farklı olmalıdır.

2.1.2 Doğrudan Yöntemler

Bazı optimizasyon problemleri için, x değeri gerçel sayı olmayıp yalnızca belirli ayrık değerler alabilmektedir. Bu tür süreksiz fonksiyonlar için gradyan bilgisi her noktada mevcut olmayacak ve arama algoritması, fonksiyonun minimum değerine ulaşmak için yalnızca fonksiyona ait hesaplamaları kullanarak ilerlemek zorunda kalacaktır. Bu gibi belirli noktalarda süreksiz fonksiyonlara ait problemlerin çözümleri için gradyan temelli yöntemler kullanılamamaktadır. Bu yöntemler yerine gradyan temelli olmayan, doğrudan yöntemleri kullanmak gerekmektedir (İplikçi 2018).

2.1.2.1 Altın Bölme Yöntemi

Altın oran yöntemi, süreksiz fonksiyonlara ait problemler için oldukça etkili bir çözüm tekniğidir. Bu yöntem sürekli fonksiyonlara da uygulanabilmektedir. Altın orana sahip p ve q sayılarını aşağıdaki gibi ifade edelim.

$$\frac{p+q}{p} = \frac{p}{q} = \tau \tag{2.5}$$

Denklem (2.5)'i aşağıdaki gibi yazabiliriz.

$$1 + \frac{q}{p} = \tau \tag{2.6}$$

Denklem (2.5) şu şekilde de ifade edebiliriz.

$$1 + \frac{1}{\tau} = \tau \tag{2.7}$$

Denklem (2.7)'yi aşağıdaki gibi düzenleyebiliriz.

$$\tau^2 - \tau - 1 = 0 \tag{2.8}$$

Denklem (2.8)'deki ikinci derece denklemi çözersek, aşağıdaki eşitliği elde ederiz.

$$\tau = \frac{1 + \sqrt{5}}{2} = 1.618033 \tag{2.9}$$

Burada τ, estetikte önemi olan altın sayı olarak adlandırılmaktadır. Daha önce anlatılan arama yöntemlerinde gradyan bilgisine ihtiyaç duyuluyordu. Altın oran yönteminde, arama sadece fonksiyon hesaplamalarına dayalı olarak yapılmaktadır. Altın oran yönteminde gradyan hesaplamasına gerek duyulmamaktadır. Bu yöntemin diğer yöntemlere göre iki önemli avantajı bulunmaktadır. Bu avantajlar aşağıdaki gibi sıralanabilir.

- Altın oran yönteminde, her adımda sadece tek yeni fonksiyon hesaplaması gerekmektedir.
- Her adımda sabit olan bir azaltma faktörü bulunmaktadır.

Altın oran yöntemine ait algoritma, Algoritma 2.3'te verilmiştir (Arora 2015).

Algoritma 2.3: Altın Oran Yöntemi Algoritması

Adım 1: x_{alt}, $x_{\text{üst}}$, ε ve τ değerlerini belirle

Adım 2: Hesapla

$$x_1 = x_{alt}(1 - \tau) + \tau x_{\text{üst}}$$

$$x_2 = \tau x_{alt} + x_{\text{üst}}(1 - \tau)$$

Adım 3: Eğer $f(x_1) > f(x_2)$

ise $x_{alt} = x_1$, $x_1 = x_2$, $x_2 = \tau x_{alt} + x_{\text{üst}}(1 - \tau)$

değilse $x_{alt} = x_2$, $x_2 = x_1$, $x_1 = x_{alt}(1 - \tau) + \tau x_{\text{üst}}$

Adım 4: $|f(x_1) - f(x_2)| < \varepsilon$ şartı sağlanana kadar Adım 3'ü tekrarla

Adım 5: $x^* = x_1, f(x^*) = f(x_1)$

2.2 Çok-Boyutlu Doğrusal Olmayan Optimizasyon

Çok-boyutlu doğrusal olmayan kısıtsız optimizasyon problemleri için geliştirilmiş çözüm teknikleri bu bölümde ele alınmıştır. Gerçek hayatta, optimizasyon problemlerinin çoğu kısıtlıdır ve kısıtsız optimizasyon problemleri oldukça azdır. Bu bölümde ele alınan yöntemler, kısıtlı optimizasyon problemleri çözmek için de kullanılabilmektedir. İlerleyen alt bölümlerde anlatılan yöntemlerin tamamı gradyan tabanlı arama yöntemleridir. Bu yöntemlerle hesaplanan arama yönü, gradyan bilgisini, Hessian bilgisini veya bu ikisinin bir kombinasyonunu kullanır. Bazı yöntemlerde ise Hessian matrisine ait bir yaklaşıklık kullanılmaktadır. Arama yönü belirlendikten sonra, amaç fonksiyonunu minimize etmek için için o yönde ne kadar hareket edileceğini hesaplamak gerekmektedir. Bu da tek-boyutlu optimizasyon problemidir (İplikçi 2018).

2.2.1 Birinci Dereceden Yöntemler

Birinci dereceden yöntemler, Taylor açılımında sadece birinci dereceden türev bilgisini kullandıkları için bu ismi almışlardır. Bu yöntemleri uygulamak için fonksiyonun sadece gradyan vektörünü bilmek yeterli olmaktadır. Devam eden alt başlıklarda, birinci dereceden yöntemlerin en bilinenleri ele alınmaktadır.

2.2.1.1 Dik-İniş (Steepest-Descent) Yöntemi

Fonksiyonun değerini azaltan $\mathbf{s}_i$ arama yönü bir iniş yönüdür. Gradyan yönü boyunca ilerlendiğinde, fonksiyon değeri maksimize olmaktadır. Buradan yola çıkarak gradyanın tersi yönde ilerlendiğinde ise fonksiyonun minimize olacağı çıkarılabilir. Negatif gradyan yönüne dik-iniş yönü denilmektedir. Dik-iniş yönü aşağıdaki gibi ifade edilebilir.

$$\mathbf{s}_i = -\nabla f(\mathbf{x}_i) \tag{2.10}$$

Ardışık iterasyonlarla tasarım değişkenleri, Denklem (2.11) kullanılarak güncellenebilir.

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i) \tag{2.11}$$

Denklem (2.11)'deki α parametresi, bir-boyutlu optimizasyon yöntemleri kullanılarak belirlenebilen scalar bir parametredir. Dik-iniş yöntemi, her bir iterasyonda fonksiyon değerinde bir azalma sağlar. Başlangıç noktası minimumdan uzaksa, gradyan daha büyük değerli olacak ve her bir iterasyonda fonksiyonun azalması maksimum düzeyde olacaktır. Dik-iniş yönteminde arama her bir adımda çok fazla değişebilmektedir ve minimuma yaklaşma düzensiz olmaktadır. Bu nedenle yöntem, diğer gradyan tabanlı algoritmalar için bir başlangıç kabul edilmektedir. Dik-iniş yöntemine ait algoritma, Algoritma 2.4'te açıklanmıştır (Arora 2015).

Algoritma 2.4: Dik-İniş Yöntemi Algoritması

Adım 1: Tasarım değişkeninin başlangıç değeri $\mathbf{x}_i$ değerini belirle

ε_1 ve ε_2 değerlerini belirle

Adım 2: $f(\mathbf{x}_i)$ ve $\nabla f(\mathbf{x}_i)$ değerlerini hesapla

$\mathbf{s}_i = -\nabla f(\mathbf{x}_i)$ ilerleme yönünü seç

α parametresini bir boyutlu optimizasyon kullanarak bul

$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$ kuralı ile tasarım vektörünü güncelle

Adım 3: Eğer $|f(\mathbf{x}_{i+1}) - f(\mathbf{x}_i)| > \varepsilon_1$ veya $\|\nabla f(\mathbf{x}_i)\| > \varepsilon_2$

ise Adım 2'ye git

değilse Adım 4'e git

Adım 4: $\mathbf{x}^* = \mathbf{x}_{i+1}, f(\mathbf{x}^*) = f(\mathbf{x}_{i+1})$

2.2.1.2 Conjugate-Gradient (Fletcher-Reeves) Yöntemi

Conjugate-Gradient yöntemi, dik-iniş yönteminin modifiye edilmiş bir versiyonudur. Yönteme ait algoritma, Algoritma 2.5'te ifade edilmiştir (Arora 2015).

Algoritma 2.5: Conjugate-Gradient Yöntemi Algoritması

Adım 1: Tasarım değişkeninin başlangıç değeri $\mathbf{x}_i$ değerini belirle

ε_1 ve ε_2 değerlerini belirle

Adım 2: $f(\mathbf{x}_i)$ ve $\nabla f(\mathbf{x}_i)$ değerlerini hesapla

$\mathbf{s}_i = -\nabla f(\mathbf{x}_i)$ ilerleme yönünü seç

Bir boyutlu optimizasyon kullanarak $f(\mathbf{x}_{i+1})$ değerini minimum yapan α parametresini bul

$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha \mathbf{s}_i$ kuralı ile tasarım vektörünü güncelle

Adım 3: $\mathbf{s}_{i+1} = -\nabla f(\mathbf{x}_{i+1}) + \frac{\nabla^T f(\mathbf{x}_{i+1}) \nabla f(\mathbf{x}_{i+1})}{\nabla^T f(\mathbf{x}_i) \nabla f(\mathbf{x}_i)} \mathbf{s}_i$

Bir boyutlu optimizasyon kullanarak $f(\mathbf{x}_{i+2})$ değerini minimum yapan α parametresini bul

$\mathbf{x}_{i+2} = \mathbf{x}_{i+1} + \alpha \mathbf{s}_{i+1}$ kuralı ile tasarım vektörünü güncelle

Adım 4: Eğer $|f(\mathbf{x}_{i+2}) - f(\mathbf{x}_{i+1})| > \varepsilon_1$ veya $\|\nabla f(\mathbf{x}_{i+1})\| > \varepsilon_2$

ise Adım 3'e git

değilse Adım 5'e git

Adım 5: $\mathbf{x}^* = \mathbf{x}_{i+2}, f(\mathbf{x}^*) = f(\mathbf{x}_{i+2})$

2.2.2 İkinci Dereceden Yöntemler

İkinci dereceden yöntemler, Taylor açılımında birinci dereceden ve ikinci dereceden türev bilgisini kullandıkları için bu ismi almışlardır. Bu yöntemleri uygulamak için fonksiyonun gradyant vektörünü ve Hessian matrisi bilmek gerekmektedir. Devam eden alt başlıklarda, ikinci dereceden yöntemlerin en bilinenleri ele alınmaktadır.

2.2.2.1 Newton Yöntemi

Newton yönteminde, ikinci derece türevler veya Hessian matrisi kullanmaktadır (Arora 2015). Bu durum Newton yöntemi ile birinci türevleri veya

gradyan bilgilerini gerektiren birinci derece yöntemler olan dik-iniş ve conjugate-gradient yöntemleri arasındaki farklılığı oluşturmaktadır. Beklendiği gibi Newton yöntemindeki yakınsama birinci derecen yöntemlere göre daha hızlı olmaktadır. Bu yöntemdeki fikir, $f(\mathbf{x})$ fonksiyonuna ikinci dereceden bir yaklaşım oluşturmak ve ikinci dereceyi en aza indirgemektir. Dolayısıyla Newton yönteminde, fonksiyon $\mathbf{x}_i$ noktasındayken uygun ilerleme yönü $\mathbf{s}_i$ değerini bulurken, Denklem (2.12)'deki gibi ikinci dereceden Taylor yaklaşıklığı kullanılır (İplikçi 2018).

$$f(\mathbf{x}_i + \mathbf{s}_i) \cong f(\mathbf{x}_i) + [\nabla f(\mathbf{x}_i)]^T \mathbf{s}_i + \frac{1}{2}[\mathbf{s}_i]^T \nabla^2 f(\mathbf{x}_i)\mathbf{s}_i \tag{2.12}$$

Denklem (2.12)'deki $\nabla^2 f(\mathbf{x}_i)$ ifadesinin pozitif tanımlı olduğunu kabul edersek, $f(\mathbf{x}_i + \mathbf{s}_i)$ fonksiyonun minimum değerini bulabilmek için, fonksiyonun iniş yönüne göre türevi alınıp sıfıra eşitlenirse aşağıdaki denklem elde edilir.

$$\nabla^2 f(\mathbf{x}_i)\mathbf{s}_i = -\nabla f(\mathbf{x}_i) \tag{2.13}$$

Denklem (2.13)'den de görülebileceği gibi, i. iterasyona ait ilerleme yönü $\mathbf{s}_i$, Denklem (2.13)'deki lineer denklem sistemi çözülerek elde edilebilir.

2.2.2.2 Değiştirilmiş Newton Yöntemi

Newton yönteminde çözülmesi gerek problemler bulunmaktadır. Eğer fonksiyon yüksek derecede doğrusal olmayan bir fonksiyon ise, ikinci dereceden bir yaklaşım bile fonksiyonun zayıf bir yaklaşımı olabilir. Bu gibi problemleri çözmek, Newton yöntemini daha güvenilir kılmak ve işlemsel olarak karmaşıklığını azaltmak için yöntem üzerinde iki adet modifikasyon yapmak gerekmektedir (İplikçi 2018).

İlk olarak, güncelleme kuralına α_i adım-aralığı parametresini ekleyebiliriz ve güncelleme kuralını aşağıdaki gibi ifade edebiliriz.

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{s}_i \tag{2.14}$$

Denklem (2.14)'deki $\mathbf{s}_i$ ise, arama yönünü ifade etmektedir. Adım aralığı da $f(\mathbf{x}_{i+1}) < f(\mathbf{x}_i)$ olacak şekilde seçilmelidir.

İkinci değişiklik ise, $\mathbf{s}_i$ arama yönünün $\mathbf{x}_i$ noktasındaki f fonksiyonunun bir iniş yönü olmasını sağlamaktır. Bu nedenle aşağıdaki şartı sağlamak gerekmektedir.

$$[\nabla f(\mathbf{x}_i)]^T \mathbf{s}_i < 0 \tag{2.15}$$

veya başka bir deyişle,

$$-[\nabla f(\mathbf{x}_i)]^T [\nabla^2 f(\mathbf{x}_i)]^{-1} \nabla f(\mathbf{x}_i) < 0 \tag{2.16}$$

Denklem (2.16)'daki iniş yönü şartı ancak $\nabla^2 f(\mathbf{x}_i)$ matrisinin (eşdeğer olarak $[\nabla^2 f(\mathbf{x}_i)]^{-1}$ matrisinin) pozitif tanımlı olması ile mümkün olmaktadır. Eğer $\nabla^2 f(\mathbf{x}_i)$ matrisi pozitif tanımlı değilse kullanılabilecek bir diğer yöntem, Hessian matrisi yerine simetrik pozitif tanımlı bir $\mathbf{F}_i$ matrisi koymaktır. Bu $\mathbf{F}_i$ matrisini aşağıdaki gibi tanımlayabiliriz.

$$\mathbf{F}_i = \nabla^2 f(\mathbf{x}_i) + \gamma \mathbf{I} \tag{2.17}$$

Denklem (2.17)'de ifade edilen $\boldsymbol{F}_i$ matrisinin özdeğerleri $(\lambda_k + \gamma)$ şeklinde olmaktadır. $\mathbf{F}_i$ matrisi, tüm k değerleri için $(\lambda_k + \gamma) > 0$ olacak şekilde γ değeri artırılarak pozitif tanımlı hale getirilebilir. Böylece pozitif tanımlı hale getirilen $\mathbf{F}_i$ matrisinin tersi alınabilir. İniş yönü $\mathbf{s}_i$ ise, aşağıdaki denklemin çözülmesi ile bulunur.

$$[\mathbf{F}_i]\mathbf{s}_i = \nabla f(\mathbf{x}_i) \tag{2.18}$$

γ değerinin belirlenmesi, Hessian matrisinin $\mathbf{LDL}^{\boldsymbol{T}}$ çarpanlarına ayırma yöntemini kullanan bir teknikle, $\nabla^2 f(\mathbf{x}_i)$ ifadesinin en düşük özdeğerinin gerçek bir tanımlamasına dayanarak yapılabilir. Bu teknikte, **D** diyagonal bir matristir. Tüm diyagonal elemanlar $D_{kk} > 0$ ise, Hessian matrisi pozitif tanımlıdır. Aksi durumda, herhangi bir negatif D_{kk} elemanını pozitif olacak şekilde değiştirebiliriz. Bu da $\mathbf{F}_i$ matrisinin pozitif tanımlı olmasını sağlar Değiştirilmiş Newton Yöntemi Algoritması Algoritma 2.6'da gösterilmiştir (Arora 2015).

Algoritma 2.6: Değiştirilmiş Newton Yöntemi Algoritması

Adım 1: Tasarım değişkeninin başlangıç değeri $\mathbf{x}_i$ değerini belirle

ε_1 ve ε_2 değerlerini belirle

Adım 2: $\mathbf{x}_i$ noktasındaki $\nabla f(\mathbf{x}_i)$ gradyan vektörünü hesapla

$\mathbf{x}_i$ noktasındaki $\nabla^2 f(\mathbf{x}_i)$ Hessian matrisini hesapla

Eğer $\nabla^2 f(\mathbf{x}_i)$ matrisi pozitif tanımlı ise ilerleme yönü olarak $\mathbf{s}_i = -[\nabla^2 f(\mathbf{x}_i)]^{-1}\nabla f(\mathbf{x}_i)$ seç

Eğer $\nabla^2 f(\mathbf{x}_i)$ matrisi pozitif tanımlı değilse uygun bir matris ekleme ile pozitif tanımlı hale getir ve ilerleme yönü olacak $\mathbf{s}_i = -[\nabla^2 f(\mathbf{x}_i) + \gamma \mathbf{I}]^{-1}\nabla f(\mathbf{x}_i)$ seç

Bir boyutlu optimizasyon kullanarak $f(\mathbf{x}_i + \alpha_i \mathbf{s}_i)$ değerini minimum yapan α_i adım-aralığı değerini bul

$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{s}_i$ kuralı ile tasarım vektörünü güncelle

Adım 3: Eğer $|f(\mathbf{x}_{i+1}) - f(\mathbf{x}_i)| > \varepsilon_1$ veya $\|\nabla f(\mathbf{x}_{i+1})\| > \varepsilon_2$

ise Adım 2'ye git

değilse Adım 4'e git

Adım 4: $\mathbf{x}^* = \mathbf{x}_{i+1}$, $f(\mathbf{x}^*) = f(\mathbf{x}_{i+1})$

2.2.3 İkinci Dereceden Yaklaşık Yöntemler

İkinci dereceden yaklaşık yöntemler, birinci dereceden türev bilgisini kullanarak Hessian matrisini belirli bir yaklaşıklıkla bulup ikinci dereceden bir yakınsama sağlamaya çalışmaktadırlar. Eğer minimize edilecek $f(\mathbf{x})$ fonksiyonu, belirli sayıda karelerin toplamı şeklinde ifade edilebiliyorsa bu durum aşağıdaki gibi yazılabilir (İplikçi 2018).

$$\begin{aligned} f(\mathbf{x}) &= r_1^2(\mathbf{x}) + r_2^2(\mathbf{x}) + \cdots + r_N^2(\mathbf{x}) \\ &= \sum_{k=1}^{N} r_k^2(\mathbf{x}) \\ &= \boldsymbol{r}^T(\mathbf{x})\boldsymbol{r}(\mathbf{x}) \end{aligned} \tag{2.19}$$

Denklem (2.19)'da belirtilen $\mathbf{r}(\mathbf{x}) = [r_1(\mathbf{x}) \quad r_2(\mathbf{x}) \quad \cdots \quad r_N(\mathbf{x})]^T$ biçiminde bir vektördür. Bu durumda $f(\mathbf{x})$ fonksiyonunun gradyan vektörünün i^{inci} elemanı aşağıdaki gibi ifade edilebilir.

$$[\nabla f(\mathbf{x})]_i = \frac{\partial f(\mathbf{x})}{\partial x_i} = 2\sum_{k=1}^{N} r_k(\mathbf{x}) \frac{\partial r_k(\mathbf{x})}{\partial x_i} \tag{2.20}$$

Denklem (2.20)'deki gradyan vektörünü de şu şekilde yazabiliriz.

$$\nabla f(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\boldsymbol{r}(\mathbf{x}) \tag{2.21}$$

Denklem (2.21)'de ifade edilen $\mathbf{J}(\mathbf{x})$ matrisi Jacobian matrisidir ve aşağıdaki gibi yazılabilir.

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial r_1(\mathbf{x})}{\partial x_1} & \frac{\partial r_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial r_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial r_2(\mathbf{x})}{\partial x_1} & \frac{\partial r_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial r_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_N(\mathbf{x})}{\partial x_1} & \frac{\partial r_N(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial r_N(\mathbf{x})}{\partial x_n} \end{bmatrix} \tag{2.22}$$

$f(\mathbf{x})$ fonksiyonuna ait Hessian matrisinin $j^{inci}i^{inci}$ elemanı aşağıdaki gibi ifade edilebilir.

$$[\nabla^2 f(\mathbf{x})]_{ji} = \frac{\partial^2 f(\mathbf{x})}{\partial x_j \partial x_i} = 2\sum_{k=1}^{N} \left(\frac{\partial r_k(\mathbf{x})}{\partial x_j} \frac{\partial r_k(\mathbf{x})}{\partial x_i} + r_k(\mathbf{x}) \frac{\partial^2 r_k(\mathbf{x})}{\partial x_j \partial x_i} \right) \tag{2.23}$$

Jacobian matrisi kullanılarak $f(\mathbf{x})$ fonksiyonuna ait Hessian matrisi aşağıdaki gibi yazılabilir.

$$\nabla^2 f(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) + 2\mathbf{E}(\mathbf{x}) \tag{2.24}$$

Denklen (2.24)'deki $\mathbf{E}(\mathbf{x})$ matrisinin $j^{inci}i^{inci}$ elemanı da aşağıdaki gibidir.

$$[\mathbf{E}(\mathbf{x})]_{ji} = \sum_{k=1}^{N} r_k(\mathbf{x}) \frac{\partial^2 r_k(\mathbf{x})}{\partial x_j \partial x_i} \tag{2.25}$$

Eğer $\mathbf{E}(\mathbf{x})$ matrisinin elemanlarının yeteri kadar küçük olduğu düşünülürse, Hessian matrisi aşağıdaki gibi yaklaşık olarak ifade edilebilir.

$$\nabla^2 f(\mathbf{x}) \cong 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) \tag{2.26}$$

Denklem (2.26)'dan da görülebileceği gibi ikinci dereceden türev bilgisine sahip Hessian matrisi, birinci dereceden türev bilgisine sahip Jacobian matrisi kullanılarak kabul edilebilir bir hata ile hesaplanabilmektedir.

2.2.3.1 Gauss-Newton (GN) Yöntemi

Newton yönteminde ilerleme yönü $\mathbf{s}_i = -[\nabla^2 f(\mathbf{x}_i)]^{-1}\nabla f(\mathbf{x}_i)$ şeklinde ifade edilmekteydi. Eğer ilerleme yönünü bulabilmek için gerekli olan büyüklüklerin Jacobian matrisi karşılıkları kullanılırsa, ilerleme yönü aşağıdaki gibi hesaplanabilir (İplikçi 2018).

$$\begin{aligned} \mathbf{s}_i &= -[\nabla^2 f(\mathbf{x}_i)]^{-1}\nabla f(\mathbf{x}_i) \\ &\cong -[2\mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i)]^{-1}2\mathbf{J}^T(\mathbf{x}_i)\mathbf{r}(\mathbf{x}_i) \\ &\cong -[\mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i)]^{-1}\mathbf{J}^T(\mathbf{x}_i)\mathbf{r}(\mathbf{x}_i) \end{aligned} \tag{2.27}$$

Denklem (2.27)'de verilen $\mathbf{s}_i$ ilerleme yönü kullanılan yönteme Gauss-Newton yöntemi denilmektedir. Fakat bu yöntem, gerçek hayatta çok fazla tercih edilmemektedir. Çünkü uygulama esnasındaki her bir iterasyonda $\mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i)$ matrisinin tersinin alınabiliyor olması gerekmektedir. Ancak bu her zaman mümkün olmamakta, ilgili matris tekil olabilmektedir. Böyle bir durum oluştuğunda Gauss-Newton yöntemi kullanılamamaktadır. Bunu ortadan kaldırmak amacıyla Levenberg-Marquardt (LM) yöntemi önerilmiştir.

2.2.3.2 Levenberg-Marquardt (LM) Yöntemi

Gauss-Newton yönteminde bahsedilen ve yöntemin uygulamada tercih edilmemesinin sebebi olan $\mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i)$ matrisinin tersinin olmaması durumu, Levenberg-Marquardt yönteminde, bu matrise $\gamma_i\mathbf{I}$ terimi ilave edilerek giderilebilir. Bu durumda ilerleme yönü aşağıdaki gibi hesaplanabilmektedir.

$$\mathbf{s}_i = -[\mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i) + \gamma_i \mathbf{I}]^{-1}\mathbf{J}^T(\mathbf{x}_i)\mathbf{r}(\mathbf{x}_i) \quad (2.28)$$

Denklem (2.28)'den de görülebileceği gibi ilerleme yönündeki γ_i değeri her bir iterasyonda değişmektedir. γ_i değerinin ayarlanması Algoritma 2.7'de verilen Levenberg-Marquardt yöntemi algoritması ile gerçekleştirilmektedir (İplikçi 2018).

Algoritma 2.7: Levenberg-Marquardt Yöntemi Algoritması

Adım 1: Tasarım değişkeninin başlangıç değeri $\mathbf{x}_i$ ve γ_i başlangıç değerini belirle

γ değerinin değişimi için $\gamma_{scal} > 0$, γ_{min} ve γ_{max} değerlerini belirle

ε_1 ve ε_2 değerlerini belirle

$i \leftarrow 0$

Adım 2: $f(\mathbf{x}_i)$ fonksiyon değerini, hata vektörü $\mathbf{r}(\mathbf{x}_i)$ ve Jacobian matrisi $\mathbf{J}(\mathbf{x}_i)$ değerlerini hesapla

Adım 3: Aday ilerleme yönü $\mathbf{c}_i = -[\mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i) + \gamma_i \mathbf{I}]^{-1}\mathbf{J}^T(\mathbf{x}_i)\mathbf{r}(\mathbf{x}_i)$

Eğer $f(\mathbf{x}_i + \mathbf{c}_i) < f(\mathbf{x}_i)$ ise

$\mathbf{s}_i \leftarrow \mathbf{c}_i$

$f(\mathbf{x}_i + \alpha_i \mathbf{s}_i)$ değerini minimum yapan α_i değerini bul

$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{s}_i$ kuralı ile tasarım vektörünü güncelle

$\gamma_i \leftarrow \gamma_i / \gamma_{scal}$

Adım 5'e git

Adım 4: Eğer $f(\mathbf{x}_i + \mathbf{c}_i) > f(\mathbf{x}_i)$ ise

$\gamma_i \leftarrow \gamma_i \cdot \gamma_{scal}$

Eğer $\gamma_i < \gamma_{max}$ ve $\gamma_{min} < \gamma_i$

ise Adım 3'e git

değilse Adım 5'e git

Adım 5: $k \leftarrow k + 1$

Adım 6: Eğer $|f(\mathbf{x}_{i+1}) - f(\mathbf{x}_i)| > \varepsilon_1$ veya $\|\nabla f(\mathbf{x}_{i+1})\| > \varepsilon_2$

ise Adım 2'ye git

değilse Adım 7'ye git

Adım 7: $\mathbf{x}^* = \mathbf{x}_{i+1}, f(\mathbf{x}^*) = f(\mathbf{x}_{i+1})$

Algoritma 2.7'den de görülebileceği gibi fonksiyonu azaltan bir ilerleme yönü bulunduğu durumda γ_i büyüklüğü azaltılarak $\mathbf{s}_i$ ilerleme yönü Gauss-Newton yönüne dönüşmektedir. Bu durumda yakınsama hızlanmaktadır. Öte yandan, fonksiyonu azaltan uygun bir ilerleme yönü bulunamadığında, γ_i değeri fonksiyonu azaltan uygun bir ilerleme yönü bulunana kadar artırılmaktadır. Böylece $\mathbf{s}_i$ ilerleme yönü, Dik-İniş yönüne benzemeye başlamaktadır. Bu durumda yakınsamanın yavaşlaması göze alınarak fonksiyonun azalması sağlanmaya çalışılmaktadır. Sonuç olarak, Levenberg-Marquardt algoritması, yavaş ama güvenilir Dik-İniş yönü ile hızlı ama daha az güvenilir Gauss-Newton yönü arasında uygun bir geçiş sağlamaktadır. Bu da Levenberg-Marquardt algoritmasının en güçlü yanını oluşturmaktadır (İplikçi 2018).

3. KISITLI OPTİMİZASYON

Kısıtlı optimizasyonda yer alan kısıtların varlığı, kısıtsız optimizasyon problemlerinde karşılaşılmayan bir takım teknik sorunlara yol açmaktadır. Örneğin, amaç fonksiyonuna ait gradyan vektörünün negatif yönü boyunca yapılan bir araştırma, kısıtsız optimizasyon için iyi gerekçelendirilmiş bir tekniktir. Bununla birlikte, kısıtlı bir optimizasyon probleminde böyle bir yön boyunca noktalar kısıtlamaları sağlamayabilir. Böyle bir durumda da yapılan arama problemin çözümünü vermemektedir. Sonuç olarak, uygun arama yönlerini belirlemek için yeni yöntemler araştırılmalıdır. Kısıtlı optimizasyon problemleri için geliştirilen birçok güçlü teknik, kısıtsız optimizasyon yöntemlerine dayanmaktadır. Kısıtlar, parametreler üzerindeki alt ve/veya üst limitler açısından verilmişse, problem kolaylıkla kısıtsız bir optimizasyon problemine dönüştürülebilir (Antoniou ve Lu 2007).

3.1 Kısıtlar

En genel haliyle, kısıtlı bir optimizasyon problemi, ilgili problemi çözen optimal bir $\mathbf{x}^*$ vektörünü bulmaktır (Antoniou ve Lu 2007).

$$\begin{aligned} & \min f(\mathbf{x}) \\ Kısıtlar: \quad & a_i(\mathbf{x}) = 0 \quad burada\ i = 1, 2, \dots, p \\ & c_j(\mathbf{x}) \geq 0 \quad burada\ j = 1, 2, \dots, q \end{aligned} \tag{3.1}$$

Denklem (3.1)'de ifade edilen $f(\mathbf{x})$ amaç fonksiyonunun yanı sıra, kısıt olarak belirtilen $a_i(\mathbf{x})$ ve $c_j(\mathbf{x})$ fonksiyonlarının sürekli olduğunu ve sürekli ikinci dereceden türevlere sahip olduklarını varsayıyoruz. Burada, aynı boyuttaki iki $\mathbf{A} = \{a_{ij}\}$ ve $\mathbf{B} = \{b_{ij}\}$ matrisleri için, tüm i ve j'ler için $a_{ij} \geq b_{ij}$ durumunu belirtmek üzere $\mathbf{A} \geq \mathbf{B}$ ifadesini kullanırız. Sonuç olarak, $\mathbf{A} \geq 0$ tüm i ve j'ler için $a_{ij} \geq 0$ anlamına gelir. $\mathbf{A}$ matrisinin pozitif tanımlı, pozitif yarı tanımlı, negatif tanımlı ve negatif yarı tanımlı olduğunu belirtmek için sırasıyla $\mathbf{A} \succ 0$, $\mathbf{A} \succcurlyeq 0$, $\mathbf{A} \prec 0$ ve $\mathbf{A} \preccurlyeq 0$ yazarız.

3.1.1 Eşitlik Kısıtları

Aşağıda verilen eşitlik kısıtları kümesi, R^n'de bir hiper yüzeyi tanımlar.

$$\begin{aligned} a_1(\mathbf{x}) &= 0 \\ &\vdots \\ a_p(\mathbf{x}) &= 0 \end{aligned} \tag{3.2}$$

Vektör gösterimini kullanarak Denklem (3.2)'yi aşağıdaki gibi ifade edebiliriz.

$$\mathbf{a}(\mathbf{x}) = [a_1(\mathbf{x}) \quad a_2(\mathbf{x}) \quad \cdots \quad a_p(\mathbf{x})] \tag{3.3}$$

Denklem (3.2)'den aşağıdaki sonucu da çıkartabiliriz.

$$\mathbf{a}(\mathbf{x}) = 0 \tag{3.4}$$

x noktası, Denklem (3.2)'yi sağlarsa ve $\nabla a_1(\mathbf{x})$, $\nabla a_2(\mathbf{x})$, ..., $\nabla a_p(\mathbf{x})$ sütun vektörlerinin doğrusal olarak bağımsız olması durumunda **x** noktasına Denklem (3.2)'deki kısıtların düzenli noktası denir. Bu tanım, aslında **x** noktası, Denklem (3.2)'nin bir çözümü ise ve Jacobian matrisi, $\mathbf{J} = \left[\nabla a_1(\mathbf{x}), \nabla a_2(\mathbf{x}), \ldots, \nabla a_p(\mathbf{x})\right]^T$ satır boyunca full rank ise, kısıtların düzenli bir noktası olduğunu belirtir. Belirli bir eşitlik kısıtları kümesi için, **x** noktasının düzenli olmasının önemi, düzenli bir **x** noktasında kısıtlar tarafından belirlenen hiper yüzeyin teğet düzleminin iyi tanımlanmış olması gerçeğinde yatmaktadır. Bu bölümün ilerleyen kısımlarında, "teğet düzlem" terimi, kısıtlı optimizasyon problemleri için gerekli ve anı zamanda önemli koşulları ifade etmek ve tanımlamak için kullanılacaktır. Jacobian matrisi **J**, $p \times n$ boyutlu bir matris olduğundan, eğer $p > n$ olsaydı **x** noktası kısıtların düzenli bir noktası olmazdı. Bu, bağımsız eşitlik kısıtlarının sayısı için $p \leq n$ şeklinde bir üst sınır oluşturmaktadır. Ayrıca $p = n$ ise, birçok durumda Denklem (3.2)'yi karşılayan **x** vektörlerinin sayısı sonludur ve optimizasyon problemi önemsiz hale gelir. Bu nedenlerle bu bölümde $p < n$ olduğu varsayılacaktır (Antoniou ve Lu 2007).

Eşitlik kısıtlarının önemli bir sınıfı, $a_i(\mathbf{x})$ fonksiyonlarının hepsinin doğrusal olduğu doğrusal kısıtlar sınıfıdır. Bu durumda, Denklem (3.2) aşağıdaki şekilde ifade edilebilen bir doğrusal denklem sistemi haline gelmektedir.

$$\mathbf{Ax} = \mathbf{b} \tag{3.5}$$

Denklem (3.5)'de $\mathbf{A} \in R^{p \times n}$ matrisi nümerik olarak Jacobian matrisine eşittir. Yani $\mathbf{A} = \mathbf{J}$ ve $\mathbf{b} \in R^{p \times 1}$ şeklinde ifade edilebilir. Jacobian matrisi sabit bir matris olduğundan, Denklem (3.5)'in herhangi bir çözüm noktası, eğer $rank(\mathbf{A}) = p$ ise düzenli bir noktadır. Eğer $rank(\mathbf{A}) = p' < p$ ise Denklem (3.6) ya da Denklem (3.7) durumu ortaya çıkabilir.

$$rank([\mathbf{A} \;\; \mathbf{b}]) > rank(\mathbf{A}) \tag{3.6}$$

$$rank([\mathbf{A} \;\; \mathbf{b}]) = rank(\mathbf{A}) \tag{3.7}$$

Eğer Denklem (3.6)'daki eşitsizlik sağlanırsa, Denklem (3.5)'de çelişkilerin olduğu ve bu çelişkileri ortadan kaldırmak için Denklem (3.5)'in dikkatli bir şekilde incelenmesi gerektiği sonucuna varıyoruz. Eğer Denklem (3.7), $rank(\mathbf{A}) = p' < p$ şartını sağlarsa, Denklem (3.5)'i p' kısıtları ile aşağıdaki gibi bir eşitlik kısıtları setine indirgemek için cebirsel manipülasyonlar kullanılabilir.

$$\widehat{\mathbf{A}}\mathbf{x} = \hat{\mathbf{b}} \tag{3.8}$$

Denklem (3.8)'de $\widehat{\mathbf{A}} \in R^{p' \times n}$ matrisinin rankı p' değerine eşittir ve $\hat{\mathbf{b}} \in R^{p' \times 1}$ şeklinde ifade edilebilir. Ayrıca, problemi kısıtsız bir optimizasyon problemine dönüştürmek veya ilgili parametre sayısını azaltmak için Denklem (3.8) ile satırca full rank $\widehat{\mathbf{A}}$ matrisi arasındaki doğrusal eşitlik kısıtlamaları ortadan kaldırılabilir.

$rank(\mathbf{A}) = p' < p$ olduğunda, Denklem (3.5)'i Denklem (3.8)'e indirgemenin sayısal olarak güvenilir bir yolu, **A** matrisine tekil-değer ayrıştırmasını (SVD) uygulamaktır. Tekil-değer ayrıştırması **A** matrisine uygulandığında aşağıdaki denklemi elde ederiz.

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \tag{3.9}$$

Denklem (3.9)'da $\mathbf{U} \in R^{p \times p}$ ve $\mathbf{V} \in R^{n \times n}$ ortagonal dik matrislerdir ve $\boldsymbol{\Sigma}$ matrisi de aşağıdaki gibi ifade edilebilir.

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{S} & \vdots & \mathbf{0} \\ \cdots & \cdots & \cdots \\ \mathbf{0} & \vdots & \mathbf{0} \end{bmatrix}_{p \times n} \tag{3.10}$$

Denklem (3.10)'da $\mathbf{S} = diag\{\sigma_1\,,\ \sigma_2\,,\ \ldots,\ \sigma_{p'}\}$ ve $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{p'} > 0$ şeklinde ifade edilir. Bunu takiben **A** matrisi aşağıdaki gibidir.

$$\mathbf{A} = \mathbf{U}\begin{bmatrix} \widehat{\mathbf{A}} \\ \mathbf{0} \end{bmatrix} \tag{3.11}$$

Denklem (3.11)'de $\widehat{\mathbf{A}} = \mathbf{S}[\mathbf{v}_1\ \ \mathbf{v}_2\ \ \cdots\ \ \mathbf{v}_{p'}]^T \in R^{p' \times n}$ şeklinde ifade edilebilir ve burada $\mathbf{v}_i$, **V** matrisinin i^{inci} sütununu göstermektedir. En nihayetinde Denklem (3.5) aşağıdaki denkleme dönüşmektedir.

$$\begin{bmatrix} \widehat{\mathbf{A}} \\ \mathbf{0} \end{bmatrix}\mathbf{x} = \begin{bmatrix} \hat{\mathbf{b}} \\ \mathbf{0} \end{bmatrix} \tag{3.12}$$

Denklem (3.12)'da belirtilen eşitlik, Denklem (3.8)'e yol açmaktadır ve burada $\hat{\mathbf{b}}$, $\mathbf{U}^T\mathbf{b}$ ifadesinin ilk p' girişleri kullanılarak oluşturulur. Açıkça belirtmek gerekirse, Denklem (3.8)'in herhangi bir çözüm noktası düzenli bir noktadır (Antoniou ve Lu 2007).

3.1.2 Eşitsizlik Kısıtları

Aşağıdaki eşitsizlik kısıtlarını ele alalım.

$$\begin{aligned} c_1(\mathbf{x}) &\geq 0 \\ c_2(\mathbf{x}) &\geq 0 \\ &\vdots \\ c_q(\mathbf{x}) &\geq 0 \end{aligned} \tag{3.13}$$

Eşitlik kısıtlarının aksine, q eşitsizlik kısıtlarının sayısının n'den az olması gerekmez. Örneğin, $1 \leq j \leq q$ şartı için tüm $c_j(\mathbf{x})$ fonksiyonlarının doğrusal fonksiyonlar olduğu durumunu ele alırsak, Denklem (3.13)'deki kısıtlar q adet yüzeyi olan bir çokyüzlüyü temsil eder ve böyle bir çokyüzlüdeki yüzeylerin sayısı açıkça sınırsızdır (Antoniou ve Lu 2007).

Uygun bir $\mathbf{x}$ noktası için Denklem (3.13)'deki eşitsizlikler iki sınıfa ayrılabilir. Bunlardan ilki $c_i(\mathbf{x}) = 0$ durumunu sağlayan ve aktif kısıtlar olarak adlandırılan kısıtlar kümesi, ikinci ise $c_i(\mathbf{x}) > 0$ durumunu sağlayan ve aktif olmayan kısıtlar olarak adlandırılan kısıtlar kümesidir. $c_i(\mathbf{x})$ fonksiyonları sürekli fonksiyonlar olduğundan, $\mathbf{x}$ noktasında aktif olmayan kısıtlar, $\mathbf{x}$ noktasının yeterince küçük bir komşuluğunda bu şekilde kalacaktır. Bu, $\mathbf{x}$'in lokal özelliklerinin aktif olmayan kısıtlardan etkilenmeyeceği anlamına gelmektedir. Öte yandan, $c_i(\mathbf{x}) = 0$ olduğunda $\mathbf{x}$ noktası aktif kısıtlar tarafından belirlenen sınır yüzeyindedir. Dolayısıyla, bu kısıtların bazılarını ihlal edecek yönler mevcuttur. Başka bir deyişle, aktif kısıtlar $\mathbf{x}$ noktasının komşuluğunun uygun bölgesini kısıtlamaktadır.

Eşitsizlik kısıtlarıyla başa çıkmak için kullanılabilecek bir başka yaklaşım ise, onları eşitlik kısıtlarına dönüştürmektir. Sadelik olması açısından, sadece eşitsizlik kısıtlarını içeren aşağıdaki problemi ele alıyoruz.

$$\begin{aligned} &\min f(\mathbf{x}) \quad \mathbf{x} \in R^n \\ Kısıtlar: \quad &c_i(\mathbf{x}) \geq 0 \quad burada\ i = 1, 2, \dots, q \end{aligned} \tag{3.14}$$

Denklem (3.14)'deki eşitsizlik kısıtları aşağıdaki gibi ifade edilebilir.

$$\begin{aligned} \hat{c}_1 &= c_1(\mathbf{x}) - y_1 = 0 \\ \hat{c}_2 &= c_2(\mathbf{x}) - y_2 = 0 \\ &\vdots \\ \hat{c}_q &= c_q(\mathbf{x}) - y_q = 0 \\ y_i &\geq 0, \quad 1 \leq i \leq q \end{aligned} \tag{3.15}$$

Denklem (3.15)'deki y_1, y_2, ..., y_q durağan değişkenler olarak adlandırılır. Buradaki kısıtlar, aşağıdaki gibi değişken yerine koyma eşitlikleri kullanılarak ortadan kaldırılabilir.

$$\begin{aligned} y_i &= \hat{y}_i^2, \quad 1 \leq i \leq q \\ \hat{\mathbf{x}} &= \left[x_1 \cdots x_n \;\; \hat{y}_1 \cdots \hat{y}_q\right]^T \end{aligned} \tag{3.16}$$

Denklem (3.16)'da verilen eşitlikler yerine konulursa, Denklem (3.14)'deki problem Denklem (3.17)'deki gibi formüle edilebilir.

$$\begin{aligned} & \min f(\hat{\mathbf{x}}) \quad \hat{\mathbf{x}} \in E^{n+q} \\ Kısıtlar: \quad & \hat{c}_i(\hat{\mathbf{x}}) \geq 0 \quad burada\ i = 1, 2, \ldots, q \end{aligned} \tag{3.17}$$

Bir optimizasyon problemini yeniden formüle etmek için durağan değişkenlerin tanıtılması fikri, özellikle doğrusal programlamada, standart olmayan bir problemi standart bir probleme dönüştürmek için başarılı bir şekilde kullanılmıştır.

Denklem (3.13)'deki eşitsizlik kısıtları ile karakterize edilen bölgenin dışbükey olup olmadığının belirlenmesi her zaman kolay olmasa da, doğrusal $c_i(\mathbf{x})$ fonksiyonu ile Denklem (3.13) tarafından tanımlanan uygun bir bölgenin dışbükey bir çokyüzlü olduğu kolayca gösterilebilir. Bu durumun gerçekten böyle olduğunu göstermek için, doğrusal eşitsizlik kısıtlarını aşağıdaki gibi yazabiliriz.

$$\mathbf{Cx} \geq \mathbf{d} \tag{3.18}$$

Denklem (3.18)'deki $\mathbf{C} \in R^{q \times n}$ ve $\mathbf{d} \in R^{q \times 1}$ şeklindedir. Denklem (3.18)'de $\mathcal{R} = \{\mathbf{x} : \mathbf{Cx} \geq \mathbf{d}\}$ olsun ve $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{R}$ olduğunu varsayalım. Denklem (3.18), $\lambda \in [0, 1]$ ve $\mathbf{x} = \lambda\mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2$ noktası için aşağıdaki gibi ifade edilebilir.

$$\begin{aligned} \mathbf{Cx} &= \lambda\mathbf{Cx}_1 + (1 - \lambda)\mathbf{Cx}_2 \\ &\geq \lambda\mathbf{d} + (1 - \lambda)\mathbf{d} = \mathbf{d} \end{aligned} \tag{3.19}$$

Bu nedenle, $\mathbf{Cx} \geq \mathbf{d}$ ifadesi dışbükey bir kümeyi tanımlamaktadır. Literatürde eşitsizlik kısıtları bazen aşağıdaki biçimde verilmektedir.

$$\begin{aligned} c_1(x) &\leq 0 \\ &\vdots \\ c_q(x) &\leq 0 \end{aligned} \tag{3.20}$$

Denklem (3.20)'deki, $1 \leq i \leq q$ şartı için $c_i(x)$ fonksiyonlarının tümü doğrusal fonksiyonlarsa, Denklem (3.20) tarafından tanımlanan uygun bölgenin dışbükey olduğunu göstermek için benzer bir argüman kullanılabilir (Antoniou ve Lu 2007).

3.2 Kısıtlı Optimizasyon Problemlerinin Sınıflandırılması

Kısıtlı optimizasyon problemleri, amaç fonksiyonunun doğasına ve kısıtlarına göre sınıflandırılabilir. Belirli problem sınıfları için, hızlı ve güvenilir bir şekilde çözüm elde etmek için uygun yöntemler vardır. Örneğin, doğrusal programlama problemleri için Danzig'in simpleks yönteminin ve primal-dual iç nokta yönteminin oldukça verimli olduğu kanıtlanmıştır. Genel dışbükey programlama problemleri için, özellikle etkili olan birkaç iç nokta yöntemi geliştirilmiştir (Antoniou ve Lu 2007).

Aşağıdaki tanımlarda $\mathcal{R}$, Denklem (3.1)'de belirtilen problemin uygun bölgesini ifade eder ve $\delta > 0$ şartı ile $\{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^*\| \leq \delta\}$ nokta kümesinin $\mathbf{x}^*$ merkezli bir top olduğu söylenir.

Tanım 3.1 Eğer $\mathbf{x}^*$ noktası, $\mathcal{D}_{x^*} = \mathcal{B}_{x^*} \cap \mathcal{R}$ ifadesinin boş ve $f(\mathbf{x}^*) = min\{f(\mathbf{x}) : \mathbf{x} \in \mathcal{D}_{x^*}\}$ olacak şekilde $\delta > 0$ şartını sağlayan bir $\mathcal{B}_{x^*} = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^*\| \leq \delta\}$ topu varsa, Denklem (3.1)'deki problemin kısıtlı bir yerel minimumudur.

Tanım 3.2 Eğer $\mathbf{x}^*$ noktası, $\mathbf{x}^* \in \mathcal{R}$ ve $f(\mathbf{x}^*) = min\{f(\mathbf{x}) : \mathbf{x} \in \mathcal{R}\}$ ise, Denklem (3.1)'deki problemin kısıtlı bir global minimumudur.

Tanım 3.3 Eğer bir $\mathcal{B}_{x^*}$ topu varsa, $\mathcal{D}_{x^*} = \mathcal{B}_{x^*} \cap \mathcal{R}$ boş değilse ve $\mathcal{D}_{x^*}$'deki $\mathbf{x}^*$ kısıtlı tek minimumsa, bu kısıtlı minimum $\mathbf{x}^*$, güçlü bir yerel minimum olarak adlandırılır.

3.2.1 Doğrusal Programlama (LP)

Bir doğrusal programlama (LP) probleminin standart şekli Denklem (3.21)'deki gibi ifade edilebilir (Antoniou ve Lu 2007).

$$\begin{aligned} &\min f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \\ &Kısıtlar: \quad \mathbf{Ax} = \mathbf{b} \\ &\qquad\qquad \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{3.21}$$

Denklem (3.21)'deki $\mathbf{c} \in R^{n\times 1}$, $\mathbf{A} \in R^{p\times n}$ ve $\mathbf{b} \in R^{p\times 1}$ şeklinde verilmiştir. Denklem (3.21)'deki doğrusal eşitlik kısıtlarına ve negatif olmayan sınırlara tabi olan

doğrusal bir amaç fonksiyonunu minimize eden bir $\mathbf{x}^*$ vektörü bulmamız gerekir. Doğrusal programlama problemleri, aşağıdaki gibi de gösterilebilir.

$$\begin{aligned} &\min \mathbf{c}^T\mathbf{x} \\ &Kısıtlar: \quad \mathbf{Ax} \geq \mathbf{b} \end{aligned} \tag{3.22}$$

$\mathbf{y}$ vektörü cinsinden durağan değişkenleri Denklem (3.23)'deki gibi gösterebiliriz.

$$\mathbf{y} = \mathbf{Ax} - \mathbf{b} \tag{3.23}$$

Denklem (3.22)'deki kısıtlar,

$$\mathbf{Ax} - \mathbf{y} = \mathbf{b} \tag{3.24}$$

ve

$$\mathbf{y} \geq \mathbf{0} \tag{3.25}$$

şeklinde verilebilir. Eğer $\mathbf{x}$ değişkenini, $\mathbf{x}^+ \geq 0$ ve $\mathbf{x}^- \geq 0$ şeklinde verilen iki negatif olmayan vektörün farklı olarak ifade edersek, yani,

$$\mathbf{x} = \mathbf{x}^+ - \mathbf{x}^- \tag{3.26}$$

ve

$$\hat{\mathbf{x}} = \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \\ \mathbf{y} \end{bmatrix} \tag{3.27}$$

eşitliği verilebilir. Daha sonra amaç fonksiyonu,

$$\hat{\mathbf{c}}^T\hat{\mathbf{x}} = [\mathbf{c}^T \quad -\mathbf{c}^T \quad 0]\hat{\mathbf{x}} \tag{3.28}$$

şeklinde ifade edilir ve Denklem (3.24) ve Denklem (3.25)'deki kısıtlar,

$$[\mathbf{A} \quad -\mathbf{A} \quad -\mathbf{I}]\hat{\mathbf{x}} = \mathbf{b} \tag{3.29}$$

ve

$$\hat{\mathbf{x}} \geq \mathbf{0} \tag{3.30}$$

olarak yazılabilir. Dolayısıyla, Denklem (3.22)'deki problem, standart doğrusal programlama problemi olarak aşağıdaki gibi ifade edilebilir.

$$\begin{aligned} &\min \hat{\mathbf{c}}^T \hat{\mathbf{x}} \\ \textit{Kısıtlar}: \quad &\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}} \\ &\hat{\mathbf{x}} \geq \mathbf{0} \end{aligned} \tag{3.31}$$

Denklem (3.31)'de $\hat{\mathbf{c}} = [\mathbf{c} \quad -\mathbf{c} \quad \mathbf{0}]^T$ ve $\hat{\mathbf{A}} = [\mathbf{A} \quad -\mathbf{A} \quad -\mathbf{I}]$ şeklindedir.

3.2.2 Karesel Programlama (QP)

En basit ve en sık karşılaşılan kısıtlı doğrusal olmayan optimizasyon problemleri sınıfı, karesel programlama (QP) problemleri sınıfıdır (Antoniou ve Lu 2007). Bu problemlerde, amaç fonksiyonu ikinci derecedendir ve kısıtlar doğrusaldır. Bir karesel programlama problemi,

$$\begin{aligned} &\min f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{p} + c \\ \textit{Kısıtlar}: \quad &\mathbf{A}\mathbf{x} = \mathbf{b} \\ &\mathbf{C}\mathbf{x} = \mathbf{d} \end{aligned} \tag{3.32}$$

şeklinde ifade edilebilir. Birçok uygulamada, $f(\mathbf{x})$ fonksiyonuna ait Hessian matrisi **H**, pozitif yarı tanımlıdır. Bu, $f(\mathbf{x})$'in global konveks bir fonksiyon olduğu anlamına gelmektedir. Denklem (3.32)'deki kısıtlar ile belirlenen uygun bölge her zaman konveks olduğu için pozitif yarı tanımlı Hessian matrisi **H** ile verilen karesel programlama problemleri, bir sonraki bölümde anlatılacak özel bir konveks programlama problemleri sınıfı olarak kabul edilebilir.

3.2.3 Konveks Programlama (CP)

Konveks programlama probleminde, problem için uygun bir konveks bölgeyi tanımlayan kısıtlar kümesine tabi olan konveks bir amaç fonksiyonunu minimize eden parametre vektörü aranmaktadır. Açıkça görülebileceği gibi, pozitif yarı tanımlı

Hessian matrisine sahip doğrusal programlama ve karesel programlama problemleri, konveks programlama problemleri olarak görülebilir (Antoniou ve Lu 2007).

Mühendislik alanında ve bilimde pratik önemi olan başka konveks programlama problemleri de vardır. Örnek olarak aşağıdaki problemi ele alalım.

$$\begin{aligned} & \min \ln(\det \mathbf{P}^{-1}) \\ Kısıtlar: \quad & \mathbf{P} \succ \mathbf{0} \\ & \mathbf{v}_i^T \mathbf{P} \mathbf{v}_i \leq 1, \ \ i = 1, 2, \dots, L \end{aligned} \tag{3.33}$$

Denklem (3.33)'de, $1 \leq i \leq L$ için $\mathbf{v}_i$ vektörleri verilmiştir ve $\mathbf{P} = \mathbf{P}^T$ matrisinin elemanları değişkenlerdir. Eğer $\mathbf{P} \succ \mathbf{0}$ kısıtı sağlanıyorsa ($\mathbf{P}$ matrisi pozitif tanımlı ise), $\ln(\det \mathbf{P}^{-1})$ 'nin $\mathbf{P}$ matrisinin konveks bir fonksiyonu olduğu gösterilebilir. Ek olarak, $\mathbf{p} = \mathbf{P}(:)$ eşitliği, $\mathbf{P}$ matrisinin elemanlarını sözlüksel biçimde sıralayarak elde edilen vektörü belirtirse, Denklem (3.33)'deki kısıtları karşılayan $\mathbf{p}$ vektörleri kümesi konvekstir ve bu nedenle Denklem (3.33) bir konveks programlama problemini açıklar.

3.3 Lagrange Çarpanları

Lagrange çarpanları, kısıtlı optimizasyon konusunda önemli bir yere sahiptir. Bir yandan, Lagrange çarpanlarına uygulanan koşullar her zaman çeşitli gerekli ve yeterli koşulların ayrılmaz bir parçasıdır ve diğer yandan, kısıtlı ve ilgili kısıtsız optimizasyon problemleri arasında doğal bir bağlantı sağlar; her bir Lagrange çarpanı, ilişkili kısıt fonksiyonundaki değişikliklere göre amaç fonksiyonundaki değişim oranı olarak yorumlanabilir. Basit bir ifadeyle, eğer $\mathbf{x}^*$ kısıtlı bir optimizasyon probleminin yerel bir minimumu ise, $\mathbf{x}^*$'ın uygun bir nokta olmasına ek olarak, $\mathbf{x}^*$ 'daki amaç fonksiyonunun gradyanı kısıt fonksiyonlarının gradyanlarının doğrusal bir kombinasyonu olmalıdır ve Lagrange çarpanları, bu doğrusal kombinasyonun katsayılarıdır. Ayrıca, eşitsizlik kısıtları ile ilişkili Lagrange çarpanlarının negatif olmaması ve aktif olmayan eşitsizlik kısıtları ile ilişkili çarpanların sıfır olması gerekir. Toplu olarak, bu koşullar Karush-Kuhn Tucker koşulları (KKT) olarak bilinir (Antoniou ve Lu 2007).

Aşağıda ifade edilen eşitlik kısıtlarına tabi olan $f(x_1, x_2, x_3, x_4)$ amaç fonksiyonunun minimizasyonunu ele alalım.

$$\begin{aligned} a_1(x_1, x_2, x_3, x_4) &= 0 \\ a_2(x_1, x_2, x_3, x_4) &= 0 \end{aligned} \tag{3.34}$$

Eğer Denklem (3.34)'deki kısıtlar,

$$\begin{aligned} x_3 &= h_1(x_1, x_2) \\ x_4 &= h_2(x_1, x_2) \end{aligned} \tag{3.35}$$

olarak ifade edilebilirse, Denklem (3.35)'i $f[x_1, x_2, h_1(x_1, x_2), h_2(x_1, x_2)]$ formunu alacak olan amaç fonksiyonu ile değiştirerek ortadan kaldırılabilirler. Eğer $\mathbf{x}^* = [x_1^*, x_2^*, x_3^*, x_4^*]^T$, orijinal kısıtlı optimizasyon probleminin yerel bir minimumu ise, $\hat{\mathbf{x}}^* = [x_1^*, x_2^*]^T$ eşitliği de aşağıdaki problemin yerel bir minimumudur.

$$\min f[x_1, x_2, h_1(x_1, x_2), h_2(x_1, x_2)] \tag{3.36}$$

Bu nedenle, $\hat{\mathbf{x}}^*$ için aşağıdaki ifadeyi yazabiliriz.

$$\nabla f = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \end{bmatrix} \tag{3.37}$$

Denklem (3.35)'in kısıtlarındaki x_3 ve x_4 değişkenleri x_3 ve x_3 değişkenleri ile ilişkili olduğundan, ∇f ifadesindeki kısmi türevler için zincir kuralının kullanılması aşağıdaki eşitlikleri verir.

$$\begin{aligned} \frac{\partial f}{\partial x_1} + \frac{\partial f}{\partial x_3}\frac{\partial h_1}{\partial x_1} + \frac{\partial f}{\partial x_4}\frac{\partial h_2}{\partial x_1} &= 0 \\ \frac{\partial f}{\partial x_2} + \frac{\partial f}{\partial x_3}\frac{\partial h_1}{\partial x_2} + \frac{\partial f}{\partial x_4}\frac{\partial h_2}{\partial x_2} &= 0 \end{aligned} \tag{3.38}$$

Denklem (3.34) ve Denklem (3.35)'den aşağıdaki eşitlikleri elde ederiz.

$$\begin{aligned}
&\frac{\partial a_1}{\partial x_1}+\frac{\partial a_1}{\partial x_3}\frac{\partial h_1}{\partial x_1}+\frac{\partial a_1}{\partial x_4}\frac{\partial h_2}{\partial x_1}=0\\
&\frac{\partial a_1}{\partial x_2}+\frac{\partial a_1}{\partial x_3}\frac{\partial h_1}{\partial x_2}+\frac{\partial a_1}{\partial x_4}\frac{\partial h_2}{\partial x_2}=0\\
&\frac{\partial a_2}{\partial x_1}+\frac{\partial a_2}{\partial x_3}\frac{\partial h_1}{\partial x_1}+\frac{\partial a_2}{\partial x_4}\frac{\partial h_2}{\partial x_1}=0\\
&\frac{\partial a_2}{\partial x_2}+\frac{\partial a_2}{\partial x_3}\frac{\partial h_1}{\partial x_2}+\frac{\partial a_2}{\partial x_4}\frac{\partial h_2}{\partial x_2}=0
\end{aligned} \tag{3.39}$$

Denklem (3.38) ve Denklem (3.39)'daki altı denklem aşağıdaki şekilde ifade edilebilir.

$$\begin{bmatrix} \nabla^T f(\mathbf{x}) \\ \nabla^T a_1(\mathbf{x}) \\ \nabla^T a_2(\mathbf{x}) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \dfrac{\partial h_1}{\partial x_1} & \dfrac{\partial h_1}{\partial x_2} \\ \dfrac{\partial h_2}{\partial x_1} & \dfrac{\partial h_2}{\partial x_2} \end{bmatrix} = \mathbf{0} \tag{3.40}$$

Denklem (3.40), $\nabla f(\mathbf{x}^*)$, $\nabla a_1(\mathbf{x}^*)$ ve $\nabla a_2(\mathbf{x}^*)$ ifadelerinin doğrusal bağımlı olduğunu göstermektedir. Dolayısıyla tümü sıfır olmayan α, β ve γ sabitleri ile birlikte aşağıdaki eşitlik verilebilir.

$$\alpha\nabla f(\mathbf{x}^*)+\beta\nabla a_1(\mathbf{x}^*)+\gamma\nabla a_2(\mathbf{x}^*)=\mathbf{0} \tag{3.41}$$

$\mathbf{x}^*$ 'ın kısıtların düzenli bir noktası olduğunu varsayarsak, Denklem (3.41)'deki α sabiti sıfır olamaz ve Denklem (3.41),

$$\nabla f(\mathbf{x}^*)-\lambda_1^*\nabla a_1(\mathbf{x}^*)-\lambda_2^*\nabla a_2(\mathbf{x}^*)=\mathbf{0} \tag{3.42}$$

ve

$$\nabla f(\mathbf{x}^*)=\lambda_1^*\nabla a_1(\mathbf{x}^*)+\lambda_2^*\nabla a_2(\mathbf{x}^*) \tag{3.43}$$

olacak şekilde basitleştirilebilir. Denklem (3.42) ve Denklem (3.43)'deki $\lambda_1^*=-\beta/\alpha$ ve $\lambda_2^*=-\gamma/\alpha$ şeklindedir. Buradan, kısıtlı optimizasyon probleminin yerel bir minimumunda, amaç fonksiyonunun gradyanının, kısıtların gradyanlarının doğrusal bir kombinasyonu olduğu sonucuna varabiliriz. Denklem (3.42) ve

Denklem (3.43)'deki λ_1^* ve λ_2^* sabitlerine, kısıtlı problemin Lagrange çarpanları denir.

3.3.1 Eşitlik Kısıtları

Fletcher tarafından kullanılan bir yaklaşımı takiben kısıtlı optimizasyon problemini aşağıdaki gibi değerlendirebiliriz.

$$\begin{aligned} &\min f(\mathbf{x}) \\ &Kısıtlar: \quad a_i(\mathbf{x}) = 0, \;\; i = 1, 2, \dots, p \end{aligned} \tag{3.44}$$

$\mathbf{x}^*$, Denklem (3.44)'deki problemin bir yerel minimumu olsun. Denklem (3.44)'deki $a_i(\mathbf{x})$ fonksiyonunda $\mathbf{x}$ yerine $\mathbf{x}^*$ koyup Taylor serisi kısıt fonksiyonu $a_i(\mathbf{x}^*)$'ı kullanarak, $a_i(\mathbf{x}^*) = 0$ olduğundan, aşağıdaki ifade yazılabilir.

$$\begin{aligned} a_i(\mathbf{x}^* + \mathbf{s}) &= a_i(\mathbf{x}^*) + \mathbf{s}^T \nabla a_i(\mathbf{x}^*) + o(\|\mathbf{s}\|) \\ &= \mathbf{s}^T \nabla a_i(\mathbf{x}^*) + o(\|\mathbf{s}\|) \end{aligned} \tag{3.45}$$

Eğer $\mathbf{s}$, $\mathbf{x}^*$'da uygun bir vektör ise, $a_i(\mathbf{x}^* + \mathbf{s}) = 0$ olur ve dolayısıyla Denklem (3.45),

$$\mathbf{s}^T \nabla a_i(\mathbf{x}^*) = 0, \;\; i = 1, 2, \dots, p \tag{3.46}$$

şeklinde ifade edilebilir. Başka bir deyişle, $\mathbf{s}$, kısıt fonksiyonlarının gradyanlarına dik ise uygun bir vektördür. Şimdi $\nabla f(\mathbf{x}^*)$ gradyanının, $\{\nabla a_1(\mathbf{x}^*), \nabla a_2(\mathbf{x}^*), \dots, \nabla a_p(\mathbf{x}^*)\}$ ifadesinin kapsadığı alana dik olarak iz düşümünü alıyoruz. Bu izdüşümü,

$$\sum_{i=0}^{p} \lambda_i^* \nabla a_i(\mathbf{x}^*) \tag{3.47}$$

olarak belirtirsek, $\nabla f(\mathbf{x}^*)$ gradyanı,

$$\nabla f(\mathbf{x}^*) = \sum_{i=0}^{p} \lambda_i^* \nabla a_i(\mathbf{x}^*) + \mathbf{d} \tag{3.48}$$

şeklinde ifade edebiliriz. Burada $\mathbf{d}$, $\nabla a_i(\mathbf{x}^*),\ \ i = 1, 2, \ldots, p$ ifadesine diktir.

Aşağıda, $\mathbf{x}^*$ yerel bir minimum ise $\mathbf{d}$ 'nin sıfır olması gerektiği gösterilmektedir. $\mathbf{d} \neq \mathbf{0}$ olduğunu varsayalım ve $\mathbf{s} = -\mathbf{d}$ olsun. $\mathbf{s}$, Denklem (4.46)'ya binaen $\nabla a_i(\mathbf{x}^*)$ ifadesine dik olduğundan $\mathbf{s}$, $\mathbf{x}^*$ 'da uygundur. Şimdi aşağıdaki denklemi elde edebilmek için Denklem (3.48)'i kullanabiliriz.

$$\mathbf{s}^T \nabla f(\mathbf{x}^*) = \mathbf{s}^T \left(\sum_{i=0}^{p} \lambda_i^* \nabla a_i(\mathbf{x}^*) + \mathbf{d} \right) = -\|\mathbf{d}\|^2 < 0 \tag{3.49}$$

Bu, $\mathbf{s}$'nin $\mathbf{x}^*$'da bir iniş yönü olduğu anlamına gelir, bu da $\mathbf{x}^*$'ın bir minimum olduğu gerçeğiyle çelişmektedir. Bu sebeple $\mathbf{d} = \mathbf{0}$ ve Denklem (3.48) aşağıdaki gibi olur.

$$\nabla f(\mathbf{x}^*) = \sum_{i=0}^{p} \lambda_i^* \nabla a_i(\mathbf{x}^*) \tag{3.50}$$

Gerçekte, eşitlik kısıtları ile keyfi olarak kısıtlanmış bir problem için, yerel bir minimumdaki amaç fonksiyonun gradyanı, eşitlik kısıt fonksiyonlarının gradyanlarının katsayı olarak Lagrange çarpanları ile doğrusal kombinasyonuna eşittir (Antoniou ve Lu 2007).

Denklem (3.1)'deki hem eşitlik hem de eşitsizlik kısıtları olan problem için, Denklem (3.50)'nin $\mathbf{x}^*$ 'da aktif olan eşitsizlik kısıtlarını içerecek şekilde değiştirilmesi gerekmektedir.

Örnek 3.1 Aşağıdaki optimizasyon problemi için Lagrange çarpanlarını belirleyelim.

$$\begin{aligned} &\min f(\mathbf{x}) \\ &Kısıtlar: \quad \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned} \tag{3.51}$$

Denklem (3.51)'deki $A \in R^{p \times n}$ ifadesinin satırca full rank olduğu varsayılır. Ayrıca problem için, kısıtların doğrusal olmadığı durumu da tartışalım.

Çözüm Bu durumda Denklem (3.50) aşağıdaki gibi olur.

$$\mathbf{g}^* = \mathbf{A}^T \boldsymbol{\lambda}^* \tag{3.52}$$

Denklem (3.52)'de $\boldsymbol{\lambda}^* = \begin{bmatrix}\lambda_1^* & \lambda_2^* & \cdots & \lambda_p^*\end{bmatrix}$ ve $\mathbf{g}^* = \nabla f(\mathbf{x}^*)$ şeklindedir. Denklem (3.52)'ye binaen, Lagrange çarpanları eşsiz olarak aşağıdaki gibi belirlenir.

$$\boldsymbol{\lambda}^* = (\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{g}^* = (\mathbf{A}^\mathbf{T})^+\mathbf{g}^* \tag{3.53}$$

Denklem (3.53)'de $(\mathbf{A}^\mathbf{T})^+$, $\mathbf{A}^T$ değerinin Moore-Penrose pseudo-inverse ifadesidir.

Doğrusal olmayan eşitlik kısıtları söz konusu olduğunda, benzer bir sonuca Denklem (3.44)'deki kısıtlardan Jacobian açısından varılabilir. Aşağıdaki denklemi tanımlarsak,

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix}\nabla a_1(\mathbf{x}) & \nabla a_2(\mathbf{x}) & \cdots & \nabla a_p(\mathbf{x})\end{bmatrix}^T \tag{3.54}$$

$\mathbf{J}(\mathbf{x})$'in $\mathbf{x}^*$'da satırca full rank olması koşuluyla, Denklem (3.50)'deki $1 \leq i \leq p$ için λ_i^* Lagrange çarpanları,

$$\boldsymbol{\lambda}^* = [\mathbf{J}^T(\mathbf{x}^*)]^+\mathbf{g}^* \tag{3.55}$$

olarak belirlenir.

Lagrange çarpanları kavramı farklı bir perspektiften açıklanabilir. Optimizasyon probleminin Lagrangian'ı olarak,

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{i=1}^{p} \lambda_i a_i(\mathbf{x}) \tag{3.56}$$

fonksiyonunu tanımlarsak, Denklem (3.50)'deki koşul ve Denklem (3.44)'deki kısıtlar sırasıyla,

$$\nabla_x L(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}, \;\; \{\mathbf{x}, \boldsymbol{\lambda}\} = \{\mathbf{x}^*, \boldsymbol{\lambda}^*\} \tag{3.57}$$

ve

$$\nabla_\lambda L(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}, \;\; \{\mathbf{x}, \boldsymbol{\lambda}\} = \{\mathbf{x}^*, \boldsymbol{\lambda}^*\} \tag{3.58}$$

olarak yazılabilir. Denklem (3.57) ve Denklem (3.58)'deki denklemlerin sayısı sırasıyla n ve p'dir ve toplam denklem sayısı, $\mathbf{x}$ ve $\boldsymbol{\lambda}$'daki parametre sayısı ile tutarlıdır, yani $n + p$'dir. Şimdi ∇ gradient operatörünü,

$$\nabla = \begin{bmatrix} \nabla_x \\ \nabla_\lambda \end{bmatrix} \tag{3.59}$$

olarak tanımlarsak, Denklem (3.57) ve Denklem (3.58) aşağıdaki gibi ifade edilebilir.

$$\nabla L(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}, \;\; \{\mathbf{x}, \boldsymbol{\lambda}\} = \{\mathbf{x}^*, \boldsymbol{\lambda}^*\} \tag{3.60}$$

Yukarıdaki analizden, Lagrangian'ın kısıtları değiştirilmiş bir amaç fonksiyonuna, kısıtlı bir minimum $\mathbf{x}^*$'ın, p Lagrange çarpanları ile arttırmanın elde edildiği artırılmış amaç fonksiyonu $L(\mathbf{x}, \boldsymbol{\lambda})$ için kısıtsız bir minimum $\{\mathbf{x}^*, \boldsymbol{\lambda}^*\}$'a bağlanacak şekilde dahil edildiğini görüyoruz.

Örnek 3.2 Aşağıdaki optimizasyon probleminin çözümünü inceleyelim.

$$\begin{aligned} \min f(\mathbf{x}) &= \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{p} \\ Kısıtlar: \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned} \tag{3.61}$$

Denklem (3.61)'de $\mathbf{H} \succ \mathbf{0}$ ve $\mathbf{A} \in R^{p \times n}$ ifadesi satırca full ranktır.

Çözüm Burada,

$$\nabla L(\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{p} - \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{3.62}$$

Lagrange ifadesini tanımlamak, ve

$$\begin{aligned} \nabla L(\mathbf{x}, \boldsymbol{\lambda}) &= \begin{bmatrix} \mathbf{H}\mathbf{x} + \mathbf{p} - \mathbf{A}^T\boldsymbol{\lambda} \\ -\mathbf{A}\mathbf{x} + \mathbf{b} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{H} & -\mathbf{A}^T \\ -\mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} + \begin{bmatrix} \mathbf{p} \\ \mathbf{b} \end{bmatrix} = \mathbf{0} \end{aligned} \tag{3.63}$$

denklemini elde etmek için Denklem (3.60)'daki koşul uygulanır. $\mathbf{H} \succ \mathbf{0}$ ve $rank(\mathbf{A}) = p$ olduğundan,

$$\begin{bmatrix} \mathbf{H} & -\mathbf{A}^T \\ -\mathbf{A} & \mathbf{0} \end{bmatrix} \tag{3.64}$$

matrisinin tekil olmayan bir matris olduğunu ve bu nedenle Denklem (3.63)'ün,

$$\begin{bmatrix} \mathbf{x}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = -\begin{bmatrix} \mathbf{H} & -\mathbf{A}^T \\ -\mathbf{A} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{p} \\ \mathbf{b} \end{bmatrix} \tag{3.65}$$

benzersiz çözümüne sahip olduğunu gösterebiliriz.

Denklem (3.65)'teki $\mathbf{x}^*$ ve $\boldsymbol{\lambda}^*$,

$$\mathbf{x}^* = \mathbf{H}^{-1}(\mathbf{A}^T\boldsymbol{\lambda}^* - \mathbf{p}) \tag{3.66}$$

$$\boldsymbol{\lambda}^* = (\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^T)^{-1}(\mathbf{A}\mathbf{H}^{-1}\mathbf{p} + \mathbf{b}) \tag{3.67}$$

şeklinde ifade edilebilir.

3.4 Konvekslik

Konvekslik kavramı, kısıtlı optimizasyonda oldukça önemlidir. Kısıtlı bir optimizasyonda, amaç fonksiyonu, uygulanan kısıtlar ile karakterize edilen uygun bölgeye göre minimize edilir. Beklenebileceği gibi, hem amaç fonksiyonu hem de uygun bölge konveks olduğunda faydalı optimizasyon sonuçları elde etmek için konvekslik kavramı tamamen kullanılabilir (Antoniou ve Lu 2007). Bölüm 3.2'de bu problemlere konveks programlama problemi adı verilmişti. Bu sınıfa ait tipik bir problem aşağıdaki gibi formüle edilebilir.

$$\begin{aligned} & \min f(\mathbf{x}) \\ Kısıtlar: \quad & a_i(\mathbf{x}) = \mathbf{a}_i^T\mathbf{x} - b_i, \quad 1 \le i \le p \\ & c_i(\mathbf{x}) \ge 0, \qquad\quad 1 \le j \le q \end{aligned} \tag{3.68}$$

Denklem (3.68)'de $f(\mathbf{x})$ ve $1 \le j \le q$ için $-c_i(\mathbf{x})$ konveks fonksiyonlardır. Burada ana sonuçlar, sonraki iki teorem ile açıklanacaktır.

Teorem 3.1 Konveks programlama problemlerinde minimumların globalliği ve konveksliği

a) Eğer $\mathbf{x}^*$, bir konveks programlama probleminin bir yerel minimumu ise, $\mathbf{x}^*$ aynı zamanda bir global minimumdur.
b) S olarak ifade edilen bir konveks programlama probleminin minimumlarının kümesi konvekstir.
c) Eğer amaç fonksiyonu $f(\mathbf{x})$, uygun bölge $\mathcal{R}$ üzerinde kesin konveks ise, global minimum benzersizdir.

İspat

a) Eğer $\mathbf{x}^*$ global olmayan bir yerel minimum ise, $f(\hat{\mathbf{x}}) < f(\mathbf{x}^*)$ olacak şekilde uygun bir $\hat{\mathbf{x}}$ vardır. Eğer $0 < \tau < 1$ için $\mathbf{x}_\tau = \tau\hat{\mathbf{x}} + (1-\tau)\mathbf{x}^*$ eşitliğini yazarsak, $\mathbf{x}_\tau$, $\mathbf{x}^*$'a ne kadar yakın olursa olsun $f(\mathbf{x})$'in konveksliği aşağıdaki gibi ifade edilir.

$$f(\mathbf{x}_\tau) \leq \tau f(\hat{\mathbf{x}}) + (1-\tau) f(\mathbf{x}^*) < f(\mathbf{x}^*) \tag{3.69}$$

Bu durum, $f(x^*)$ fonksiyonunun, $\mathbf{x}^*$'ın yeterince küçük bir komşuluğunda en küçük değeri alması gerektiği için $\mathbf{x}^*$ 'ın bir yerel minimum olduğu varsayımıyla çelişmektedir. Dolayısıyla, $\mathbf{x}^*$ bir global minimumdur.

b) $\mathbf{x}_a, \mathbf{x}_b \in S$ olsun. Yukarıdaki a) kısmından, $\mathbf{x}_a$ ve $\mathbf{x}_b$'nin global minimumlar oldukları anlaşılmaktadır. Eğer $0 < \tau < 1$ için $\mathbf{x}_\tau = \tau\mathbf{x}_a + (1-\tau)\mathbf{x}_b$ ise, $f(\mathbf{x})$'in konveksliği aşağıdaki ifadeye yol açar.

$$f(\mathbf{x}_\tau) \leq \tau f(\mathbf{x}_a) + (1-\tau) f(\mathbf{x}_b) = f(\mathbf{x}_a) \tag{3.70}$$

$\mathbf{x}_a$ bir global minimum olduğundan, $f(\mathbf{x}_\tau) \geq f(\mathbf{x}_a)$ 'dır. Dolayısıyla, $f(\mathbf{x}_\tau) = f(\mathbf{x}_a)$'dır, yani her τ değeri için $\mathbf{x}_\tau \in S$ olduğundan, S konvekstir.

c) Çözüm kümesi S'nin $\mathbf{x}_a$ ve $\mathbf{x}_b$ şeklinde iki farklı nokta içerdiğini ve $0 < \tau < 1$ için $\mathbf{x}_\tau$'nun b) kısmında tanımlandığını varsayalım. $\mathbf{x}_a \neq \mathbf{x}_b$ ve $\tau \in (0,1)$ olduğundan, $\mathbf{x}_\tau \neq \mathbf{x}_a$ ifadesine sahibiz. $f(\mathbf{x})$ fonksiyonunun kesin

konveksliğini kullanarak, $\mathbf{x}_a \in S$ varsayımıyla çelişen $f(\mathbf{x}_\tau) < f(\mathbf{x}_a)$ sonucuna varabiliriz. Bu nedenle, global minimum benzersizdir.

3.5 Duallik

Optimizasyona uygulandığı gibi duallik kavramı esasen dolaylı fakat bazen daha verimli bir çözüm yöntemine yol açan bir problem dönüşümüdür. Duallik tabanlı bir yöntemde, primal problem olarak adlandırılan orijinal problem, parametrelerin primalin Lagrange çarpanları olduğu bir probleme dönüştürülür. Dönüştürülen probleme dual problem adı verilir. Eşitsizlik kısıtlarının $\mathbf{x}$ 'in boyutundan çok daha fazla olduğu durumda, Lagrange çarpanlarını bulmak için dual problemi çözmek ve daha sonra primal problem için $\mathbf{x}^*$'ı bulmak çekici bir alternatif haline gelir. Doğrusal programlama problemleri için, modern primal-dual iç nokta yöntemlerinin temelini oluşturmak üzere bir duallik teorisi geliştirilmiştir (Antoniou ve Lu 2007).

Popüler bir duallik tabanlı yöntem, Denklem (3.68)'deki konveks programlama problemiyle ilgilenen Wolfe dualidir (Wolfe 1961). Wolfe dualinin ana sonuçları aşağıdaki teoremde açıklanmaktadır.

Teorem 3.2 Konveks programlamada duallik

$\mathbf{x}^*$ bir minimum ve $\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*$ Denklem (3.68)'deki problemin ilgili Lagrange çarpanları olsun. Eğer $\mathbf{x}^*$ kısıtların düzenli bir noktasıysa, $\mathbf{x}^*, \boldsymbol{\lambda}^*$ ve $\boldsymbol{\mu}^*$ aşağıdaki dual problemi çözer.

$$\begin{aligned} &\max_{\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\mu}} L(\mathbf{x}, \boldsymbol{\lambda},) \\ Kısıtlar: \quad &\nabla_x L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{0} \\ &\boldsymbol{\mu} \geq \mathbf{0} \end{aligned} \tag{3.71}$$

Ayrıca $f(\mathbf{x}^*) = L(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$.

İspat

Burada, $f(\mathbf{x}^*) = L(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ ve $\boldsymbol{\mu}^* \geq \mathbf{0}$ olduğunu biliyoruz. Denklem (3.71)'deki problem için uygun olan bir $\{\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}\}$ için $\boldsymbol{\mu} \geq \mathbf{0}$ ve $\nabla_x L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{0}$ ifadelerine sahibiz. Dolayısıyla aşağıdaki denklemi yazabiliriz.

$$\begin{aligned} L(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) &= f(\mathbf{x}^*) \\ &\geq f(\mathbf{x}^*) - \sum_{i=1}^{p} \lambda_i a_i(\mathbf{x}^*) - \sum_{j=1}^{q} \mu_j c_j(\mathbf{x}^*) - L(\mathbf{x}^*, , \boldsymbol{\mu}) \end{aligned} \tag{3.72}$$

$\boldsymbol{\mu} \geq \mathbf{0}$ için Lagrange $L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ konvekstir ve

$$L(\mathbf{x}^*, \boldsymbol{\lambda}, \boldsymbol{\mu}) \geq L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) + (x^* - x)^T \nabla_x L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{3.73}$$

olur. Bu nedenle, $L(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \geq L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$, yani $\{\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*\}$ seti, Denklem (3.71)'deki problemi çözer.

Örnek 3.3 Aşağıda standart formda verilmiş doğrusal programlama problemi için Wolfe dualliğini bulun.

$$\begin{aligned} & \min \mathbf{c}^T \mathbf{x} \\ Kısıtlar: \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathrm{A} \in R^{p \times n} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{3.74}$$

Çözüm Lagrange ifadesi aşağıdaki gibi verilir.

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{c}^T \mathbf{x} - (\mathbf{A}\mathbf{x} - \mathbf{b})^T \boldsymbol{\lambda} - \mathbf{x}^T \boldsymbol{\mu} \tag{3.75}$$

Teorem 3.2'den, Denklem (3.74)'teki problemin Wolfe dualliği aşağıdaki maksimizasyon problemidir.

$$\begin{aligned} & \max_{\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}} \mathbf{x}^T (\mathbf{c} - \mathbf{A}^T \boldsymbol{\lambda} - \boldsymbol{\mu}) + \mathbf{b}^T \boldsymbol{\lambda} \\ Kısıtlar: \quad & (\mathbf{c} - \mathbf{A}^T \boldsymbol{\lambda} - \boldsymbol{\mu}) = \mathbf{0} \\ & \boldsymbol{\mu} \geq \mathbf{0} \end{aligned} \tag{3.76}$$

Denklem (3.76)'daki eşitlik kısıtı kullanılarak, Denklem (3.76)'daki amaç fonksiyonu basitleştirilebilir ve dual problem aşağıdaki gibi ifade edilebilir.

$$\begin{aligned} &\max_{\boldsymbol{\lambda},\boldsymbol{\mu}} \mathbf{b}^T\boldsymbol{\lambda} \\ Kısıtlar: \quad &(\mathbf{c} - \mathbf{A}^T\boldsymbol{\lambda} - \boldsymbol{\mu}) = \mathbf{0} \\ &\boldsymbol{\mu} \geq \mathbf{0} \end{aligned} \tag{3.77}$$

4. VERİ MADENCİLİĞİ

Veri madenciliğinin amacı, geçmişi anlayıp geleceği tahmin etmeye çalışmaktır. Bir büyüklüğün gelecekteki değerini tahmin etmek için geliştirilen yapay sinir ağları, destek vektör makineleri gibi yöntemler "açıklayıcı (explanatory)" olmaktan çok giriş-çıkış verisini doğru bir şekilde temsil etmeye dayalı modeller olduğu için genellikler kara-kutu (black box) modeller gibi davranırlar. Oysaki veri madenciliği yöntemleri, veriler içerisindeki örüntüleri bularak insanlar tarafından kolayca anlaşılabilecek ve yorumlanabilecek sonuçlar üretmeyi hedeflemektedir. Veri içerisindeki örüntüleri bulmak için literatürde önerilmiş olan yöntemler, bulunacak örüntülerin tipine göre çeşitlilik kazanmaktadır. Yaygın olarak kullanılan bazı örüntü tipleri olarak, kümeler (clusters), öğe-kümeleri (itemsets), eğilimler (trends) ve aykırılıklar (outliers) verilebilir.

4.1 Sık Öğeseti Madenciliği (Frequent Itemset Mining – FIM)

Veri içindeki örüntülerin bulunması fikri ilk olarak Agrawal ve Srikant (1994) tarafından ortaya atılmıştır. Buna ilk olarak büyük örüntü madenciliği denilmiş olsa da artık günümüzde buna Sık Öğeseti Madenciliği (Frequent Itemset Mining – FIM) denmektedir. FIM ilk olarak market sepeti verilerinin analizinde kullanıldığı için, FIM kavramını tanımlamak için de market sepeti uygulamasından yararlanarak şu şekilde tanımlama yapılabilir: müşterilerin yaptıkları alış-veriş ya da işlemlerin (transactions) olduğu bir veritabanı (database) verildiğini düşünelim. Burada FIM birlikte satın alınan öğe veya öğe-kümelerinden en sık görülenleri bulmaya çalışır. Örnek olarak, bir FIM algoritmasının sonucunda, çok sayıda müşteri tarafından kurabiye ile baharatın bir arada satın alındığı gibi bir örüntü ortaya çıkabilir. Bu öğeler arasındaki ilişkilerin ortaya çıkarılması, müşteri davranışlarının analiz edilmesinde, pazarlama konusunda (bir arada çokça satılan ürünlerin rafta yan yana konması veya kampanya yapılması gibi) stratejik kararlar alınmasında oldukça faydalıdır (İplikçi 2020).

FIM her ne kadar ilk olarak müşteri davranışlarının analiz edilmesi için önerilmiş olsa da artık günümüzde pek çok alana uygulanabilecek bir veri madenciliği işi olarak görülmektedir. Zira, müşteri hareketlerinin olduğu veritabanı daha da genelleştirilerek FIM daha farklı alanlarda kullanılabilmektedir. Artık bu daha genel veritabanında, müşteri hareketleri yerine belli özelliklere (attribute) sahip nesneleri (objects) tanımlayan örnekler (instances) bulunacaktır. Böylece, FIM, veritabanında bir arada sık görülen özellikleri bulma işi olarak daha genel bir şekilde tanımlanabilir. Pek çok veri tipi, işlem veritabanı (transaction database) şeklinde temsil edilebildiği için, FIM, bioinformatics, image classification, network traffic analysis, analyzing customer reviews, activity monitoring, malware detection ve e-learning gibi çok çeşitli alanlarda uygulanabilmektedir. Ayrıca, FIM, rare patterns, correlated patterns, patterns in sequences and graphs and patterns that generate a high profit gibi özel tip örüntülerin bulunmasında da kullanılabilir. FIM, sürekli yeni algoritmaların geliştirildiği çok aktif bir araştırma alanıdır (İplikçi 2020).

FIM daha formel olarak şu şekilde tanımlanabilir: Bir dizi öğe (set of items) $I = \{i_1, i_2 \dots i_m\}$ şeklinde verilsin. Bir işlem veritabanı (transaction database) $D = \{T_1, T_2, \dots, T_N\}$ işlemlerinden oluşan bir veritabanıdır. Öyle ki, her bir işlem $T_q \subseteq I$ $(1 \leq q \leq m)$ bir öğe (item) ya da öğe-setinden (itemset) oluşmaktadır ve her bir T_q işleminin q indisi ile gösterilen ve Transaction Identifier (TID) adı verilen kendine has bir kimlik numarası vardır. Örnek olarak Tablo 1'de verilen işlem veritabanını ele alalım. Bu veritabanında beş adet işlem bulunmaktadır. Burada a, b, c, d ve e harfleri müşteriler tarafından satın alınan ürünleri ya da öğeleri (items) temsil etmektedir. Mesela, T_1 işlemi, a, c ve d ürünlerinin bir arada satın alındığı bir işlemdir.

Tablo 4.1: İşlem veritabanı (transaction database).

TID	İşlem (Transaction)
T_1	$\{a, c, d\}$
T_2	$\{b, c, e\}$
T_3	$\{a, b, c, e\}$
T_4	$\{b, e\}$
T_5	$\{a, b, c, e\}$

X gibi bir öğe-seti (itemset) öğelerden oluşan bir kümedir öyle ki, $X \subseteq I$. $|X|$ notasyonu öğe-setinin eleman sayısını (cardinality) ya da, başka bir deyişle, X öğe-setindeki öğe sayısını göstersin. Eğer $|X| = k$ ise, yani X öğe-setinin öğe sayısı k ise, o zaman bu X öğe-setine k -uzunluklu veya k -öğeli (k -itemset) denir. FIM algoritmalarının amacı, verilen bir veritabanındaki önemli/ilginç örüntüleri ortaya çıkarmaktır. Genel olarak, bir örüntünün önemliliği ya da ilginçliği için çeşitli ölçütler ortaya konsa da FIM algoritmalarında önemlilik/ilginçlik ölçütü olarak destek (support) kavramı kullanılır. Verilen bir D gibi bir veritabanında, X gibi bir öğe-setinin mutlak-desteği (absolute support) $AbsSupp(X)$ ile gösterilir ve X öğe setini içeren işlemlerin (transactions) sayısı olarak tanımlanır yani,

$$AbsSupp(X) = |\{T|\ X \subseteq T \wedge T \in D\}| = X \tag{4.1}$$

Benzer şekilde, verilen bir D gibi bir veritabanında, X gibi bir öğe-setinin bağıl-desteği (relative support) $RelSupp(X)$ ile gösterilir ve N işlem sayısı (number of transactions) olmak üzere, X öğe-setini içeren işlemlerin (transactions) sayısının işlem sayısına (N) oranı olarak tanımlanır yani,

$$RelSupp(X) = \frac{AbsSupp(X)}{N} \tag{4.2}$$

Örnek olarak, Tablo 4.1'de verilen veritabanı için $\{a, b\}$ öğe-setinin mutlak-desteği 2, bağıl-desteği 0.4'tür, çünkü $\{a, b\}$ öğe-seti sadece T_3 ve T_5 işlemlerinde yer almaktadır, yani

$$AbsSupp(\{a, b\}) = 2 \qquad\qquad RelSupp(\{a, b\}) = 0.4$$

Şimdi de sık (frequent) öğe-seti kavramını ele alalım. $minsup$ gibi verilen bir minimum destek (eşit) değeri için, X gibi bir öğe-setinin destek değeri bu eşik değerine eşit veya daha fazlaysa, o zaman bu X gibi öğe-setine sık öğe seti (frequent itemset) denir. Yani eğer $RelSupp(X) \geq minsup$ (veya $AbsSupp(X) \geq minsup$) ise, o zaman X bir sık öğe-setidir (frequent itemset).

FIM algoritmalarının amacı, verilen bir veritabanındaki tüm sık öğe-kümelerini bulmaktır. Örnek olarak, Tablo 4.1'de verilen veritabanı için $minsup =$

3 olmak üzere, tüm sık öğe-kümeleri, destek değerleri karşılarında belirtildiği gibi, şu şekildedir:

$\{a\}:3 \quad \{b\}:4 \quad \{c\}:4 \quad \{e\}:4 \quad \{a,c\}:3 \quad \{b,c\}:3 \quad \{b,e\}:4 \quad \{c,e\}:3 \quad \{b,c,e\}:3$

Dolayısıyla, FIM bir sayma (enumeration) problemidir. Amaç, minimum support şartını sağlayan tüm öğe-kümelerini ortaya çıkarmaktır. Böylece, oldukça zor olan FIM probleminin her zaman tek bir çözüm kümesi vardır. FIM problemini çözmek için izlenecek en basit (naive) yöntem, önce olası tüm öğe-kümelerini bulup ardından bunların destek (support) değerlerini bularak $minsup$ eşiğini geçenleri sık öğe-seti (frequent itemset) döndürmek olacaktır. Ancak, böyle basit bir yaklaşım şu sebepten dolayı kullanışsızdır: diyelim ki veritabanında m farklı öğe olsun, o zaman bu veritabanında $2^m - 1$ farklı öğe-seti olacaktır. Örneğin a, b, c, d ve e öğelerinin bulunduğu bir veritabanının öğe-seti uzayı'nda (itemset space) aşağıdaki gibi $2^5 - 1 = 31$ farklı öğe-seti olacaktır:

$\{a\}$	$\{a,b\}$	$\{b,d\}$	$\{a,b,c\}$	$\{a,d,e\}$	$\{a,b,c,d\}$	$\{a,b,c,d,e\}$
$\{b\}$	$\{a,c\}$	$\{b,e\}$	$\{a,b,d\}$	$\{b,c,d\}$	$\{a,b,c,e\}$	
$\{c\}$	$\{a,d\}$	$\{c,d\}$	$\{a,b,e\}$	$\{b,c,e\}$	$\{a,b,d,e\}$	
$\{d\}$	$\{a,e\}$	$\{c,e\}$	$\{a,c,d\}$	$\{b,d,e\}$	$\{a,c,d,e\}$	
$\{e\}$	$\{b,c\}$	$\{d,e\}$	$\{a,c,e\}$	$\{c,d,e\}$	$\{b,c,d,e\}$	

Eğer bir veritabanında 100 farklı öğe olursa, o zaman $2^{100} - 1$ farklı öğe-seti olacaktır ve basit yaklaşımla sık öğe-kümelerini bulması neredeyse imkansız hale gelecektir. Bu yaklaşım, çok küçük veritabanlarında bile kullanışsızdır. Örneğin, $minsup = 1$ olmak üzere, 100 farklı öğeden oluşan tek bir işlemin (transaction) olduğu bir veritabanı için bile $2^{100} - 1$ farklı öğe-seti oluşturmaya çalışacaktır. Dolayısıyla, genel olarak, veritabanındaki öğe sayısı, veritabanındaki işlem sayısından daha önemli hale gelmektedir. Peki, arama uzayındaki öğe-seti sayısını hangi faktörler etkilemektedir? Öğe-seti sayısı, hem $minsup$ değerine, hem de veritabanındaki öğelerin birbirine ne kadar benzediğine bağlıdır.

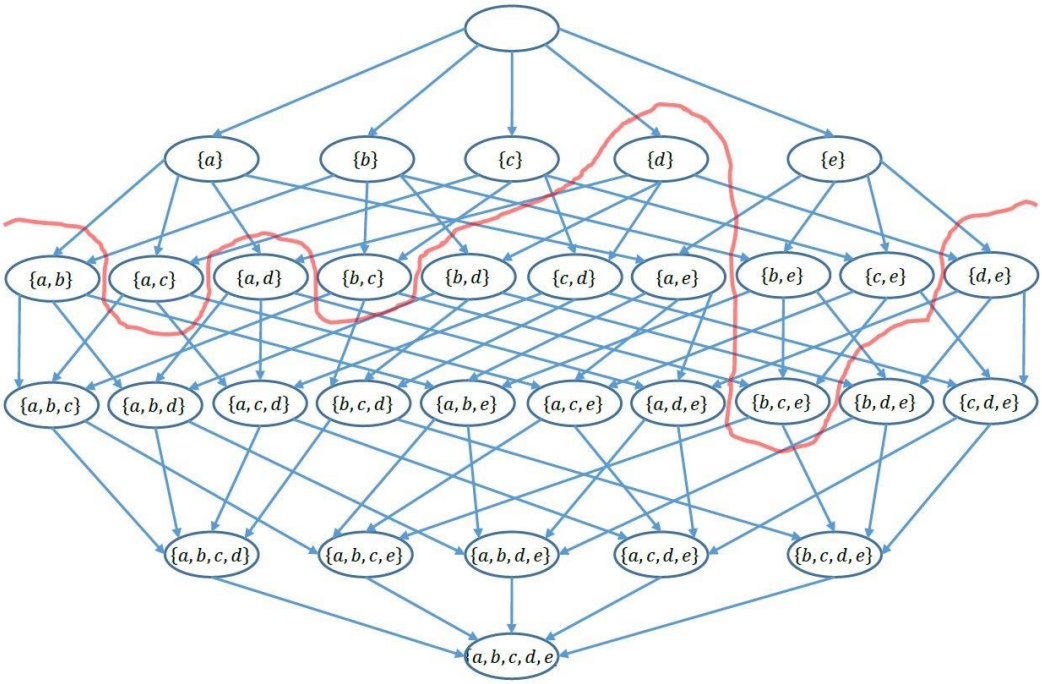
Sık öğe-kümelerini bulmak için literatürde etkili yöntemler geliştirilmiştir. Bu algoritmalar, öğe-seti uzayındaki olası tüm öğe-kümelerini araştırmaktan kaçınarak sık öğe-kümelerini olabildiğince etkili bir şekilde bulmaya çalışırlar. Bunlardan bazıları, Apriori, FP-Growth, ECLAT, H-Mine ve LCM algoritmalarıdır. Tüm bu

algoritmalar, aynı veritabanı ve $minsup$ değerine karşı aynı sık öğe-seti ve destek değeri çıkışını üretirler. Bu algoritmalar arasındaki farklar algoritmaların stratejileri ve kullandıkları veri yapılarıdır. Daha özele inilecek olursa, FIM algoritmaları şu şekilde ayrılırlar:

1. Hangi tip arama yöntemini kullanmaktadır? İki tip arama yöntemi vardır. genişlik öncelikli arama (breadth-first search) ve derinlik öncelikli arama (depth-first search)
2. Kullanılan veritabanının tipi. Yatay (horizontal) veya dikey (vertical) olabilir.
3. Arama uzayında aranacak bir sonraki öğe-setinin oluşturulma biçimi.
4. Bir öğe-setinin sık olup olmadığını anlamak için destek (support) değerinin nasıl hesaplandığı.

4.1.1 Genişlik Öncelikli Arama (Breadth-First Search)

Varsayalım ki, veritabanında m farklı öğe bulunsun. Bir breadth-first search tipindeki algoritma (aynı zamanda level-wise algoritma da denir) öğe-seti uzayındaki aramayı şu şekilde yapar: Önce 1 uzunluklu öğe-kümelerini yani 1-öğe-kümelerini (1-itemsets) bulur. Ardından, 2 uzunluklu öğe-kümelerini yani 2-öğe-kümelerini (2-itemsets) bulur ve bu şekilde m-öğe-kümelerine (m-itemsets) kadar devam eder. Örnek olarak, Tablo 4.1 ile verilen veritabanına ilişkin olası tüm öğe-kümelerinin arama uzayı Şekil 4.1'de görülmektedir (İplikçi 2020). Şekilde bu arama uzayı bir Hasse diyagramı şeklinde verilmiştir. Bir Hasse diyagramı, ancak ve ancak $X \subseteq Y$ ve $|X| + 1 = |Y|$ ise, X öğe-setinden Y öğe-setine bir ok çizer.

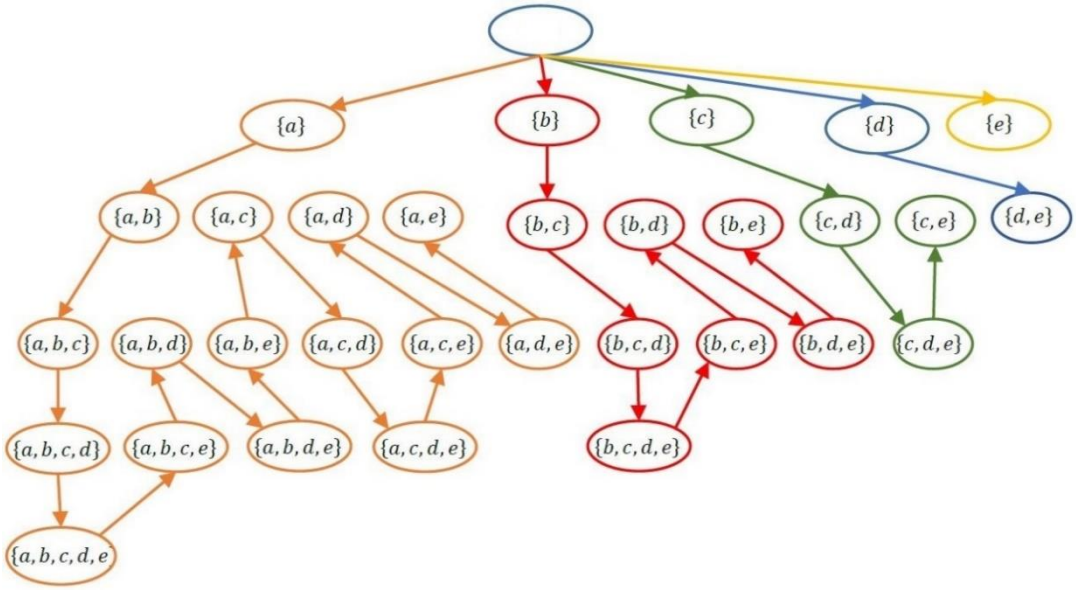


Şekil 4.1: $\boldsymbol{I = \{a, b, c, d, e\}}$ kümesi için genişlik öncelikli arama uzayı (Hasse Diyagramı).

4.1.2 Derinlik Öncelikli Arama (Depth-First Search)

Bilindiği gibi, bir breadth-first search algoritması ilk olarak 1-öğe kümeleri olan a, b, c, d ve e öğe-kümelerini bulur, ardından $\{a, b\}, \{a, c\}, \{a, d\}$ gibi 2-öğe-kümelerini bulur, sonra 3-öğe-kümelerini bulur ve bu işlem tüm öğeleri içeren ve son öğe-seti olan $\{a, b, c, d, e\}$ öğe-setine kadar devam eder. Diğer taraftan, tipik bir depth-first search algoritması her bir 1-öğe-setinden başlar ve sonra tekrarlı bir şekilde (recursively) o anki öğe-setine öğeler ekleyerek daha büyük öğe-kümeleri oluşturur. Örnek olarak, Tablo 4.1 ile verilen örnekteki veritabanı ele alınırsa, tipik bir depth-first search algoritması, Şekil 4.2'de görüldüğü gibi, öğe-kümelerini aşağıdaki sıra ile oluşturur (İplikçi 2020).

$\{a\}, \{a, b\}, \{a, b, c\}, \{a, b, c, d\}, \{a, b, c, d, e\}, \{a, b, c, e\}, \{a, b, d\}, \{a, b, d, e\}, \{a, b, e\},$
$\{a, c\}, \{a, c, d\}, \{a, c, d, e\}, \{a, c, e\}, \{a, d\}, \{a, d, e\}, \{a, e\}, \{b\}, \{b, c\}, \{b, c, d\},$
$\{b, c, d, e\}, \{b, c, e\}, \{b, d\}, \{b, d, e\}, \{b, e\}, \{c\}, \{, d\}, \{c, d, e\}, \{c, e\}, \{d\}, \{d, e\}, \{e\}.$



Şekil 4.2: $\boldsymbol{I = \{a, b, c, d, e\}}$ kümesinin derinlik öncelikli arama uzayı.

4.1.3 Arama Uzayı Daraltma (Search Space Pruning)

Etkili bir FIM algoritması geliştirmek için, algoritmanın, tüm arama uzayını taramasından kaçınması oldukça önemlidir. Çünkü önceden de belirtildiği gibi $2^m - 1$ farklı öğe-seti barındıran arama uzayı çok büyük olabilir. Arama uzayını daraltmak ya da küçültmek için, bazı arama uzayı daraltma teknikleri (search space pruning techniques) kullanılmaktadır. FIM probleminde, arama uzayını daraltmak için en kritik tespit şudur:

Destek (support) monotone bir ölçüttür, yani X ve Y gibi iki öğe-seti için, eğer $X \subset Y$ ise, o zaman $AbsSupp(X) \geq AbsSupp(Y)$ olacaktır. Bunun anlamı şudur: eğer bir öğe-seti sık değilse (infrequent), o zaman onun tüm üst-öğe-kümeleri (supersets) de sık olmayacaktır ve artık bu üst-öğe-kümelerinin araştırılmasına gerek yoktur. Örnek olarak, $minsup = 3$ olsun ve $\{a, b\}$ öğe-setinin destek değeri 2 olduğu için sık bir öğe-seti değildir. Dolayısıyla, onun üst-öğe-setleri olan $\{a, b\}, \{a, b, c\}, \{a, b, d\}, \{a, b, e\}, \{a, b, c, d\}, \{a, b, c, e\}, \{a, b, d, e\}$ ve $\{a, b, c, d, e\}$ öğe-setleri de sık olmayacaktır ve bunların araştırılmasına gerek yoktur. Bu özelliğe, downward-closure property, anti-monotonicity property ve Apriori property gibi isimler verilmektedir. Downward-closure property (aşağı doğru kapanma özelliği) arama uzayını önemli ölçüde daraltmaktadır. Bunu, Tablo 4.1 ile verilen veritabanı örneği ile görmek mümkündür. Normalde 31 öğe-setinden oluşan arama uzayı,

downward-closure property dikkate alındığında, Şekil 4.1'den de görüleceği gibi, sadece 9 öğe-setine indirgenmektedir.

4.2 Sık Öğeseti Madenciliği (FIM) Algoritmaları

4.2.1 Apriori Algoritması

Yatay (horizontal) veritabanı ve genişlik öncelikli arama yaklaşımı kullanan Apriori algoritması, literatürdeki ilk FIM algoritmasıdır (Agrawal ve Srikant 1994). Giriş olarak işlem veritabanı (transaction database) ve $minsup$ eşik değerini alır ve buna göre sık öğe-setlerini ve bunların destek değerlerini çıkış olarak döndürür. Apriori algoritması Tablo 4.1'de verildiği gibi, yatay veritabanı da (horizontal database) adı verilen standart veritabanı gösterilimini kullanır.

Apriori algoritmasının sözde-kodu (pseudocode) Algoritma 4.1'de verilmiştir (Agrawal ve Srikant 1994). Apriori algoritması ilk olarak, satır 1'de görüldüğü gibi, her bir öğenin değerini hesaplamak için veritabanını tarar. Ardından, buradan gelen sonuçlara göre, sık 1-öğe-setlerini (frequent 1-itemsets) bulur, bu öğe-setleri F_1 ile gösterilmektedir. Ardından Apriori algoritması, satır 4-10'da görüldüğü gibi, daha büyük öğe-setlerini oluşturmak için tekrarlı bir şekilde genişlik öncelikli arama yapar. Bu arama esnasında, F_{k-1} ile gösterilen $k-1$ uzunluklu sık öğe-setlerini kullanarak, C_k ile gösterilen k-uzunluklu potansiyel sık öğe-seti aday kümesini oluşturur. F_{k-1} kümesinden C_k kümesini oluşturmak için, satır 5'te görüldüğü gibi, F_{k-1} kümesindeki, bir öğe hariç diğer tüm öğeleri aynı olan $k-1$ uzunluklu öğe-seti çiftleri birleştirilerek C_k kümesi oluşturulur. Örnek olarak, F_1 kümesi içerisinde yer alan $\{a\},\{b\},\{c\}$ ve $\{e\}$ öğe-setleri birleştirilerek 2-uzunluklu aday öğe-setleri barındıran C_2 kümesi $\{a,b\},\{a,c\},\{a,e\},\{b,c\},\{b,e\}$ ve $\{c,e\}$ şeklinde elde edilir. Devam edersek, C_k kümesi oluşturulduktan sonra, Apriori algoritması, C_k kümesinin $(k-1)$ uzunluklu her bir alt kümesinin sık olup olmadığına bakar. Eğer, C_k kümesindeki k-uzunluklu bir X öğe-setinin $(k-1)$ uzunluklu alt-kümelerinden herhangi biri sık değilse, o zaman, downward-closure property özelliğine göre, X öğe-seti de sık değildir ve C_k kümesinden çıkarılır. Bu işlem, satır 7'de görüldüğü

gibi, C_k kümesindeki tüm aday öğe-setleri için yapılır. Satır 8'de görüldüğü gibi, $minsup$ eşiğini geçen her bir aday öğe-seti F_k kümesine alınır. Artık yeni aday öğe-seti bulunmayana kadar bu işleme devam edilir. Son olarak, tüm sık öğe-setleri ve destek değerleri çıkış olarak döndürülür.

Algoritma 4.1: Apriori algoritması.

Girişler: D: yatay işlem veritabanı, $minsup$: kullanıcı tanımlı eşik değer

Çıkışlar: F: sık öğe-seti kümesi

1 I içindeki tüm öğelerin destek değerini hesaplamak için veritabanı taranır
2 $F_1 = \{i | i \in I \wedge \sup(\{i\}) \geq minsup\}$ F_1: sık 1-öğe-setleri
3 $k = 2$
4 **while** $F_k \neq \emptyset$
5 $C_k = CandidateGeneration(F_{k-1})$ C_k: aday k-öğeleri
6 F_{k-1} içerisinde bulunmayan $(k-1)$ uzunluklu öğe-kümelerini içeren her aday $X \in C_k$ kaldırılır.
7 Her $X \in C_k$ adayının desteği hesaplanmak üzere veri tabanı taranır.
8 $F_k = \{X | X \in C_k \wedge \sup(X) \geq minsup\}$ F_k: sık k-öğe-setleri
9 $k \leftarrow k + 1$
10 end
11 $F = \bigcup F_k$

4.2.2 ECLAT Algoritması

ECLAT (Equivalence Class Transformation) algoritması dikey veritabanı (vertical database) ve hafızayı daha iyi kullanmak için derinlik öncelikli arama yaklaşımını kullanır (Zaki 2000). Dikey veritabanında, bir öğenin bulunduğu işlemlerin listesi vardır. Örnek olarak, Tablo 4.1'deki gibi verilen bir yatay veritabanını ele alalım:

X gibi bir öğe-seti için bu öğeyi içeren işlemlerin listesine X öğe-setinin işlem-listesi (TID-list) denir ve $tid(x)$ ile gösterilir. Tablo 4.1 ile verilen veritabanının dikey gösterimi Tablo 4.2'deki gibidir.

Tablo 4.2: Dikey işlem veritabanı.

X	$tid(X)$
$\{a\}$	$\{T_1, T_3, T_5\}$
$\{b\}$	$\{T_2, T_3, T_4, T_5\}$
$\{c\}$	$\{T_1, T_2, T_3, T_5\}$
$\{d\}$	$\{T_1\}$
$\{e\}$	$\{T_2, T_3, T_4, T_5\}$

Bu dikey veritabanı, verilen yatay veritabanı sadece bir kez taranarak elde edilebilir. Tabi, bunun tersi de mümkündür, yani, dikey veritabanından yatay veritabanına geçmek de mümkündür. Dikey veritabanı, iki özelliğinden dolayı veri madenciliğinden çokça kullanılmaktadır: İlki, X ve Y gibi herhangi iki öğe-seti için, bu ikisinin birleşiminden oluşan $X \cup Y$ öğe-seti, veritabanı tekrar taranmadan, $tid(X)$ ve $tid(Y)$ kümelerinin kesişiminden bulunabilir, yani,

$$tid(X \cup Y) = tid(X) \cap tid(Y) \tag{4.3}$$

Örnek olarak, $\{a\}$ ve $\{b\}$ öğelerinden oluşan $\{a, b\}$ öğe-setinin işlem-listesi şu şekilde bulunur:

$$tid(\{a\} \cup \{b\}) = tid(\{a\}) \cap tid(\{b\}) = \{T_1, T_5\}$$

İkinci özellik, X öğe-setinin destek değeri, veritabanı tekrar taranmadan, işlem-listesi olan $tid(X)$ kullanılarak şu şekilde bulunabilir:

$$\sup(X) = |tid(X)| \tag{4.4}$$

Örnek olarak, $\{a, b\}$ öğe-setinin destek değeri şu şekilde bulunur:

$$\sup(\{a, b\}) = |tid(\{a, b\})| = 2$$

Böylece, bu iki özelliği kullanarak ECLAT algoritması gibi dikey veritabanı kullanan algoritmalar veritabanını sadece bir kez tarayarak ilk işlem-listelerini oluştururlar (İplikçi 2020). Artık bundan sonra, aday öğe-seti oluşturma ve destek hesabı tamamen bu dikey veritabanından gerçekleştirilir. ECLAT algoritmasının sözde-kodu Algoritma 4.2'de verilmiştir (Zaki 2000).

Algoritma 4.2: ECLAT algoritması.

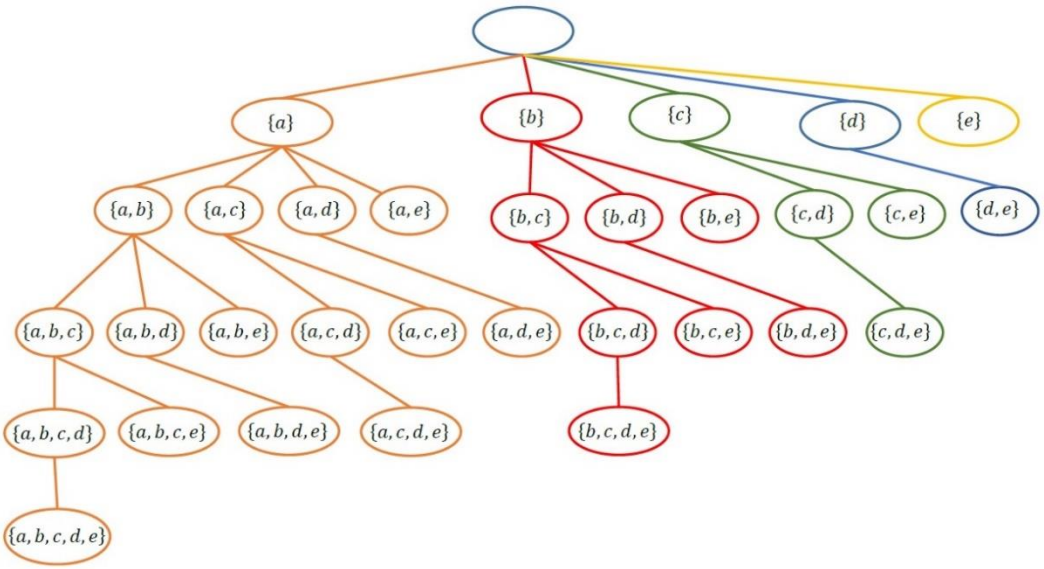
Girişler: R: İşlem-listeleri ile birlikte bir öğe-seti kümesi,
$minsupp$: kullanıcı tanımlı eşik değeri

Çıkışlar: sık öğe-seti kümesi

1 **for** her öğe-seti $X \in R$, $|tid(X)| \geq minsupp$ olacak şekilde seçilir
2 Çıkış X — X sık öğe-setidir
3 $E = \emptyset$
4 **for** X ile sonuncusu hariç tüm öğeleri aynı olan $Y \in R$ öğeseti
5 $tid(X \cup Y) = tid(X) \cap tid(Y)$ — $X \cup Y$'nin işlem listesi
6 **if** $|tid(X \cup Y)| \geq minsupp$ **then** — Eğer $X \cup Y$ sık-öğe ise
7 $E = E \cup \{X \cup Y\}$
8 **end**
9 $ECLAT(E, minsupp)$ — ECLAT Algoritmasını E için yinelemeli çağır.
10 **end**
11 **end**

ECLAT algoritmasının girişi, dikey veritabanı R (Tablo 4.2'de görüldüğü gibi) ve $minsupp$ eşik değeridir. ECLAT algoritması, dikey veritabanı R 'nin içerisinde sık olan her bir X öğe-setini dikkate alarak bir döngü gerçekleştirir (satır 2-10). X öğe-seti ilk çıktıdır. Daha sonra, X öğe-setine bir öğe ekleyerek genişletilerek sık öğe-seti bulmak için arama yapılır. Bu şu şekilde yapılır: R içerisinde, X öğe-setiyle biri hariç tüm elemanları aynı olan her bir Y öğe-setini birleştirerek $X \cup Y$ öğe-seti elde edilir (satır 4-10). Örnek olarak, eğer $X = \{a\}$ ise, o zaman ECLAT X öğe-setini $\{b\}, \{c\}, \{d\}$ ve $\{e\}$ öğe-setleriyle birleştirerek sırasıyla $\{a, b\}, \{a, c\}, \{a, d\}$ ve $\{a, e\}$ öğe-setlerini oluşturacaktır. Bu işlem esnasında, $X \cup Y$ 'nin işlem-listesi $tid(X \cup Y) = tid(X) \cap tid(Y)$ ile bulunur (satır 6). Eğer $X \cup Y$ sık bir öğe-setiyse o zaman, X öğe-setinin genişletilmiş sık öğe-seti kümesi olan E'ye eklenir (satır 6 ve 7). Ardından, ECLAT algoritması, $X \cup Y$'nin tüm uzantılarını bulmak için E kümesi ile birlikte kendi kendini çağırır. Bu döngü, R'nin içindeki tüm öğe setleri için tekrarlanır. Algoritma bittiğinde elde edilen sık öğe-setleri ve onların destek değerleri

çıkışı oluşturur. ECLAT algoritmasının arama uzayı Şekil 4.3'te görülmektedir (İplikçi 2000).



Şekil 4.3: ECLAT algoritması için $\boldsymbol{I} = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}\}$ kümesinin depth-first search arama uzayı.

ECLAT algoritması çıktıları derinlik öncelikli arama yaklaşımındaki sıraya göre ürettiği için bir derinlik öncelikli arama algoritması olarak görülebilir. Veritabanını çok sayıda taramadığı için Apriori algoritmasından her zaman daha hızlıdır. Ancak, ECLAT algoritmasının bazı dezavantajları vardır. ECLAT, aday öğe-setlerini bulurken veritabanını taramadığı için veritabanında bulunmayan öğe-seti adayları oluşturabilir ki bu da zaman kaybına yol açar. Diğer bir dezavantajı, her ne kadar işlem-listeleri (tid-list) faydalı olsa da, özellikle yoğun (tüm öğelerin hemen hemen her işlemde yer aldığı veritabanları) veritabanları için hafızada çok yer tutar. Yine de, işlem-listelerinin (tid-list) boyutunu azaltan yapıların literatürde önerildiğini söylemekte fayda vardır. İşlem-listelerini bit-vektörleri şeklinde kodlayarak boyutları azaltılabilir ki böylece işlem hızı artar. İşlem-listeleri, ayrıca, Apriori-TID gibi genişlik öncelikli arama algoritmalarında da kullanılabilir.

4.3 İlişkisel Kural Madenciliği (Association Rule Mining – ARM)

FIM algoritmaları ile bulunan sık öğe-setleri arasındaki ilişkileri bulmak için İlişkisel Kural Madenciliği (Association Rule Mining – ARM) yöntemleri kullanılmaktadır (İplikçi 2020). Bu kurallar sayesinde, örüntüler arasındaki ilişkileri

belli sayısal değerlerle ifade etmek mümkün olmaktadır. Örnek olarak, Tablo 4.1'deki gibi verilen bir yatay veritabanını ele alalım:

Bu veritabanı ve $minsupp = 0.4$ için sık öğe-setleri Apriori ve ECLAT gibi bir algoritmayla aşağıdaki şekilde bulunmuş olsun:

#1	$\{a\}$	$RelSupp$: 0.6	#8	$\{b,c\}$	$RelSupp$: 0.6
#2	$\{b\}$	$RelSupp$: 0.8	#9	$\{b,e\}$	$RelSupp$: 0.8
#3	$\{c\}$	$RelSupp$: 0.8	#10	$\{c,e\}$	$RelSupp$: 0.6
#4	$\{e\}$	$RelSupp$: 0.8	#11	$\{a,b,c\}$	$RelSupp$: 0.4
#5	$\{a,b\}$	$RelSupp$: 0.4	#12	$\{a,b,e\}$	$RelSupp$: 0.4
#6	$\{a,c\}$	$RelSupp$: 0.6	#13	$\{a,c,e\}$	$RelSupp$: 0.4
#7	$\{a,e\}$	$RelSupp$: 0.4	#14	$\{b,c,e\}$	$RelSupp$: 0.6
			#15	$\{a,b,c,e\}$	$RelSupp$: 0.4

Bu şekilde verilen sık öğe-setleri arasındaki ilişkiyi ifade etmek için şu notasyon kullanılacaktır: Verilen bir veritabanı ve $minsupp$ değeri için, X ve Y, destek değerleri sıfır olmayan, yani $Supp(X) \neq 0$ ve $Supp(Y) \neq 0$, ve kesişim kümesi boş-küme ($X \cap Y \neq 0$) olan herhangi iki sık öğe-seti olmak üzere bu iki öğe-seti arasındaki ilişkisel kural (rule veya implication) $X \rightarrow Y$ notasyonu ile gösterilir ve "X öğe-setinin olduğu bir işlemde Y öğe-seti de vardır" anlamına gelir. Bu notasyon X öncül (premise) ve Y ise ardıl (consequent) olarak adlandırılmaktadır.

Örneğin, yukarıda bulunan sık öğe-setleri için $\{a,e\} \rightarrow \{c\}$ gibi bir kural, "$\{a,e\}$ öğe setinin olduğu bir işlemde $\{c\}$ öğe-seti vardır" anlamına gelir.

Ancak böyle bir kural için akla şu sorular gelmektedir: $\{a,e\}$ öğe-setinin olduğu bir işlemde $\{c\}$ öğe-setinin de bulunması ne kadar olasıdır? Biz bu kurala ne kadar güvenebiliriz? Burada, kuralın güvenilirliğini ifade etmek için olasılıksal bir değer vermek gerekir ki bu da bizi kural ile ilgili olarak "güven (confidence) kavramına götürür.

$X \rightarrow Y$ şeklindeki bir kuralın güvenirliğini ifade etmek için, literatürde "güven (cofidence)" kavramı önerilmiştir. $X \rightarrow Y$ şeklindeki bir kuralın güven değeri, $Conf(X \rightarrow Y)$ ile gösterilir ve şu şekilde hesaplanır:

$$Conf(X \rightarrow Y) = \frac{Supp(X \cup Y)}{Supp(X)} \tag{4.5}$$

Güven değerinin hesabında, destek değeri $Supp(.)$ olarak bağıl destek $RelSupp$ kullanılabileceği gibi, mutlak destek $AbsSupp$ değeri de kullanılabilir ve aynı sonucu üretirler. $Conf(X \rightarrow Y)$ güven değeri, X ve Y öğe-setlerinin birleşiminden (union) oluşan $X \cup Y$ öğe-setinin destek değerinin, X öğe-setinin destek değerine oranı ile hesaplanır. Downward-closure özelliğinden hatırlanacağı gibi, her zaman $Supp(X \cup Y) \leq Supp(X)$ olmaktadır. Dolayısıyla, güven değeri her zaman [0 1] aralığında olacaktır, yani,

$$0 \leq Conf(X \rightarrow Y) \leq 1 \tag{4.6}$$

Bilindiği gibi, X öğe-setinin sık öğe-seti olabilmesi için $minsupp \leq Supp(X)$ şartının sağlanması gerekir. Benzer şekilde, $X \rightarrow Y$ gibi bir ilişkinin bir ilişkisel kural (association rule) olabilmesi için şu iki koşulu aynı anda sağlaması gerekir:

$$\begin{aligned} &(1)\ minsupp \leq Supp(X \cup Y) \\ &(2)\ minconf \leq Conf(X \cup Y) \end{aligned} \tag{4.7}$$

Burada, $minconf$ değeri, kuralın sahip olması gereken en küçük güven değeridir. Bu güven değerinin altında bir güven değerine sahip bir ilişki, ilişkisel kural olmaz. Böylece, işlemlerden (transactions) oluşan bir veritabanından ilişkisel kurallar çıkarabilmek için hem minimum destek değeri $minsupp$, hem de minimum güven değeri $minconf$ verilmesi gerekir ki buna literatürde destek-güven çerçevesi (support-confidence framework) denmektedir.

Örneğe devam edilirse, $minsupp = 0.4$ değeri için bulunan sık öğe-setlerini ele alalım. Bu sık öğe-setleri arasındaki ilişkisel kıralları bulmak için, minimum güven değeri $minconf = 0.5$ olarak belirlenirse, çok sayıda ilişkisel kural bulunabilir. Bunlardan bir tanesi de $\{a, c\} \rightarrow \{b, e\}$ şeklindedir. Bu kuralın öncülünün destek değeri $Supp(\{a, c\}) = 0.6$, ardılının destek değeri $Supp(\{b, e\}) = 0.8$ şeklinde olup bunların birleşimi olan öğe-setinin destek değeri $Supp(\{a, b, c, e\}) = 0.4$ şeklinde olup kuralın güven değeri Denklem (4.8)'deki gibidir.

$$Conf(\{a,e\} \rightarrow \{b,e\}) = \frac{Supp(\{a,b,c,e\})}{Supp(\{a,c\})} = \frac{0.4}{0.6} = 0.66 \quad (4.8)$$

Bu kuralın anlamı şudur: "$\{a,c\}$ öğe-setinin olduğu bir işlemde, $\{b,e\}$ öğe-setinin de bulunma olasılığı 0.66'dır." Görüldüğü gibi, güven değeri, kuralın hangi olasılıkla geçerli olduğunu göstermektedir; dolayısıyla, uygulamalarda olabildiğince yüksek seçilir. Bu örnekte, minimum güven değeri $minconf = 0.95$ olarak seçilirse aşağıdaki Tablo 4.3'te verilen kurallar elde edilir.

Tablo 4.3: Örnek kural tablosu.

Öncül		**Ardıl**	**Güven**
$\{e\}$	$\rightarrow$	$\{b\}$	1.00
$\{b\}$	$\rightarrow$	$\{e\}$	1.00
$\{a\}$	$\rightarrow$	$\{c\}$	1.00
$\{c,b\}$	$\rightarrow$	$\{e\}$	1.00
$\{e,c\}$	$\rightarrow$	$\{b\}$	1.00
$\{a,b\}$	$\rightarrow$	$\{e,c\}$	1.00
$\{a,e\}$	$\rightarrow$	$\{c,b\}$	1.00
$\{a,e\}$	$\rightarrow$	$\{c\}$	1.00
$\{a,c,b\}$	$\rightarrow$	$\{e\}$	1.00
$\{a,e,b\}$	$\rightarrow$	$\{c\}$	1.00
$\{a,e,c\}$	$\rightarrow$	$\{b\}$	1.00
$\{a,b\}$	$\rightarrow$	$\{e\}$	1.00
$\{a,e\}$	$\rightarrow$	$\{b\}$	1.00
$\{a,b\}$	$\rightarrow$	$\{c\}$	1.00

5. MODELLEME VE ZAMAN SERİSİ TAHMİNİ

5.1 Modelleme Kavramı

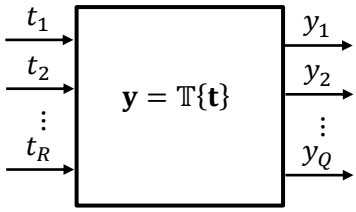
Modelleme kavramı, gerçek dünyadaki bir sürecin davranışının analitik olarak ifade edilemediği, ya da başka bir deyişle temel yasalarla ifade edilemediği durumlarda başvurulan bir yöntemdir. Burada süreç ile, biyolojik, fiziksel, kimyasal, elektriksel, mekanik, meteorolojik, sosyal, finansal vs her türlü dinamik sistem kastedilmektedir. Gerçek dünyada bu tarzda pek çok sistem bulunmaktadır. Örneğin, hava durumu analitik olarak ifade edilemeyecek kadar karmaşık dinamik bir sistemdir ve dolayısıyla modellenmesi gerekir. Benzer şekilde, borsa da içinde çok sayıda değişken barındıran karmaşık bir sistemdir. Bunlar kadar karmaşık olmasa da diğer alanlarda da karşımıza pek çok sistem çıkmaktadır ki bunların analitik olarak ifade edilmeleri mümkün değildir. O yüzden, gerekli durumlarda, bu tip süreçlerden yeterli miktarda veri toplanıp bu verilerle güvenilir bir model elde etme yoluna gidilir (İplikçi 2018).

Veriler, modellenecek sürecin doğasına bağlı olarak en genel halde karşımıza iki değişik biçimde çıkmaktadır:

5.2 Veri Tipleri

5.2.1 Giriş-Çıkış Verisi

Giriş-çıkış verisi, genellikle dinamik bir sistemin modellenmesinde karşımıza çıkan bir veri tipidir. En genel halde bir sistemi Şekil 5.1'deki gibi bir şekilde gösterelim (İplikçi 2018).



Şekil 5.1: Giriş-çıkış verisi.

Burada, $i = 1, \dots, R$ olmak üzere t_i 'ler sistemin giriş işaretlerini temsil etmektedir ve daha kolay bir gösterilim için tüm girişler $\mathbf{t} = [t_1 \;\; t_2 \;\; \cdots \;\; t_R]^T$ şeklinde bir vektörle gösterilmektedir. Benzer şekilde, $i = 1, \dots, Q$ olmak üzere y_i 'ler de sistemin çıkış işaretlerini temsil etmekte olup daha kolay bir gösterilim için tüm çıkışlar $\mathbf{y} = [y_1 \;\; y_2 \;\; \cdots \;\; y_Q]^T$ şeklinde bir vektörle gösterilmektedir. Diğer taraftan, $\mathbb{T}\{\,\}$ operatörü ise sistemin giriş çıkış ilişkisini temsil etmektedir. Yani, **t** gibi bir girişe karşı sistemin **y** gibi bir çıkışı nasıl üreteceğini ifade etmektedir. $\mathbb{T}\{\,\}$ operatörü sistemin doğasına bağlı olarak bir diferansiyel denklem veya bir fark denklemi olabilir. Belli bir fiziksel süreç için modelleme ihtiyacını ortaya çıkaran sorun ise bu $\mathbb{T}\{\,\}$ operatörünün matematiksel ifadesinin bilinmemesidir. Modelleme yapabilmek için sistemin giriş ve çıkış büyüklükleri belli aralıklarla ölçülerek kaydedilir ve Tablo 5.1'dekine benzer bir veri seti elde edilir.

Tablo 5.1: Giriş-çıkış veri seti.

$\mathfrak{D}_{\text{MIMO}}$	**Giriş verisi**				**Çıkış verisi**			
i	$\mathbf{t}_i$				$\mathbf{y}_i$			
	t_{i1}	t_{i2}	…	t_{iR}	y_{i1}	y_{i2}	…	y_{iQ}
1	t_{11}	t_{12}	…	t_{1R}	y_{11}	y_{12}	…	y_{1Q}
2	t_{21}	t_{22}	…	t_{2R}	y_{21}	y_{22}	…	y_{2Q}
⋮	⋮				⋮			
N	t_{N1}	t_{N2}	…	t_{NR}	y_{N1}	y_{N2}	…	y_{NQ}

Burada, N toplam veri sayısıdır. Bu veri seti $\mathfrak{D}_{\text{MIMO}}$ ile gösterilmiştir ki burada MIMO ifadesi veri setinin Çok Girişli-Çok Çıkışlı (Multiple Input-Multiple Output) olduğunu göstermektedir.

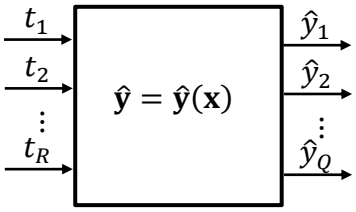
5.3 Modelleme ve Model Seçme

Tablo 5.2'de görüldüğü gibi, $\mathbf{t}_i \in \mathbb{R}^R$ ve $\mathbf{y}_i \in \mathbb{R}^Q$ olmak üzere $\mathfrak{D}_{\text{MIMO}} = \{\mathbf{t}_i, \mathbf{y}_i\}_{i=1}^N$ şeklinde verilen bir MIMO veriyi ele alalım:

Tablo 5.2: MIMO giriş-çıkış verisi.

$\mathfrak{D}_{\text{MIMO}}$	**Giriş Verisi**				**Çıkış Verisi**				**Model Çıkışı**			
i	$\mathbf{t}_i$				$\mathbf{y}_i$				$\hat{\mathbf{y}}_i$			
	t_{i1}	t_{i2}	...	t_{iR}	y_{i1}	y_{i2}	...	y_{iQ}	$\hat{y}_{i1}$	$\hat{y}_{i2}$	...	$\hat{y}_{iQ}$
1	t_{11}	t_{12}	...	t_{1R}	y_{11}	y_{12}	...	y_{1Q}	$\hat{y}_{11}$	$\hat{y}_{12}$	...	$\hat{y}_{1Q}$
2	t_{21}	t_{22}	...	t_{2R}	y_{21}	y_{22}	...	y_{2Q}	$\hat{y}_{21}$	$\hat{y}_{22}$	...	$\hat{y}_{2Q}$
⋮	⋮				⋮				⋮			
N	t_{N1}	t_{N2}	...	t_{NR}	y_{N1}	y_{N2}	...	y_{NQ}	$\hat{y}_{N1}$	$\hat{y}_{N2}$	...	$\hat{y}_{NQ}$

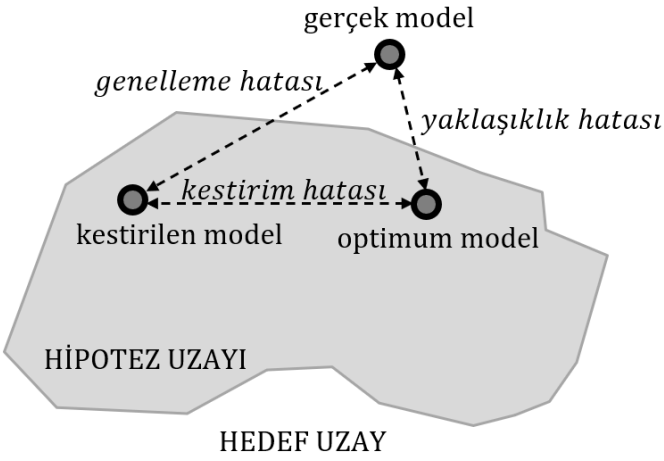
Bu veriler, $\hat{\mathbf{y}} = \hat{\mathbf{y}}(\mathbf{x})$ gibi bir yapı ile modellenecektir. Öncelikle elde etmek istediğimiz modelin yapısına bir göz atalım: En genel halde model Şekil 5.2'teki gibi gösterilebilir (İplikçi 2018).



Şekil 5.2: MIMO model.

Tablo 5.2'de $i = 1, \dots, R$ olmak üzere t_i'ler modelin giriş işaretlerini temsil etmektedir ve daha kolay bir gösterilim için tüm girişler $\mathbf{t} = [t_1 \;\; t_2 \;\; \cdots \;\; t_R]^T$ şeklinde bir vektörle gösterilmektedir. Benzer şekilde, $i = 1, \dots, Q$ olmak üzere $\hat{y}_i$'ler de modelin çıkış işaretlerini temsil etmekte olup daha kolay bir gösterilim için tüm çıkışlar $\hat{\boldsymbol{y}} = [\hat{y}_1 \;\; \hat{y}_2 \;\; \cdots \;\; \hat{y}_Q]^T$ şeklinde bir vektörle gösterilmektedir. Diğer taraftan, $\hat{\mathbf{y}}(\mathbf{x})$ operatörü ise modelin giriş-çıkış ilişkisini temsil etmektedir ve buradaki $\mathbf{x}$ vektörü de modelin parametrelerini temsil eden parametre vektörüdür. Modelleme problemi, bu giriş-çıkış verisini en iyi şekilde temsil edecek modelin giriş-çıkış ilişkisinin $\hat{\mathbf{y}}(\mathbf{x})$ ve bu ilişkideki parametrelerin $(\mathbf{x})$ belirlenmesidir.

Verilen bir giriş-çıkış veri setini en iyi temsil eden model dendiğinde aklımıza "verileri iyi bir genelleme yaparak giriş-çıkış ilişkisini en iyi şekilde öğrenen en basit model" gelmelidir. Aslında bu durum Şekil 5.3 ile daha iyi anlaşılabilir (İplikçi 2018).



Şekil 5.3: Hedef uzayı-hipotez uzayı.

Şekil 5.3'e göre, gözlemlenen sürecin gerçek modeli hedef uzayı (target space) denilen bir uzayda bulunmaktadır. Bu sürecin zamanda belli aralıklarla gözlenerek elde edilen verilerin ışığı altında bir model (hipotez) geliştirilecektir ancak bu model sadece hipotez uzayında (hypothesis space) yer alabilir. Hipotez uzayı ise hedef uzayın bir alt-uzayıdır ve maalesef gerçek model hipotez uzayının çok dışındadır. Bunun nedeni süreçten kısıtlı sayıda veri toplanması ve model geliştirilirken yapılan bazı varsayımlar ve yaklaşıklıklardır. Neticede, elde edilebilecek en optimum model bile gerçek model olmayacak ve gerçek modelle arasında "yaklaşıklık hatası" bulunan bir model olacaktır. Hipotez uzayında bulunan modellerden optimum olanı bulmak için geliştirilmiş yöntemler (model selection methods) kullanılsa bile neticede elde edilen model (kestirilen model) optimum modelden farklı olacaktır. Kestirilen model ile optimum model arasındaki hataya da "kestirim hatası" denmektedir. En iyi modeli seçmek için kullanılacak yöntemlerle genelleme hatası en aza indirilmeye çalışılacaktır. Yapısal Risk Minimizasyonu (Structural Risk Minimization) prensibine göre bir modelin yaptığı genelleme hatası ($R[\hat{\mathbf{y}}]$) Denklem 5.1'deki gibi tanımlanmaktadır:

$$R[\hat{\mathbf{y}}] \leq R_{emp}[\hat{\mathbf{y}}] + \Omega\left(\frac{N}{h}\right) \tag{5.1}$$

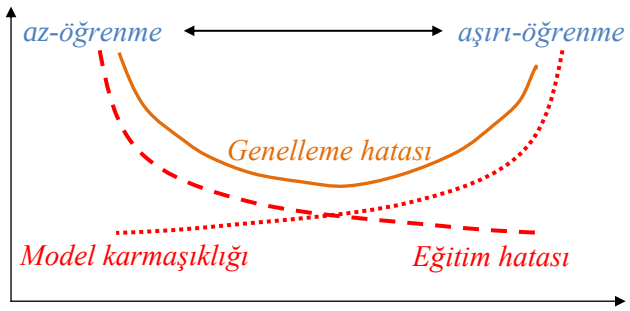
burada $\hat{\mathbf{y}}$ modeli temsil etmektedir ki gerçekte $\hat{\mathbf{y}} = \hat{\mathbf{y}}(\mathbf{x})$ gibi bir yapıdır ve buradaki $\mathbf{x}$ vektörü modelin parametrelerini içeren optimize edilecek parametre vektörüdür. $R_{emp}[\hat{\mathbf{y}}]$ ise,

$$R_{emp}[\hat{\mathbf{y}}] = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{5.2}$$

şeklinde tanımlanan "Ampirik Risk (Emprical Risk)" olup modelin yaptığı eğitim hatasıdır ve limit durumda,

$$\lim_{N\to\infty} R_{emp}[\hat{\mathbf{y}}] = R[\hat{\mathbf{y}}] \tag{5.3}$$

olduğu varsayılmaktadır. Diğer taraftan, $\Omega\left(\frac{N}{h}\right)$ ise modelin karmaşıklığını temsil etmektedir ki buradaki N veri sayısıdır, h ise Vapnik-Chervonenkis boyutu (VC dimension) adı verilen ve modelin öğrenme kapasitesini temsil eden bir büyüklüktür. $R[\hat{\mathbf{y}}]$ büyüklüğünü doğrudan minimize etmek mümkün değildir. Onun yerine $R_{emp}[\hat{\mathbf{y}}] + \Omega\left(\frac{N}{h}\right)$ toplamı minimize edilmeye çalışılır ama bu toplamdaki iki terim aşağıdaki şekilde görüldüğü gibi birbiriyle çelişmektedir. Yani eldeki veriler için $R_{emp}[\hat{\mathbf{y}}]$ büyüklüğünü en az yapacak modelin karmaşıklığı da oldukça fazla olmaktadır. Aradaki ilişkiler Şekil 5.4'te gösterilmektedir (İplikçi 2018).



Şekil 5.4: Hatalar arasındaki ilişkiler.

Modelin karmaşıklığı ile amprik hata arasında bir denge kurmak için en basit modelden başlanarak her bir modelin amprik hatası hesaplanır ve en az amprik hatayı veren en basit model bulunmaya çalışılır.

Diğer taraftan, $R_{emp}[\hat{\mathbf{y}}]$ büyüklüğünün çok büyük olması zaten istenen bir durum (az-öğrenme) değilken, çok küçük olması durumu (aşırı-öğrenme) da istenen

bir durum değildir. Bunu en iyi şekilde ayarlamak için kullanılacak yöntemlerin en yaygını,

$$N = N_{tra} + N_{val} + N_{tst} \tag{5.4}$$

olacak şekilde verilen rasgele bir şekilde eğitim (training), doğrulama (validation) ve test olarak üç kısma ayrılmasıdır, burada N_{tra}, N_{val} ve N_{tst} sırasıyla eğitim, doğrulama ve test verilerinin sayısıdır. Bu arada, eğitim verilerinin indekslerinin bulunduğu küme TRA, doğrulama verilerinin indekslerinin bulunduğu küme VAL ve test verilerinin indekslerinin bulunduğu küme de TST ile gösterilsin.

Önce eğitim verileri kullanılarak modelin parametreleri bulunur yani model elde edilir. Modelin parametrelerinin bulunması aslında şu şekilde tanımlanabilecek bir kısıtsız optimizasyon problemidir: $\hat{\mathbf{y}}(\mathbf{x})$ modelinin parametreleri olan $\mathbf{x}$ vektörü öyle ayarlanmalı ki modelin her bir eğitim verisi için yaptığı hataların karelerinin toplamı minimum olsun, yani,

$$\begin{aligned} \min_{\mathbf{x}} F_{tra}(\mathbf{x}) &= \sum_{j=1}^{Q} \sum_{i \in \mathrm{TRA}} \left(y_{i,j} - \hat{y}_{i,j}(\mathbf{x}) \right)^2 \\ &= \sum_{j=1}^{Q} \sum_{i \in \mathrm{TRA}} e_{ij}^2(\mathbf{x}) \end{aligned} \tag{5.5}$$

burada her bir veri ve çıkış için modelin yaptığı hata $e_{ij}(\mathbf{x}) = y_{ij} - \hat{y}_{ij}(\mathbf{x})$ şeklindedir.

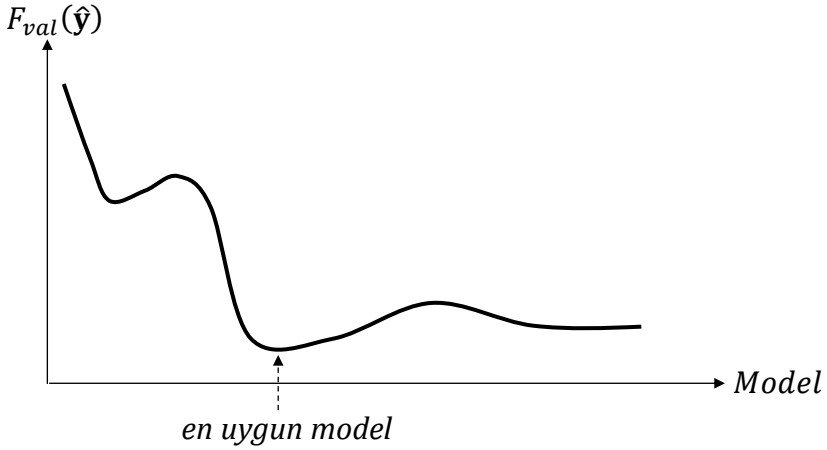
Ardından, eğitim verileriyle elde edilen modelin ne kadar iyi genelleme yapabildiğini anlamak için modele daha önceden hiç kullanılmayan doğrulama verileri uygulanarak modelin doğrulama (validation) performansı bulunur. Doğrulama performansı şu şekilde tanımlanmaktadır:

$$F_{val}(\hat{\mathbf{y}}) = \frac{1}{N_{val}} \sum_{j=1}^{Q} \sum_{i \in \mathrm{VAL}} \left(y_{i,j} - \hat{y}_{i,j}(\mathbf{x}) \right)^2 \tag{5.6}$$

Doğrulama performansı modelin genelleme yeteneğinin sayısal bir göstergesidir. Birden fazla tipte model arasında bir seçim yapılacaksa o zaman her

bir tipe ilişkin en iyi genelleme yapan modellerin test performansına bakılır. Eğer tek bir model tipi varsa test verilerine gerek yoktur, sadece eğitim ve doğrulama olarak ikiye ayrılması yeterlidir.

Sonuç olarak eldeki kısıtlı sayıda veriyle en optimum modeli bulmak için şu yöntem izlenebilir: Tek tip model olduğu varsayımı altında, önce veriler eğitim ve doğrulama olmak üzere ikiye ayrılır. Ardından, aynı eğitim verileriyle elde edilen her bir modelin doğrulama performansı elde edilerek Şekil 5.5'e benzer bir grafik elde edilir (İplikçi 2018).



Şekil 5.5: Validasyona göre en uygun model.

Şekil 6.5'ten de görüldüğü gibi, "en uygun model" olarak, doğrulama hatası yeterince düşük olan en basit model seçilir.

5.3.1 Bir-Boyutlu Veri Modelleme

Eğer veri setinde $R = 1$ ve $Q = 1$ ise o zaman problem çok basit bir form alır. Aşağıdaki tabloda görüldüğü gibi, $t_i, y_i \in \mathbb{R}$ olmak üzere $\mathfrak{D}_{\text{SISO}} = \{t_i, y_i\}_{i=1}^{N}$ şeklinde verilen bir Tek Girişli-Tek Çıkışlı (Single Input-Single Output) veri Tablo 5.3'te gösterilmiştir.

Tablo 5.3: Tek girişli-tek çıkışlı veri.

$\mathfrak{D}_{\text{SISO}}$	**Giriş Verisi**	**Çıkış Verisi**	**Model Çıkışı**
i	t_i	y_i	$\hat{y}_i$
1	t_1	y_1	$\hat{y}_1$

2	t_2	y_2	$\hat{y}_2$
⋮	⋮	⋮	⋮
N	t_N	y_N	$\hat{y}_N$

Tablo 6.4'teki veriler, $\hat{y} = \hat{y}(\mathbf{x})$ gibi bir yapı ile modellenecektir. Bu durumda, ilk olarak model parametrelerinin ayarlanması için eğitim verileri kullanılır. $\hat{y}(\mathbf{x})$ modelinin parametreleri olan **x** vektörü öyle ayarlanmalı ki modelin her bir eğitim verisi için yaptığı hataların karelerinin toplamı minimum olsun, yani,

$$\min_{\mathbf{x}} F_{tra}(\mathbf{x}) = \sum_{i \in \mathrm{TRA}} e_i^2(\mathbf{x}) \tag{5.7}$$

burada, her bir veri için modelin yaptığı hata $e_i(\mathbf{x}) = y_i - \hat{y}_i(\mathbf{x})$ şeklindedir. Ardından elde edilen her bir modelin doğrulama performansı,

$$F_{val}(\hat{\mathbf{y}}) = \frac{1}{N_{val}} \sum_{i \in \mathrm{VAL}} \left(y_{i,j} - \hat{y}_{i,j}(\mathbf{x}) \right)^2 \tag{5.8}$$

ile bulunarak en iyi doğrulama performansını veren en basit model seçilir.

5.3.2 Çok-Boyutlu Veri Modelleme

Tablo 5.4'te görüldüğü gibi, $\mathbf{t}_i \in \mathbb{R}^R$ ve $y_i \in \mathbb{R}$ olmak üzere $\mathfrak{D}_{\mathrm{MISO}} = \{\mathbf{t}_i, y_i\}_{i=1}^N$ şeklinde verilen bir Çok Girişli-Tek Çıkışlı (Multiple Input-Single Output) veriyi ele alalım:

Tablo 5.4: Çok girişli-tek çıkışlı veri.

$\mathfrak{D}_{\mathrm{MISO}}$	**Giriş Verisi**				**Çıkış Verisi**	**Model Çıkışı**
i	$\mathbf{t}_i$				y_i	$\hat{y}_i$
	t_{i1}	t_{i2}	...	t_{iR}		
1	t_{11}	t_{12}	...	t_{1R}	y_1	$\hat{y}_1$
2	t_{21}	t_{22}	...	t_{2R}	y_2	$\hat{y}_2$
⋮	⋮				⋮	⋮
N	t_{N1}	t_{N2}	...	t_{NR}	y_N	$\hat{y}_N$

Tablo 6.4'teki veriler, $\hat{y} = \hat{y}(\mathbf{x})$ gibi bir yapı ile modellenecektir. Bu durumda, Çok Girişli-Tek Çıkışlı (Multiple Input-Single Output) modelleme probleminin tanımı şu şekildeki bir optimizasyon problemi ile yapılabilir: $\hat{y}(\mathbf{x})$ modelinin parametreleri olan $\mathbf{x}$ vektörü öyle ayarlanmalı ki modelin her bir veri için yaptığı hataların karelerinin toplamı minimum olsun, yani,

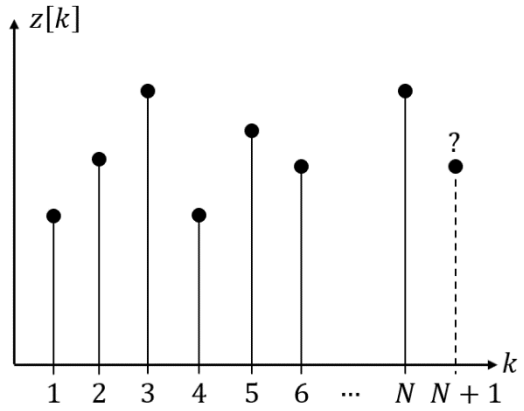
$$\min_{\mathbf{x}} f(\mathbf{x}) = e_1^2(\mathbf{x}) + e_2^2(\mathbf{x}) + \cdots + e_N^2(\mathbf{x}) \tag{5.9}$$

burada, her bir veri için modelin yaptığı hata $e_i(\mathbf{x}) = y_i - \hat{y}_i(\mathbf{x})$ şeklindedir. Eğer istenen çıkış vektörü $\mathbf{y} = [y_1 \quad y_2 \quad \cdots \quad y_N]^T$ ve modelin ürettiği çıkış vektörü $\hat{\mathbf{y}}(\mathbf{x}) = [\hat{y}_1(\mathbf{x}) \quad \hat{y}_2(\mathbf{x}) \quad \ldots \quad \hat{y}_N(\mathbf{x})]^T$ olmak üzere hata vektörü $\mathbf{e}(\mathbf{x}) = \mathbf{y} - \hat{\mathbf{y}}(\mathbf{x})$ şeklinde yazılırsa, problem Denklem (5.10)'daki hale gelir:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \mathbf{e}^T(\mathbf{x})\mathbf{e}(\mathbf{x}) \tag{5.10}$$

5.4 Zaman Serisi

Zaman serisi, genellikle belli bir girişin ve çıkışın olmadığı ve sadece gözlenebilen büyüklüklerin olduğu süreçlerde karşımıza çıkmaktadır. Şekil 5.6'yı ele alalım: burada $z[k]$ büyüklüğü k gözlem anında sürecin gözlenen büyüklüğünün değeridir.



Şekil 5.6: Zaman serisi örneği.

Sürecin gözlenen bu büyüklüğüne örnek olarak borsadaki bir kağıdın günlük değeri, bir barajın su seviyesinin haftalık değeri, kimyasal bir süreçteki bir

elenmentin saatlik miktarı veya biyolojik bir süreçteki bir canlı bakteri sayısının saatlik değerleri vb olabilir. Süreçten eşit aralıklarla N adet gözlem yapılarak henüz gözlenemeyen $N+1$ anındaki değerin tahmin edilmesi problemi zaman serilerinde en çok karşılaşılan problemdir. Bu zaman serisini, $k = 1, \dots, N$ için $z[k] \in \mathbb{R}$ için $z[k] \in \mathbb{R}$ olmak üzere $\mathfrak{D}_{\text{TS}} = \{z[k]\}_{k=1}^{N}$ şeklinde göstermek mümkündür. Burada k zaman indeksini göstermektedir. Bu zaman serisi herhangi bir büyüklüğün herhangi bir zaman aralığında yapılmış ölçümlerle elde edilmiş değerler olabilir. Bu zaman serisini modelleme problemi Tablo 5.5'teki gibi bir giriş-çıkış veri setine dönüştürülerek bir çok-boyutlu modelleme problemi olarak çözülebilir.

Tablo 5.5: Zaman serisi giriş-çıkış verisi.

$\mathfrak{D}_{\text{TS}}$	**Giriş Verisi**				**Çıkış Verisi**
i	$\mathbf{t}_i$				y_i
	t_{i1}	t_{i2}	...	t_{iR}	
1	$z[1]$	$z[2]$	...	$z[R]$	$z[R+1]$
2	$z[2]$	$z[3]$	...	$z[R+1]$	$z[R+2]$
⋮	⋮				⋮
$N-R$	$z[N-R]$	$z[N-R+1]$	...	$z[N-1]$	$z[N]$

Tablo 5.5'teki R parametresi, modelde kaç tane giriş olacağını ya da başka bir deyişle zaman serisinin belli bir andaki çıkışının o andan itibaren kaç adım öncesine kadar bağlı olduğunu göstermektedir (İplikçi 2018).

Zaman serisinin bir sonraki değeri olan $z[n+1]$ değerini tahmin etmek için, yapay öğrenme modellerinin girişlerine Denklem (5.11)'deki vektör uygulanır.

$$\mathbf{t}_{i+1} = \begin{bmatrix} z[N-R+1] \\ z[N-R+2] \\ \vdots \\ z[N] \end{bmatrix} \tag{5.11}$$

5.5 Çok-Adımlı Zaman Serisi Tahmini

Toplam N adet gözleme sahip olan $[x_{t_1}, x_{t_2}, \dots, x_{t_N}]$ şeklindeki tek değişkenli bir zaman serisi göz önüne alındığında, çok-adımlı ileri tahmin, zaman serisinin gelecekteki sonraki H adet $[x_{t_{N+1}}, x_{t_{N+2}}, \dots, x_{t_{N+H}}]$ veri noktalarını tahmin etmek için N adet kayıtlı veri noktasını kullanmaktır. Burada $H > 1$ parametresi tahmin ufkudur. Çok-adımlı zaman serisi tahmininde 3 adet strateji kullanılmaktadır. Bunlar; özyinelemeli strateji, doğrudan strateji ve çok girişli-çok çıkışlı (MIMO) stratejidir.

5.5.1 Özyinelemeli Strateji

En sezgisel ve geleneksel tahmin stratejisi, tek bir tahmin modeli $f(.)$ ile bir-adım ileri zaman serisi tahmin yönteminin uygulandığı özyinelemeli stratejidir. İlgili model Denklem (5.12)'de gösterilmiştir.

$$x_{t+1} = f(x_t, x_{t-1}, \dots, x_{t-d+1}) + \epsilon \tag{5.12}$$

Denklem (5.12)'deki $t \in \{d, d+1, \dots, N\}$ şeklinde ifade edilir. Denklem (5.12)'de d, tahmincinin boyutudur, ϵ toplam gürültüdür, $f: \mathbf{R}^d \rightarrow \mathbf{R}$ tahmincidir ve $\mathbf{R}$ gerçek alanı belirtmektedir.

H adım ilerisini tahmin etmek için öncelikle Denklem (5.12)'yi kullanarak bir-adım ileri $x_{t_{N+1}}$ tahminini yaparız. Ardından bir sonraki adım, giriş zaman serisinin bir parçası olarak tahmin edilen $x_{t_{N+1}}$ ile, Denklem (5.12)'deki aynı bir-adım ileri tahmin modelini kullanarak $x_{t_{N+2}}$ değerini tahmin etmektir. Bu prosedür, $x_{t_{N+H}}$ tahmin edilene kadar özyinelemeli olarak çalıştırılır.

Özyinelemeli strateji sezgisel ve uygulaması kolay olmasına karşılık, özellikle tahmin ufku geniş olduğunda tahmin hatalarının birikmesine karşı hassastır. Özyinelemeli stratejide, önceki adımlardaki tahmin hataları, sonraki tahmin doğruluğunu bozmak için yayılır ve biriktirilir (Zheng ve diğ. 2017).

5.5.2 Doğrudan Strateji

Çok-adımlı tahmin için başka bir strateji doğrudan stratejidir (Zheng ve diğ. 2017). Özyinelemeli stratejiden farklı olarak doğrudan strateji, gözlemlenen zaman serisi verilerine dayalı olarak her tahmin ufku için H farklı tahmin modeli oluşturur. İlgili model Denklem (5.13)'te gösterilmiştir.

$$x_{t+h} = f_h(x_t, x_{t-1}, \dots, x_{t-d+1}) + \epsilon_h \tag{5.13}$$

Denklem (5.13)'teki $h \in \{1,2, \dots, H\}$ şeklinde ifade edilir. Denklem (5.13)'te f_h h'inci tahmin modelidir ve ϵ_h h'inci modelle ilişkili gürültüdür.

Doğrudan strateji, tahmin için girdi olarak herhangi bir tahmin edilen değeri kullanmadığı için birikmiş hatalara eğilimli değildir. Bununla birlikte, H tahmin modelleri birbirinden ayrı ve bağımsız olarak eğitilir, bu da H tahmin değerleri arasında koşullu bağımsızlığa neden olabilir. Böyle bir bağımsızlık etkisi, tahmin yöntemlerinin tahmin edilen veriler arasındaki istatistiksel bağımlılığı yansıtmasını engelleyecek ve bu da tahmin performansını düşürecektir. Bu gibi sebeplerden dolayı bu tez kapsamında doğrudan strateji kullanılmamıştır (Zheng ve diğ. 2017).

5.5.3 Çok Girişli-Çok Çıkışlı (MIMO) Strateji

Hem özyinelemeli strateji hem de doğrudan strateji, birden çok girdiyi tek bir çıktıya eşleştirdikleri için çok girişli-tek çıkışlı (MISO) strateji olarak kabul edilirler. Buna karşılık, MIMO stratejisi, birden çok çıktı oluşturmak için birden çok girdi kullanan bir tahmin stratejisidir. MIMO stratejisi ile tahmin sonucu skaler yerine bir zaman serisidir yani bir vektördür. Bu çıktı vektöründeki tüm veriler, aynı gözlemlenen zaman serisi verileri kullanılarak eğitilen aynı model tarafından üretilir. İlgili model Denklem (5.14)'te gösterilmiştir.

$$[x_{t+1}, x_{t+2}, \dots, x_{t+H}] = F(x_t, x_{t-1}, \dots, x_{t-d+1}) + \boldsymbol{\epsilon} \tag{5.14}$$

Denklem (5.14)'te ϵ gürültü vektörüdür ve $F: \mathbf{R}^d \rightarrow \mathbf{R}^H$ şeklinde ifade edilir.

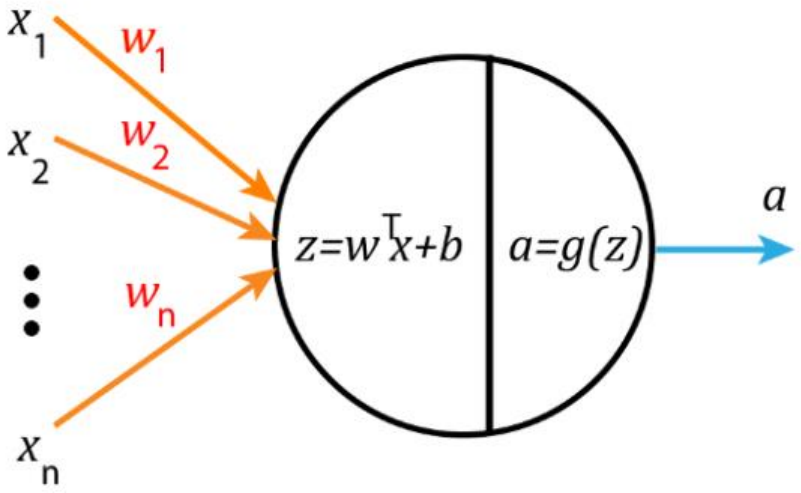
MISO stratejileri ile karşılaştırıldığında, MIMO strateji koşullu bağımsızlık problemini azaltma yeteneğine sahiptir. Tahmin edilen zaman serilerinde zamansal istatistiksel bağımlılığı koruma avantajına sahiptir. Ancak MIMO strateji tüm verileri aynı tahmin modeli ile tahmin ettiği için esnekliği ve değişkenliği diğer tahmin stratejileri kadar güçlü olmayabilir (Zheng ve diğ. 2017).

6. YAPAY SİNİR AĞLARI

6.1 İleri Beslemeli Yapay Sinir Ağları

6.1.1 Nöron Modeli

Bir nöron beynimizin temel birimidir. Beynin yaklaşık 100 milyar nörona sahip olduğu tahmin ediliyor ve bu devasa biyolojik ağ, çevremizdeki dünya hakkında düşünmemizi ve bu dünyayı algılamamızı sağlıyor. Temel olarak bir nöronun yaptığı, diğer nöronlardan bilgi almak, bu bilgiyi işlemek ve sonucu diğer nöronlara göndermektir. Bu girdiler hücre gövdesinde toplanır ve akson yoluyla diğer nöronlara gönderilen bir sinyale dönüştürülür. Akson, sinapslarla diğer nöronların dendritlerine bağlanır. Sinaps bir ağırlık görevi görebilir ve bu bağlantının ne sıklıkla kullanıldığına bağlı olarak içinden geçen sinyali daha güçlü veya daha zayıf hale getirebilir. Nöronun bu biyolojik anlayışı, Şekil 6.1'de gösterildiği gibi matematiksel bir modele çevrilebilir (Bagheri 2020).



Şekil 6.1: Bir nöronun matematiksel modeli.

Yapay nöron, $x_1, x_2, \dots, x_n$ giriş özelliklerinin bir vektörünü alır ve her biri belirli bir ağırlık vektörü $w_1, w_2, \dots, w_n$ ile çarpılır. Ağırlıklı girişler toplanır ve

Denklem (6.1)'de belirtildiği gibi nöronun net girişini üretmek için bunlara bias (b) adı verilen sabit bir değer eklenir.

$$z = \sum_{i=1}^{n} x_i w_i + b \tag{6.1}$$

Net giriş daha sonra bir g aktivasyon fonksiyonundan geçirilerek $a = g(z)$ çıktısı üretilir ve bu daha sonra diğer nöronlara iletilir. Bu durum Denklem (6.2)'de belirtildiği gibidir.

$$a = g(z) = g(\sum_{i=1}^{n} x_i w_i + b) \tag{6.2}$$

Aktivasyon fonksiyonu tasarımcı tarafından seçilir, ancak w_i ve b sinir ağının eğitim sürecinde bazı öğrenme kuralları ile ayarlanır.

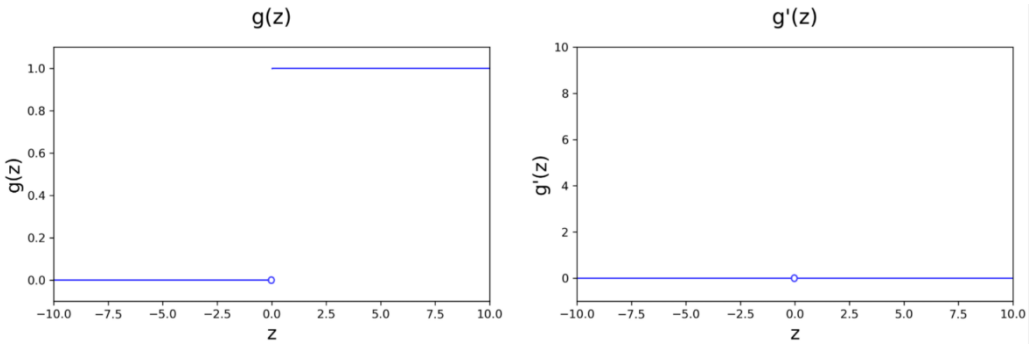
6.1.2 Aktivasyon Fonksiyonları

Bir sinir ağında kullanabilecek farklı aktivasyon fonksiyonları vardır ve bu aktivasyon fonksiyonlarından bazıları aşağıda tanımlanmıştır.

1. **İkili Adım Fonksiyonu**

   İkili adım fonksiyonu, eşik-tabanlı bir aktivasyon fonksiyonudur. Fonksiyonun girişi (z) sıfırdan küçük veya sıfıra eşitse nöronun çıkışı sıfır, fonksiyonun girişi (z) sıfırın üzerindeyse nöronun çıkışı 1'dir. İkili adım fonksiyonuna ait denklem ve grafik, Denklem (6.3)'de ve Şekil 6.2'de gösterildiği gibidir (Bagheri 2020).

$$a = g(z) = \begin{cases} 0, eğer\ z \leq 0 \\ 1, eğer\ z > 0 \end{cases} \tag{6.3}$$

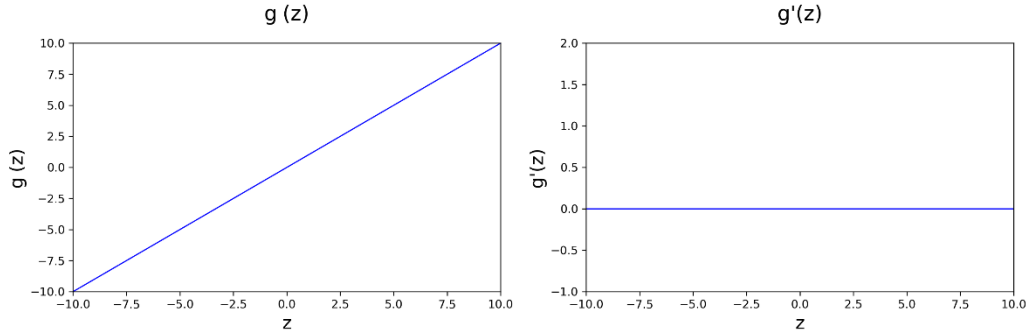


Şekil 6.2: İkili adım fonksiyonu ve türevi.

2. Lineer Fonksiyon

Doğrusal aktivasyon fonksiyonunun çıkışı, girişinin c sabiti ile çarpımına eşittir. Lineer aktivasyon fonksiyonuna ait denklem ve grafik, Denklem (6.4)'de ve Şekil 6.3'te gösterildiği gibidir (Bagheri 2020).

$$a = g(z) = cz \tag{6.4}$$



Şekil 6.3: Lineer fonksiyon ve türevi.

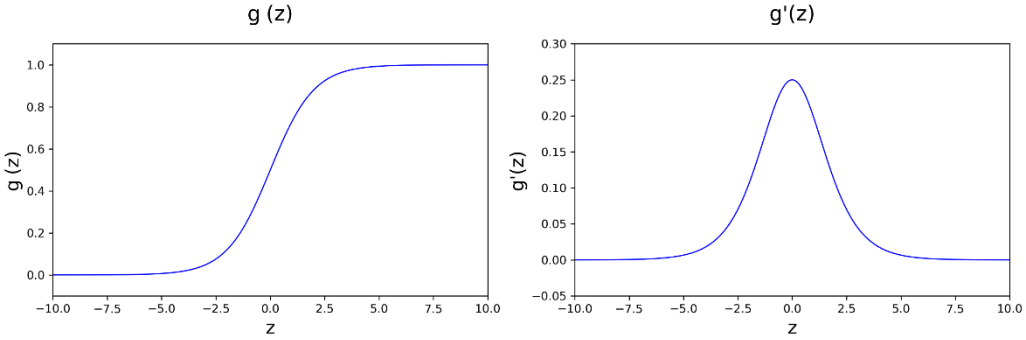
3. Sigmoid Fonksiyonu

Sigmoid fonksiyonu, 0 ile 1 aralığında sürekli çıkış veren doğrusal olmayan bir aktivasyon fonksiyonudur. Denklem (6.5)'teki gibi tanımlanmaktadır.

$$a = g(z) = \frac{1}{e^{-z}} \tag{6.5}$$

Sigmoid, adım fonksiyonuna benzer olma özelliğine sahiptir ancak süreklidir ve adım fonksiyonunda var olan çıkış değerlerindeki

sıçramayı engeller. Sigmoid fonksiyonuna ait grafik Şekil 6.4'te gösterildiği gibidir (Bagheri 2020).

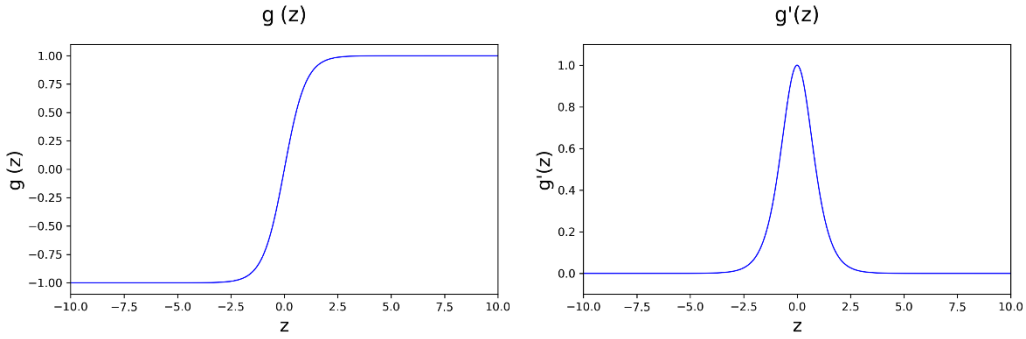


Şekil 6.4: Sigmoid fonksiyonu ve türevi.

4. Tanjant Hiperbolik Fonksiyonu

Tanjant hiperbolik fonksiyonu sigmoide benzer, ancak -1 ile 1 aralığında sürekli bir çıkış veren doğrusal olmayan bir aktivasyon fonksiyonudur. Tanjant hiperbolik aktivasyon fonksiyonuna ait denklem ve grafik, Denklem (6.6)'da ve Şekil 6.5'te gösterildiği gibidir (Bagheri 2020).

$$a = g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{6.6}$$



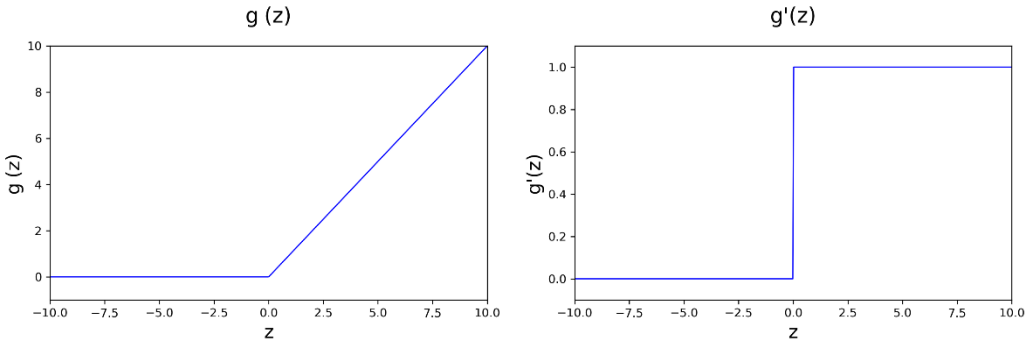
Şekil 6.5: Tanh fonksiyonu ve türevi.

5. Doğrultulmuş Lineer Birim (ReLU) Fonksiyonu

ReLU, derin sinir ağlarında çok popüler bir aktivasyon fonksiyonudur. Denklem (6.7)'deki gibi tanımlanmaktadır.

$$a = g(z) = \max(0, z) \tag{6.7}$$

Doğrusal bir fonksiyon gibi görünse de ReLU aslında doğrusal olmayan bir fonksiyondur. ReLU fonksiyonuna ait grafik Şekil 6.6'da gösterildiği gibidir (Bagheri 2020).



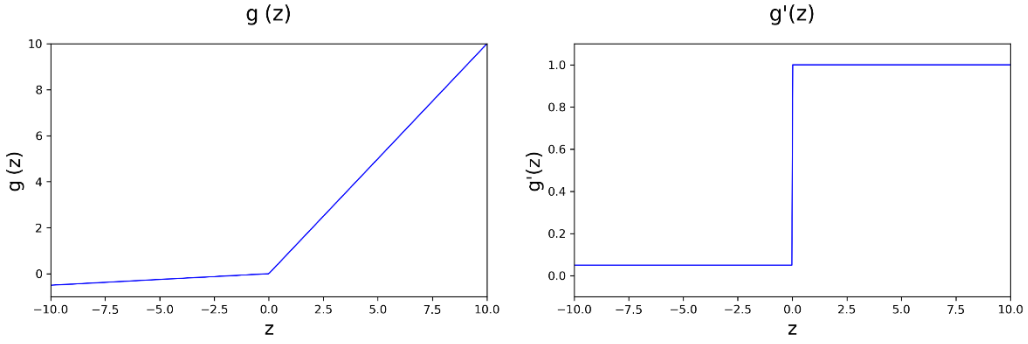
Şekil 6.6: ReLu fonksiyonu ve türevi.

6. Leaky ReLU Fonksiyonu

ReLU'nun Leaky ReLU olarak tanımlanan başka bir versiyonu daha vardır. Denklem (6.8)'deki gibi tanımlanmaktadır.

$$a = g(z) = \max(cz, z) \tag{6.8}$$

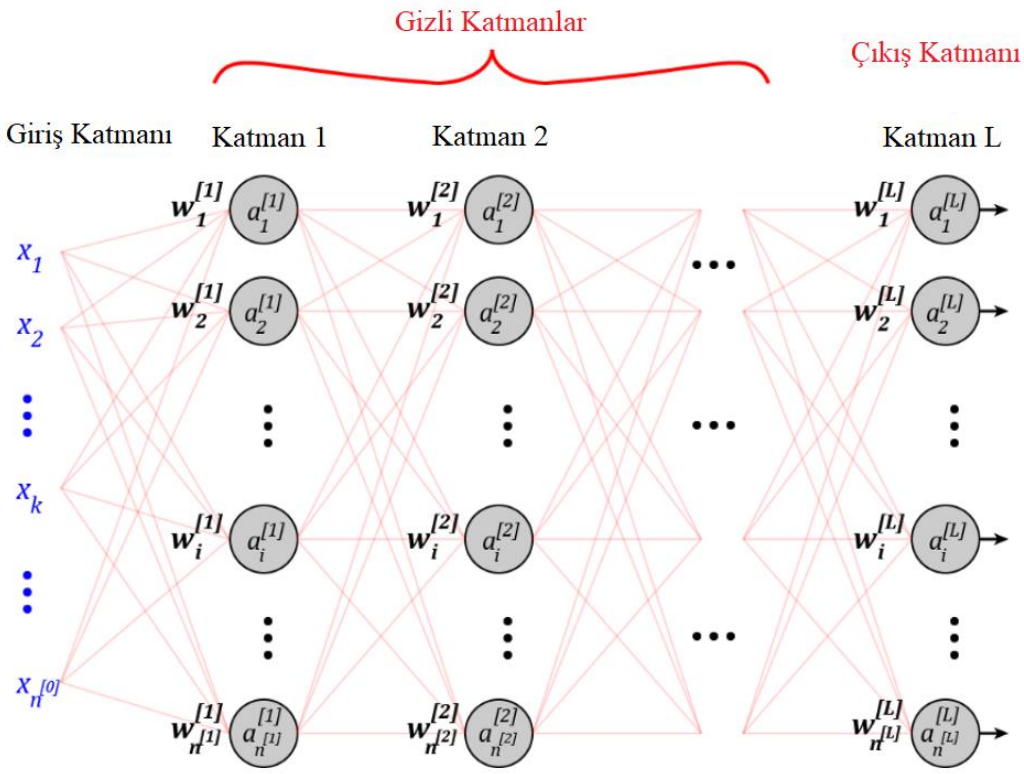
burada c, 0.001 gibi küçük bir sabittir. Leaky ReLU fonksiyonuna ait grafik Şekil 6.7'de gösterildiği gibidir (Bagheri 2020).



Şekil 6.7: Leaky ReLu fonksiyonu ve türevi.

6.1.3 Çok Katmanlı Ağ Yapısı

Çok katmanlı ağlarda, ağın son katmanına çıkış katmanı ve aralarda bulunan katmanlara gizli katmanlar denir. Çok katmanlı ağ yapısı Şekil 6.8'de gösterildiği gibidir (Bagheri 2020).



Şekil 6.8: Çok katmanlı ileri beslemeli yapay sinir ağı yapısı.

İleri beslemeli bir ağda, bilgi yalnızca ileri yönde, giriş katmanından gizli katmanlar (varsa) boyunca ve çıkış katmanına doğru hareket eder. Bu ağda çevrim veya döngü yoktur. İleri beslemeli sinir ağları bazen çok katmanlı algılayıcılar olarak adlandırılır. Katman l'deki nöron sayısı $n^{[l]}$ şeklinde gösterilir. Katman l'deki nöronların girişi Denklem (6.9)'daki vektör ile temsil edilir.

$$\mathbf{z}^{[l]} = \begin{bmatrix} z_1^{[l]} \\ z_2^{[l]} \\ \vdots \\ z_{n^{[l]}}^{[l]} \end{bmatrix} \tag{6.9}$$

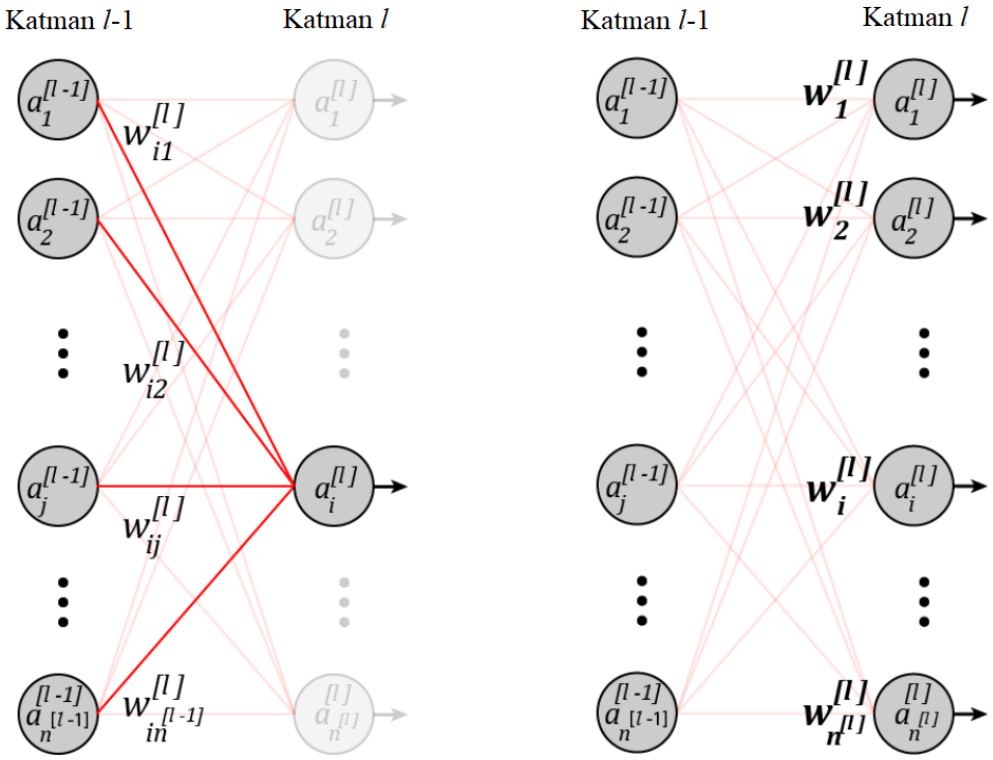
Benzer şekilde, katman l'deki nöronların aktivasyonu Denklem (6.10)'daki aktivasyon vektörü ile temsil edilebilir.

$$\mathbf{a}^{[l]} = \begin{bmatrix} a_1^{[l]} \\ a_2^{[l]} \\ \vdots \\ a_{n^{[l]}}^{[l]} \end{bmatrix} \tag{6.10}$$

Katman l'deki i nöronunun ağırlıkları Denklem (6.11)'de gösterildiği gibidir.

$$\mathbf{w}_i^{[l]} = \begin{bmatrix} w_{i1}^{[l]} \\ w_{i2}^{[l]} \\ \vdots \\ w_{in^{[l-1]}}^{[l]} \end{bmatrix} \tag{6.11}$$

Burada $w_{i1}^{[l]}$, l katmanındaki i nöronuna giden j girişinin ($l-1$ katmanındaki j nöronundan gelen) ağırlığını temsil eder. Bu durum Şekil 6.9'da gösterilmektedir (Bagheri 2020).



Şekil 6.9: Tam bağlantılı katman yapısı.

Şekil 6.9'da görülebileceği gibi l katmanındaki tüm girişler o katmandaki tüm nöronlara bağlıdır. Böyle bir katmana yoğun veya tam bağlantılı katman denir.

Şimdi, Denklem (6.12)'deki ağırlık ve giriş vektörlerini kullanarak katman l'deki bir nöronun aktivasyonunun hesaplayabiliriz.

$$\begin{aligned} z_i^{[l]} &= \sum_{k=1}^{n^{[l-1]}} w_{ik}^{[l]} a_k^{[l-1]} + b_i^{[l]} = \mathbf{w}_i^{[l]^T} \mathbf{a}^{[l-1]} + b_i^{[l]} \\ a_i^{[l]} &= g^{[l]}\left(z_i^{[l]}\right) = g^{[l]}(\mathbf{w}_i^{[l]^T} \mathbf{a}^{[l-1]} + b_i^{[l]}) \end{aligned} \tag{6.12}$$

Burada, $b_i^{[l]}$, katman l'deki i nöronunun bias değeridir ve $g^{[l]}$, katman l'deki her nöron için aktivasyon fonksiyonudur. Her katmanın farklı bir aktivasyon fonksiyonuna sahip olabileceğine dikkat etmek önemlidir, ancak bir katmandaki nöronlar genellikle aynı aktivasyon fonksiyonuna sahiptir. Tutarlı bir gösterime sahip olabilmek için, $\mathbf{a}^{[0]} = \mathbf{x}$ olduğunu varsayabiliriz.

6.1.3.1 Aktivasyonların Vektörleştirilmesi

Bir katmanın tüm ağırlıklarını, o katman için Denklem (6.13)'deki ağırlık matrisinde birleştirebiliriz.

$$\mathbf{W}^{[l]} = \begin{bmatrix} w_1^{[l]T} \\ w_2^{[l]T} \\ \vdots \\ w_{n^{[l]}}^{[l]T} \end{bmatrix} \tag{6.13}$$

Bu bölümlenmiş matrisin i-inci elemanı bir satır vektörüdür ve bu satır vektörü, katman l'deki i nöronu için ağırlıkları veren bir sütun vektörünün transpoze halidir. Denklem (6.13)'deki matrisi genişletirsek, Denklem (6.14)'ü elde ederiz.

$$\mathbf{W}^{[l]} = \begin{bmatrix} w_{11}^{[l]} & w_{12}^{[l]} & \cdots & w_{1n^{[l-1]}}^{[l]} \\ w_{21}^{[l]} & w_{22}^{[l]} & \cdots & w_{2n^{[l-1]}}^{[l]} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n^{[l]}1}^{[l]} & w_{n^{[l]}1}^{[l]} & \cdots & w_{n^{[l]}n^{[l-1]}}^{[l]} \end{bmatrix} \tag{6.14}$$

Yani bu matrisin (i, j) öğesi, $l-1$ katmanındaki j nöronundan l katmanındaki i nöronuna giden bağlantının ağırlığını verir. Bias vektörü de Denklem (6.15)'deki gibi ifade edilebilir.

$$\mathbf{b}^{[l]} = \begin{bmatrix} b_1^{[l]} \\ b_2^{[l]} \\ \vdots \\ b_{n^{[l]}}^{[l]} \end{bmatrix} \tag{6.15}$$

burada i-inci eleman, katman l'deki nöron i için bias terimidir. Denklem (6.12)' yi kullanarak Denklem (6.16)'yı yazabiliriz.

$$\begin{aligned}\mathbf{W}^{[l]}\mathbf{a}^{[l-1]}+\mathbf{b}^{[l]}&=\begin{bmatrix}\mathbf{w}_1^{[l]^T}\\ \mathbf{w}_2^{[l]^T}\\ \vdots\\ \mathbf{w}_{n^{[l]}}^{[l]^T}\end{bmatrix}\mathbf{a}^{[l-1]}+\mathbf{b}^{[l]}=\begin{bmatrix}\mathbf{w}_1^{[l]^T} & \mathbf{a}^{[l-1]}\\ \mathbf{w}_2^{[l]^T} & \mathbf{a}^{[l-1]}\\ \vdots & \vdots\\ \mathbf{w}_{n^{[l]}}^{[l]^T} & \mathbf{a}^{[l-1]}\end{bmatrix}+\mathbf{b}^{[l]}\\ &=\begin{bmatrix}\mathbf{w}_1^{[l]^T} & \mathbf{a}^{[l-1]}+b_1^{[l]}\\ \mathbf{w}_2^{[l]^T} & \mathbf{a}^{[l-1]}+b_2^{[l]}\\ \vdots & \vdots\\ \mathbf{w}_{n^{[l]}}^{[l]^T} & \mathbf{a}^{[l-1]}+b_{n^{[l]}}^{[l]}\end{bmatrix}=\begin{bmatrix}z_1^{[l]}\\ z_2^{[l]}\\ \vdots\\ z_{n^{[l]}}^{[l]}\end{bmatrix}=\mathbf{z}^{[l]}\end{aligned} \tag{6.16}$$

Denklem (6.10) ve Denklem (6.12)'yi kullanarak l katmanındaki vektörleştirilmiş aktivasyon fonksiyonunu önceki denkleme uygularsak, Denklem (6.17)'yi elde ederiz.

$$g^{[l]}\left(\mathbf{W}^{[l]}\mathbf{a}^{[l-1]}+\mathbf{b}^{[l]}\right)=g^{[l]}\left(\mathbf{z}^{[l]}\right)=\begin{bmatrix}g^{[l]}(z_1^{[l]})\\ g^{[l]}(z_2^{[l]})\\ \vdots\\ g^{[l]}(z_{n^{[l]}}^{[l]})\end{bmatrix}=\begin{bmatrix}a_1^{[l]}\\ a_2^{[l]}\\ \vdots\\ a_{n^{[l]}}^{[l]}\end{bmatrix}=\mathbf{a}^{[l]} \tag{6.17}$$

Denklem (6.17)'yi yeniden düzenleyerek Denklem (6.18)'i elde ederiz.

$$\mathbf{a}^{[l]}=g^{[l]}\left(\mathbf{z}^{[l]}\right)=g^{[l]}\left(\mathbf{W}^{[l]}\mathbf{a}^{[l-1]}+\mathbf{b}^{[l]}\right) \tag{6.18}$$

Denklem (6.18), ileri beslemeli bir sinir ağı için ileri yayılma denklemidir. Bu denklemi kullanarak bir önceki katmanın aktivasyonlarını kullanarak bir katmanın aktivasyonlarını hesaplayabiliriz. Denklem (6.16) ve Denklem (6.18)'i ilk katmana ($l=1$) uygularsak, önceki katman giriş katmanıdır, yani Denklem (6.19)'yi elde etmiş oluruz.

$$\begin{aligned}\mathbf{z}^{[1]}&=\mathbf{W}^{[1]}\mathbf{x}+\mathbf{b}^{[1]}\\ \mathbf{a}^{[1]}&=g^{[l]}\left(\mathbf{z}^{[1]}\right)=g^{[l]}\left(\mathbf{W}^{[1]}\mathbf{x}+\mathbf{b}^{[1]}\right)\end{aligned} \tag{6.19}$$

6.1.3.2 Giriş Vektörleri Üzerinde Vektörleştirme

m örnekli bir eğitim setimiz olduğunu varsayalım. Yani tüm eğitim seti $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}$ için bir dizi giriş vektörümüz vardır. Her örnek, Denklem (6.20)'deki gibi $n^{[0]}$ giriş özelliklerine sahiptir.

$$\mathbf{x}^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_{n^{[0]}}^{(i)} \end{bmatrix} \tag{6.20}$$

Artık eğitim setindeki tüm girdi vektörlerini Denklem (6.21)'daki gibi bir giriş matrisine sahip olacak şekilde birleştirebiliriz.

$$\mathbf{X} = [\mathbf{x}^{(1)} \quad \mathbf{x}^{(2)} \quad \ldots \quad \mathbf{x}^{(m)}] \tag{6.21}$$

Bu bölümlenmiş matrisin her elemanı bir sütun vektörüdür ve eğitim kümesinin i-inci örneğinin giriş vektörüne eşittir, dolayısıyla bu matrisin her bir elemanı $[\mathbf{X}]_{ij} = x_i^{(j)}$ şeklindedir.

Bu örneklerin her biri sinir ağı için giriş katmanı olarak kullanılabilir ve her eğitim örneği için her katmanın aktivasyonlarını tahmin etmek için Denklem (6.16) ve Denklem (6.18)'i kullanabiliriz.

$$\begin{aligned} \mathbf{z}^{[l](i)} &= \mathbf{W}^{[l]}\mathbf{a}^{[l-1](i)} + \mathbf{b}^{[l]} \\ \mathbf{a}^{[l](j)} &= g^{[l]}\left(\mathbf{z}^{[l](j)}\right) = g^{[l]}\left(\mathbf{W}^{[l]}\mathbf{a}^{[l-1](j)} + \mathbf{b}^{[l]}\right) \end{aligned} \tag{6.22}$$

Burada üst simge $[l]$ katman numarasını ve üst simge (j) eğitim örneği numarasını ifade etmektedir. Denklem (6.20)'nin ilk eşitliği Denklem (6.23)'deki gibi yazılabilir.

$$z_i^{[l](j)} = \sum_{k=1}^{n^{[l-1]}} w_{ik}^{[l]} a_k^{[l-1](j)} + b_i^{[l]} \tag{6.23}$$

Dolayısıyla Denklem (6.23) j numaralı eğitim örneği için l katmanının net girişini ve aktivasyon vektörünü verir. İlk katman için Denklem (6.24)'yi yazabiliriz.

$$\mathbf{a}^{[1](j)} = g^{[1]}\left(\mathbf{z}^{[1](j)}\right) = g^{[1]}\left(\mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}\right) \tag{6.24}$$

Buradan yola çıkarak Denklem (6.25)'ü yazabiliriz.

$$\begin{aligned}\mathbf{W}^{[l]}\boldsymbol{X} + \mathbf{b}^{[1]} &= \mathbf{W}^{[l]}[\mathbf{x}^{(1)} \quad \mathbf{x}^{(2)} \quad \dots \quad \mathbf{x}^{(m)}] + \mathbf{b}^{[1]} \\ &= [\mathbf{W}^{[1]}\mathbf{x}^{(1)} \quad \mathbf{W}^{[1]}\mathbf{x}^{(2)} \quad \dots \quad \mathbf{W}^{[1]}\mathbf{x}^{(m)}] + \mathbf{b}^{[1]} \\ &= [\mathbf{W}^{[1]}\mathbf{x}^{(1)} + \mathbf{b}^{[1]} \quad \mathbf{W}^{[1]}\mathbf{x}^{(2)} + \mathbf{b}^{[1]} \quad \dots \quad \mathbf{W}^{[1]}\mathbf{x}^{(m)} + \mathbf{b}^{[1]}] \\ &= [\mathbf{z}^{[1](1)} \quad \mathbf{z}^{[1](2)} \quad \dots \quad \mathbf{z}^{[1](m)}]\end{aligned} \tag{6.25}$$

Net girdi matrisini Denklem (6.26)'teki gibi tanımlayabiliriz.

$$\mathbf{Z}^{[l]} = [\mathbf{z}^{[l](1)} \quad \mathbf{z}^{[l](2)} \quad \dots \quad \mathbf{z}^{[l](m)}] \tag{6.26}$$

Böylece $\left[\mathbf{Z}^{[l]}\right]_{ij} = z_i^{(j)}$ ifadesine sahip oluruz. Burada $\mathbf{Z}^{[l]}$'nin i. sütunu, i numaralı eğitim örneği için l katmanının net girdisidir. Benzer şekilde aktivasyon matrisini Denklem (6.27)'deki gibi tanımlayabiliriz.

$$\begin{gathered}\mathbf{A}^{[l]} = [\mathbf{a}^{[l](1)} \quad \mathbf{a}^{[l](2)} \quad \dots \quad \mathbf{a}^{[l](m)}] \\ \left[\mathbf{A}^{[l]}\right]_{ij} = a_i^{(j)}\end{gathered} \tag{6.27}$$

Şimdi Denklem (6.25)'ü $l = 1$ için, Denklem (6.28)'daki gibi yeniden yazmak için Denklem (6.26)'ü kullanabiliriz.

$$\mathbf{Z}^{[1]} = \mathbf{W}^{[1]}\boldsymbol{X} + \mathbf{b}^{[1]} \tag{6.28}$$

Vektörleştirilmiş aktivasyon fonksiyonunu Denklem (6.26)'ya uygularsak, Denklem (6.24) ve Denklem (6.27)'nin ilk eşitliğini kullanarak Denklem (6.29)'u elde ederiz.

$$\begin{aligned}g^{[l]}(\mathbf{Z}^{[l]}) &= g^{[l]}([\mathbf{z}^{[l](1)} \quad \mathbf{z}^{[l](2)} \quad \dots \quad \mathbf{z}^{[l](m)}]) \\ &= \left[g^{[l]}\left(\mathbf{z}^{[l](1)}\right) \quad g^{[l]}\left(\mathbf{z}^{[l](2)}\right) \quad \dots \quad g^{[l]}\left(\mathbf{z}^{[l](m)}\right)\right] \\ &= [\mathbf{a}^{[l](1)} \quad \mathbf{a}^{[l](2)} \quad \dots \quad \mathbf{a}^{[l](m)}] \\ &= \mathbf{A}^{[l]}\end{aligned} \tag{6.29}$$

Denklem (6.28) ve Denklem (6.29)'u $l = 1$ için birleştirerek Denklem (6.30)'u elde ederiz.

$$\mathbf{A}^{[1]} = g^{[1]}(\mathbf{Z}^{[1]}) = g^{[1]}\left(\mathbf{W}^{[1]}\mathbf{X} + \mathbf{b}^{[1]}\right) \tag{6.30}$$

Denklem (6.22)'nin ilk eşitliğini, Denklem (6.26)'yı ve Denklem (6.27)'nin ilk eşitliğini kullanarak Denklem (6.31)'i yazabiliriz.

$$\begin{aligned}\mathbf{W}^{[l]}\mathbf{A}^{[l-1]} + \mathbf{b}^{[1]} &= \mathbf{W}^{[l]}[\mathbf{a}^{[l-1](1)} \quad \mathbf{a}^{[l-1](2)} \quad \dots \quad \mathbf{a}^{[l-1](m)}] + \mathbf{b}^{[1]} \\ &= [\mathbf{z}^{[l](1)} \quad \mathbf{z}^{[l](2)} \quad \dots \quad \mathbf{z}^{[l](m)}] \\ &= \mathbf{Z}^{[l]}\end{aligned} \tag{6.31}$$

Sonuç olarak Denklem (6.29) ve Denklem (6.31)'i birleştirirsek, Denklem (6.32)'u elde ederiz.

$$\mathbf{A}^{[l]} = g^{[l]}(\mathbf{Z}^{[l]}) = g^{[l]}\left(\mathbf{W}^{[l]}\mathbf{A}^{[l-1]} + \mathbf{b}^{[l]}\right) \tag{6.32}$$

Denklem (6.32), tüm eğitim seti için vektörleştirilmiş ileri yayılma denklemidir. Ayrıca $\mathbf{A}^{[0]} = \mathbf{X}$ olduğunu da varsayabiliriz. Böylece Denklem (6.31) $l = 1$ olduğunda Denklem (6.30)'a dönüşmektedir.

6.1.3.3 Çıkış Katmanı

$\mathbf{a}^{[L]}$ 'nin sinir ağının son katmanının aktivasyon vektörü olduğunu hatırlayalım. Ancak, ağın son çıktısı olduğu için genellikle çıkış katmanının aktivasyonu için $\hat{\mathbf{y}}$ vektörünü kullanırız. Yani j örneği için Denklem (6.33)'i yazabiliriz.

$$\hat{\mathbf{y}}^{(j)} = \mathbf{a}^{[L](j)} = g^{[L]}(\mathbf{z}^{[L](j)}) \tag{6.33}$$

Bir regresyon problemi için, her eğitim örneği için gerçek bir değer etiketine sahibiz, bu nedenle genellikle doğrusal aktivasyon fonksiyonuna sahip tek bir nöron kullanırız.

6.1.3.4 Maliyet Fonksiyonu

Ağın çıkışının $\hat{\mathbf{y}}$ olduğunu hatırlayalım. Her $\mathbf{x}^{(i)}$ örneği için çıktı veya ağın tahmini $\hat{\mathbf{y}}^{(i)}$'dir ve $\mathbf{y}^{(i)} = \hat{\mathbf{y}}^{(i)}$ olması istenmektedir. Kayıp (veya hata) fonksiyonu, çıkış hatasını ölçen bir fonksiyondur. Kayıp fonksiyonu, ağ çıkışı $\hat{\mathbf{y}}^{(i)}$'nin gerçek değer $\mathbf{y}^{(i)}$'den ne kadar uzakta olduğunu ifade etmektedir. İkinci dereceden kayıp fonksiyonu Denklem (6.34)'deki gibi tanımlanmaktadır.

$$\mathcal{L}\left(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}\right) = \frac{1}{2}\left\|\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}\right\|^2 \tag{6.34}$$

Kayıp fonksiyonu bize belirli bir örnek için hatayı vermektedir. Ancak, ağın hepsini birlikte öğrenmesini istediğimiz için tüm örnekler için ortalama hataya ihtiyacımız vardır. Bu nedenle, ikinci dereceden maliyet fonksiyonunu, tüm örneklerin kayıp fonksiyonunun ortalaması olarak tanımlanmaktadır ve Denklem (6.35)'de gösterilmektedir.

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}\right) = \frac{1}{2m}\sum_{i=1}^{m}\left\|\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}\right\|^2 \tag{6.35}$$

Ortalama karesel hata (MSE) maliyet fonksiyonu olarak da bilinir. J, $\mathbf{y}^{(i)}$ ve $\hat{\mathbf{y}}^{(i)}$'nin bir fonksiyonudur. Ancak ağ çıkışı $\hat{\mathbf{y}}^{(i)}$'nin ağ parametrelerinin kendisinin bir fonksiyonu olduğunu biliyoruz. Yani maliyet fonksiyonu aslında bu parametrelerin bir fonksiyonudur. Burada, w ve b (indeksler olmadan), ağdaki tüm nöronların ağırlıklarının ve bias terimlerinin toplanmasını ifade eder. Hem kayıp fonksiyonunun hem de maliyet fonksiyonunun skaler fonksiyonlar olduğuna dikkat edilmelidir (skaler bir değer döndürürler). Çıkış katmanında sadece bir nöron varsa Denklem (6.34) ve Denklem (6.35), Denklem (6.36)'daki gibi olmaktadır.

$$\begin{aligned} \mathcal{L}\left(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}\right) &= \frac{1}{2}\left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}\right)^2 \\ J(w, b) &= \frac{1}{2m}\sum_{i=1}^{m}\left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}\right)^2 \end{aligned} \tag{6.36}$$

w ve b parametreleri ayarlanabilir parametrelerdir ve bir sinir ağını eğittiğimizde amaç, $J(w, b)$ maliyet fonksiyonunu en aza indiren ağırlıkları ve bias

terimlerini bulmaktır. Maliyet fonksiyonu minimize edildiğinde, eğitim seti için minimum hata olmasını bekleriz. Bir fonksiyonu pozitif bir a çarpanı ile çarparsak, $aJ(w,b)$ 'nin minimumu $J(w,b)$ 'nin minimumu ile aynı w ve b değerlerinde gerçekleşir. Dolayısıyla Denklem (6.35)'deki $1/2$ çarpanının maliyet fonksiyonunun minimizasyonu üzerinde hiçbir etkisi yoktur ve genellikle hesaplamaları kolaylaştırmak için eklenmektedir. MSE maliyet fonksiyonu, regresyon problemleri için varsayılan maliyet fonksiyonudur (Bagheri 2020).

6.1.3.5 Hatayı Geriye Yayma

Optimizasyon bölümünde tanımlanan gradyan azalan yöntemini kullanmak için, w ve b 'ye göre maliyet fonksiyonunun kısmi türevini veya gradyanını hesaplamamız ve bunu yapmak için hatayı geriye yayma adlı bir algoritma kullanmak gerekmektedir. Yani hatayı geriye yaymanın asıl amaç Denklem (6.37)'de ifade edilenleri hesaplamaktır.

$$\frac{\partial C}{\partial w_{ij}^{[l]}}, \frac{\partial C}{\partial b_{i}^{[l]}} \tag{6.37}$$

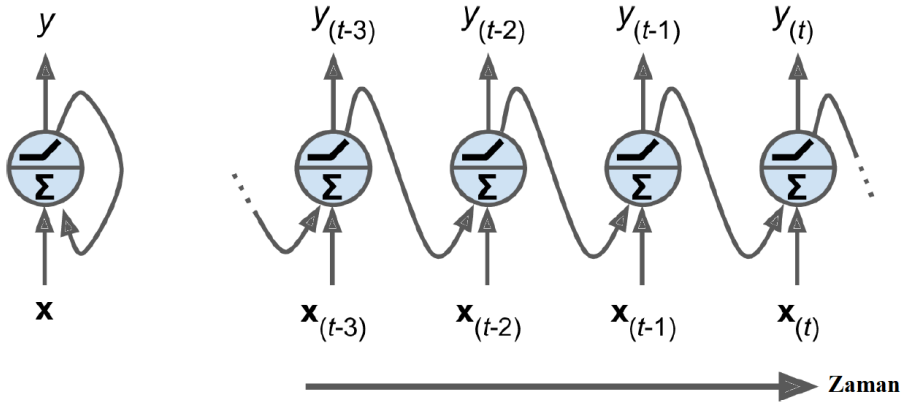
Burada sanki hata yapay sinir ağının çıkışından girdiye, geriye doğru yayıldığı için bu algoritmaya hatayı geriye yayma adı verilmiştir (Alpaydın 2014).

6.2 Yinelemeli Sinir Ağları

Yinelemeli sinir ağları, geleceği tahmin edebilen bir ağ sınıfıdır. Hisse senedi fiyatları gibi zaman serisi verilerini analiz edebilir ve ne zaman alınıp satılacağını söyleyebilirler. Otonom sürüş sistemlerinde, araba yönergelerini tahmin edebilir ve kazaların önlenmesine yardımcı olabilirler. Daha genel olarak, diğer ağlar gibi sabit boyutlu girdiler yerine rastgele uzunluktaki diziler üzerinde çalışabilirler. Örneğin, girdi olarak cümleler, belgeler veya ses örnekleri alabilirler, bu da onları otomatik çeviri veya konuşmadan metne gibi doğal dil işleme uygulamaları için son derece kullanışlı hale getirir (Géron 2019).

6.2.1 Yinelemeli Nöronlar ve Katmanlar

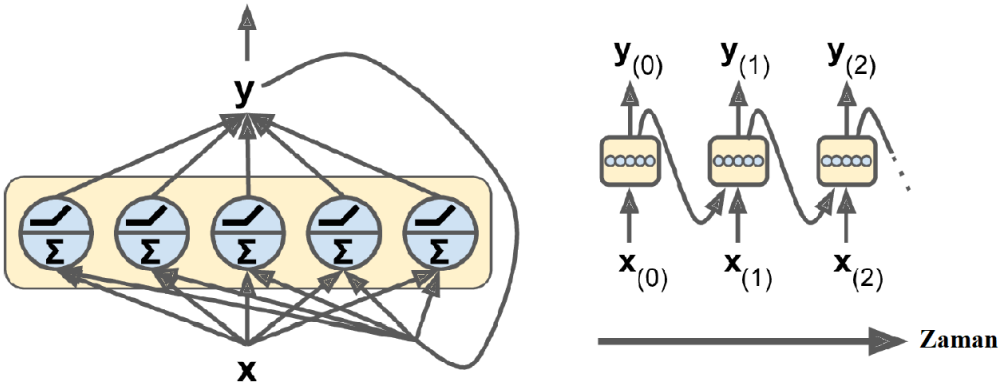
Yinelemeli sinir ağları, aynı zamanda geriye dönük bağlantılara sahip olması dışında, ileri beslemeli sinir ağlarına oldukça benzemektedir. Şekil 6.10'da girdileri alan, bir çıktı üreten ve bu çıktıyı kendisine geri gönderen bir nörondan oluşan mümkün olan en basit RNN'i yapısı gösterilmiştir (Géron 2019).



Şekil 6.10: a) Yinelemeli nöron, b) Zaman içinde açılmış yapısı.

Her t zaman adımında, bu yinelemeli nöron, $\mathbf{x}_{(t)}$ girişlerini ve ayrıca önceki $y_{(t-1)}$ zaman adımından kendi çıktısını alır. İlk zaman adımından önceki çıkış olmadığından genellikle 0 olarak ayarlanır. Bu küçük ağ, Şekil 6.10.b'de gösterildiği gibi zaman eksinine göre temsil edilebilmektedir. Buna ağın zaman içinde açılması denir. Bu durum Şekil 6.11'de gösterilmiştir (Géron 2019).

Buna göre kolayca yenilebilir nöronlardan oluşan bir katman oluşturulabilmektedir. Her t zaman adımında, her nöron, Şekil 6.11'de gösterildiği gibi, hem giriş vektörü $\mathbf{x}_{(t)}$ 'yi hem de önceki zaman adımından $\mathbf{y}_{(t-1)}$ çıkış vektörünü alır. Burada hem girdiler hem de çıktılar vektör olarak tanımlanmaktadır.



Şekil 6.11: a) Yinelemeli nöron katmanı, b) Zaman içinde açılmış yapısı.

Her yinelemeli nöronun iki ağırlık seti vardır: biri $\mathbf{x}_{(t)}$ girdileri için, diğeri ise önceki adımın çıktıları $\mathbf{y}_{(t-1)}$ içindir. Bu ağırlık vektörlerine $\mathbf{w}_x$ ve $\mathbf{w}_y$ denilmektedir. Tek bir yinelemeli nöron yerine tüm yinelemeli katmanı göz önünde bulundurursak, tüm ağırlık vektörleri $\mathbf{W}_x$ ve $\mathbf{W}_y$ olmak üzere iki ağırlık matrisi ile temsil edilebilir. Tüm yinelemeli katmanın çıkış vektörü, Denklem (6.38)'deki gibi hesaplanabilir.

$$\mathbf{y}_{(t)} = \phi(\mathbf{W}_x^T \mathbf{x}_{(t)} + \mathbf{W}_y^T \mathbf{y}_{(t-1)} + \mathbf{b}) \tag{6.38}$$

Burada $\mathbf{b}$ bias vektörü ve $\phi(.)$ aktivasyon fonksiyonudur.

Tıpkı ileri beslemeli sinir ağlarında olduğu gibi, t zamanındaki tüm girdileri bir $\mathbf{X}_{(t)}$ girdi matrisine yerleştirerek tüm bir mini-batch için tek seferde yinelemeli bir katmanın çıktısı Denklem (6.39)'daki gibi hesaplanabilmektedir.

$$\begin{aligned} \mathbf{Y}_{(t)} &= \phi(\mathbf{X}_{(t)}\mathbf{W}_x + \mathbf{Y}_{(t-1)}\mathbf{W}_y + \mathbf{b}) \\ &= \phi([\mathbf{X}_{(t)} \quad \mathbf{Y}_{(t-1)}]\mathbf{W} + \mathbf{b}), \mathbf{W} = \begin{bmatrix} \mathbf{W}_x \\ \mathbf{W}_y \end{bmatrix} \end{aligned} \tag{6.39}$$

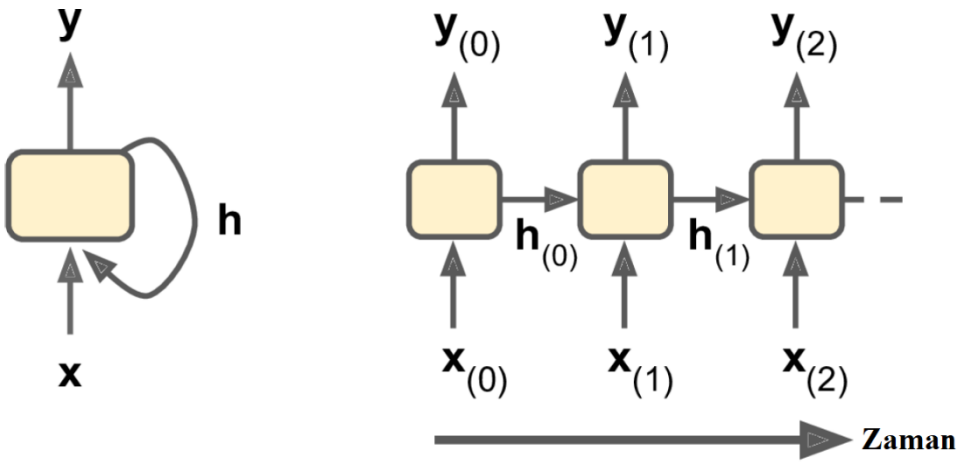
Burada $\mathbf{Y}_{(t)}$, mini-batchdeki her örnek için t zaman adımındaki katmanın çıktılarını içeren $m \times n_{nöron}$ boyutlu bir matristir (m, mini yığındaki örneklerin sayısı ve $n_{nöron}$, nöronların sayısıdır). $\mathbf{X}_{(t)}$, tüm örnekler için girdileri içeren $m \times n_{girdi}$ boyutlu bir matristir (n_{girdi}, girdi özelliklerinin sayısıdır). $\mathbf{W}_x$, mevcut zaman adımının girişleri için bağlantı ağırlıklarını içeren $n_{girdi} \times n_{nöron}$ boyutlu bir matristir. $\mathbf{W}_y$, önceki zaman adımının çıktıları için bağlantı ağırlıklarını içeren $n_{nöron} \times n_{nöron}$ boyutlu bir matristir. $\mathbf{b}$, her nöronun bias terimini içeren $n_{nöron}$ büyüklüğünde bir vektördür. Ağırlık matrisleri $\mathbf{W}_x$ ve $\mathbf{W}_y$ genellikle dikey olarak $(n_{girdi} \times n_{nöron}) \times n_{nöron}$ boyutundaki bir $\mathbf{W}$ ağırlık matrisinde birleştirilir. $[\mathbf{X}_{(t)} \quad \mathbf{Y}_{(t-1)}]$ notasyonu, $\mathbf{X}_{(t)}$ ve $\mathbf{Y}_{(t-1)}$ matrislerinin yatay olarak sıralanmasını temsil etmektedir.

$\mathbf{Y}_{(t)}$'nin $\mathbf{X}_{(t)}$ ve $\mathbf{Y}_{(t-1)}$'in bir fonksiyonu olduğu, $\mathbf{X}_{(t-1)}$ ve $\mathbf{Y}_{(t-2)}$'nin de $\mathbf{X}_{(t-2)}$ ve $\mathbf{Y}_{(t-3)}$'ün bir fonksiyonu olduğu unutulmamalıdır. Bu durumda $\mathbf{Y}_{(t)}$, $t = 0$ zamanından itibaren (yani $\mathbf{X}_{(0)}, \mathbf{X}_{(1)}, \ldots, \mathbf{X}_{(t)}$) tüm girdilerin fonksiyonudur. İlk

zaman adımında ($t = 0$) önceki çıktılar yoktur, dolayısıyla bunların tamamen sıfır oldukları varsayılmaktadır.

6.2.2 Bellek Hücreleri

t zaman adımında yenilenen bir nöronun çıktısı, önceki zaman adımlarından gelen tüm girdilerin bir fonksiyonu olduğundan, onun bir hafıza biçimine sahip olduğu söylenebilmektedir. Bir sinir ağının zaman adımlarında bazı durumları koruyan bir parçasına bellek hücresi (ya da sadece hücre) denir. Tek bir yenilenen nöron veya bir yenilenen nöron tabakası, sadece kısa patternleri öğrenebilen (tipik olarak yaklaşık 10 adım uzunluğundadır, ancak bu gövdeye bağlı olarak değişir) çok temel bir hücredir. Genel olarak, bir hücrenin $\mathbf{h}_{(t)}$ (h gizli anlamına gelir) ile gösterilen, t zaman adımındaki bir hücrenin durumu, o zaman adımındaki bazı girdilerin ve önceki zaman adımındaki durumun $\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, \mathbf{x}_{(t)})$ şeklide ifade edilen bir fonksiyonudur. $\mathbf{y}_{(t)}$ ile gösterilen t zaman adımındaki çıktısı da önceki durumun ve mevcut girdilerin bir fonksiyonudur. Temel hücrelerde, çıktı basitçe duruma eşittir ancak daha karmaşık hücrelerde bu, Şekil 6.12'de gösterildiği gibi her zaman böyle değildir (Géron 2019).



Şekil 6.12: Bir hücrenin gizli durumu ve çıktısının farklılığı.

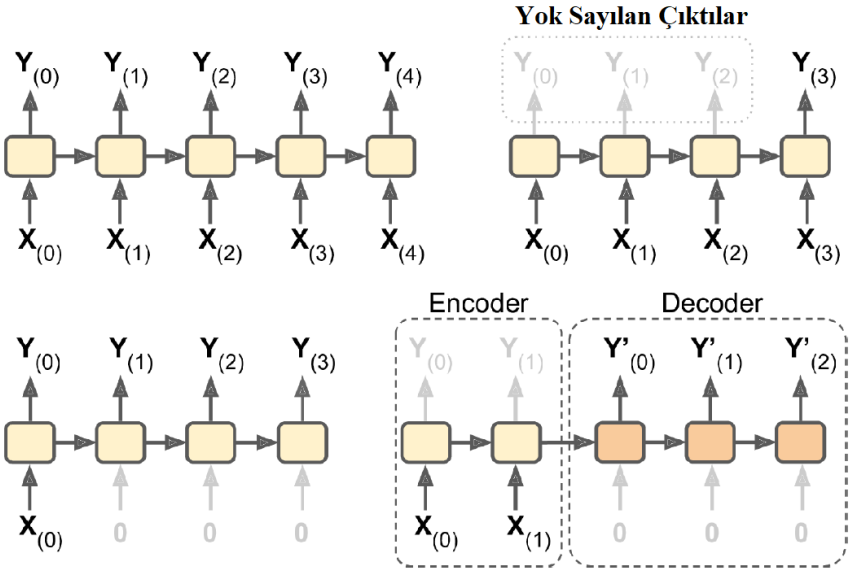
6.2.3 Giriş ve Çıkış Sıraları

Bir yinelemeli sinir ağı aynı anda bir sıra girdi ve bir sıra çıktı üretebilmektedir. Bu durum Şekil 6.13.a'da gösterilir (Géron 2019). Bu tür sıradan sıraya ağ, hisse senedi fiyatları gibi zaman serilerini tahmin etmek için kullanışlıdır.

Alternatif olarak, ağ bir sıra girdi ile beslenebilir ve sonuncusu dışındaki tüm çıktılar yok sayılabilmektedir. Bu durum Şekil 6.13.b'de gösterilmektedir. Başka bir deyişle, bu sıradan vektöre bir ağdır. Örneğin, ağ bir film incelemesine karşılık gelen bir kelime dizisi ile beslenebilir ve ağ buna karşılık bir duyarlılık puanı verir.

Tam tersi, tüm zaman adımlarında ağ aynı girdi vektörü ile tekrar tekrar beslenebilir ve buna karşılık bir sıra çıktı üretir. Bu durum Şekil 6.13.c'de gösterilmektedir. Bu, vektörden sıraya bir ağdır. Örneğin, girdi bir resim olabilir ve çıktı o resim için bir resim yazısı olabilir.

Son olarak encoder adı verilen bir sıradan vektöre ağ ve ardından decoder adı verilen vektörden sıraya ağ yapıları vardır. Bu durum Şekil 6.13.d'de gösterilmektedir. Örneğin bu yapı, bir cümleyi bir dilden başka bir dile çevirmek için kullanılabilir. Ağı bir dilde bir cümle ile beslersiniz, encoder bu cümleyi tek bir vektör temsiline dönüştürür ve ardından decoder bu vektörün kodunu başka bir dilde bir cümleye dönüştürür. Encoder-Decoder adı verilen bu iki aşamalı model, tek bir sıradan sıraya RNN ile anında çeviri yapmaya çalışmaktan çok daha iyi çalışır.

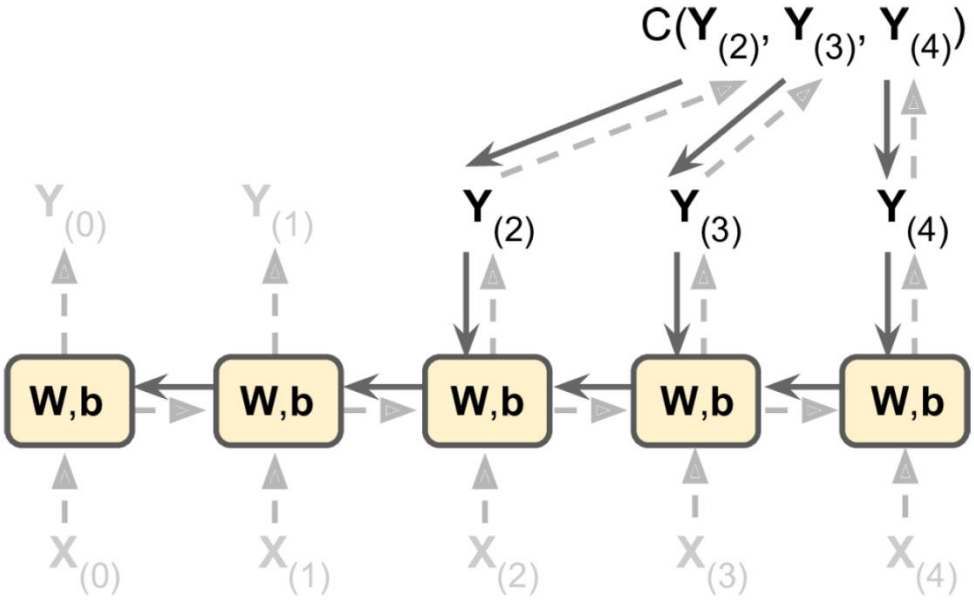


Şekil 6.13: a) Sıradan sıraya, b) Sıradan vektöre, c) Vektörden sıraya, d) Encoder-Decoder Ağları.

6.2.4 Yinelemeli Sinir Ağlarının Eğitimi

Bir yinelemeli sinir ağını eğitmek için işin püf noktası, ağı zaman içinde açmak ve ardından hatayı geri yayma algoritmasını kullanmaktır. Bu strateji backpropagation through time (BPTT) olarak adlandırılır. Bu durum Şekil 6.14'te gösterilmiştir (Géron 2019).

Normal geri yayılımda olduğu gibi, yuvarlanmamış ağda bir ilk ileri geçiş vardır ve Şekil 6.14'te kesikli oklarla gösterilmiştir. Daha sonra çıkış sırası bir maliyet fonksiyonu $C(\mathbf{Y}_{(0)}, \mathbf{Y}_{(1)}, \ldots, \mathbf{Y}_{(T)})$ kullanılarak değerlendirilir. Bu maliyet fonksiyonun Şekil 6.14'te gösterildiği gibi bazı çıktıları görmezden gelebileceği unutulmamalıdır (örneğin, sıradan vektöre RNN'de, en sonuncusu dışında tüm çıktılar yok sayılır). Bu maliyet fonksiyonunun gradyanları daha sonra açılmamış ağ üzerinden geriye doğru yayılır ve Şekil 6.14'te düz oklarla gösterilmiştir. Son olarak model parametreleri, BPTT sırasında hesaplanan gradyanlar kullanılarak güncellenir. Gradyanların, yalnızca nihai çıktıdan değil, maliyet fonksiyonu tarafından kullanılan tüm çıktılardan geriye doğru aktığı unutulmamalıdır. Ayrıca, her zaman adımında aynı **W** ve **b** parametreleri kullanıldığından, geri yayılım doğru olanı yapacak ve tüm zaman adımlarını toplayacaktır.



Şekil 6.14: Backpropagation Through Time.

6.3 Aşırı Öğrenme Makineleri

ELM, Huang ve diğ. (2006) tarafından, tek gizli katmanlı ileri beslemeli ağların (SLFN'ler) mimarisine dayalı olarak geliştirilmiştir. Bu yeni makine öğrenmesi algoritması, öznitelik öğrenme, boyut indirgeme, sınıflandırma ve regresyon gibi çok çeşitli alanlarda başarıyla kullanılmıştır. Geleneksel sinir ağları ile karşılaştırıldığında, başarısı temel olarak aşağıdaki üç özellikten kaynaklanmaktadır.

- Giriş katmanında rastgele oluşturulmuş ağırlıklarla ELM, mükemmel genelleme performansı gösterir ve bu durum gerçek dünya uygulamalarına uygundur.
- Parametreleri öğrenme hızı, öğrenme epokları ve yerel minimumları yinelemeli olarak ayarlanmış geleneksel sinir ağlarıyla karşılaştırıldığında, ELM, son derece hızlı öğrenme hızıyla giriş ağırlıklarını düzeltir.
- ELM, hem en küçük eğitim hatasını hem de ağırlıkların en küçük normunu elde etmek için kolayca uygulanabilir.

Tek gizli katmanlı ileri beslemeli ağların topolojik yapısına göre, genel bir ELM ağı oluşturulabilir. N adet veri örneği $(\mathbf{x}_i, t_i)$ için, burada $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \cdots, x_{i,c}] \in \mathbf{R}^c$ ifadesinin giriş vektörü ve t_i 'nin hedef olduğunu varsayarsak, L gizli düğümlü ELM aşağıdaki gibi modellenebilir:

$$o_i = \sum_{k=1}^{L} \beta_k g_k(\mathbf{x}_i) = \sum_{k=1}^{L} \beta_k g(\mathbf{x}_i \cdot \mathbf{w}_k + b_k); i = 1,2, \dots, N \tag{6.40}$$

burada, $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_L]$ giriş katmanı ve gizli katman arasındaki ağırlık matrisini, $\mathbf{b} = [\mathrm{b}_1, \mathrm{b}_2, \dots, \mathrm{b}_L]$ bias vektörünü, $g(*)$ doğrusal veya doğrusal olmayan aktif fonksiyonu, $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \dots, \boldsymbol{\beta}_L]^T$ çıkışın ağırlık matrisini, o_i ise i'inci ELM çıkışını ifade eder. Matris formuna dönüştürülerek Denklem (6.40) aşağıdaki gibi yeniden yazılır.

$$\mathbf{O} = \mathbf{H}\boldsymbol{\beta} \tag{6.41}$$

burada $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_N]^T$ nihai çıkış matrisini, $\mathbf{H}$ Denklem (6.42)'daki gibi gösterilen gizli katman çıkış matrisini ifade eder.

$$\mathbf{H} = \begin{bmatrix} g(\mathbf{x}_1 \cdot \mathbf{w}_1 + b_1) & \cdots & g(\mathbf{x}_1 \cdot \mathbf{w}_L + b_L) \\ \vdots & \ddots & \vdots \\ g(\mathbf{x}_N \cdot \mathbf{w}_1 + b_1) & \cdots & g(\mathbf{x}_N \cdot \mathbf{w}_L + b_L) \end{bmatrix}_{N \times L} \tag{6.42}$$

Optimum ELM ağını eğitmek için amacın, model çıktıları ile hedefler arasındaki Denklem (6.43)'de ifade edilen hatayı en aza indirmek olduğunu varsayıyoruz.

$$Minimize \|\mathbf{O} - \mathbf{T}\|^2 = \sum_{i=1}^{N} \|\mathbf{o}_i - \mathbf{t}_i\|^2 \tag{6.43}$$

burada, $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N]^T$ hedef matrisidir. Denklem (6.41)'deki amaç fonksiyonu Denklem (6.43)'de yerine yazılarak ve çözüm için en küçük kareler yöntemi kullanılarak, çıkış ağırlıkları aşağıdaki gibi hesaplanabilir:

$$\min_{\beta} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2 \rightarrow \boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{6.44}$$

burada $\mathbf{H}^{\dagger}$, $\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$ olarak hesaplanabilen, gizli katman çıkışı $\mathbf{H}$ matrisinin Moore-Penrose genelleştirilmiş tersidir (Ouyang ve diğ. 2019).

7. KARAR AĞAÇLARI

7.1 Karar Ağaçları

Karar Ağaçları (Decision Trees), sınıflandırma ve regresyon için kullanılan parametrik olmayan denetimli bir öğrenme yöntemidir (Breiman ve diğ. 1984). Amaç, veri özelliklerinden çıkarılan basit karar kurallarını öğrenerek bir hedef değişkenin değerini tahmin eden bir model oluşturmaktır. Bir ağaç, parçalı sabit bir yaklaşım olarak görülebilir (Scikit-learn 2020).

Örneğin, Şekil 7.1'de, karar ağaçları verilerden bir dizi if-then-else karar kuralıyla bir sinüs eğrisine yaklaşmayı öğrenir. Ağaç ne kadar derin olursa, karar kuralları o kadar karmaşık ve model o kadar uygun olur.

Şekil 7.1: Karar Ağaçları ile sinüs yaklaşıklama.

Karar ağaçlarının bazı avantajları şunlardır:

- Anlaması ve yorumlaması basittir. Ağaçlar görselleştirilebilir.
- Çok az veri hazırlığı gerektirir.
- Ağacı kullanmanın maliyeti (yani verileri tahmin etme), ağacı eğitmek için kullanılan veri noktalarının sayısına göre logaritmiktir.
- Hem sayısal hem de kategorik verileri işleyebilir.

- Çok çıkışlı problemleri çözebilir.
- Beyaz kutu modeli kullanır. Bir modelde belirli bir durum gözlemlenebilirse, durumun açıklaması boole mantığıyla kolayca açıklanabilir. Buna karşılık, bir kara kutu modelinde (örneğin, yapay sinir ağında), sonuçların yorumlanması daha zor olabilir.
- İstatistiksel testler kullanarak bir modeli doğrulamak mümkündür. Bu, modelin güvenilirliğini hesaba katmayı mümkün kılar.
- Varsayımları, verileri üretildiği gerçek model tarafından bir şekilde ihlal edilse bile iyi performans gösterir (Scikit-learn 2020).

Karar ağaçlarının bazı dezavantajları şunlardır:

- Karar ağaçlarını kullananlar, verileri iyi genellemeyen aşırı karmaşık ağaçlar oluşturabilirler. Buna aşırı öğrenme denir. Bu problemden kaçınmak için budama, bir yaprak düğümünde gereken minimum örnek sayısının ayarlanması veya ağacın maksimum derinliğinin ayarlanması gibi mekanizmalar gereklidir.
- Karar ağaçları kararsız olabilir. Çünkü verilerdeki küçük değişiklikler tamamen farklı bir ağacın üretilmesine neden olabilir. Bu sorun, bir topluluk içinde karar ağaçları kullanılarak azaltılabilir.
- Karar ağaçlarının tahminleri ne düzgün ne de süreklidir. Şekil 7.1'de görüldüğü gibi parçalı ve sabit yaklaşımlardır. Bu nedenle, ekstrapolasyonda iyi değillerdir.
- Optimal karar ağacını öğrenme probleminin, optimalliğin çeşitli yönleri altında ve hatta basit kavramlar için bile NP-tam olduğu bilinmektedir. Sonuç olarak, pratik karar ağacı öğrenme algoritmaları, her düğümde yerel olarak optimal kararların alındığı açgözlü algoritma gibi sezgisel algoritmalara dayanmaktadır. Bu tür algoritmalar, global olarak optimal karar ağacını döndürmeyi garanti edemez. Bu durum, özelliklerin ve örneklerin değiştirilerek rastgele örneklendiği bir topluluk öğrenicisinde birden fazla ağaç eğiterek azaltılabilir (Scikit-learn 2020).

7.1.1 Karar Ağacı Algoritmaları

Literatürdeki çeşitli karar ağacı algoritmaları şunlardır:

- ID3 algoritması, Quinlann (1986) tarafından geliştirilmiştir. Algoritma, her düğüm için, kategorik hedegler için en büyük bilgi kazancını sağlayacak kategorik özelliği bularak çok yollu bir ağaç oluşturur. Ağaçlar maksimum boyutlarına büyütülür ve ardından ağacın görünmeyen verilere genelleme yapma yeteneğini geliştirmek için genellikle bir budama adımı uygulanır.
- C4.5 algoritması, ID3'ün halefidir ve sürekli öznitelik değerini ayrık bir aralık kümesine bölen ayrı bir özniteliği (sayısal değişkenlere dayalı olarak) dinamik olarak tanımlayarak özniteliklerin kategorik olması gerektiği kısıtlamasını kaldırmıştır. C4.5, eğitilmiş ağaçları (yani ID3 algoritmasının çıktısı) if-then kuralları kümelerine dönüştürür. Her kuralın bu doğruluğu daha sonra uygulanmaları gereken sırayı belirlemek için değerlendirilir. Budama, kuralın doğruluğu onsuz iyileşirse, kuralın ön koşulu kaldırılarak yapılır (Quinlann 1993).
- C5.0 algoritması, Quinlan'ın tescilli lisansı altındaki en son sürümüdür. Daha az bellek kullanır ve daha doğru olmakla birlikte C4.5'ten daha küçük kural kümeleri oluşturur.
- CART (Sınıflandırma ve Regresyon Ağaçları), C4.5'e oldukça benzer, ancak sayısal hedef değişkenlerini (regresyon) desteklemesi ve kural kümelerinin hesaplanması bakımından farklıdır. CART, her düğümde en büyük bilgi kazancını sağlayan özelliği ve eşiği kullanarak ikili ağaçlar oluşturur (Breiman ve diğ. 1984).

Verilen eğitim vektörleri $\mathbf{x}_i \in \mathbf{R}^n, i = 1, \ldots, l$ ve bir etiket vektörü $\mathbf{y} \in \mathbf{R}^l$, karar ağacı öznitelik uzayını aynı etiketlere veya benzer hedef değerlere sahip örneklerin birlikte gruplanacağı şekilde özyinelemeli olarak bölümlere ayırır.

m düğümündeki veriler, N_m örnekleriyle $\mathbf{Q}_m$ ile temsil edilir. Öznitelik j ve eşik t_m 'den oluşan her bir aday $\theta = (j, t_m)$ için, veriler $Q_m^{sol}(\theta)$ ve $Q_m^{sağ}(\theta)$ alt kümelerine ayrılır. Bu alt kümeler Denklem (7.1)'de gösterilmiştir.

$$\begin{aligned} Q_m^{sol}(\theta) &= \{(x, y) | x_j \leq t_m\} \\ Q_m^{sağ}(\theta) &= Q_m \backslash Q_m^{sol}(\theta) \end{aligned} \tag{7.1}$$

m düğümünün aday bölünmesinin kalitesi daha sonra seçimi çözülmekte olan göreve (sınıflandırma veya regresyon) bağlı olan bir safsızlık fonksiyonu veya kayıp fonksiyonu **H** kullanılarak hesaplanır.

$$G(Q_m, \theta) = \frac{N_m^{sol}}{N_m} H(Q_m^{sol}(\theta)) + \frac{N_m^{sağ}}{N_m} H(Q_m^{sağ}(\theta)) \tag{7.2}$$

Safsızlığı en aza indiren parametreler seçilir. Bu durum Denklem (7.3)'te gösterilmiştir.

$$\theta^* = argmin_\theta G(Q_m, \theta) \tag{7.3}$$

İzin verilen maksimum derinliğe ulaşana kadar $Q_m^{sol}(\theta^*)$ ve $Q_m^{sağ}(\theta^*)$ alt kümeleri için yineleme, $N_m < min_{örnek}$ veya $N_m = 1$ koşulları altındadır.

Hedef sürekli bir değerse, o zaman m düğümü için, gelecekteki bölmeler için konumların belirlenmesine ilişkin olarak en aza indirilecek ortak kriterler, Ortalama Karesel Hata (MSE), Poisson sapması ve Ortalama Mutlak Hatadır (MAE).

MSE ve Poisson sapması, terminal düğümlerinin tahmin edilen değerini düğümün öğrenilen ortalama $\bar{y}_m$ değerine ayarlarken, MAE, terminal düğümlerinin tahmin edilen değerini medyan $median(y)_m$ 'e ayarlar. Ortalama Karesel Hata Denklem (7.4)'teki gibidir.

$$\begin{aligned} \bar{y}_m &= \frac{1}{N_m} \sum_{y \in Q_m} y \\ H(Q_m) &= \frac{1}{N_m} \sum_{y \in Q_m} (y - \bar{y}_m)^2 \end{aligned} \tag{7.4}$$

Yarım Poisson sapması Denklem (7.5)'te gösterilmiştir.

$$H(Q_m) = \frac{1}{N_m} \sum_{y \in Q_m} (y \log \frac{y}{\bar{y}_m} - y + \bar{y}_m) \tag{7.5}$$

Hedef bir sayı veya bir frekans ise, Poisson sapması iyi bir seçim olabilmektedir. Her durumda $y \geq 0$, bu kriteri kullanmak için gerekli bir koşuldur.

Ortalama Mutlak Hata ise Denklem (7.6)'da belirtilmiştir.

$$\begin{aligned} median(y)_m &= \underset{y \in Q_m}{median}(y) \\ H(Q_m) &= \frac{1}{N_m} \sum_{y \in Q_m} |y - median(y)_m| \end{aligned} \tag{7.6}$$

7.2 Rastgele Ormanlar

Rastgele orman, $\{\theta\}$'nın bağımsız, aynı şekilde dağıtılmış rastgele vektörler olduğu ve her ağacın x girişindeki en popüler sınıf için bir birim oy kullandığı, ağaç yapılı $\{h(\mathbf{x}, \theta_k), k = 1, \ldots\}$ sınıflandırıcılarının bir koleksiyonundan oluşan bir sınıflandırıcıdır (Breiman 2001).

Regresyon için rastgele ormanlar, ağaçların rastgele bir θ vektörüne bağlı olarak büyütülmesiyle oluşturulur, öyle ki ağaç tahmincisi $h(\mathbf{x}, \theta_k)$, sınıf etiketlerinin aksine sayısal değerler alır. Çıktı değerleri sayısaldır ve eğitim setinin bağımsız olarak Y, $\mathbf{X}$ rastgele vektörünün dağılımından çekildiğini varsayıyoruz. Herhangi bir sayısal öngörücü $h(x)$ için ortalama genelleştirilmiş karesel hata Denklem (7.7)'deki gibidir.

$$E_{\mathbf{X},Y}(Y - h(\mathbf{X}))^2 \tag{7.7}$$

Rastgele orman tahmincisi, $\{h(\mathbf{x}, \theta_k)\}$ ağaçlarının k'den fazlasının ortalaması alınarak oluşturulur. Ormandaki ağaçların sayısı sonsuza giderken, neredeyse kesin olarak Denklem (7.8)'deki gibidir.

$$E_{\mathbf{X},Y}(Y - av_k h(\mathbf{x}, \theta_k))^2 \rightarrow E_{\mathbf{X},Y}(Y - E_\theta h(\mathbf{X}, \theta))^2 \tag{7.8}$$

Denklem (7.8)'in sağ tarafını A-ormanının genelleme hatası olarak gösterilmektedir. Bir ağacın ortalama genelleştirilmiş karesel hatası Denklem (7.9) gibi tanımlanmaktadır.

$$PE^*(tree) = E_\theta E_{\mathbf{X},Y}(Y - h(\mathbf{X},\theta))^2 \tag{7.9}$$

Tüm θ değerleri için, $EY = E_{\mathbf{X}}h(\mathbf{X},\theta)$'dır. Sonra,

$$PE^*(forest) \leq \bar{\rho}PE^*(tree) \tag{7.10}$$

Denklem (7.10)'da $\bar{\rho}$, $Y - h(\mathbf{X},\theta)$ ve $Y - h(\mathbf{X},\theta')$ ifadelerinin artık değerleri arasındaki ağırlıklı korelasyondur, burada θ ve θ' bağımsızdır.

$$\begin{aligned} PE^*(forest) &= E_{\mathbf{X},Y}[E_\theta(Y - h(\mathbf{X},\theta))]^2 \\ &= E_\theta E_{\theta'} E_{\mathbf{X},Y}(Y - h(\mathbf{X},\theta))(Y - h(\mathbf{X},\theta')) \end{aligned} \tag{7.11}$$

Denklem (7.11)'deki sağ taraftaki terim bir kovaryanstır ve Denklem (7.12)'deki gibi yazılabilir.

$$E_\theta E_{\theta'}(\rho(\theta,\theta')sd(\theta)sd(\theta')) \tag{7.12}$$

Burada $sd(\theta) = \sqrt{E_{\mathbf{X},Y}(Y - h(\mathbf{X},\theta))^2}$ şeklindedir. Ağırlıklı korelasyon Denklem (7.13)'deki gibi tanımlanır.

$$\bar{\rho} = \frac{E_\theta E_{\theta'}(\rho(\theta,\theta')sd(\theta)sd(\theta'))}{(E_\theta sd(\theta))^2} \tag{7.13}$$

Daha sonra,

$$PE^*(forest) = \bar{\rho}(E_\theta sd(\theta))^2 \leq \bar{\rho}PE^*(tree) \tag{7.14}$$

ifadesi yazılabilir.

Rastgele orman, $\bar{\rho}$ faktörü tarafından kullanılan ağaçların ortalama hatasını azaltır. Kullanılan randomizasyonun, düşük korelasyonu hedeflemesi gerekmektedir (Breiman 2001).

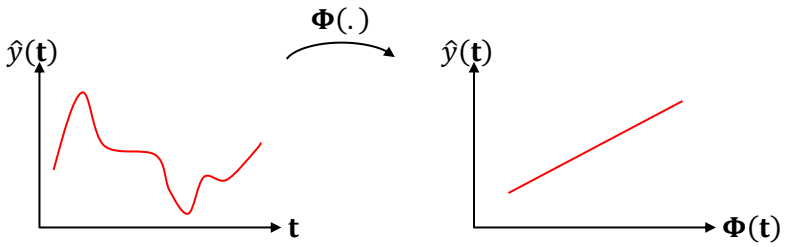
8. DESTEK VEKTÖR MAKİNELERİ

Destek Vektör Makineleri (SVM), istatistiksel öğrenme ve yapısal riski minimuma indirme ilkesiyle, sınıflandırma ve regresyon problemlerinin çözümü için Vapnik tarafından geliştirilmiş bir yapay öğrenme yöntemidir (Vapnik 1995).

$\mathbf{t}_i \in \mathbb{R}^R$ ve $y_i \in \mathbb{R}$ olmak üzere $\mathfrak{D}_{\text{MISO}} = \{\mathbf{t}_i, y_i\}_{i=1}^N$ şeklinde verilen bir Çok-Girişli Tek-Çıkışlı (Multi-Input Single-Output-MISO) veriyi ele alalım. Regresyon problemi olarak ele alınan problemde amaç bu verilerin giriş-çıkış ilişkisini uygun bir model ile temsil etmektir. SVM yaklaşımı ile bu regresyon problemi çözülebilir (Smola ve Schölkopf 2004). Sınıflandırma problemine benzer şekilde, regresyon problemlerinin çözümünde de öznitelik uzayında doğrusal olan bir modelden yararlanılmaktadır. Bu model,

$$\hat{y}(\mathbf{t}) = \mathbf{w}^T \mathbf{\Phi}(\mathbf{t}) + b \tag{8.1}$$

ile verilmektedir, burada $\mathbf{\Phi}(.)$ sınıflandırma probleminde olduğu gibi giriş uzayından öznitelik uzayına bir dönüşüm fonksiyonudur. Şekil 8.1'den de görüldüğü gibi bu model giriş uzayında doğrusal değildir ancak öznitelik uzayında doğrusaldır (İplikçi 2019).

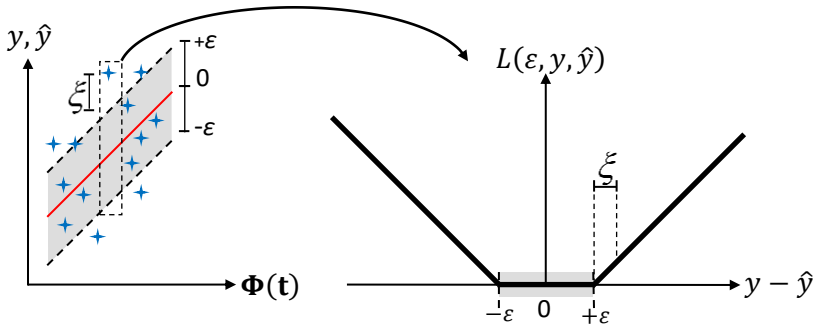


Şekil 8.1: Giriş uzayı ve öznitelik uzayı arasındaki ilişki.

Böylece, giriş uzayında doğrusal-olmayan bir veri, öznitelik uzayında doğrusal olan bir veriye dönüşmekte ve doğrusal bir modelle modellenebilmektedir.

8.1 ε-SVR Algoritması

ε-SVR algoritması, $\hat{y}(\mathbf{t}) = \mathbf{w}^T\boldsymbol{\Phi}(\mathbf{t}) + b$ şeklindeki regresyon modelinin belli bir giriş verisine karşı çıkışta oluşacak hatayı bulmak için Şekil 8.2 ve Denklem (8.2)'deki gibi Vapnik's ε-insensitive loss function kullanır (İplikçi 2019).



Şekil 8.2: Vapnik's ε-insensitive loss function.

$$L(\varepsilon, y, \hat{y}) = \begin{cases} 0 & |y - \hat{y}| \le \varepsilon \\ |y - \hat{y}| & |y - \hat{y}| > \varepsilon \end{cases} \tag{8.2}$$

ε-SVR algoritması, bu regresyon modelini kullanarak regresyon problemini bir optimizasyon problemi olarak şu şekilde ifade eder:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} \ & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ \text{Kıs.:} \ & y_i - \mathbf{w}^T\boldsymbol{\Phi}(\mathbf{t}_i) - b \le \varepsilon + \xi_i, \quad i = 1, \dots, N \\ & \mathbf{w}^T\boldsymbol{\Phi}(\mathbf{t}_i) + b - y_i \le \varepsilon + \xi_i^*, \quad i = 1, \dots, N \\ & \xi_i, \xi_i^* \ge 0 \qquad\qquad i = 1, \dots, N \end{aligned} \tag{8.3}$$

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\mu}, \boldsymbol{\mu}^*) = & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) - \sum_{i=1}^{N}\beta_i[\varepsilon + \xi_i - y_i + \mathbf{w}^T\boldsymbol{\Phi}(\mathbf{t}_i) + b] \\ & - \sum_{i=1}^{N}\beta_i^*[\varepsilon + \xi_i^* - \mathbf{w}^T\boldsymbol{\Phi}(\mathbf{t}_i) - b + y_i] - \sum_{i=1}^{N}\mu_i\xi_i - \sum_{i=1}^{N}\mu_i^*\xi_i^* \\ = & \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T\sum_{i=1}^{N}(\beta_i - \beta_i^*)\boldsymbol{\Phi}(\mathbf{t}_i) - \varepsilon\sum_{i=1}^{N}(\beta_i + \beta_i^*) \\ & + \sum_{i=1}^{N}y_i(\beta_i - \beta_i^*) + b\sum_{i=1}^{N}(\beta_i^* - \beta_i) + \sum_{i=1}^{N}\xi_i(C - \beta_i - \mu_i) \\ & + \sum_{i=1}^{N}\xi_i^*(C - \beta_i^* - \mu_i^*) \end{aligned}$$

burada, β_i, β_i^*, μ_i ve μ_i^*'ler Lagrange çarpanları, ξ_i büyüklüğü $\mathbf{t}_i$ verisi için modelin yaptığı hata miktarıdır ve gevşek değişken olarak adlandırılmaktadır, C büyüklüğü ise hataları cezalandıran bir ceza parametresidir. Bu Lagrange ifadesinin birincil değişkenlere $\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*$ göre türevi alınıp sıfıra eşitlenirse Denklem (8.4)'teki eşitlikler elde edilir:

$$\begin{aligned}
&\frac{\partial \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\mu}, \boldsymbol{\mu}^*)}{\partial \mathbf{w}} = \mathbf{0} \mapsto \mathbf{w} = \sum_{i=1}^{N} (\beta_i - \beta_i^*) \boldsymbol{\Phi}(\mathbf{t}_i) \\
&\frac{\partial \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\mu}, \boldsymbol{\mu}^*)}{\partial b} = 0 \mapsto \sum_{i=1}^{N} (\beta_i^* - \beta_i) = 0 \\
&\frac{\partial \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\mu}, \boldsymbol{\mu}^*)}{\xi_i} = 0 \mapsto C - \beta_i - \mu_i = 0, \quad i = 1, \dots, N \\
&\frac{\partial \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\mu}, \boldsymbol{\mu}^*)}{\xi_i^*} = 0 \mapsto C - \beta_i^* - \mu_i^* = 0, \quad i = 1, \dots, N
\end{aligned} \tag{8.4}$$

Burada Lagrange çarpanlarının $\boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\mu}, \boldsymbol{\mu}^* \geq \mathbf{0}$ şartlarını sağlamaları gerektiği unutulmamalıdır. Yukarıda bulunan eşitlikler Lagrange ifadesinde yerine konduğunda,

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\beta}^*) = &-\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\beta_i - \beta_i^*)\left(\beta_j - \beta_j^*\right) \boldsymbol{\Phi}^T(\mathbf{t}_i) \boldsymbol{\Phi}\left(\mathbf{t}_j\right) \\
&-\varepsilon \sum_{i=1}^{N} (\beta_i + \beta_i^*) + \sum_{i=1}^{N} y_i (\beta_i - \beta_i^*)
\end{aligned} \tag{8.5}$$

elde edilir. Böylece, regresyon probleminin ikinci formülasyonu şu şekilde yazılabilir:

$$\begin{aligned}
\min_{\boldsymbol{\beta}, \boldsymbol{\beta}^*} \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\beta}^*) = &\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\beta_i - \beta_i^*)\left(\beta_j - \beta_j^*\right) \boldsymbol{\Phi}^T(\mathbf{t}_i) \boldsymbol{\Phi}\left(\mathbf{t}_j\right) \\
&-\varepsilon \sum_{i=1}^{N} (\beta_i + \beta_i^*) + \sum_{i=1}^{N} y_i (\beta_i - \beta_i^*) \\
\text{Kıs.:} \quad &\sum_{i=1}^{N} (\beta_i^* - \beta_i) = 0 \\
&0 \leq \beta_i, \beta_i^* \leq C, \quad i = 1, \dots, N
\end{aligned} \tag{8.6}$$

Bu bir karesel programlama (QP) problemidir ve bu problemin çözümü ile tüm β_i ve β_i^* değerleri elde edilir. $\mathbf{w} = \sum_{i=1}^{N}(\beta_i - \beta_i^*)\mathbf{\Phi}(\mathbf{t}_i)$ olduğu hatırlanırsa, $\hat{y}(\mathbf{t}) = \mathbf{w}^T\mathbf{\Phi}(\mathbf{t}) + b$ şeklinde olan regresyon modeli artık ikincil formda,

$$\hat{y}(\mathbf{t}) = \sum_{i=1}^{N}(\beta_i - \beta_i^*)\mathbf{\Phi^T}(\mathbf{t}_i)\mathbf{\Phi}(\mathbf{t}) + b \tag{8.7}$$

şeklinde ifade edilebilir. İkincil formdaki regresyon modelinde görülen ve öznitelik uzayında bir iç-çarpım olan $\mathbf{\Phi^T}(\mathbf{t}_i)\mathbf{\Phi}(\mathbf{t})$ ifadesi dönüşüm fonksiyonuna karşı düşen kernel fonksiyonu ile,

$$\mathbf{\Phi^T}(\mathbf{t}_i)\mathbf{\Phi}(\mathbf{t}) = K(\mathbf{t}_i, \mathbf{t}) \tag{8.8}$$

şeklinde ifade edilebileceği için regresyon modeli

$$\hat{y}(\mathbf{t}) = \sum_{i=1}^{N}(\beta_i - \beta_i^*)K(\mathbf{t}_i, \mathbf{t}) + b \tag{8.9}$$

olarak yazılabilir. QP probleminin çözümünden elde edilen β_i ve β_i^* değerleri, $\alpha_i = \beta_i - \beta_i^*$ şeklinde tek bir değişken ile ifade edildiğinde model

$$\hat{y}(\mathbf{t}) = \sum_{i=1}^{N}\alpha_i K(\mathbf{t}_i, \mathbf{t}) + b \tag{8.10}$$

haline gelir. b teriminin hesabı şu şekildedir: $0 \leq \beta_i, \beta_i^* \leq C$ şartını sağlayan her bir destek vektörü için

$$|y_i - \hat{y}(\mathbf{x}_i)| = \varepsilon \tag{8.11}$$

şartı sağlanmalıdır, yani

$$y_i - \sum_{j\in\mathcal{S}}\alpha_j K(\mathbf{t}_j, \mathbf{x}_i) - b = \varepsilon \tag{8.12}$$

koşulu her destek vektörü için sağlanacak şekilde ortalama bir b terimi bulunur. Sadece destek vektörleri dikkate alındığında model,

$$\hat{y}(\mathbf{t}) = \sum_{i \in \mathcal{S}} \alpha_i K(\mathbf{t}_i, \mathbf{t}) + b \tag{8.13}$$

şeklinde ifade edilebilir.

9. DENEYSEL SONUÇLAR

Bu bölümde, Enerji Piyasaları İşletme A.Ş. (EPİAŞ)'ye ait internet sitesinde bulunan Şeffaflık Platformu'ndan alınan, Türkiye geneline ait gerçekleşen gerçek zamanlı tüketim verileri kullanılarak yapay sinir ağları, karar ağaçları ve destek vektör makineleriyle oluşturulan toplamda 13 farklı çok-adımlı ileri tahmin yöntemi ile 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini gerçekleştirilmiştir.

9.1 Veri Seti

Bu tez çalışmasında, Enerji Piyasaları İşletme A.Ş. (EPİAŞ) tarafından, internet sitelerinde bulunan Şeffaflık Platformu üzerinden açık kaynaklı olarak yayınlanan Türkiye geneline ait gerçekleşen gerçek zamanlı tüketim verileri kullanılmıştır (EPİAŞ 2021).

Kullanılan veri seti, Türkiye geneline ait, 31 Aralık 2015 saat 00:00 ile 28 Temmuz 2021 saat 23:00 arasındaki toplam 48888 adet saatlik elektrik enerjisi tüketim verisidir.

Bu tez çalışmasında, zamanda 24 adım geriye gidilerek MISO ve MIMO zaman serisi verileri oluşturulmuştur. Yapay öğrenme modellerini eğitmek için; MISO yapay öğrenme modellerinde eğitim için 29304 adet, doğrulama için 19536 adet veri ve MIMO yapay öğrenme modellerinde eğitim için 29290 adet, doğrulama için 19527 adet veri kullanılmıştır. Modellerin başarılarını karşılaştırmak için ise 28 Temmuz 2021 gününe ait 24 adet veri kullanılmıştır.

9.2 Eğitimde Kullanılan Yazılımlar ve Donanımlar

Yapay öğrenme modellerini oluşturmak, bu modelleri eğitmek ve modeller arasındaki karşılaştırmaları yapmak amacıyla açık kaynak kodlu bir programlama dili olan Python kullanılmıştır.

Yapay öğrenme modellerinin eğitilmesinde ve karşılaştırılmasında kullanılan kişisel bilgisayar Intel i7-9750H işlemci ve 16 GB RAM'e sahiptir. Python ile geliştirilen kodlar bu bilgisayar üzerinde çalıştırılmıştır.

Yapay öğrenme ve makine öğrenmesi konularında Python kullanılarak geliştirilmiş pek çok açık kaynaklı kütüphane bulunmaktadır. Açık kaynaklı bu kütüphanelerden bazıları şunlardır: TensorFlow, scikit-learn, Theano, PyTorch. Bu tez çalışmasında TensorFlow kütüphanesi ile birlikte bu kütüphane üzerinde çalışan üst düzey bir kütüphane olan Keras ve scikit-learn kullanılmıştır.

9.3 Yapay Sinir Ağlarına Ait Tahmin Sonuçları

Bu tez çalışmasında, 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini yapmak üzere üç yapay sinir ağı yapısı kullanılmıştır. Bu yapılar; İleri Beslemeli Yapay Sinir Ağları (FFNN), Yinelemeli Yapay Sinir Ağları (RNN) ve Aşırı Öğrenme Makineleri (ELM)'dir.

9.3.1 İleri Beslemeli Yapay Sinir Ağlarına Ait Tahmin Sonuçları

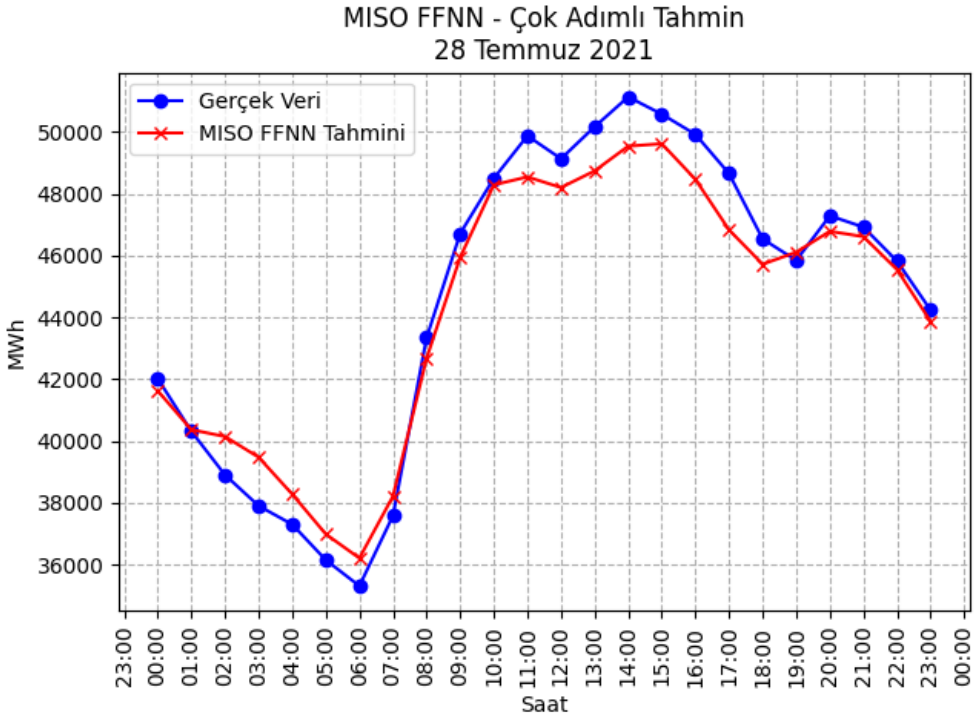
24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini yapmak üzere Çok Girişli-Tek Çıkışlı İleri Beslemeli Yapay Sinir Ağı (MISO FFNN) ve Çok Girişli-Çok Çıkışlı İleri Beslemeli Yapay Sinir Ağı (MIMO FFNN) yapıları kullanılmıştır.

MISO FFNN'yi eğitmek amacıyla 4 katmanlı bir yapay sinir ağı modeli oluşturulmuştur. Zamanda 24 adım geri gidildiği için yapay sinir ağının giriş boyutu 24'tür.

Birinci gizli katmanda 96 adet nöron kullanılmış ve aktivasyon fonksiyonu olarak tanh seçilmiştir. İkinci gizli katmanda ise 48 nöron kullanılmış ve aktivasyon fonksiyonu olarak tanh seçilmiştir. Gizli katmanlardaki nöron sayıları Grid Search kullanılarak, en düşük RMSE değerini verecek şekilde belirlenmiştir. Çıkış katmanında ise 1 nöron kullanılmış ve aktivasyon fonksiyonu olarak lineer aktivasyon fonksiyonu kullanılmıştır.

Oluşturulan model 28 Temmuz 2021 tarihine ait 24 adet saatlik elektrik enerjisi tüketim verisi üzerinde test edilmiştir. Yapılan test sonucunda RMSE değeri 978,37 ve MAPE değeri %1,9 olarak tespit edilmiştir.

MISO FFNN'ye ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini sonucu Şekil 9.1'de gösterilmiştir.



Şekil 9.1: MISO FFNN 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini.

MISO FFNN'ye ait 24 saatlik çok adımlı ileri elektrik enerjisi tüketim tahmini değerleri, gerçek değerler ve değerler arasındaki hataların mutlak değerleri Tablo 9.1'deki gibidir.

Tablo 9.1: MISO FFNN 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini sonuçları.

MISO FFNN			
28 Temmuz 2021			
Saat	**Gerçek Değerler (MWh)**	**Tahmin Değerleri (MWh)**	**Hata (MWh)**
00:00	42010,09	41618,63	391,46
01:00	40305,75	40360,24	54,49
02:00	38898,23	40138,63	1240,40
03:00	37896,78	39470,34	1573,56
04:00	37295,62	38273,79	978,17

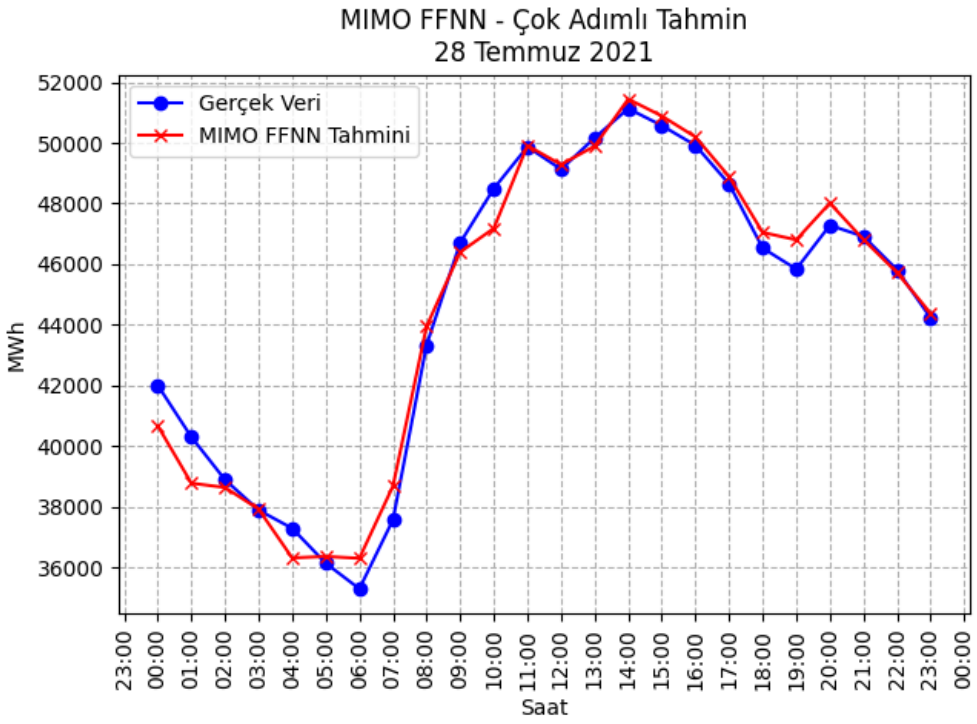
05:00	36144,84	36980,74	835,90
06:00	35301,97	36203,74	901,77
07:00	37577,15	38207,06	629,91
08:00	43334,98	42672,73	662,25
09:00	46709,36	45939,13	770,23
10:00	48493,17	48282,92	210,25
11:00	49870,84	48541,90	1328,94
12:00	49129,19	48197,33	931,86
13:00	50163,34	48735,43	1427,91
14:00	51130,40	49536,30	1594,10
15:00	50569,96	49614,88	955,08
16:00	49917,37	48461,03	1456,34
17:00	48648,69	46827,69	1821,00
18:00	46533,60	45715,30	818,30
19:00	45838,97	46104,88	265,91
20:00	47282,90	46781,19	501,71
21:00	46916,85	46607,48	309,37
22:00	45816,06	45540,34	275,72
23:00	44227,09	43861,73	365,36

Diğer bir ileri beslemeli yapay sinir ağı modeli olan MIMO FFNN'yi eğitmek amacıyla 4 katmanlı bir yapay sinir ağı modeli oluşturulmuştur. Zamanda 24 adım geri gidildiği için yapay sinir ağının giriş boyutu 24'tür.

Birinci gizli katmanda 168 adet nöron kullanılmış ve aktivasyon fonksiyonu olarak tanh seçilmiştir. İkinci gizli katmanda ise 48 nöron kullanılmış ve aktivasyon fonksiyonu olarak tanh seçilmiştir. Gizli katmanlardaki nöron sayıları Grid Search kullanılarak, en düşük RMSE değerini verecek şekilde belirlenmiştir. Çıkış katmanında ise 24 nöron kullanılmış ve aktivasyon fonksiyonu olarak lineer aktivasyon fonksiyonu kullanılmıştır.

Oluşturulan model 28 Temmuz 2021 tarihine ait 24 adet saatlik elektrik enerjisi tüketim verisi üzerinde test edilmiştir. Yapılan test sonucunda RMSE değeri 701,88 ve MAPE değeri %1,26 olarak tespit edilmiştir.

MIMO FFNN'ye ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini sonucu Şekil 9.2'de gösterilmiştir.



Şekil 9.2: MIMO FFNN 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini.

MIMO FFNN'ye ait 24 saatlik çok adımlı ileri elektrik enerjisi tüketim tahmini değerleri, gerçek değerler ve değerler arasındaki hataların mutlak değerleri Tablo 9.2'deki gibidir.

Tablo 9.2: MIMO FFNN 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini sonuçları.

MIMO FFNN			
28 Temmuz 2021			
Saat	**Gerçek Değerler (MWh)**	**Tahmin Değerleri (MWh)**	**Hata (MWh)**
00:00	42010,09	40674,85	1335,24
01:00	40305,75	38782,74	1523,01
02:00	38898,23	38642,72	255,51
03:00	37896,78	37928,16	31,38
04:00	37295,62	36320,87	974,75
05:00	36144,84	36367,89	223,05
06:00	35301,97	36308,77	1006,80
07:00	37577,15	38689,29	1112,14

08:00	43334,98	43956,00	621,02
09:00	46709,36	46400,22	309,14
10:00	48493,17	47182,34	1310,83
11:00	49870,84	49909,69	38,85
12:00	49129,19	49284,53	155,34
13:00	50163,34	49892,31	271,03
14:00	51130,40	51443,98	313,58
15:00	50569,96	50884,45	314,49
16:00	49917,37	50195,47	278,10
17:00	48648,69	48889,46	240,77
18:00	46533,60	47047,94	514,34
19:00	45838,97	46803,95	964,98
20:00	47282,90	48013,62	730,72
21:00	46916,85	46813,70	103,15
22:00	45816,06	45720,35	95,71
23:00	44227,09	44380,78	153,69

9.3.2 Yinelemeli Yapay Sinir Ağlarına Ait Tahmin Sonuçları

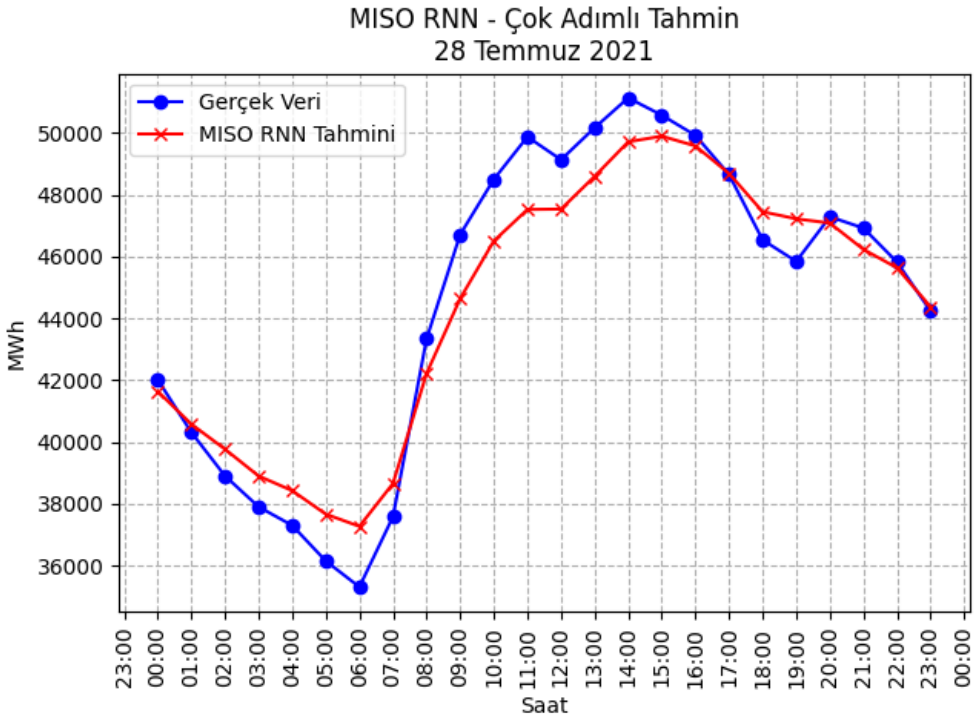
24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini yapmak üzere Çok Girişli Tek-Çıkışlı Yinelemeli Yapay Sinir Ağı (MISO RNN) ve Çok Girişli-Çok Çıkışlı Yinelemeli Yapay Sinir Ağı (MIMO RNN) yapıları kullanılmıştır.

MISO RNN'yi eğitmek amacıyla 4 katmanlı bir yapay sinir ağı modeli oluşturulmuştur. Zamanda 24 adım geri gidildiği için yapay sinir ağının giriş boyutu 24'tür.

Birinci gizli katmanda 24 adet nöron kullanılmış ve aktivasyon fonksiyonu olarak tanh seçilmiştir. İkinci gizli katmanda ise 24 nöron kullanılmış ve aktivasyon fonksiyonu olarak tanh seçilmiştir. Gizli katmanlardaki nöron sayıları Grid Search kullanılarak, en düşük RMSE değerini verecek şekilde belirlenmiştir. Çıkış katmanında ise 1 nöron kullanılmış ve aktivasyon fonksiyonu olarak lineer aktivasyon fonksiyonu kullanılmıştır.

Oluşturulan model 28 Temmuz 2021 tarihine ait 24 adet saatlik elektrik enerjisi tüketim verisi üzerinde test edilmiştir. Yapılan test sonucunda RMSE değeri 1233,33 ve MAPE değeri %2,36 olarak tespit edilmiştir.

MISO RNN'ye ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini sonucu Şekil 9.3'te gösterilmiştir.



Şekil 9.3: MISO RNN 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini.

MISO RNN'ye ait 24 saatlik çok adımlı ileri elektrik enerjisi tüketim tahmini değerleri, gerçek değerler ve değerler arasındaki hataların mutlak değerleri Tablo 9.3'teki gibidir.

Tablo 9.3: MISO RNN 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini sonuçları.

MISO RNN			
28 Temmuz 2021			
Saat	**Gerçek Değerler (MWh)**	**Tahmin Değerleri (MWh)**	**Hata (MWh)**
00:00	42010,09	41633,95	376,14
01:00	40305,75	40571,37	265,62
02:00	38898,23	39762,52	864,29
03:00	37896,78	38893,72	996,94
04:00	37295,62	38421,08	1125,46

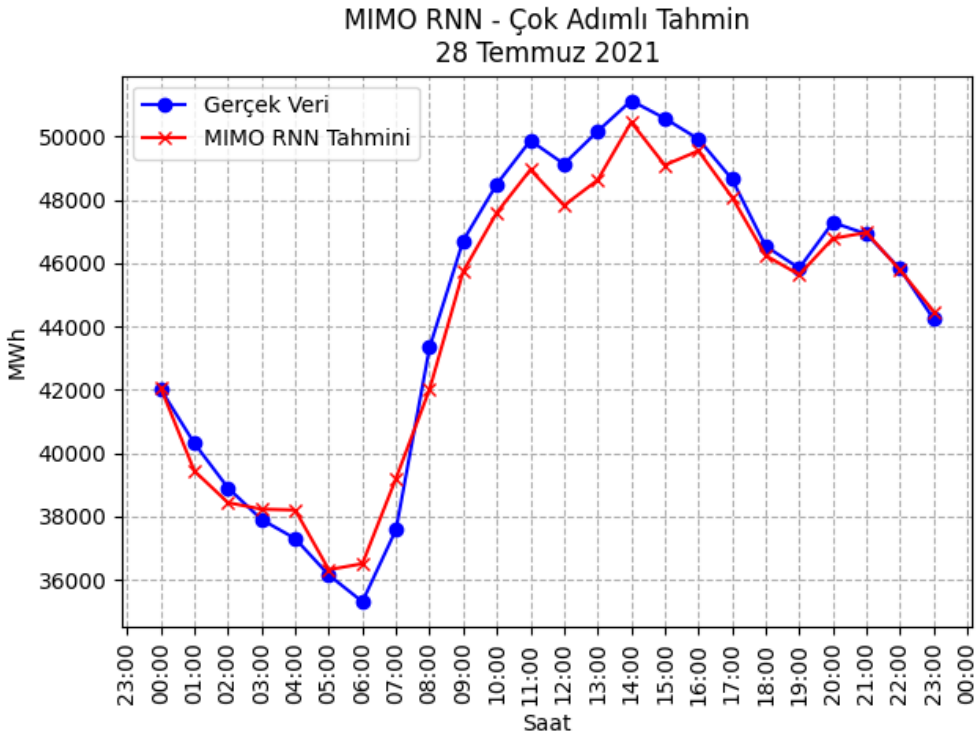
05:00	36144,84	37659,39	1514,55
06:00	35301,97	37264,68	1962,71
07:00	37577,15	38651,62	1074,47
08:00	43334,98	42214,23	1120,75
09:00	46709,36	44637,80	2071,56
10:00	48493,17	46498,74	1994,43
11:00	49870,84	47519,07	2351,77
12:00	49129,19	47530,89	1598,30
13:00	50163,34	48589,73	1573,61
14:00	51130,40	49709,50	1420,90
15:00	50569,96	49892,06	677,90
16:00	49917,37	49575,26	342,11
17:00	48648,69	48684,57	35,88
18:00	46533,60	47445,62	912,02
19:00	45838,97	47217,18	1378,21
20:00	47282,90	47079,32	203,58
21:00	46916,85	46227,29	689,56
22:00	45816,06	45629,15	186,91
23:00	44227,09	44367,49	140,40

Diğer bir yinelemeli yapay sinir ağı modeli olan MIMO RNN'yi eğitmek amacıyla 4 katmanlı bir yapay sinir ağı modeli oluşturulmuştur. Zamanda 24 adım geri gidildiği için yapay sinir ağının giriş boyutu 24'tür.

Birinci gizli katmanda 72 adet nöron kullanılmış ve aktivasyon fonksiyonu olarak tanh seçilmiştir. İkinci gizli katmanda ise 96 nöron kullanılmış ve aktivasyon fonksiyonu olarak tanh seçilmiştir. Gizli katmanlardaki nöron sayıları Grid Search kullanılarak, en düşük RMSE değerini verecek şekilde belirlenmiştir. Çıkış katmanında ise 24 nöron kullanılmış ve aktivasyon fonksiyonu olarak lineer aktivasyon fonksiyonu kullanılmıştır.

Oluşturulan model 28 Temmuz 2021 tarihine ait 24 adet saatlik elektrik enerjisi tüketim verisi üzerinde test edilmiştir. Yapılan test sonucunda RMSE değeri 866 ve MAPE değeri %1,60 olarak tespit edilmiştir.

MIMO RNN'ye ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini sonucu Şekil 9.4'te gösterilmiştir.



Şekil 9.4: MIMO RNN 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini.

MIMO RNN'ye ait 24 saatlik çok adımlı ileri elektrik enerjisi tüketim tahmini değerleri, gerçek değerler ve değerler arasındaki hataların mutlak değerleri Tablo 9.4'teki gibidir.

Tablo 9.4: MIMO RNN 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini sonuçları.

MIMO RNN			
28 Temmuz 2021			
Saat	**Gerçek Değerler (MWh)**	**Tahmin Değerleri (MWh)**	**Hata (MWh)**
00:00	42010,09	42056,08	45,99
01:00	40305,75	39431,18	874,57
02:00	38898,23	38432,08	466,15
03:00	37896,78	38225,42	328,64
04:00	37295,62	38197,02	901,40
05:00	36144,84	36313,23	168,39
06:00	35301,97	36501,74	1199,77
07:00	37577,15	39196,79	1619,64

08:00	43334,98	42016,35	1318,63
09:00	46709,36	45734,31	975,05
10:00	48493,17	47590,40	902,77
11:00	49870,84	48948,87	921,97
12:00	49129,19	47819,35	1309,84
13:00	50163,34	48620,86	1542,48
14:00	51130,40	50451,42	678,98
15:00	50569,96	49093,17	1476,79
16:00	49917,37	49545,61	371,76
17:00	48648,69	48080,73	567,96
18:00	46533,60	46237,41	296,19
19:00	45838,97	45633,41	205,56
20:00	47282,90	46776,00	506,90
21:00	46916,85	46963,34	46,49
22:00	45816,06	45774,90	41,16
23:00	44227,09	44472,68	245,59

9.3.3 Aşırı Öğrenme Makinelerine Ait Tahmin Sonuçları

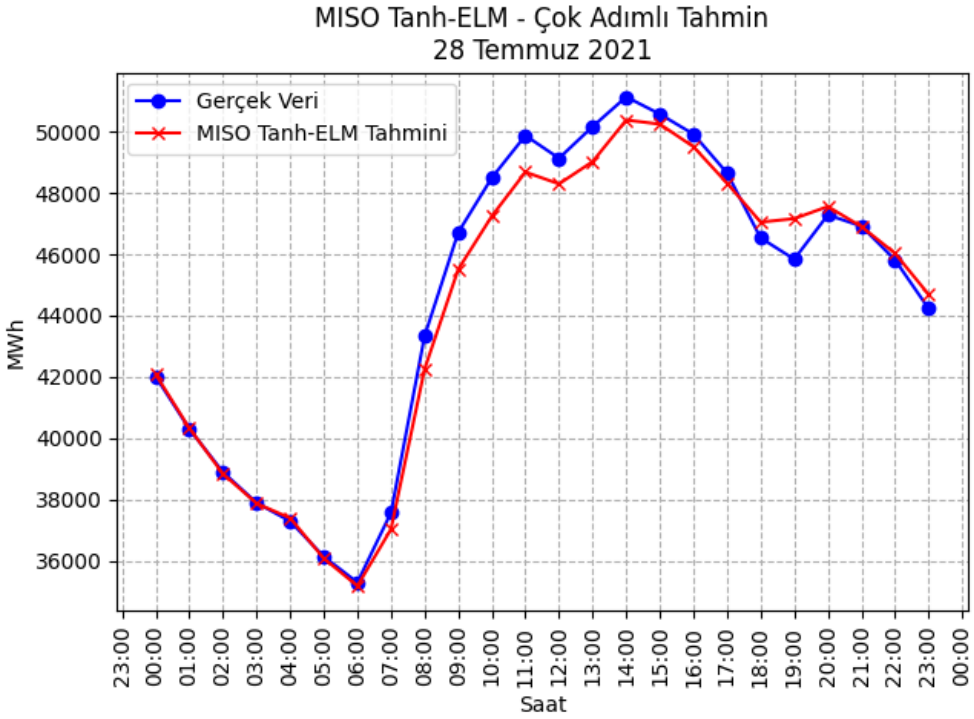
24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini yapmak üzere Çok Girişli-Tek Çıkışlı Tanh Aşırı Öğrenme Makinesi (MISO Tanh-ELM), Çok Girişli-Çok Çıkışlı Tanh Aşırı Öğrenme Makinesi (MIMO Tanh-ELM), Çok Girişli-Tek Çıkışlı RBF Aşırı Öğrenme Makinesi (MISO RBF-ELM) ve Çok Girişli-Çok Çıkışlı RBF Aşırı Öğrenme Makinesi (MIMO RBF-ELM) yapıları kullanılmıştır.

MISO Tanh-ELM'yi eğitmek amacıyla 3 katmanlı bir yapay sinir ağı modeli oluşturulmuştur. Zamanda 24 adım geri gidildiği için yapay sinir ağının giriş boyutu 24'tür.

Gizli katmanda 380 adet nöron kullanılmış ve aktivasyon fonksiyonu olarak tanh seçilmiştir. Gizli katmandaki nöron sayısı Grid Search kullanılarak, en düşük RMSE değerini verecek şekilde belirlenmiştir. Çıkış katmanında ise 1 nöron kullanılmış ve aktivasyon fonksiyonu olarak lineer aktivasyon fonksiyonu kullanılmıştır.

Oluşturulan model 28 Temmuz 2021 tarihine ait 24 adet saatlik elektrik enerjisi tüketim verisi üzerinde test edilmiştir. Yapılan test sonucunda RMSE değeri 683,89 ve MAPE değeri %1,09 olarak tespit edilmiştir.

MISO Tanh-ELM'ye ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini sonucu Şekil 9.5'te gösterilmiştir.



Şekil 9.5: MISO Tanh-ELM 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini.

MISO Tanh-ELM'ye ait 24 saatlik çok adımlı ileri elektrik enerjisi tüketim tahmini değerleri, gerçek değerler ve değerler arasındaki hataların mutlak değerleri Tablo 9.5'teki gibidir.

Tablo 9.5: MISO Tanh-ELM 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini sonuçları.

MISO Tanh-ELM			
28 Temmuz 2021			
Saat	**Gerçek Değerler (MWh)**	**Tahmin Değerleri (MWh)**	**Hata (MWh)**
00:00	42010,09	42079,41	69,32
01:00	40305,75	40327,63	21,88
02:00	38898,23	38843,96	54,27
03:00	37896,78	37876,58	20,20

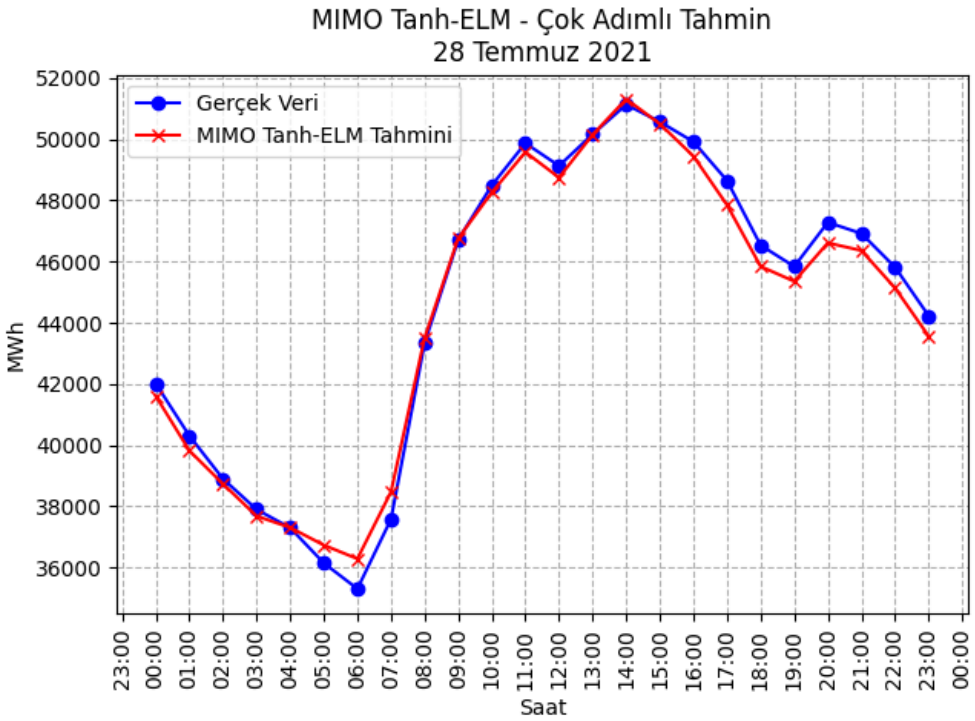
04:00	37295,62	37400,55	104,93
05:00	36144,84	36088,12	56,72
06:00	35301,97	35186,36	115,61
07:00	37577,15	37038,71	538,44
08:00	43334,98	42254,68	1080,30
09:00	46709,36	45507,29	1202,07
10:00	48493,17	47242,09	1251,08
11:00	49870,84	48679,31	1191,53
12:00	49129,19	48293,95	835,24
13:00	50163,34	49020,77	1142,57
14:00	51130,40	50376,36	754,04
15:00	50569,96	50242,95	327,01
16:00	49917,37	49507,09	410,28
17:00	48648,69	48318,28	330,41
18:00	46533,60	47048,46	514,86
19:00	45838,97	47160,94	1321,97
20:00	47282,90	47556,61	273,71
21:00	46916,85	46901,20	15,65
22:00	45816,06	46029,39	213,33
23:00	44227,09	44688,22	461,13

İkinci aşırı öğrenme makineleri modeli olan MIMO Tanh-ELM'yi eğitmek amacıyla 3 katmanlı bir yapay sinir ağı modeli oluşturulmuştur. Zamanda 24 adım geri gidildiği için yapay sinir ağının giriş boyutu 24'tür.

Gizli katmanda 204 adet nöron kullanılmış ve aktivasyon fonksiyonu olarak tanh seçilmiştir. Gizli katmandaki nöron sayısı Grid Search kullanılarak, en düşük RMSE değerini verecek şekilde belirlenmiştir. Çıkış katmanında ise 24 nöron kullanılmış ve aktivasyon fonksiyonu olarak lineer aktivasyon fonksiyonu kullanılmıştır.

Oluşturulan model 28 Temmuz 2021 tarihine ait 24 adet saatlik elektrik enerjisi tüketim verisi üzerinde test edilmiştir. Yapılan test sonucunda RMSE değeri 507,26 ve MAPE değeri %0,98 olarak tespit edilmiştir.

MIMO Tanh-ELM'ye ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini sonucu Şekil 9.6'da gösterilmiştir.



Şekil 9.6: MIMO Tanh-ELM 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini.

MIMO Tanh-ELM'ye ait 24 saatlik çok adımlı ileri elektrik enerjisi tüketim tahmini değerleri, gerçek değerler ve değerler arasındaki hataların mutlak değerleri Tablo 9.6'daki gibidir.

Tablo 9.6: MIMO Tanh-ELM 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini sonuçları.

MIMO Tanh-ELM			
28 Temmuz 2021			
Saat	**Gerçek Değerler (MWh)**	**Tahmin Değerleri (MWh)**	**Hata (MWh)**
00:00	42010,09	41604,84	405,25
01:00	40305,75	39819,66	486,09
02:00	38898,23	38732,19	166,04
03:00	37896,78	37680,91	215,87
04:00	37295,62	37303,58	7,96
05:00	36144,84	36730,17	585,33
06:00	35301,97	36284,54	982,57
07:00	37577,15	38470,26	893,11

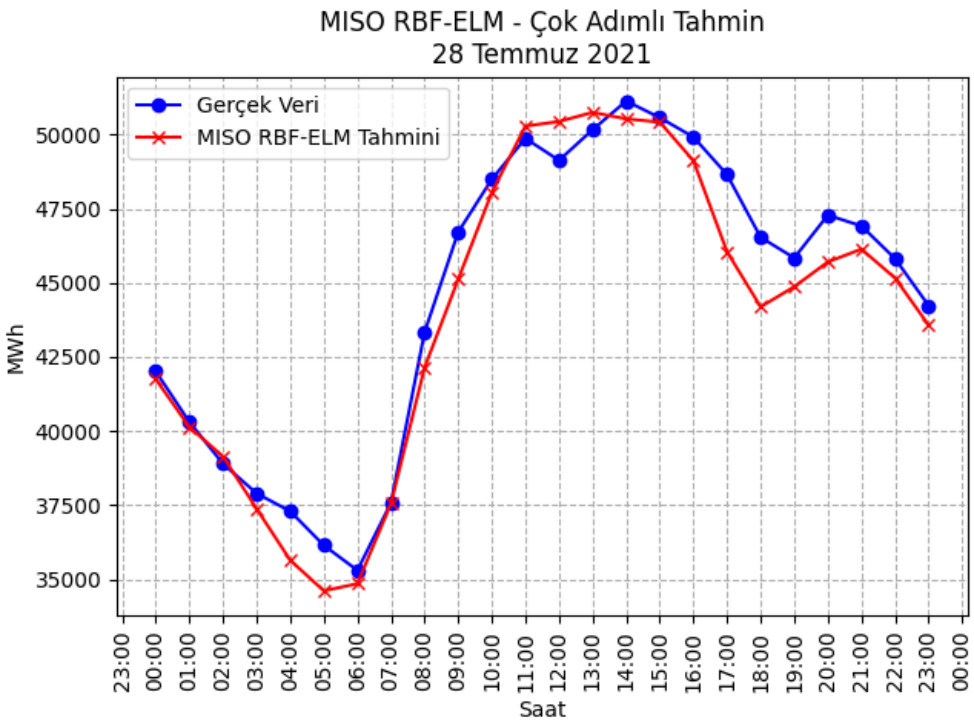
08:00	43334,98	43503,76	168,78
09:00	46709,36	46757,92	48,56
10:00	48493,17	48260,71	232,46
11:00	49870,84	49578,99	291,85
12:00	49129,19	48747,47	381,72
13:00	50163,34	50147,65	15,69
14:00	51130,40	51313,34	182,94
15:00	50569,96	50501,63	68,33
16:00	49917,37	49447,37	470,00
17:00	48648,69	47849,44	799,25
18:00	46533,60	45841,17	692,43
19:00	45838,97	45353,86	485,11
20:00	47282,90	46610,79	672,11
21:00	46916,85	46359,86	556,99
22:00	45816,06	45136,01	680,05
23:00	44227,09	43562,06	665,03

Üçüncü aşırı öğrenme makineleri modeli olan MISO RBF-ELM'yi eğitmek amacıyla 3 katmanlı bir yapay sinir ağı modeli oluşturulmuştur. Zamanda 24 adım geri gidildiği için yapay sinir ağının giriş boyutu 24'tür.

Gizli katmanda 292 adet nöron kullanılmış ve aktivasyon fonksiyonu olarak RBF seçilmiştir. Gizli katmandaki nöron sayısı Grid Search kullanılarak, en düşük RMSE değerini verecek şekilde belirlenmiştir. Çıkış katmanında ise 1 nöron kullanılmış ve aktivasyon fonksiyonu olarak lineer aktivasyon fonksiyonu kullanılmıştır.

Oluşturulan model 28 Temmuz 2021 tarihine ait 24 adet saatlik elektrik enerjisi tüketim verisi üzerinde test edilmiştir. Yapılan test sonucunda RMSE değeri 1123,34 ve MAPE değeri %2,01 olarak tespit edilmiştir.

MISO RBF-ELM'ye ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini sonucu Şekil 9.7'de gösterilmiştir.



Şekil 9.7: MISO RBF-ELM 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini.

MISO RBF-ELM'ye ait 24 saatlik çok adımlı ileri elektrik enerjisi tüketim tahmini değerleri, gerçek değerler ve değerler arasındaki hataların mutlak değerleri Tablo 9.7'deki gibidir.

Tablo 9.7: MISO RBF-ELM 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini sonuçları.

MISO RBF-ELM			
28 Temmuz 2021			
Saat	**Gerçek Değerler (MWh)**	**Tahmin Değerleri (MWh)**	**Hata (MWh)**
00:00	42010,09	41747,69	262,40
01:00	40305,75	40135,28	170,47
02:00	38898,23	39144,79	246,56
03:00	37896,78	37379,86	516,92
04:00	37295,62	35644,15	1651,47
05:00	36144,84	34610,55	1534,29
06:00	35301,97	34850,24	451,73
07:00	37577,15	37555,36	21,79
08:00	43334,98	42142,94	1192,04
09:00	46709,36	45120,02	1589,34
10:00	48493,17	48045,02	448,15

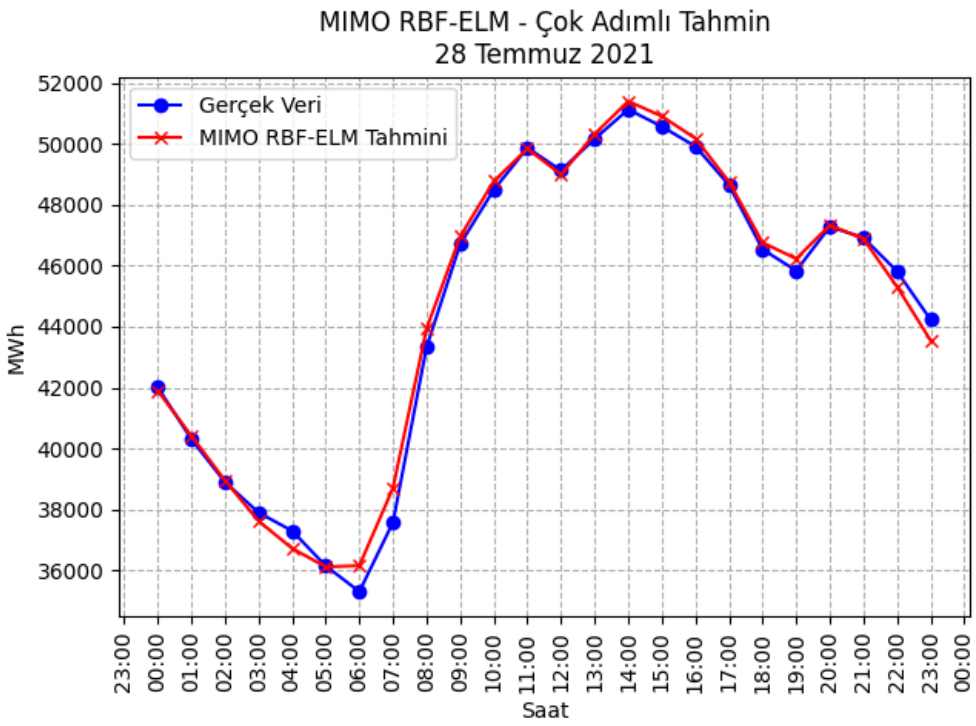
11:00	49870,84	50285,46	414,62
12:00	49129,19	50440,92	1311,73
13:00	50163,34	50748,25	584,91
14:00	51130,40	50529,77	600,63
15:00	50569,96	50418,61	151,35
16:00	49917,37	49118,68	798,69
17:00	48648,69	46046,79	2601,90
18:00	46533,60	44190,37	2343,23
19:00	45838,97	44866,47	972,50
20:00	47282,90	45703,89	1579,01
21:00	46916,85	46136,70	780,15
22:00	45816,06	45159,44	656,62
23:00	44227,09	43570,92	656,17

Dördüncü aşırı öğrenme makineleri modeli olan MIMO RBF-ELM'yi eğitmek amacıyla 3 katmanlı bir yapay sinir ağı modeli oluşturulmuştur. Zamanda 24 adım geri gidildiği için yapay sinir ağının giriş boyutu 24'tür.

Gizli katmanda 304 adet nöron kullanılmış ve aktivasyon fonksiyonu olarak RBF seçilmiştir. Gizli katmandaki nöron sayısı Grid Search kullanılarak, en düşük RMSE değerini verecek şekilde belirlenmiştir. Çıkış katmanında ise 24 adet nöron kullanılmış ve aktivasyon fonksiyonu olarak lineer aktivasyon fonksiyonu kullanılmıştır.

Oluşturulan model 28 Temmuz 2021 tarihine ait 24 adet saatlik elektrik enerjisi tüketim verisi üzerinde test edilmiştir. Yapılan test sonucunda RMSE değeri 416,94 ve MAPE değeri %0,73 olarak tespit edilmiştir.

MIMO RBF-ELM'ye ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini sonucu Şekil 9.8'de gösterilmiştir.



Şekil 9.8: MIMO RBF-ELM 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini.

MIMO RBF-ELM'ye ait 24 saatlik çok adımlı ileri elektrik enerjisi tüketim tahmini değerleri, gerçek değerler ve değerler arasındaki hataların mutlak değerleri Tablo 9.8'deki gibidir.

Tablo 9.8: MIMO RBF-ELM 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini sonuçları.

MIMO RBF-ELM			
28 Temmuz 2021			
Saat	**Gerçek Değerler (MWh)**	**Tahmin Değerleri (MWh)**	**Hata (MWh)**
00:00	42010,09	41900,14	109,95
01:00	40305,75	40414,69	108,94
02:00	38898,23	38951,35	53,12
03:00	37896,78	37619,98	276,80
04:00	37295,62	36720,54	575,08
05:00	36144,84	36113,93	30,91
06:00	35301,97	36155,17	853,20
07:00	37577,15	38703,90	1126,75
08:00	43334,98	43937,19	602,21
09:00	46709,36	46960,23	250,87
10:00	48493,17	48775,58	282,41

11:00	49870,84	49840,95	29,89
12:00	49129,19	48996,64	132,55
13:00	50163,34	50319,67	156,33
14:00	51130,40	51401,03	270,63
15:00	50569,96	50907,21	337,25
16:00	49917,37	50180,28	262,91
17:00	48648,69	48751,76	103,07
18:00	46533,60	46758,18	224,58
19:00	45838,97	46228,82	389,85
20:00	47282,90	47332,92	50,02
21:00	46916,85	46882,45	34,40
22:00	45816,06	45309,60	506,46
23:00	44227,09	43547,31	679,78

9.4 Karar Ağaçları Yapılarına Ait Tahmin Sonuçları

Bu tez çalışmasında, 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini yapmak üzere iki adet karar ağacı yapısı kullanılmıştır. Bu yapılar; Karar Ağaçları (Decision Trees) ve Rastgele Ormanlar (Random Forest)'dır.

9.4.1 Karar Ağaçlarına Ait Tahmin Sonuçları

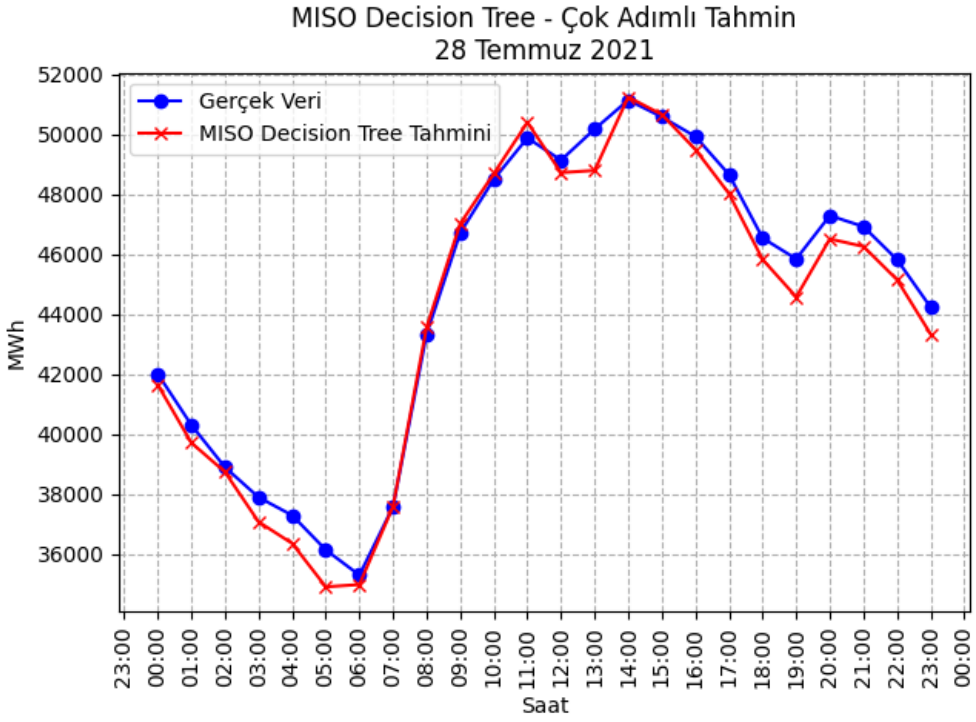
24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini yapmak üzere Çok Girişli-Tek Çıkışlı Karar Ağacı (MISO Decision Tree) ve Çok Girişli-Çok Çıkışlı Karar Ağacı (MIMO Decision Tree) yapıları kullanılmıştır.

MISO Decision Tree modelinin giriş boyutu zamanda 24 adım geri gidildiği için 24'tür.

Her düğümde bölünmeyi seçmek için kullanılan ayırıcı parametresi en iyi olarak seçilmiştir. Bir dahili düğümü bölmek için gereken minimum örnek sayısı 2 olarak belirlenmiştir. Bir yaprak düğümünde olması gereken minimum örnek sayısı ise 1 olarak seçilmiştir. Modelin çıkış boyutu ise 1'dir.

Oluşturulan model 28 Temmuz 2021 tarihine ait 24 adet saatlik elektrik enerjisi tüketim verisi üzerinde test edilmiştir. Yapılan test sonucunda RMSE değeri 683,13 ve MAPE değeri %1,30 olarak tespit edilmiştir.

MISO Decision Tree'ye ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini sonucu Şekil 9.9'da gösterilmiştir.



Şekil 9.9: MISO Decision Tree 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini.

MISO Decision Tree'ye ait 24 saatlik çok adımlı ileri elektrik enerjisi tüketim tahmini değerleri, gerçek değerler ve değerler arasındaki hataların mutlak değerleri Tablo 9.9'daki gibidir.

Tablo 9.9: MISO Decision Tree 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini sonuçları.

MISO Decision Tree			
28 Temmuz 2021			
Saat	**Gerçek Değerler (MWh)**	**Tahmin Değerleri (MWh)**	**Hata (MWh)**
00:00	42010,09	41645,56	364,53
01:00	40305,75	39712,94	592,81
02:00	38898,23	38738,57	159,66
03:00	37896,78	37078,08	818,70
04:00	37295,62	36359,22	936,40

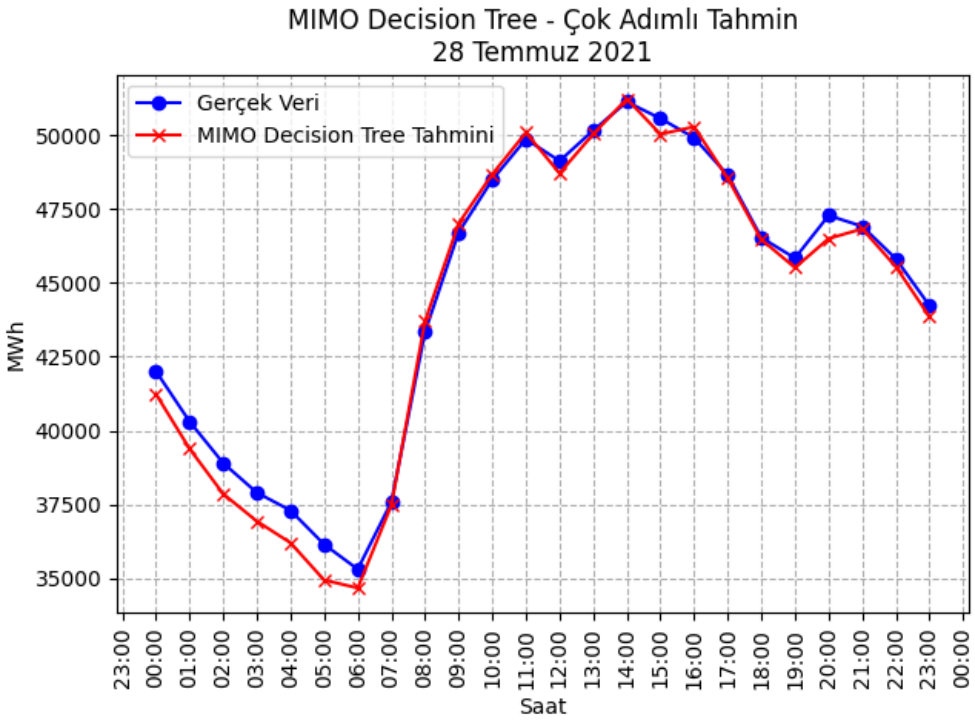
05:00	36144,84	34913,71	1231,13
06:00	35301,97	34987,67	314,30
07:00	37577,15	37585,42	8,27
08:00	43334,98	43559,68	224,70
09:00	46709,36	47012,23	302,87
10:00	48493,17	48681,12	187,95
11:00	49870,84	50389,47	518,63
12:00	49129,19	48714,86	414,33
13:00	50163,34	48785,84	1377,50
14:00	51130,40	51232,46	102,06
15:00	50569,96	50649,85	79,89
16:00	49917,37	49485,28	432,09
17:00	48648,69	48012,19	636,50
18:00	46533,60	45818,69	714,91
19:00	45838,97	44572,71	1266,26
20:00	47282,90	46504,95	777,95
21:00	46916,85	46254,14	662,71
22:00	45816,06	45147,14	668,92
23:00	44227,09	43333,74	893,35

İkinci karar ağaçları modeli olan MIMO Decision Tree modelinin giriş boyutu zamanda 24 adım geri gidildiği için 24'tür.

Her düğümde bölünmeyi seçmek için kullanılan ayırıcı parametresi en iyi olarak seçilmiştir. Bir dahili düğümü bölmek için gereken minimum örnek sayısı 2 olarak belirlenmiştir. Bir yaprak düğümünde olması gereken minimum örnek sayısı ise 1 olarak seçilmiştir. Modelin çıkış boyutu ise 24'tür.

Oluşturulan model 28 Temmuz 2021 tarihine ait 24 adet saatlik elektrik enerjisi tüketim verisi üzerinde test edilmiştir. Yapılan test sonucunda RMSE değeri 592,42 ve MAPE değeri %1,14 olarak tespit edilmiştir.

MIMO Decision Tree'ye ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini sonucu Şekil 9.10'da gösterilmiştir.



Şekil 9.10: MIMO Decision Tree 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini.

MIMO Decision Tree'ye ait 24 saatlik çok adımlı ileri elektrik enerjisi tüketim tahmini değerleri, gerçek değerler ve değerler arasındaki hataların mutlak değerleri Tablo 9.10'daki gibidir.

Tablo 9.10: MIMO Decision Tree 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini sonuçları.

MIMO Decision Tree			
28 Temmuz 2021			
Saat	**Gerçek Değerler (MWh)**	**Tahmin Değerleri (MWh)**	**Hata (MWh)**
00:00	42010,09	41237,93	772,16
01:00	40305,75	39383,38	922,37
02:00	38898,23	37854,97	1043,26
03:00	37896,78	36928,88	967,90
04:00	37295,62	36198,78	1096,84
05:00	36144,84	34937,16	1207,68
06:00	35301,97	34666,59	635,38
07:00	37577,15	37468,74	108,41
08:00	43334,98	43698,99	364,01
09:00	46709,36	47012,23	302,87
10:00	48493,17	48681,12	187,95

11:00	49870,84	50089,97	219,13
12:00	49129,19	48714,86	414,33
13:00	50163,34	50039,27	124,07
14:00	51130,40	51232,46	102,06
15:00	50569,96	50037,61	532,35
16:00	49917,37	50288,24	370,87
17:00	48648,69	48548,33	100,36
18:00	46533,60	46471,05	62,55
19:00	45838,97	45515,96	323,01
20:00	47282,90	46500,71	782,19
21:00	46916,85	46831,56	85,29
22:00	45816,06	45546,27	269,79
23:00	44227,09	43855,79	371,30

9.4.2 Rastgele Ormanlara Ait Tahmin Sonuçları

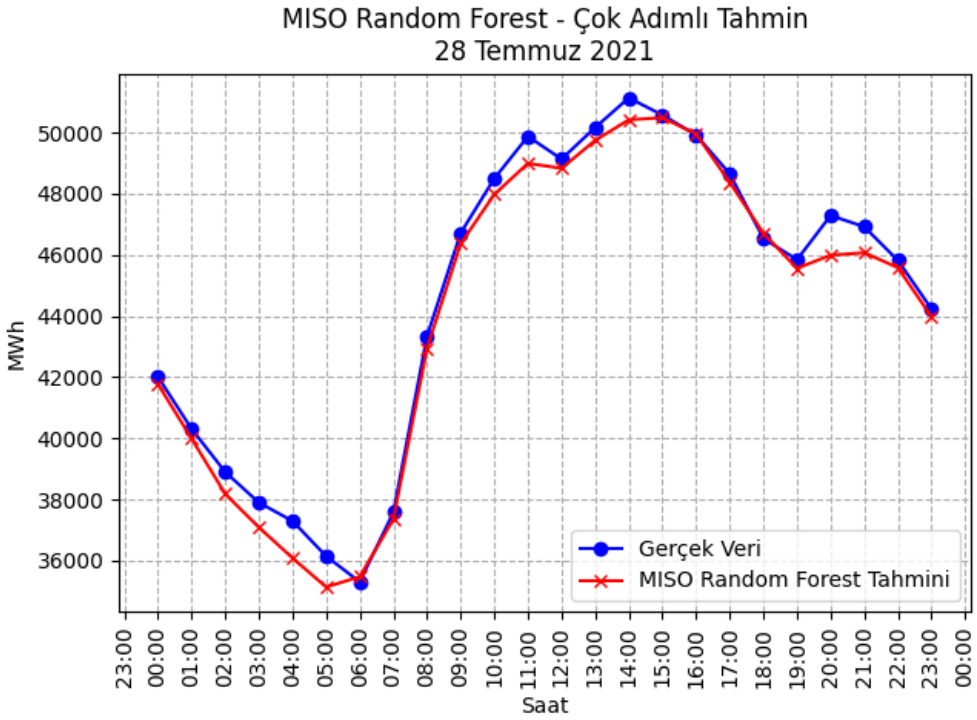
24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini yapmak üzere Çok Girişli Tek-Çıkışlı Rastgele Orman (MISO Random Forest) ve Çok Girişli-Çok Çıkışlı Rastgele Orman (MIMO Random Forest) yapıları kullanılmıştır.

MISO Random Forest modelinin giriş boyutu zamanda 24 adım geri gidildiği için 24'tür.

En iyi bölünmeyi ararken göz önünde bulundurulması gereken özelliklerin sayısı otomatik olarak seçilmiştir. Bir dahili düğümü bölmek için gereken minimum örnek sayısı 2 olarak belirlenmiştir. Bir yaprak düğümünde olması gereken minimum örnek sayısı ise 1 olarak seçilmiştir. Ormandaki ağaç sayısı 20 olarak belirlenmiştir. Ormandaki ağaç sayısı Grid Search kullanılarak, en düşük RMSE değerini verecek şekilde seçilmiştir. Modelin çıkış boyutu ise 1'dir.

Oluşturulan model 28 Temmuz 2021 tarihine ait 24 adet saatlik elektrik enerjisi tüketim verisi üzerinde test edilmiştir. Yapılan test sonucunda RMSE değeri 602,29 ve MAPE değeri %1,13 olarak tespit edilmiştir.

MISO Random Forest'a ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini sonucu Şekil 9.11'de gösterilmiştir.



Şekil 9.11: MISO Random Forest 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini.

MISO Random Forest'a ait 24 saatlik çok adımlı ileri elektrik enerjisi tüketim tahmini değerleri, gerçek değerler ve değerler arasındaki hataların mutlak değerleri Tablo 9.11'deki gibidir.

Tablo 9.11: MISO Random Forest 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini sonuçları.

MISO Random Forest			
28 Temmuz 2021			
Saat	**Gerçek Değerler (MWh)**	**Tahmin Değerleri (MWh)**	**Hata (MWh)**
00:00	42010,09	41771,50	238,59
01:00	40305,75	39989,24	316,51
02:00	38898,23	38180,51	717,72
03:00	37896,78	37088,46	808,32
04:00	37295,62	36085,08	1210,54
05:00	36144,84	35136,13	1008,71
06:00	35301,97	35469,38	167,41
07:00	37577,15	37357,43	219,72

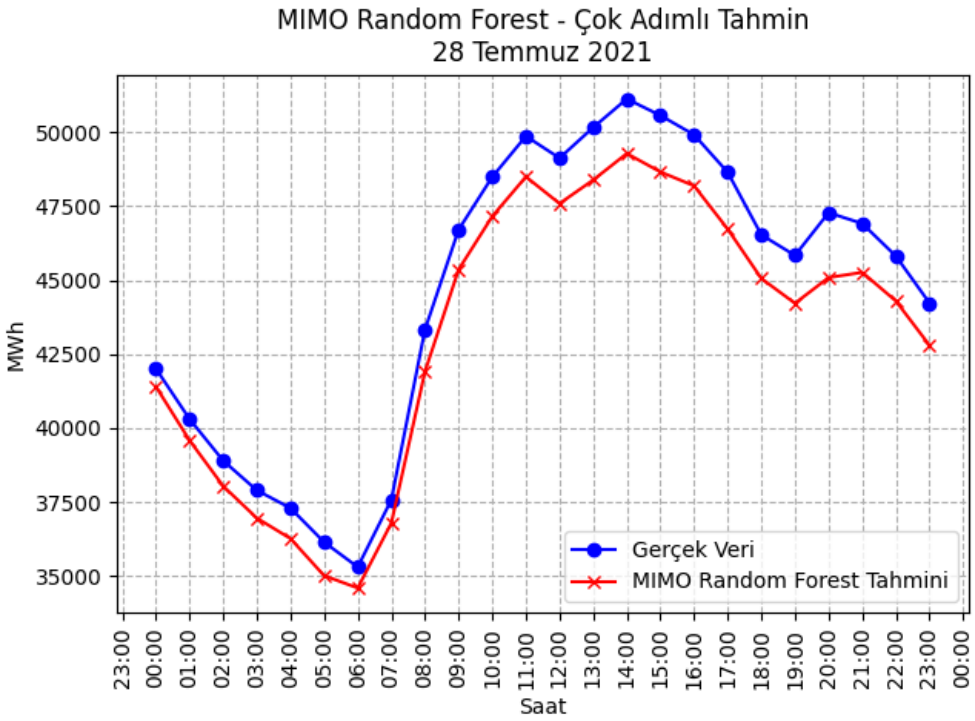
08:00	43334,98	42940,36	394,62
09:00	46709,36	46363,51	345,85
10:00	48493,17	47972,18	520,99
11:00	49870,84	48989,76	881,08
12:00	49129,19	48828,08	301,11
13:00	50163,34	49741,39	421,95
14:00	51130,40	50415,12	715,28
15:00	50569,96	50482,87	87,09
16:00	49917,37	49950,43	33,06
17:00	48648,69	48368,61	280,08
18:00	46533,60	46708,46	174,86
19:00	45838,97	45555,47	283,50
20:00	47282,90	45987,53	1295,37
21:00	46916,85	46068,23	848,62
22:00	45816,06	45577,93	238,13
23:00	44227,09	43971,89	255,20

İkinci rastgele ormanlar modeli olan MIMO Random Forest modelinin giriş boyutu zamanda 24 adım geri gidildiği için 24'tür.

En iyi bölünmeyi ararken göz önünde bulundurulması gereken özelliklerin sayısı otomatik olarak seçilmiştir. Bir dahili düğümü bölmek için gereken minimum örnek sayısı 2 olarak belirlenmiştir. Bir yaprak düğümünde olması gereken minimum örnek sayısı ise 1 olarak seçilmiştir. Ormandaki ağaç sayısı 160 olarak belirlenmiştir. Ormandaki ağaç sayısı Grid Search kullanılarak, en düşük RMSE değerini verecek şekilde seçilmiştir. Modelin çıkış boyutu ise 24'tür.

Oluşturulan model 28 Temmuz 2021 tarihine ait 24 adet saatlik elektrik enerjisi tüketim verisi üzerinde test edilmiştir. Yapılan test sonucunda RMSE değeri 1434,78 ve MAPE değeri %3,01 olarak tespit edilmiştir.

MISO Random Forest'a ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini sonucu Şekil 9.12'de gösterilmiştir.



Şekil 9.12: MIMO Random Forest 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini.

MIMO Random Forest'a ait 24 saatlik çok adımlı ileri elektrik enerjisi tüketim tahmini değerleri, gerçek değerler ve değerler arasındaki hataların mutlak değerleri Tablo 9.12'deki gibidir.

Tablo 9.12: MIMO Random Forest 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini sonuçları.

MIMO Random Forest			
28 Temmuz 2021			
Saat	**Gerçek Değerler (MWh)**	**Tahmin Değerleri (MWh)**	**Hata (MWh)**
00:00	42010,09	41402,44	607,65
01:00	40305,75	39584,24	721,51
02:00	38898,23	38038,57	859,66
03:00	37896,78	36957,57	939,21
04:00	37295,62	36265,18	1030,44
05:00	36144,84	35011,08	1133,76
06:00	35301,97	34593,97	708,00
07:00	37577,15	36791,26	785,89
08:00	43334,98	41912,53	1422,45
09:00	46709,36	45361,22	1348,14
10:00	48493,17	47158,43	1334,74

11:00	49870,84	48497,98	1372,86
12:00	49129,19	47592,71	1536,48
13:00	50163,34	48387,16	1776,18
14:00	51130,40	49282,97	1847,43
15:00	50569,96	48661,21	1908,75
16:00	49917,37	48191,48	1725,89
17:00	48648,69	46729,06	1919,63
18:00	46533,60	45058,72	1474,88
19:00	45838,97	44219,33	1619,64
20:00	47282,90	45089,44	2193,46
21:00	46916,85	45265,29	1651,56
22:00	45816,06	44296,98	1519,08
23:00	44227,09	42798,08	1429,01

9.5 Destek Vektör Makinelerine Ait Tahmin Sonuçları

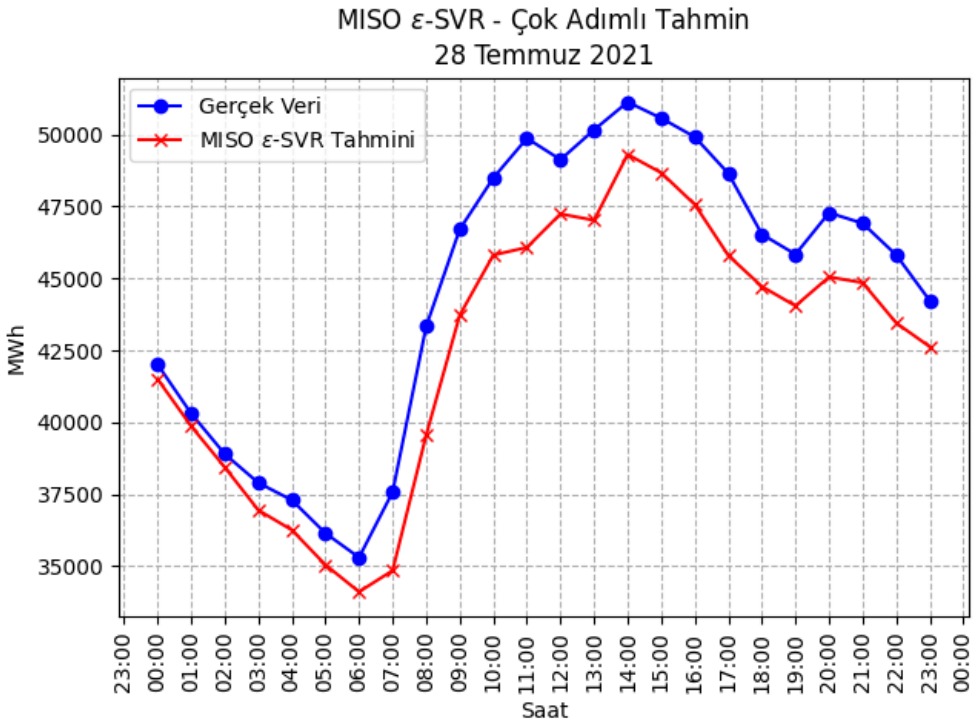
Bu tez çalışmasında, 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini yapmak üzere Çok Girişli-Tek Çıkışlı ε-Destek Vektör Regressor (MISO ε-SVR) yapısı kullanılmıştır.

MISO ε-SVR modelinin giriş boyutu zamanda 24 adım geri gidildiği için 24'tür.

Çekirdek olarak RBF çekirdek seçilmiştir. Modelde kullanılan C parametresi 1000 olarak belirlenmiştir. Modeldeki ε parametresi 0,01 olarak seçilmiştir. Modelin çıkış boyutu ise 1'dir.

Oluşturulan model 28 Temmuz 2021 tarihine ait 24 adet saatlik elektrik enerjisi tüketim verisi üzerinde test edilmiştir. Yapılan test sonucunda RMSE değeri 2192,26 ve MAPE değeri %4,34 olarak tespit edilmiştir.

MISO ε-SVR'ye ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketim tahmini sonucu Şekil 9.13'te gösterilmiştir.



Şekil 9.13: MISO ***ε***-SVR 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini.

MISO ε-SVR'ye ait 24 saatlik çok adımlı ileri elektrik enerjisi tüketim tahmini değerleri, gerçek değerler ve değerler arasındaki hataların mutlak değerleri Tablo 9.13'teki gibidir.

Tablo 9.13: MISO ***ε***-SVR 24 saatlik çok-adımlı elektrik enerjisi tüketim tahmini sonuçları.

MISO *ε*-SVR			
28 Temmuz 2021			
Saat	**Gerçek Değerler (MWh)**	**Tahmin Değerleri (MWh)**	**Hata (MWh)**
00:00	42010,09	41501,82	508,27
01:00	40305,75	39864,77	440,98
02:00	38898,23	38439,11	459,12
03:00	37896,78	36946,84	949,94
04:00	37295,62	36256,00	1039,62
05:00	36144,84	35045,77	1099,07
06:00	35301,97	34108,91	1193,06
07:00	37577,15	34847,98	2729,17
08:00	43334,98	39557,81	3777,17
09:00	46709,36	43726,13	2983,23
10:00	48493,17	45813,26	2679,91
11:00	49870,84	46077,78	3793,06
12:00	49129,19	47242,89	1886,30
13:00	50163,34	47023,05	3140,29

14:00	51130,40	49309,53	1820,87
15:00	50569,96	48673,49	1896,47
16:00	49917,37	47569,35	2348,02
17:00	48648,69	45795,75	2852,94
18:00	46533,60	44718,91	1814,69
19:00	45838,97	44047,03	1791,94
20:00	47282,90	45051,01	2231,89
21:00	46916,85	44851,83	2065,02
22:00	45816,06	43452,39	2363,67
23:00	44227,09	42626,64	1600,45

10. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında 8 farklı yapay sinir ağı modeli, 4 farklı karar ağacı modeli ve 1 destek vektör makineleri modeli kullanılmış ve oluşturulan bu yapay öğrenme modellerinin Türkiye'nin 28 Temmuz 2021 tarihine ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketimi tahmini üzerindeki tahmin başarıları değerlendirilmiştir.

Ayrıca kullanılan veri seti üzerinde Apriori ve ECLAT veri madenciliği yöntemleri ile birliktelik kural çıkarımı analizi yapılmış fakat veri içerisinde herhangi bir ilişki bulunamamıştır.

Kullanılan yapay öğrenme modellerinde kullanılan eğitim ve doğrulama verileri, karşılaştırmanın doğru yapılabilmesi için toplam veri içerisinden aynı rastgelelikte olacak şekilde seçilmiştir. Test verisi çıkarıldıktan sonra geri kalan 48864 adet verinin %60'ı eğitim verisi, %40'ı doğrulama verisi olarak kullanılmıştır.

Yapılan deneysel testler sonucunda, en iyi tahmin başarısını Hataların Karelerinin Ortalamasının Karekökü (RMSE) değeri 416,94 ve Ortalama Mutlak Yüzde Hata (MAPE) değeri %0,73 olarak tespit edilen MIMO RBF-ELM modeli göstermiştir. Tüm yapay öğrenme modellerinin tahmin başarısı RMSE ve MAPE değerleri ile Tablo 10.1'de gösterilmiştir.

Tablo 10.1: Oluşturulan yapay öğrenme modellerinin RMSE ve MAPE sonuçları.

Model	RMSE Değeri	MAPE Değeri
MISO FFNN	978,37	%1,90
MIMO FFNN	701,88	%1,26
MISO RNN	1233,33	%2,36
MIMO RNN	866	%1,60
MISO Tanh-ELM	683,89	%1,09
MIMO Tanh-ELM	507,26	%0,98
MISO RBF-ELM	1123,34	%2,01

MIMO RBF-ELM	416,94	%0,73
MISO Decision Tree	683,13	%1,30
MIMO Decision Tree	592,42	%1,14
MISO Random Forest	602,29	%1,13
MIMO Random Forest	1434,78	%3,01
MISO ε-SVR	2192,26	%4,34

Türkiye'nin 28 Temmuz 2021 tarihine ait 24 saatlik çok-adımlı ileri elektrik enerjisi tüketimi tahmini için MIMO RBF-ELM yöntemi en başarılı yöntem olmuştur. Fakat yeni bir güne ait tahminler yapılmak istendiğinde bu yöntemin en başarılı olacağını baştan söyleyebilmek güçtür. 28 Temmuz 2021 günü için en başarısız olan yöntem bile başka bir güne ait tahminler yapılmak istendiğinde en başarılı yöntem olabilmektedir. Dolayısıyla 24 saatlik çok-adımlı ileri elektrik enerjisi tüketimi tahmininde en başarılı yöntemin hangisi olacağı ile ilgili bir belirsizlik vardır. Bu belirsizliğin önüne geçmek ve model seçimini kullanıcıya bırakmayı ortadan kaldırmak için tüm yöntemlerin başarıları oranında nihai tahmin sonucuna katkı sağlayacağı bir ensemble learning (topluluk öğrenmesi) metodu uygulanabilir. Bu metotla birlikte en başarılı olan tahmin yöntemi nihai tahmin sonucuna en çok katkıyı yapacak ve en başarısız yöntem ise nihai tahmin sonucuna en az katkıyı yapacaktır.

İlerleyen dönemlerde yapılacak olan çalışmalarda bu tez kapsamında kullanılan yapay öğrenme metotları çoğaltılarak ve yukarıda bahsedilen topluluk öğrenmesi metodu ile tahmin sonuçları iyileştirilmeye ve model seçimi konusundaki belirsizlik ortadan kaldırılmaya çalışılacaktır.

11. KAYNAKLAR

Agrawal, R. and Srikant, R., "Fast Algorithms for Mining Association Rules in Large Databases", *Proceedings of the 20th International Conference on Very Large Data Bases*, 487-599, (1994).

Ahmad, A. S., Hassan, M. Y., Abdullah, M. P., Rahman, H. A., Hussin, F., Abdullah, H. and Saidur, R., "A review on applications of ANN and SVM for building electrical energy consumption forecasting", *Renewable and Sustainable Energy Reviews*, 33, 102-109, (2014).

Alpaydın, E., *Introduction to Machine Learning*, MIT Press LTD, (2014).

Antoniou, A. and Lu, W., *Practical Optimization - Algorithms and Engineering Applications*, Springer, (2007).

Arora, R. K., *Optimization: Algorithms and Applications*, Chapman and Hall/CRC, (2015).

Aydın, D. and Toros, H., "Prediction of Short-Term Electricity Consumption by Artificial Neural Networks Using Temperature Variables", *European Journal of Science and Technology*, 14, 393-398, (2018).

Azadeh, A., Ghader, S. F. and Sohrabkhani, S., "A simulated-based neural network algorithm for forecasting electrical energy consumption in Iran", *Energy Policy*, 36 (7), 2637-2644, (2008).

Bagheri, R., "An Introduction to Deep Feedforward Neural Networks [online]", (2021), https://towardsdatascience.com/an-introduction-to-deep-feedforward-neural-networks-1af281e306cd, (2020).

Bulut, M. and Başoğlu, B., "Kısa Dönem Elektrik Talep Tahminleri İçin Yapay Sinir Ağları ve Uzman Sistemler Tabanlı Hibrid Tahmin Sistemi Geliştirilmesi", *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, 32 (2), 575-583, (2017).

Bianco, V., Manca, O. and Nardini, S., "Electricity consumption forecasting in Italy using linear regression models", *Energy*, 34 (9), 1413-1421, (2009).

Breiman, L., "Random Forests", *Machine Learning*, 45, 5-32, (2001).

Breiman, L., Friedman, J., Stone, C. J. and Olshen, R. A., *Classification and Regression Trees*, Chapman & Hall/CRC, (1984).

Chou, J. S. and Truong, D. N., "Multistep energy consumption forecasting by metaheuristic optimization of time-series analysis and machine learning", *International Journal of Energy Research*, 45 (3), 4581-4612, (2020).

Divina, F., Gilson, A., Gomez-Vela, F., Torres, M. G. and Torres, J. F., "Stacking Ensemble Learning for Short-Term Electricity Consumption Forecasting", *Energies*, 11 (4), 949, (2018).

Enerji Piyasaları İşletme A.Ş. (EPİAŞ), "Gerçek Zamanlı Tüketim [online]", (2021), https://seffaflik.epias.com.tr/transparency/tuketim/gerceklesen-tuketim/gercek-zamanli-tuketim.xhtml, (2021).

Géron, A., *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, O'Reilly Media, (2019).

Hamzaçebi, C., "Forecasting of Turkey's net electricity energy consumption on sectoral bases", *Energy Policy*, 35 (3), 2009-2016, (2007).

Hamzaçebi, C. and Kutay, F., "Yapay Sinir Ağları ile Türkiye Elektrik Enerjisi Tüketiminin 2010 Yılına Kadar Tahmini", *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, 19 (3), 227-233, (2004).

Huang, G. B., Zhu, Q. Y. and Siew, C. K., "Extreme learning machine: Theory and applications", *Neurocomputing*, 70 (1-3), 489-501, (2006).

İplikçi, S., *Optimizasyon Teknikleri Ders Notları*, Pamukkale Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü, Denizli, (2018).

İplikçi, S., *Makine Öğrenmesine Giriş Ders Notları*, Pamukkale Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü, Denizli, (2019).

İplikçi, S., *Veri Madenciliği Ders Notları*, Pamukkale Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü, Denizli, (2020).

Johannesen, N. J., Kolhe, M. and Goodwin, M., "Relative evaluation of regression tools for urban area electrical energy demand forecasting", *Journal of Cleaner Production*, 218, 555-564, (2019).

Kaboli, S. H. A., Fallahpour, A., Selvaraj, J. and Rahim, N. A., "Long-term electrical energy consumption formulating and forecasting via optimized gene expression programming", *Energy*, 126, 144-164, (2017).

Kavaklıoğlu, K., "Modeling and prediction of Turkey's electricity consumption using Support Vector Regression", *Applied Energy*, 88 (1), 368-375, (2011).

Ouyang, T., Wang, C., Yu, Z., Stach, R., Mizaikoff, B., Liedberg, B., Huang, G. B. and Wang, Q. J., "Quantitative Analysis of Gas Phase IR Spectra Based on Extreme Learning Machine Regression Model", *Sensors*, 19, 5535, (2019).

Özden, S. and Öztürk, A., "Yapay Sinir Ağları ve Zaman Serileri Yöntemi ile Bir Endüstri Alanının (İvedik OSB) Elektrik Enerjisi İhtiyaç Tahmini", *Bilişim Teknolojileri Dergisi*, 11 (3), 255-261, (2018).

Özkan, E., Güler, E. and Aladağ, Z., "Elektrik Enerjisi Tüketim Verileri için Uygun Tahmin Yöntemi Seçimi", *Endüstri Mühendisliği*, 31 (2), 198-214, (2020).

Quinlan, J. R., "Induction of decision trees", *Machine Learning*, 1, 81-106, (1986).

Quinlan J. R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann, (1993).

Ruiz, L. G. B., Rueda, R., Cuéllar, M. P. and Pegalajar, M. C., "Energy consumption forecasting based on Elman neural networks with evolutive optimization", *Expert Systems with Applications*, 92, 380-389, (2018).

Scikit-learn, "Decision Trees [online]", (2021), https://scikit-learn.org/stable/modules/tree.html, (2020).

Smola, A. J. and Schölkopf, B., "A tutorial on support vector regression", *Statistics and Computing*, 14, 199-222, (2004).

Suganthia, L. and Samuel, A. A., "Energy models for demand forecasting—A review", *Renewable and Sustainable Energy Reviews*, 16 (2), 1223-1240, (2012).

Uzlu, E. and Dede, T., "Jaya algoritması ile optimize edilmiş yapay sinir ağlarını kullanarak Türkiye'de elektrik enerjisi tüketiminin tahmini", *Gazi Üniversitesi Fen Bilimleri Dergisi Part C: Tasarım ve Teknoloji*, 8 (3), 511-528, (2020).

Vapnik, V., *The Nature of Statistical Learning Theory*, Springer, (1995).

Wang, L., Lv, S. X. and Zeng, Y. R., "Effective sparse adaboost method with ESN and FOA for industrial electricity consumption forecasting in China", *Energy*, 155, 1013-1031, (2018).

Wolfe, P., "A duality theorem for non-linear programming", *Quarterly of Applied Mathematics*, 19 (3), 239-244, (1961).

Zaki, M. J., "Scalable algorithms for association mining", *IEEE Transactions on Knowledge and Data Engineering*, 12 (3), 372-390, (2000).

Zheng, J., Xu, C., Zhang, Z. and Li, X., "Electric load forecasting in smart grids using Long-Short-Term-Memory based Recurrent Neural Network", *51st Annual Conference on Information Sciences and Systems (CISS)*, USA, (2017).

Zor, K., Timur, O. and Teke, A., "A state-of-the-art review of artificial intelligence techniques for short-term electric load forecasting", *6th International Youth Conference on Energy (IYCE)*, Hungary, (2017).

Zor, K., Çelik, Ö., Timur, O. and Teke, A., "Short-Term Building Electrical Energy Consumption Forecasting by Employing Gene Expression Programming and GMDH Networks", *Energies*, 13 (5), 1102, (2020).