T.C.
AYDIN ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MATEMATİK ANABİLİM DALI

BAZI MATEMATİKSEL EPİDEMİYOLOJİ MODELLERİNİN YAPAY SİNİR AĞI ÇÖZÜMLERİ ÜZERİNE

Muhammad Jalil AHMAD

YÜKSEK LİSANS TEZİ

DANIŞMAN

Dr. Öğr. Üyesi Korhan GÜNEL

AYDIN-2021

KABUL VE ONAY

ÖZET

BAZI MATEMATİK EPİDEMİYOLOJİ MODELLERİNİN YAPAY SİNİR AĞI ÇÖZÜMLERİ ÜZERİNE

Muhammad Jalil AHMAD

Yüksek Lisans Tezi, Matematik Anabilim dalı

Tez Danışmanı: Dr. Öğr. Üyesi Korhan GÜNEL

2021, 68 Sayfa

Amaç: Bu çalışma Türkiye genelinde COVID-19 pandemisi ile ilintili olarak popülasyon içinde pozitif vaka oranı, hasta ve iyileşen birey oranlarını tahminleyecek bir yapay sinir ağı modeli geliştirmek amacı ile yapılmıştır.

Materyal ve Yöntem: Çalışma 2020 yılı Kasım ayı ile 2021 Mayıs ayları aralığında Türkiye genelinde T.C. Sağlık Bakanlığı web sitesinde ilan edilen onaylanmış COVID-19 verileri kullanılarak gerçekleştirilmiştir. Normalize edilen veriler, literatürde sunulan bazı matematiksel epidemik modellere yakınsayacak biçimde nümerik çözüm üreten çok katmanlı algılayıcı ve artık yapay sinir ağlarının eğitiminde kullanılmıştır. Ağın ağırlıklarının belirlenmesi için türev içermeyen bir optimizasyon yaklaşımı olan Örgü Uyarlamalı Doğrudan Arama Yöntemi kullanılmıştır.

Bulgular: Araştırmada geliştirilen yapay sinir ağlarının hem SI (Suspectible – Infected) ve SIS (Suspectible – Infected – Suspectible) epidemik modellerine hem de gözlemlenen gerçek verilere uyum sağladığı görülmüştür. Bununla birlikte SIR ((Suspectible – Infected – Recovered) modeli için aynı bulguyu elde etmek mümkün olmamıştır. Bunun temel nedeninin günlük vaka sayılarında zaman zaman görülen ani ve büyük miktardaki değişim oranları olduğu kanısına varılmıştır. Yapay sinir ağlarının bu değişimlere hızlı bir biçimde adapte olamadığı görülmüştür. Ağın eğitimi için, arama uzayının tamamını tarama yeteneğine sahip optimizasyon algoritmalarının tercih edilmesi gerektiği düşünülmektedir.

Sonuç: Bu çalışmada yapay sinir ağları ile pandemik hastalıkların seyrini tahminleyebilecek modellemeler yapılabileceği ve böylece hastalığın yayılmasını önleyici tedbirlerin zamanında alınabileceği sonucuna ulaşılmıştır.

Anahtar Kelimeler: COVID-19, Çok Katmanlı Algılayıcılar, Artık Yapay Sinir Ağları, Optimizasyon, Örgü Uyarlamalı Doğrudan Arama Algoritması

ABSTRACT

ON THE NEURAL NETWORK SOLUTIONS OF SOME MATHEMATICAL EPIDEMIOLOGICAL MODELS

Muhammad Jalil AHMAD

M.Sc. Thesis, Department of Mathematics

Supervisor: Asst. Prof. Korhan GÜNEL

2021, 68 Pages

Objective: This study was carried out with the aim of developing an artificial neural network model that will predict the rate of positive cases, infected and recovered individuals in the population with respect to the COVID-19 pandemic in Turkey.

Material and Methods: The study is carried out Turkey between November 2020 and May 2021 using confirmed COVID-19 data announced on the Ministry of Health website. The normalized data is used in the training of multilayer perceptron and residual artificial neural networks that produce numerical solutions in a way that converges to some mathematical epidemic models presented in the literature. The Mesh Adaptive Direct Search Method, which is a derivative-free optimization approach, is used to determine the weights of the neural network.

Results: It has been observed that the artificial neural networks developed in the research are compatible with both the SI (Suspectible - Infected) and SIS (Suspectible - Infected - Suspectible) epidemic models and the real data observed. However, it was not possible to obtain the same finding for the SIR (Suspectible - Infected - Recovered) model. It was concluded that the main reason for this was the sudden and large change rates seen from time to time in the number of daily cases. It was observed that artificial neural networks could not adapt to these changes quickly. For the training of the network, it is thought that optimization algorithms with the ability to scan the entire search space should be preferred.

Conclusions: In this study, it has been concluded that models that can predict the course of pandemic diseases can be made with artificial neural networks and thus, measures to prevent the spread of the disease can be taken in a timely manner.

Keywords: COVID-19, Multilayer Perceptron, Residual Neural Network, Optimization, Mesh Adaptive Direct Search Algorithm

ÖNSÖZ

Tanıştığımızdan bugüne kadar hiçbir zaman desteğini esirgemeyen, akademik ve bilimsel hayatımda kendimi geliştirmeme yardımcı olan saygıdeğer danışman hocam Sayın Dr. Öğr. Üyesi Korhan GÜNEL'e şükranlarımı sunarım. Tüm yaşamım boyunca bana destek veren ve bugünlere gelmemde büyük payı olan aileme sonsuz teşekkür ederim.

Muhammad Jalil AHMAD

İÇİNDEKİLER

KISALTMALAR DİZİNİ

GPS : Genelleştirilmiş Desen Arama (Generalized Patern Search)

MADS : Örgü Uyarlamalı Doğrudan Arama (Mesh Adaptive Direct Search)

SI : Yatkın - Enfekte (Susceptible - Infected)

SIS : Yatkın - Enfekte - Yatkın (Susceptible - Infected - Susceptible)

SIR : Yatkın - Enfekte - İyileşen (Susceptible - Infected - Recovered)

SEIR : Yatkın - Maruz Kalan - Enfekte - İyileşen (Susceptible-Exposed-Infected-Recovered)

ResNET : Artık Yapay Sinir Ağı (Residual Neural Network)

LSTM : Uzun Kısa Vadeli Bellek (Long-Short Term Memory)

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

EKLER DİZİNİ

1. GİRİŞ

İnsanoğlu yüzyıllardır salgın hastalıklarla mücadele sürdürmüş ve bu mücadelede milyonlarca insan hayatını kaybetmiş ve bununla birlikte büyük ekonomik zararlara girmiştir. Bilim insanları ve sağlık personelleri bu salgın hastalıklarla savaşmak için büyük çaba sarf etmektedirler. Tabii ki bu çabalara matematikçiler de büyük katkılarda bulunmuştur. Salgınların tanımlanması ve davranışının modellenmesi yönündeki çalışmalar matematiksel yöntemlerle başarılı bir şekilde gerçekleştirilmektedir.

Hastalıkların yayılmasının matematiksel modellenmesi, biyoloji anabiliminin bir dalıdır ve epidemiyoloji olarak adlandırılır. Hastalıkların yayılması uzun yıllardır sorgulanmakta ve incelenmektedir. Bulaşıcı hastalıklarını tanımlayıp inceleyen ilk matematiksel model Daniel Bernoulli tarafından verilmiştir. 1760 yılında yaygın olan çiçek virüsünün yayılımını modellemişti (Bernoulli ve Daniel, 1760).

Dünya şu anda tehlikeli, ölümcül ve insandan insana yayılan bir koronavirüs (COVID-19) hastalığıyla karşı karşıyadır. Bu virüs solunum yoluyla kontrolsüz bir şekilde yayılmaktadır. İlk COVID-19 vakası 31 Aralık 2019 tarihinde Çin'in Wuhan kentinde görülmüştür. Halihazırda bu virüs için bir takım aşı çalışmalar başarılı bir şekilde tamamlanmış olup çeşitli ülkelerde aşılama aşamasına geçilmiştir. Bu aşılama çalışmalarına rağmen, salgın henüz kontrol altına alınamamıştır. Ayrıca farklı ülkelerde görülen çeşitli mutasyonlu COVID-19 varyantları, bu aşılama programlarını tehlikeye atmış durumdadır.

Araştırmacılar, bu pandemiyi kendi alanlarında kontrol altına almak ve olası çözümleri elde etmek için ısrarlı çabalar sarf etmektedir. Chimmula ve Zhang (2020), Uzun Kısa Süreli Bellek (LSTM) ağları kullanarak Kanada'daki COVID-19 vakalarının zaman serisi analizini yapan bir çalışma gerçekleşmiştir. Zeroual ve arkadaşları (2020), Tekrarlayan Sinir Ağları (RNN), Uzun Kısa Süreli Bellek (LSTM), Çift Yönlü Tekrarlayan Sinir Ağları (BiLSTM), Geçitli Tekrarlayan Birimler (GRUs) ve Varyasyonel Oto Kodlayıcı (VAE) ağlarını kullanarak İtalya, İspanya, Fransa, Çin, Amerika Birleşik Devletleri ve Avustralya'daki günlük COVID-19 vakalarını ve iyileşen kişi sayılarını tahminlemeye çalışan model üretmiştir.

Araştırmacılar tarafından uygulanan en yaygın ve etkili yöntemlerden biri, akciğer görüntülerini COVID-19 için analiz etmek amacıyla CT Taramaları ve X-ışınlarının kullanmaktır. Bununla birlikte, bir pandemideki en zorlayıcı unsurlar her hasta için oluşturulacak raporu bireysel olarak inceleyecek birkaç radyoloji uzmanı ihtiyacının bulunması ve bu işlem için kısıtlı zaman sorunuyla karşılaşılmasıdır. Alternatif bir hızlı tarama yöntemi olarak; hastaların göğüs radyografileri üzerinde işlem yaparak ayırt edici göstergeleri belirleyecek ve X ışınlarını analiz ederek COVID-19 tanısı gerçekleştirebilecek bir derin öğrenme sinir ağı tabanlı yöntemi olan nCOVnet geliştirilmiştir (Panwar vd., 2020).

Dünya Sağlık Örgütü (WHO) tarafından, COVID-19'a karşı korunma yöntemi olarak maske kullanımı gerekliliği vurguladıktan sonra; maskeler, günlük hayatta daha yaygın olarak kullanmaya başlanmıştır. Maske takmanın COVID-19'a karşı tamamen koruma sağladığını söylemek, elimizdeki verilerle mümkün değildir. Ancak maske takmak COVID-19 salgının yayılmasını engelleme konusunda az da olsa yardımcı olmuştur. Gondim (2021), COVID-19 salgını için farmasötik olmayan bir kontrol stratejisi olarak yüz maskelerinin yaygın kullanımını $SEIR$ (Suspected-Exposed-Infected-Recovered) modeli ile incelemiştir. Söz konusu çalışmada, nüfusu maske takan ve takmayanlara göre ikiye ayırarak $SEIR$ modeli ile elde edilen nümerik sonuçları Amerika Birleşik Devletleri, Brezilya ve İtalya'daki gerçek verilerle karşılaştırmıştır (Gondim, 2021).

Sabit bir popülasyon üzerinde Kermack ve Mckendrick tarafından verilen, doğrusal olmayan diferansiyel denklemlerle tanımlanmış SIR (Suspected-Infected-Recovered) ve SIS (Suspected-Infected-Suspected) epidemik modelleri için analitik çözüm başarıyla elde edilmiştir (Shabbir vd., 2014).

SIR salgın modelinin analitik çözümü parametrik bir formda elde edilmiştir. Analitik çözümü kullanarak, parametrelerin sabit değerlerine karşılık gelen bazı açık modeller araştırılmış ve sayısal çözümün, analitik çözümü tam olarak yeniden ürettiği gösterilmiştir (Harko vd., 2014).

Cooper vd. (2020), COVID-19 hastalığının yayılma sürecini analiz etmek amacıyla teorik bir çerçeve sağlayan SIR pandemi modelleme yaklaşımının etkililiğini inceleyen bir çalışma gerçekleştirmişlerdir. Çalışmada yazarlar, önemli tahminler sunarak hastalığın etkisinin

değerlendirilmesine yardımcı olunabileceğini göstermiş ve gerekli önlemler alınarak hastalığın kontrol altına alınabileceğini iddia etmişlerdir.

Derin öğrenme yaklaşımları, Covid-19'un tespitinde oldukça ilgi görmüştür ve transfer öğrenme yaklaşımı en yaygın kullanılan yaklaşım olmuştur. Bununla birlikte, önceden eğitilmiş modellerin çoğu, X-ışınları ve renkli göğüs tomografisi görüntüleri kullanılarak eğitilmiştir. Eğitim setinde kullanılan verilerin, genellikle gri tonlamalı olan Covid-19 görüntülerinde modellerin ince ayarı yapılırken verimsizliğe neden olduğu belirtilmektedir (Aminu vd., 2021). Bu sorunu çözmek için, nispeten daha az sayıda parametre gerektiren CovidNet adlı bir derin öğrenme mimarisi önerilmiştir. CovidNet, gri tonlamalı görüntüleri girdi olarak kabul eder ve sınırlı eğitim veri seti ile eğitim için uygundur. Deneysel sonuçlar, CovidNet'in Covid-19 tespiti için diğer son teknoloji derin öğrenme modellerinden daha iyi performans gösterdiğini göstermiştir (Aminu vd., 2021).

Bu tezin amacı; SI (Susceptible-Infected), SIS (Susceptible-Infected- Susceptible) ve SIR (Susceptible-Infected- Recovered) matematiksel epidemiyoloji modellerini COVID-19 dinamiklerine uyarlayıp bu modellerin nümerik çözümlerini çeşitli yapay sinir ağları ile elde etmektir. Literatürdeki çalışmalardan farklı olarak, zaman serisi analizi kullanmadan modellenen yapay sinir ağları, türev içermeyen bir optimizasyon yöntemi olan Örgü Uyarlamalı Doğrudan Arama (MADS) yaklaşımı kullanılarak eğitilmişlerdir. Ağın girdileri sadece günlük pozitif vaka sayısı, infekte olan ve iyileşen hasta sayıları olarak belirlenmiştir.

Bir sonraki bölümde tezde kullanılan epidemiyoloji modellerinin temelleri ve teorik alt yapısı özetlenmiştir.

2. EPİDEMİYOLOJİ MODELLERİ

Bu bölümde çalışmada kullanılan çeşitli matematiksel epidemiyoloji modellerinden söz edilecektir.

2.1 *SI* (Susceptible - Infectious) Modeli

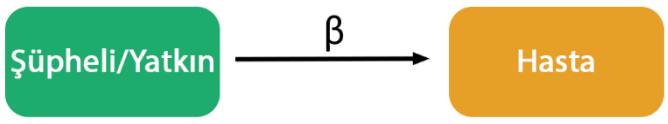
S, *t* anındaki hastalığı henüz bulaşmamış ama bulaşma ihtimali olan birey sayısını (daha sonra şüpheli bireyler olarak adlandırılacak), *I* ise *t* anındaki hastalığı bulaşmış bireyler olsun. *SI* modeli adi diferansiyel denklemlerinin sistemi olarak aşağıdaki gibi verilebilir.

$$\frac{dS(t)}{dt} = -\frac{\beta S(t) I(t)}{N} \tag{2.1}$$

$$\frac{dI(t)}{dt} = \frac{\beta S(t) I(t)}{N} \tag{2.2}$$

Burada $N = S(t) + I(t)$ popülasyondaki toplam birey sayısı, β ise hastalığın yayılma katsayısıdır. Ayrıca $S(0) = s_0 > 0$ ve $I(0) = i_0 > 0$ başlangıç değer koşullarını belirtmektedir.

Bu modelde doğum ve ölüm oranı dikkate alınmadığı (veya eşit alındığı yani toplam popülasyon sabit olduğu) için bütün şüpheli (Susceptible) bireyler zamanla enfekte olacaklardır.



Şekil 2.1 *SI* modelinin yapısı

(2.1) ve (2.2) de verilen denklem sisteminin analitik çözümü aşağıdaki gibidir.

$$I(t) = \frac{i_0}{i_0 + (1 - i_0)e^{-\beta t}} \tag{2.3}$$

$$S(t) = N - I(t) \tag{2.4}$$

2.2 *SIS* (Susceptible – Infectious - Susceptible) Modeli

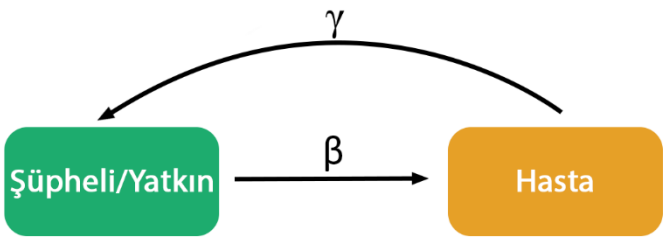
SIS modelindeki toplam popülasyon *N* olmak üzere $N = S(t) + I(t)$ dır. Bu modelde bazı enfekte olmuş bireylerin iyileşip tekrar yatkın (Susceptible) durumuna geçme ihtimali de dikkate alınacaktır. *SI* modelinde olduğu gibi *SIS* modelinde de doğum ve ölüm oranı dikkate alınmayacak (veya eşit alınacak yani toplam popülasyon sabit olacaktır).

Kermach ve Mckendrick tarafından verilen *SIS* modelinin diferansiyel denklem sistemi aşağıdaki gibidir.

$$\frac{dS(t)}{dt} = -\beta S(t)I(t) + \gamma I(t) \tag{2.5}$$

$$\frac{dI(t)}{dt} = \beta S(t)I(t) - \gamma I(t) \tag{2.6}$$

β hastalığın yayılma katsayısı, γ ise iyileşme katsayısıdır. $S(0) = s_0$ ve $I(0) = i_0$ başlangıç değer koşullarını belirtmekte olup $\beta > 0$, $i_0 > 0$ ve $s_0 > 0$ dır. Bu model *SIR* modelinden farklıdır çünkü iyileşen bireyler tekrar hastalığa bulaşabilir ve dolayısıyla $i\gamma$ oranında şüpheli (Susceptible) durumuna geçmiş olurlar.



Şekil 2.2 *SIS* modelinin yapısı

$s(t) = N - i(t)$ ifadesi (2.6) denkleminde yerine yerleştirilirse

$$i'(t) = \beta(N - i)i - \gamma i \tag{2.7}$$

Yukarıdaki denkleminin, $y = i^{-1}$ dönüşümü yapılarak, integrali alınırsa

$$ye^{(\beta N-\gamma)t} = \int e^{(\beta N-\gamma)t}dt + C \tag{2.8}$$

elde edilir. Burada *C* integrasyon sabitidir. (2.8) denklemi sabitleştirilip

$\alpha = \beta N - \gamma$ alınırsa

$$y = \frac{\beta + \alpha C e^{-\alpha t}}{\alpha} \tag{2.9}$$

elde edilir. $y = i^{-1}$ yerine geri yazılırsa

$$i(t) = \frac{\alpha}{\beta + \alpha C e^{-\alpha t}} \tag{2.10}$$

Yukarıdaki denklemde $I(0) = i_0$ başlangıç koşulu uygulanırsa

$$\mathrm{C} = \frac{\alpha - \mathrm{i}_0\,\beta}{\alpha\,\mathrm{i}_0} \tag{2.11}$$

$N = S(t) + I(t)$ olduğundan

$$s(t) = N - \frac{\alpha}{\beta + \alpha C e^{-\alpha t}} \tag{2.12}$$

2.3 *SIR* (Susceptible – Infectious - Recovered) Modeli

Kermach ve Mckendrick tarafından verilen *SIR* modelinin lineer olmayan diferansiyel denklem sistemi aşağıdaki gibidir.

$$\frac{ds(t)}{dt} = -\beta s(t)i(t) \tag{2.13}$$

$$\frac{di(t)}{dt} = \beta s(t)i(t) - \gamma i(t) \tag{2.14}$$

$$\frac{dr(t)}{dt} = \gamma i(t) \tag{2.15}$$

β hastalığın yayılma katsayısı, γ ise iyileşme katsayısıdır. $s(0) = s_0, i(0) = i_0$ ve $r(0) = r_0$ başlangıç değer koşullarını belirtmekte olup $\beta > 0, \gamma > 0, i_0 \geq 0, s_0 \geq 0$ ve $r_0 \geq 0$ dir. Ayrıca $N = s(t) + i(t) + r(t)$ dir.

Şekil 2.3 *SIR* modelinin yapısı

(2.14) denkleminin *t* ye göre türevi alınırsa

$$\frac{di}{dt} = -\frac{1}{\beta}\left[\frac{s''}{s} - \left(\frac{s'}{s}\right)^2\right] \tag{2.16}$$

burada s' ve s'' ifadeleri *s* nin *t* ye göre birinci ve ikinci türevlerini ifade etmektedir. (2.13) ve (2.16) ifadelerini (2.14) denkleminde yerine konursa

$$\frac{s''}{s} - \left(\frac{s'}{s}\right)^2 + \gamma\frac{s'}{s} - \beta s' = 0 \tag{2.17}$$

elde edilir. (2.13) ve (2.15) ifadelerinden *i* yok edilirse

$$\frac{dr}{dt} = -\frac{\gamma}{\beta}\left(\frac{s'}{s}\right) \tag{2.18}$$

elde edilir ve bunun integrali alındığında

$$s = ae^{-\frac{\beta}{\gamma}r} \tag{2.19}$$

olur. Burada *a* pozitif integrasyon katsayısıdır. (2.19) ifadesi $t = 0$ için hesaplandığında

$$a = s_0 e^{\frac{\beta}{\gamma} r_0} \tag{2.20}$$

elde edilir. (2.19) denkleminden

$$s' = -\frac{a\beta}{\gamma} r' e^{-\frac{\beta}{\gamma} r} \tag{2.21}$$

şimdi (2.18) denkleminin *t* ye göre türevini alalım

$$r'' = -\frac{\beta}{\gamma}\left[\frac{s''}{s} - \left(\frac{s'}{s}\right)^2\right] \tag{2.22}$$

(2.18), (2.21), (2.22) ifadelerini (2.17) denkleminde yerine koyarsak

$$r'' = a\beta r' e^{-\frac{\beta}{\gamma} r} - \gamma r' \tag{2.23}$$

(2.23) ifadesi (2.13), (2.14), (2.15) denklem sistemine denktir.

(2.23) lineer olmayan diferansiyel denkleminin çözümü için *u(t)* fonksiyonunu tanımlayalım öyle ki

$$u = e^{-\frac{\beta}{\gamma} r} \tag{2.24}$$

u nun başlangıç değeri için $t = 0$ alalım. Böylece

$$u(0) = u_0 = e^{-\frac{\beta}{\gamma} r_0} \tag{2.25}$$

(2.24) ifadesini (2.23) denkleminde yerine konursa

$$u\frac{d^2u}{dt^2} - \left(\frac{du}{dt}\right) + (\gamma - a\beta u)u\frac{du}{dt} = 0 \tag{2.26}$$

u ya göre ikinci dereceden diferansiyel denklemi elde edilir. Şimdi *φ* fonksiyonunu tanımlayalım

$$\phi = \frac{dt}{du} \tag{2.27}$$

(2.27) ifadesini (2.26) denkleminde yerine koyalım

$$\frac{d\phi}{du}+\frac{1}{u}\phi=(\gamma-a\beta u)\phi^2 \tag{2.28}$$

Bernoulli diferansiyel denklemi elde edilir ve bu denkleminin genel çözümü

$$\phi=\frac{1}{u(C_1-\gamma lnu+a\beta u)} \tag{2.29}$$

dır. Burada C_1 keyfi integrasyonu sabitidir. (2.27) ve (2.29) denklemlerine göre *t* nin integral temsili söyle verilebilir:

$$\mathrm{t}-\mathrm{t}_0=\int_{\mathrm{u}_0}^{\mathrm{u}}\frac{d\varepsilon}{\varepsilon(C_1-\gamma ln\varepsilon+a\beta u)} \tag{2.30}$$

burada t_0 keyfi integrasyon katsayısıdır ve genelliği kaybetmeden $t_0=0$ alınabilir. Böylece (2.13), (2.14), (2.15) denklem sisteminin parametrik çözümü

$$s=s_0u \tag{2.31}$$

$$i=\frac{\gamma}{\beta}lnu-au-\frac{C_1}{\beta} \tag{2.32}$$

$$r=-\frac{\gamma}{\beta}lnu \tag{2.33}$$

olur. Burada *u* parametre olarak alınır. Ayrıca (2.31), (2.32), (2.33) ifadeleri toplanırsa

$$s+i+r=-\frac{C_1}{\beta} \tag{2.34}$$

ve böylece

$$C_1=-\beta N \tag{2.35}$$

olur ve bundan C_1 bir negatif integrasyon katsayısıdır.

Bu modeller bilinen en temel modeller olmakla beraber hastalık sonucu oluşacak can kaybı sayısını, aşılama sonrası iyileşen kişi sayısını ve hastalığın yayılma katsayısını tahminlenmesini sağlayacak modeller de mevcuttur (Gao vd. 2007).

Bu tezde, bahsi geçen üç matematiksel epidemiyoloji modeli kullanılarak farklı yapay sinir ağları çözümleri elde edilmiş ve hastalığın seyri tahminlenmeye çalışılmıştır. Ayrıca bu yapay sinir ağları eğitirken bir sonraki bölümde detaylı olarak anlatılan Örgü Uyarlamalı Doğrudan Arama (MADS) algoritması kullanılmıştır.

3 ÖRGÜ UYARLAMALI DOĞRUDAN ARAMA ALGORİTMASI (MADS)

Örgü uyarlamalı doğrudan arama algoritması (Mesh Adaptive Direct Search, MADS), türev içermeyen kısıtlamalı optimizasyon problemlerinin çözümleri için geliştirilmiştir (Audet and Hare, 2014). İlk olarak kısıtlı doğrusal optimizasyon problemleri için önerilen MADS yöntemi, ileriki yıllarda lineer olmayan optimizasyon problemleri çözecek şekilde uyarlanmış ve NOMAD (Nonlinear Optimization with MADS) olarak adlandırılmıştır (Audet and Hare, 2014).

Genel olarak MADS yöntemi, arama uzayı üzerinde Tanım 3.4 ile tanımlanan bir örgü oluşturma işlemi ile başlar. Oluşturulan örgüyü örtecek şekilde Tanım 3.3 ile tanımlanan bir pozitif taban kümesi inşa edilir.

Tanım 3.1. $D \subset \mathbb{R}^n$ olsun. D kümesindeki bütün vektörlerin negatif olmayan lineer kombinasyonlarının kümesi D kümesinin pozitif kapsayan kümesi olarak adlandırılır ve $pspan(D)$ ile gösterilir.

$$pspan(D) = \left\{ \sum_{i=1}^{m} \lambda_i \boldsymbol{d}^i : \lambda_i \geq 0, \boldsymbol{d}^i \in D, m \in \mathbb{N} \right\} \subseteq \mathbb{R}^n$$

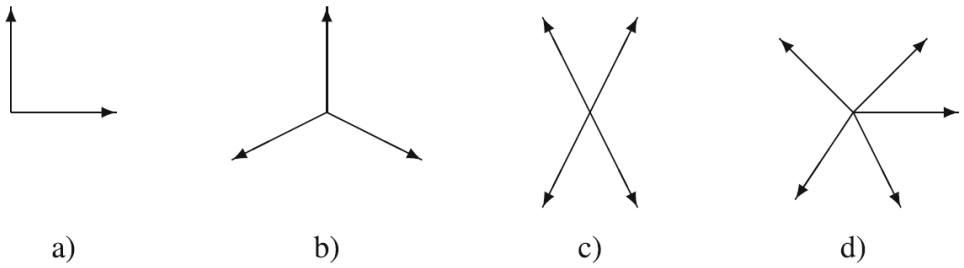
D kümesi $\mathbb{R}^n$ kümesinin pozitif kapsayan kümesidir ancak ve ancak $pspan(D) = \mathbb{R}^n$.

Tanım 3.2. D pozitif lineer bağımsız bir kümedir ancak ve ancak $\forall d \in D$ için $\mathbf{d} \notin pspan(D \backslash \{\mathbf{d}\})$.

Tanım 3.3. $D \subset \mathbb{R}^n$ kümesinin pozitif tabanı olması için gerek ve yeter koşul pozitif kapsayan küme ve pozitif lineer bağımsız olmasıdır.

Örnek 3.1.

Aşağıdaki örnekte a) pozitif kapsayan küme olmadığı için bir pozitif taban değildir**.** b), c), ve d) pozitif kapsayan kümedir ancak sadece b) ve c) pozitif tabandır çünkü d) nin dört vektörü diğer vektörlerinin pozitif kombinasyonu olarak ifade edilebilir.

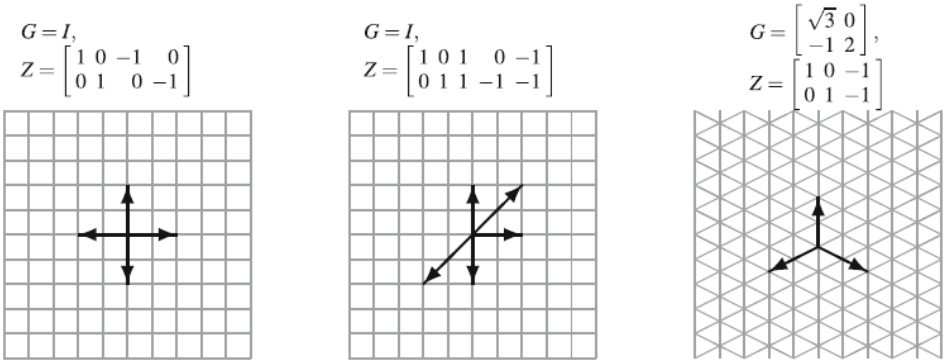


Şekil 3.1 Düzlemde bazı pozitif taban örnekleri (Audet ve Hare, 2014).

Tanım 3.4. $\mathbf{G} \in \mathbb{R}^{n\times n}$ tersinir matrisi ve $\mathbf{Z} \in \mathbb{Z}^{n\times p}$ matrisi verilsin öyle ki $\mathbf{Z}$ matrisinin sütunları $\mathbb{R}^n$ için bir pozitif kapsayan küme oluştursun ve $\mathbf{D} = \mathbf{GZ}$ olsun. M^{k} örgü kümesi (3.1) eşitliği ile tanımlanır.

$$\mathrm{M}^{\mathrm{k}} = \left\{ \mathbf{x}^{\mathbf{k}} + \delta^{\mathrm{k}}\mathrm{D}\mathbf{y} : \mathbf{x}^{\mathbf{k}} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{N}^{\mathrm{p}}, \delta^{\mathrm{k}} > 0 \right\} \subseteq \mathbb{R}^n \tag{3.1}$$

(3.1) eşitliğindeki δ^{k} değişkenine örgü adım uzunluğu parametresi denir.

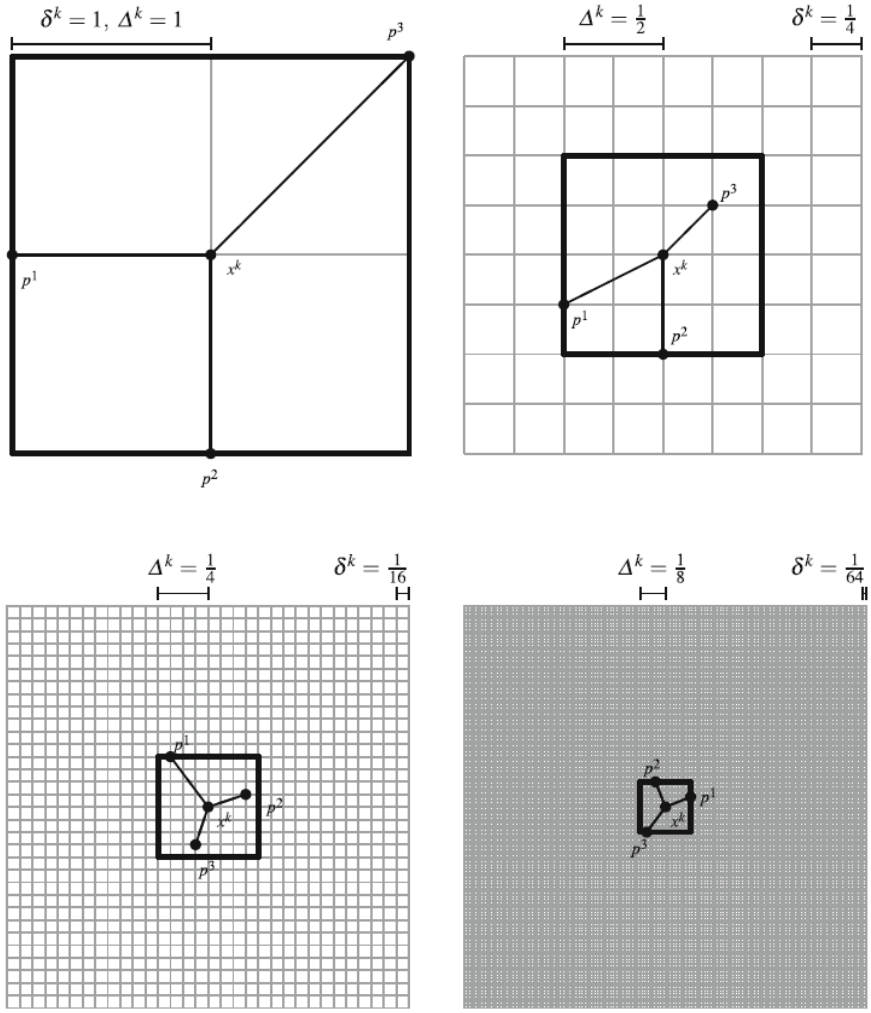


Şekil 3.2 Düzlemde bazı örgü örnekleri (Audet ve Hare, 2014).

Tanım 3.5. $\mathbf{G} \in \mathbb{R}^{n\times n}$ tersinir matrisi ve $\mathbf{Z} \in \mathbb{Z}^{n\times p}$ matrisi verilsin öyle ki $\mathbf{Z}$ matrisinin sütunları $\mathbb{R}^n$ için bir pozitif kapsayan küme oluştursun ve $\mathbf{D} = \mathbf{GZ}$ olsun. M^{k} örgü kümesi (3.1) eşitliği ile tanımlanır. $\delta^{\mathrm{k}} \leq \Delta^{\mathrm{k}}$ olmak üzere

$$\mathrm{F}^{\mathrm{k}} = \left\{ \mathrm{x} \in \mathrm{M}^{\mathrm{k}} : \left|\mathrm{x} - \mathbf{x}^{\mathbf{k}}\right|_\infty \leq \mathrm{b}\Delta^{\mathrm{k}} \right\} \tag{3.2}$$

(3.2) eşitliğinde $\mathrm{b} = \max\{|\mathbf{d}'|_\infty : \mathbf{d}' \in \mathbf{D}\}$ ve Δ^{k} değişkenine çerçeve boyutu denir.



Şekil 3.3 Düzlemde farklı Δ^k ve δ^kdeğerleri için bazı örgü ve çerçeve örnekleri (Audet ve Hare, 2014).

Örgü, pozitif taban kümesi içindeki baz vektörlerini içine alacak şekilde alt çerçevelere ayrılır. Baz vektörleri örgü üzerinde yer alan ve rassal olarak seçilen bir kuadratür düğümünden başlatılarak hareket ettirilir. Hareket doğrultusu problem boyutuna bağlı olarak baz vektörlerinin yönleri cinsinden belirlenir. Baz vektörlerinin mevcut konumunda optimize edilecek fonksiyonun uygunluk değeri ile, yeni konumundaki uygunluk değerleri karşılaştırılır. Eğer daha iyi bir sonuç elde edildiyse baz vektörü yeni konuma taşınır ve çerçeve adım uzunlukları azaltılarak yeni bir çerçeve oluşturulur. Böylelikle arama bölgesi daraltılmış olur. Aksi halde çerçeve adım uzunlukları arttırılarak arama uzayında yeni bir

çerçeve oluşturulur. Çerçeve adım uzunluğuna bağlı olarak örgü adım uzunlukları da güncellenir. Böylece arama uzayında sürekli güncellenen dinamik bir örgü yapısı oluşturulmuş olur.

MADS, havuz adı verilen optimizasyon değişkenlerin uzayındaki asimptotik olarak yörüngelerin yoğun bir kümesine yerel keşfe izin vererek Genelleştirilmiş Örüntü Arama (Generalized Pattern Search, GPS) algoritmasını genişletir. Bahsi geçen MADS yaklaşımının sözde kodu Algoritma 1 ile verilmiştir. Algoritma 1 ile görüldüğü üzere MADS yaklaşımı algoritmanın parametre değerlerine ilk değer atamalarının yapılması, parametre güncelleme ve arama adımları, istenilen hata payıyla elde edilen çözümleri içeren çözüm havuzu oluşturma adımı ve en iyi çözümün belirlenerek çalışmanın sonlandırılması adımı olmak üzere beş temel aşamada gerçekleşir.

f: $\mathbb{R}^n \rightarrow \mathbb{R}$ ve $x_0 \in \mathbb{R}^n$ başlangıç noktası verilsin. f fonksiyonunun bir yerel optimumuna, Algoritma 1 ile verilen Örgü Uyarlamalı Doğrudan Arama Algoritması (MADS) ile yaklaşılır.

Algoritma 1: MADS algoritması

Başlangıç Adımları:

1: $\Delta^0 \in (0, \infty)$	Δ: Çerçeve Boyutunun Başlangıç Değeri
2: $D = GZ$	D: Pozitif kapsayan matris
3: $\tau \in (0,1)$, $\tau \in Q$	τ: Örgü adım uzunluğu güncelleştirmen parametresi
4: $\epsilon_{stop} \in [0, \infty)$	Durma kriteri
5: $k \leftarrow 0$	İterasyon sayacı

Parametre Değişimi Adımları:

6: Örgü Adım Uzunluğunu $\delta^k = \min\{\delta^k, (\delta^k)^2\}$ olarak güncelle

Arama Adımı:

7: Eğer M^k nın sonlu alt kümesi S^k nın elemanı t için $f(t) < f(x^k)$ ise $x^k + 1 \leftarrow t$ ve $\delta^{k+1} \leftarrow \tau^{-1}\, \delta^k$ yap ve 10'a geç aksi halde 8'e geç

Havuz Oluşturma Adımı:

8: Bir pozitif kapsayan küme D^k öyle ki F^k çerçevenin Δk boyutundaki alt kümesi $P^k = \{x^k + \delta^k d : d \in D_\Delta^k\}$ olsun. Eğer $f(t) < f(x^k)$ olacak şekilde $t \in P^k$ varsa $x^{k+1} \leftarrow t$ ve $\delta^{k+1} \leftarrow \tau^{-1}\,\delta^k$ yap aksi halde x^k yerel örgü optimumlayanıdır.

9: $x^{k+1} \leftarrow x^k$ ve $\delta^{k+1} \leftarrow \tau\,\delta^k$ yap

Sonlandırma:

10: Eğer $\Delta^{k+1} \geq \epsilon_{stop}$ ise $k \leftarrow k + 1$ yap ve 7'ye geç aksi halde durdur.

Algoritma 1 ile tanımlanan Örgü Uyarlamalı Doğrudan Arama yaklaşımı tezde yapay sinir ağı modellerinin parametrelerini en iyileşmek için kullanılmıştır. Bir sonraki bölümde, Örgü Uyarlamalı Doğrudan Arama algoritmasının eğitim amacıyla kullanıldığı yapay sinir ağı modelleri tanıtılmıştır. Örgü Uyarlamalı Doğrudan Arama algoritması ile ağın ağırlıkları ve eşik değerleri optimize edilmiştir.

4. YAPAY SİNİR AĞLARI

Bu bölümde çalışmada kullanılan çeşitli yapay sinir ağlarından söz edilecektir. Yapay sinir ağların birçok çeşidi vardır. Bu çalışmada ise ileri beslemeli yapay sinir ağ ve artık yapay sinir ağları kullanılmıştır.

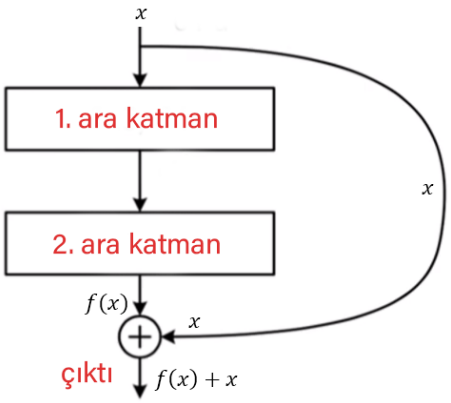
4.1 İleri Beslemeli Yapay Sinir Ağları

İleri beslemeli yapay sinir ağları tek katmanlı algılayıcı ve çok katmanlı algılayıcı olarak ikiye ayrılır. Tek katmanlı algılayıcılarda sadece girdi ve çıktı katmanı vardır. Çok katmanlı algılayıcılarda ise bir veya birden fazla ara katman bulunmaktadır. Tek katmanlı algılayıcılar sadece lineer fonksiyonların çözümünde kullanılabiliyorken çok katmanlı algılayıcılar lineer olmayan fonksiyonları için de kullanılabilir. Ayrıca çok katmanlı algılayıcılarda hem ileri hem de geri yönlü hesaplama yapılabilir.

4.2 Artık Yapay Sinir Ağları

Artık yapay sinir ağ, serebral korteksteki piramidal hücreler üzerine inşa edilen bir tür yapay sinir ağıdır. Artık yapay sinir ağlar, atlama bağlantılarını veya bazı katmanların üzerinden atlayarak çalışır. Genelde artık yapay sinir ağlarda doğrusal olmayanlar (ReLU) ve aralarında parti normalizasyonu (Batch normalization) içeren iki veya üç katmanlı atlamalar uygulanır.

Örneğin 2 ara katmanlı bir ağı ele alalım. 1. ara katmanın çıktısı 2. ara katmanına girdi olarak verilmekle birlikte 2. ara katmanın çıktısına da eklenir. İlk ara katmanın girdisi x olsun. O halde standart bir yapay sinir ağda 2. ara katmanın çıktısı $y = f(x)$ iken artık yapay sinir ağlarda ise $y = f(x) + x$ olur.



Şekil 4.1 İki ara katmanlı artık yapay sinir ağın yapısı

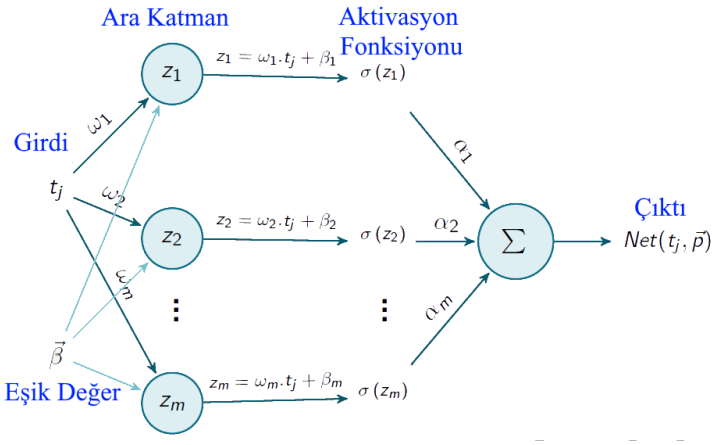
4.3 Yapay Sinir Ağları ile *SI* ve *SIR* Epidemi Modellerinin Nümerik Çözümleri

(2.1), (2.2), (2.5), (2.6)'de bahsedilen t bağımsız değişkenine bağlı başlangıç koşuluna sahip diferansiyel denklemler sistemlerini sağlayan ve $[a, b]$ aralığında t bağımsız değişkenine bağlı çözüm kümesini bulmak için yapay sinir ağının ürettiği yaklaşık çözüme bağlı ve başlangıç koşullarını sağlayan deneme fonksiyonları

$$S_T = s_0 + (t - t_0)\, Net\left(t_j, s_j, i_j, \overrightarrow{p_S}\right) \tag{4.1}$$

$$I_T = i_0 + (t - t_0)\, Net\left(t_j, s_j, i_j, \overrightarrow{p_I}\right) \tag{4.2}$$

şeklindedir. Genel olarak, $h > 0$ sabit adım uzunluğu için $t_j = t_0 + j.h, j = 0, 1, \ldots, n$ olacak şekilde $[t_0, t_n]$ aralığının parçalanışından elde edilen noktalar Net fonksiyonu ile belirtilen ağın eğitimi için kullanılır. En basit haliyle, (4.1) ve (4.2) eşitliklerinden verilen Net fonksiyonu aslında tek bir girdiye bağlı ve tek ara katmanlı bir ileri beslemeli bir yapay sinir ağı modelini belirtir ve model Şekil 4.1 ile gösterilmektedir. $\overrightarrow{p_S}$ ve $\overrightarrow{p_I}$ ağın nöronları arasındaki bağlantıların ağırlıkları ve eşik değerleri içeren ayarlanabilir parametre vektörleridir. Bu vektörlerin boyutu ağın ara katmanındaki nöron sayısına bağlıdır. Ağdaki nöron sayısı m_1 ve m_2 olmak üzere $\overrightarrow{p_S} \in \mathbb{R}^{3m_1}$ ve $\overrightarrow{p_I} \in \mathbb{R}^{3m_2}$ dir.



Şekil 4.2 İleri beslemeli yapay sinir ağın topolojik yapısı.

n toplam girdi sayısı $j = 1, 2, \ldots, n$ için, $t_j \in [t_0, t_n]$ noktasında yapay sinir ağının çıktısı, $\overrightarrow{p_S} = \overrightarrow{p_S}(\overrightarrow{\alpha_S}, \overrightarrow{w_S}, \overrightarrow{\beta_S})$ ve $\overrightarrow{p_I} = \overrightarrow{p_I}(\overrightarrow{\alpha_I}, \overrightarrow{w_I}, \overrightarrow{\beta_I})$ ağın bilinmeyen parametreleri olmak üzere yapay sinir ağın çıktısı aşağıda verilmiştir.

$$Net(t_j, s_j, i_j, \overrightarrow{p_S}) = \sum_{i=1}^{m_1} \alpha_{Si}\, \sigma(z_{1i}) \tag{4.3}$$

$$Net(t_j, s_j, i_j, \overrightarrow{p_I}) = \sum_{i=1}^{m_2} \alpha_{Ii}\, \sigma(z_{2i}) \tag{4.4}$$

Burada $z_{1i} = \omega_{Si} t_j + \beta_{Si}$ yapay sinir ağının nöronlarını temsil etmektedir. w_{Si} ve w_{Ii} yapay sinir ağının girdi katmanından ara katmandaki i. nörona giden bağlantının ağırlıklarıdır. α_{Si} ve α_{Ii} değerleri ara katmandaki i. nörondan çıktı katmandaki nörona giden ağırlık parametreleridir. β_{Si} ve β_{Ii} değerleri z_{1i} ve z_{2i} nöronlarının eşik değerleridir. Bu ağda kullanılan aktivasyon fonksiyonu (4.5) eşitliği ile tanımlanan sigmoid fonksiyonudur.

$$\sigma(z_{1i}) = \frac{1}{1 + \exp(-z_{1i})} \tag{4.5}$$

Geliştirilen modelin eğitimi için, (4.1) ve (4.2) denklemlerinin zamana göre değişimini gösteren türev değerlerinin hesaplanması gerekmektedir. (4.1) ve (4.2) eşitliklerinin türevleri (4.6) ve (4.7) eşitlikleri ile hesaplanır.

$$\frac{\partial S_T}{\partial \mathrm{t}} = Net\left(t_j, s_j, i_j, \overrightarrow{p_S}\right) + (t - t_0)\frac{\partial Net\left(t_j, s_j, i_j, \overrightarrow{p_S}\right)}{\partial \mathrm{t}} \tag{4.6}$$

$$\frac{\partial I_T}{\partial \mathrm{t}} = Net\left(t_j, s_j, i_j, \overrightarrow{p_I}\right) + (t - t_0)\frac{\partial Net\left(t_j, s_j, i_j, \overrightarrow{p_I}\right)}{\partial \mathrm{t}} \tag{4.7}$$

Başlangıçta rassal olarak belirlenen parametre değerleriyle ağın çıktısı hesaplanır. Elde edilen çıktı sonucunda ağda hata oluşacaktır. (Gözetimsiz öğrenme) Oluşan hatayı minimize etmek amacıyla, genellikle geri yayılım algoritması kullanılır ve $\overrightarrow{p_1}$ ve $\overrightarrow{p_2}$ bilinmeyen parametre değerleri güncellenerek oluşan hata tüm ağa yayılır.

Bu çalışmada, kullanılan yapay sinir ağları için farklı bir eğitim stratejisi de önerilmektedir. Üretilen ağ modelleri çok katmanlı yapıya sahiptir ve ağın eğitimi her katman için ayrı ayrı gerçekleştirilir. Ağın ağırlıkları öncelikle çıktı katmanından girdi katmanına doğru belirlenmeye çalışılır. Yani öncelikle belli bir katmandaki ağ ağırlıkları Örgü Uyarlamalı Doğrudan Arama yöntemine girdi olarak sunulur ve diğer katmanların ağırlıkları sabit bırakılır. Belirtilen katmanın ağırlık değerleri saptandıktan sonra bir önceki katman için süreç tekrarlanır. Bu bağlamda yaklaşım Geri Yayılım Algoritmasının (Backpropagation) çalışma mantığından esinlenilmiştir. Ancak girdi katmanına ulaşıldıktan sonra, ağın eğitimine bu kez girdi katmanından çıktı katmanına doğru ileri yönlü ağırlık belirleme yaklaşımı belirlenmiştir. Bu süreç bir tur olarak adlandırılmıştır. Çalışmada ağın eğitimi iki turda tamamlanmıştır. Ek olarak, ağın her katmanında elde edilen ağırlık değerleri ve ağın çıktısına bağlı maliyet fonksiyonu değerleri bir havuzda depolanır. Bu havuzdaki maliyet fonksiyonu değerini minimumlayan ağ ağırlıkları eğitilmiş ağın ağırlıkları olarak belirlenir.

Benzer şekilde (2.13), (2.14) ve (2.15) diferansiyel denklem sistemi için deneme fonksiyonu

$$S_T = s_0 + (t - t_0)\, Net\left(t_j, s_j, i_j, r_j, \overrightarrow{p_S}\right) \tag{4.8}$$

$$I_T = i_0 + (t - t_0)\, Net\left(t_j, s_j, i_j, r_j, \overrightarrow{p_I}\right) \tag{4.9}$$

$$R_T = r_0 + (t - t_0)\, Net\left(t_j, s_j, i_j, r_j, \overrightarrow{p_R}\right) \tag{4.10}$$

şeklinde olup ağın çıktısı

$$Net\left(t_j, s_j, i_j, r_j, \overrightarrow{p_S}\right) = \sum_{i=1}^{m_1} \alpha_{Si}\, \sigma(z_{1i}) \tag{4.11}$$

$$Net\left(t_j, s_j, i_j, r_j, \overrightarrow{p_I}\right) = \sum_{i=1}^{m_2} \alpha_{Ii}\, \sigma(z_{2i}) \tag{4.12}$$

$$Net\left(t_j, s_j, i_j, r_j, \overrightarrow{p_R}\right) = \sum_{i=1}^{m_3} \alpha_{Ri}\, \sigma(z_{3i}) \tag{4.13}$$

ile verilir.

(2.1), (2.2) ile verilen SI modeli aşağıdaki gibi de ifade edilebilir.

$$\frac{dS(t)}{dt} = f(\mathrm{t}, \mathrm{S(t)}, \mathrm{I(t)}) \tag{4.14}$$

$$\frac{d\mathrm{I}(t)}{dt} = g(\mathrm{t}, \mathrm{S(t)}, \mathrm{I(t)}) \tag{4.15}$$

Çalışma boyunca verilerin 90%'ı yapay sinir ağları eğitmek ve geri kalan 10%'u epidemiyoloji modellerini hesaplamak için kullanılmış olup bu çalışmada kullanılan maliyet fonksiyonu aşağıda verilmiştir.

$$E = \frac{1}{2\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \left\{ \left[\frac{dS(t_\mathrm{i})}{dt_\mathrm{i}} - f(t_\mathrm{i}, \mathrm{S}(t_\mathrm{i}), \mathrm{I}(t_\mathrm{i})) \right]^2 + \left[\frac{d\mathrm{I}(t_\mathrm{i})}{dt_\mathrm{i}} - \mathrm{g}(t_\mathrm{i}, \mathrm{S}(t_\mathrm{i}), \mathrm{I}(t_\mathrm{i})) \right]^2 \right\} \tag{4.16}$$

SIS modeli için de (4.16) ile verilen maliyet fonksiyonu kullanılabilir. Benzer şekilde (2.13), (2.14) ve (2.15) ile verilen SIR modeli aşağıdaki gibi ifade edilir

$$\frac{dS(t)}{dt} = f(\mathrm{t}, \mathrm{S(t)}, \mathrm{I(t)}, R(t)) \tag{4.17}$$

$$\frac{d\mathrm{I}(t)}{dt} = g(\mathrm{t}, \mathrm{S(t)}, \mathrm{I(t)}, \mathrm{R(t)}) \tag{4.18}$$

$$\frac{dR(t)}{dt} = \mathrm{h}(\mathrm{t}, \mathrm{S(t)}, \mathrm{I(t)}, \mathrm{R(t)}) \tag{4.19}$$

ve böylece maliyet fonksiyonu

$$E = \frac{1}{2\mathrm{n}}\sum_{\mathrm{i}=1}^{\mathrm{n}}\left\{\left[\frac{dS(t_{\mathrm{i}})}{dt_{\mathrm{i}}} - f(t_{\mathrm{i}}, \mathrm{S}(t_{\mathrm{i}}), \mathrm{I}(t_{\mathrm{i}}), \mathrm{R}(t_{\mathrm{i}}))\right]^2 + \left[\frac{d\mathrm{I}(t_{\mathrm{i}})}{dt_{\mathrm{i}}} - \mathrm{g}(t_{\mathrm{i}}, \mathrm{S}(t_{\mathrm{i}}), \mathrm{I}(t_{\mathrm{i}}), \mathrm{R}(t_{\mathrm{i}}))\right]^2 + \left[\frac{d\mathrm{R}(t_{\mathrm{i}})}{dt_{\mathrm{i}}} - \mathrm{h}(t_{\mathrm{i}}, \mathrm{S}(t_{\mathrm{i}}), \mathrm{I}(t_{\mathrm{i}}), \mathrm{R}(t_{\mathrm{i}}))\right]^2\right\} \quad (4.20)$$

Ağın maliyet fonksiyonu, eğitim setindeki örnek veriler kullanılarak optimize edilir. Optimize edilen ağ parametreleri ile ağ modeli tamamen oluşturulmuş olur. Eğitilen ağ için, test kümesi verileri ile ağın ürettiği çıktılar belirlenir. Buna bağlı olarak, $j = n + 1, n + 2, \ldots$ için t_j günündeki pozitif vaka sayısı, infekte olan birey sayısı ve iyileşen kişi sayısı sırasıyla deneme fonksiyonu cinsinden $S_T(t_j), I_T(t_j)$ ve $R_T(t_j)$ değerleriyle yaklaşık olarak belirlenir. t_j günündeki gerçekte gözlemlenen pozitif vaka sayısı, infekte olan birey sayısı ve iyileşen kişi sayısı ise sırasıyla s_j, i_j ve r_j olmak üzere mutlak hatalar

$$E(S) = |s_j - S_T(t_j)|$$

$$E(I) = |i_j - I_T(t_j)|$$

$$E(R) = |r_j - R_T(t_j)|$$

Denklemler ile verilir.

5 DENEYSEL ÇALIŞMALAR

Bu bölümde tezde bahsi geçen epidemiyoloji modelleri COVID-19 dinamiklerine uyarlanmıştır. Bu amaca ulaşmak için, Türkiye Cumhuriyeti Sağlık Bakanlığı tarafından www.covid19.saglik.gov.tr adresinde yayınladığı veriler kullanılmıştır. Verilerin web sitesinden alınabilmesi için bir web arama robotu betiği oluşturulmuştur. Oluşturulan web arama robotuna ait kodlar Ek- 1'de sunulmuştur. Ancak vurgulanması gerekir ki, üretilen kod web sayfasını her ziyaret ettiğinde sayfa üzerindeki html etiketlerinden yola çıkarak ilgili veriye ulaşmaya çalışmaktadır. Bu amaçla önce html etiketlerini temizler ve ham veri üzerinden bilgi çıkarsamaya çalışır. Haliyle an itibari ile kusursuz çalışan kod web sayfasının yapısı değiştirildiğinde etkisiz hale gelecektir. 27 Kasım 2020 tarihi ile 28 Mayıs 2021 tarihleri arasındaki toplam 182 günlük toplanan veri çalışmada dikkate alınmıştır.

Deneysel çalışmalar boyunca, S değeri günlük vaka sayısı, I değeri günlük hasta sayısı ve R değeri ise günlük iyileşen vaka sayısını göstermektedir. Tezde SI, SIS ve SIR modellerinin bahsedilen yöntemlerle nümerik çözümleri elde edilmiştir. Deneysel çalışmalarda kullanılan keyfi parametre değerleri Çizelge 5.1 ile tanımlandığı gibidir.

Önerilen modellerin tümünde, 1. gün 27 Kasım 2020 tarihini belirtmek üzere, ağ gün verisini girdi olarak kabul eder. Diğer günler ise artan sırada numaralandırılarak ağa girdi olarak sunulur. Dolayısıyla ağın girdisi t. gün olmak üzere, modele göre ağın bir veya daha fazla çıktısını belirtir. Günlük olarak gözlemlenen gerçek pozitif vaka sayısı, enfekte olan vaka sayısı ve iyileşen vaka sayılarının toplamı normalize edilmemiş popülasyon sayısını belirtir. Her gün farklı sayılarda veri edildiğinden popülasyon sayısı zamana göre değişen bir değerdir. Tezde, modellerin sağlıklı çalışabilmesi adına günlük pozitif vaka sayıları, enfekte ve iyileşen birey sayıları popülasyondaki toplam birey sayısına oranlanarak normalize edilmiştir. Dolayısıyla tüm deneysel çalışmalarda popülasyondaki birey sayısı Çizelge 5.1 ile görüleceği üzere 1 olarak alınmıştır. Normalize edilen pozitif vaka sayıları, tüm popülasyondaki Covid-19 testi pozitif çıkan birey oranını gösterir. Benzer olarak, normalize edilmiş enfekte ve iyileşen birey sayıları da oran olarak belirtilmiştir.

Çizelge 5.1 Deneysel çalışmalarda kullanılan keyfi parametreler

Parametre	Parametre Değeri
Popülasyondaki birey sayısı, N	1
Arama uzayının alt sınırı	-1
Arama uzayının üst sınırı	1
Maksimum iterasyon sayısı	1000
Arama uzayının boyutu, D	$6m$+2
Eğitim kümesi için sabit adım uzunluğu, h	0,01

Literatürdeki çoğu çalışmanın aksine, modellerde tanımlanan hastalığın yayılma katsayını belirten β değeri ile iyileşme katsayısını belirten γ değeri sabit değer olarak belirlenmemiştir. Tezde bu değerler, sırasıyla günlük vaka sayısı ve günlük iyileşen sayısını kullanarak En Küçük Kareler (Least Squares) eğri uydurma yöntemiyle elde edilmiştir. Gözlemlenen değerler için beta ve gamma katsayılarının hesaplanması sağlayan En Küçük Kareler Yöntemine ait yazılan kod

Ek- 2'de sunulmuştur. Yukarıda belirtilen kısıtlar altında elde edilen nümerik çözümler tezin devamında sunulmuştur.

5.1 SI Modeli Nümerik çözümleri

Bu kısımda, (2.1) ve (2.2) denklem sisteminin MADS algoritması ile eğitilmiş çok katmanlı ileri beslemeli yapay sinir ağı kullanarak elde edilen nümerik çözümlerinden bahsedilmiştir. Oluşturulan ağ modelinde toplam yedi tane ara katman bulunmaktadır ve her ara katmandaki nöron sayıları sırasıyla 10, 6, 6, 4, 4, 3, 3 olarak belirlenmiştir. Önerilen modelde, gün bilgisi ağa girdi olarak sunulur ve ağın çıktısı popülasyondaki pozitif vaka oranını yaklaşık olarak belirtir. Popülasyondaki hasta olan birey oranı ise olarak belirlenir.

$S(t)$ değeri gerçekte t. günde gözlemlenen pozitif vaka oranını göstermek üzere mutlak hata miktarı $E(S)$ değeri ile hesaplanır. Benzer şekilde, $I(t)$ değeri gerçekte t. günde hasta birey oranını belirtmek üzere hasta oranından elde edilen mutlak hata miktarı $E(I)$ olarak

hesaplanır. Tezin, Ek- 3 bölümünde MADS algoritması ile eğitilen hem Çok Katmanlı Algılayıcı hem de Artık Yapay Sinir Ağı modellerini üreten kodlar sunulmuştur.

Çizelge 5.2 ile ağın eğitimi için kullanılan veriler ve ağın eğitimi sonunda elde edilen nümerik çözümlerde oluşan mutlak hata miktarları özetlenmiştir. Görüleceği üzere ağın eğitimi için 172 günlük veri kullanılmıştır. Çizelge 5.3 de ise eğitilmiş ağ modelinin test edilebilmesi için bir haftalık veri kullanılmıştır. Geliştirilen tüm modellerde bu yaklaşım tercih edilmiştir. Kısaca ağ bir haftalık hastalık seyrini tahminlemek için modellenmiştir.

Şekil 5.1 ile SI modeli için önerilen çok katmanlı algılayıcı yapay sinir ağından elde edilen nümerik çözümler grafiksel olarak özetlenmiştir. Grafikte gözlemlenen gerçek değerler ile ağın ürettiği nümerik değerlerin oldukça uyumlu olduğu gözlenmiştir. Şekilde kesikli çizgili yeşil ve sarı oval simgeler ağın tahminlerini betimlemektedir. Çizelge 5.3'den elde edilen ortalama mutlak hata miktarı yaklaşık olarak 0,039 civarındadır. Bu sonuçtan da teyit edilebileceği gibi önerilen model hastalığın seyri hakkında bize oldukça yakın bilgi verebilir.

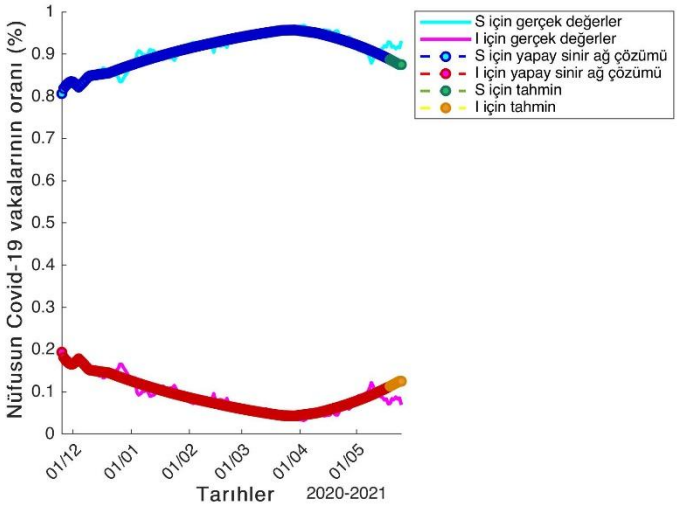
Çizelge 5.2 SI Modeli için eğitim kümesinden elde edilen ileri beslemeli yapay sinir ağı çözümleri

Tarih	t	$S(t)$	$I(t)$	$E(S)$	$E(I)$
27.11.2020	1	0.8062	0.1938	0	0
06.12.2020	10	0.8219	0.1781	0.0039	0.0039
15.12.2020	19	0.8506	0.1494	0.0100	0.0100
24.12.2020	28	0.8584	0.1416	0.0050	0.0050
02.01.2021	37	0.8727	0.1273	0.0064	0.0064
11.01.2021	46	0.8858	0.1142	0.0106	0.0106
20.01.2021	55	0.8976	0.1024	0.0109	0.0109
29.01.2021	64	0.9084	0.0916	0.0089	0.0089
07.02.2021	73	0.9181	0.0819	0.0091	0.0091

16.02.2021	82	0.9269	0.0731	0.0226	0.0226
25.02.2021	91	0.9355	0.0645	0.0005	0.0005
06.03.2021	100	0.9431	0.0569	0.0002	0.0002
15.03.2021	109	0.9499	0.0501	0.0027	0.0027
24.03.2021	118	0.9559	0.0441	0.0018	0.0018
02.04.2021	127	0.9559	0.0441	0.0097	0.0097
11.04.2021	136	0.9505	0.0495	0.0081	0.0081
20.04.2021	145	0.9406	0.0594	0.0068	0.0068
29.04.2021	154	0.9282	0.0718	0.0131	0.0131
08.04.2021	163	0.9133	0.0867	0.0101	0.0101
17.04.2021	172	0.8956	0.1044	0.0181	0.0181

Çizelge 5.3 *SI* Modeli için test kümesinden elde edilen ileri beslemeli yapay sinir ağı çözümleri

Tarih	*t*	*S(t)*	*I(t)*	*E(S)*	*E(I)*
18.05.2021	176	0.8868	0.1132	0.0408	0.0408
19.05.2021	177	0.8844	0.1156	0.0320	0.0320
20.05.2021	178	0.8821	0.1179	0.0364	0.0364
21.05.2021	179	0.8797	0.1203	0.0327	0.0327
22.05.2021	180	0.8772	0.1228	0.0397	0.0397
23.05.2021	181	0.8750	0.1250	0.0396	0.0396
24.05.2021	182	0.8750	0.1250	0.0561	0.0561



Şekil 5.1: MADS ileri beslemeli yapay sinir ağı çözümleri

Aynı sistem için MADS algoritması ile eğitilen artık yapay sinir ağının ürettiği nümerik çözümler Çizelge 5.4 ve Çizelge 5.5 ile özetlenmiştir. Bu modelde ağda yine yedi adet ara katman var olup bu ara katmanlardaki nöron sayısı sırasıyla 10, 6, 3, 10, 6, 3 ve 1'dir. Artık yapay sinir ağı modelinde atlama indeksi 3 olarak alınmıştır. Yani her katman kendisinden sonraki üçüncü katmana bağlantılıdır. Bu durum ağdaki bilinmeyen parametre sayısının artmasına ve dolayısıyla uygun çözümün belirlenmesinde güçlüklere neden olur. Ağın istenilen düzeyde eğitilebilmesi için daha çok örneğe veya iterasyona bağlı eğitim süresinin arttırılmasına ihtiyaç duyulur. Ancak tezde ağ modellerinin sağlıklı karşılaştırılabilmesi için maksimum iterasyon sayısı çok katmanlı algılayıcı modellerindeki ile aynı, 1000 olarak alınmıştır.

Çizelge 5.4 *SI* Modeli için eğitim kümesinden elde edilen artık yapay sinir ağı çözümleri

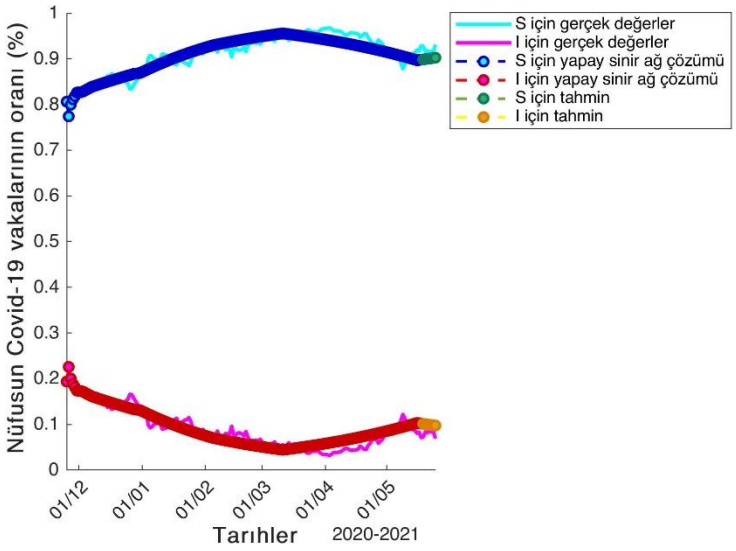
Tarih	*t*	*S(t)*	*I(t)*	*E(S)*	*E(I)*

27.11.2020	1	0.8062	0.1938	0	0
06.12.2020	10	0.8486	0.1514	0.0228	0.0228
15.12.2020	19	0.8509	0.1491	0.0102	0.0102
24.12.2020	28	0.8484	0.1516	0.0049	0.0049
02.01.2021	37	0.8782	0.1218	0.0119	0.0119
11.01.2021	46	0.8952	0.1048	0.0011	0.0011
20.01.2021	55	0.9070	0.0930	0.0203	0.0203
29.01.2021	64	0.9158	0.0842	0.0015	0.0015
07.02.2021	73	0.9223	0.0777	0.0049	0.0049
16.02.2021	82	0.9280	0.0720	0.0238	0.0238
25.02.2021	91	0.9333	0.0667	0.0017	0.0017
06.03.2021	100	0.9382	0.0618	0.0047	0.0047
15.03.2021	109	0.9428	0.0572	0.0044	0.0044
24.03.2021	118	0.9471	0.0529	0.0107	0.0107
02.04.2021	127	0.9427	0.0573	0.0229	0.0229
11.04.2021	136	0.9379	0.0621	0.0207	0.0207
20.04.2021	145	0.9326	0.0674	0.0147	0.0147
29.04.2021	154	0.9267	0.0733	0.0145	0.0145
08.04.2021	163	0.9203	0.0797	0.0172	0.0172
17.04.2021	172	0.9134	0.0866	0.0003	0.0003

Çizelge 5.5 *SI* Modeli için test kümesinden elde edilen artık yapay sinir ağı çözümleri

Tarih	*t*	*S(t)*	*I(t)*	*E(S)*	*E(I)*
17.05.2021	175	0.9110	0.0890	0.0178	0.0178

18.05.2021	176	0.9101	0.0899	0.0174	0.0174
19.05.2021	177	0.9093	0.0907	0.0071	0.0071
20.05.2021	178	0.9085	0.0915	0.0100	0.0100
21.05.2021	179	0.9076	0.0924	0.0048	0.0048
22.05.2021	180	0.9068	0.0932	0.0102	0.0102
23.05.2021	181	0.9059	0.0941	0.0087	0.0087



Şekil 5.2 *SI* modelinin MADS artık yapay sinir ağı çözümleri

5.2 *SIS* Modeli Nümerik Çözümleri

Benzer şekilde (2.5) ve (2.6) denklem sisteminin nümerik çözümlerine göz atalım. İlk olarak yine yedi ara katmanlı ve nöron sayısı sırasıyla 10, 6, 6, 4, 4, 3 ve 3 olan ileri beslemeli yapay sinir ağı ele alalım.

Çizelge 5.6 ile ağın eğitimi için kullanılan veriler ve ağın eğitimi sonunda elde edilen nümerik çözümlerde oluşan mutlak hata miktarları özetlenmiştir. Görüleceği üzere ağın eğitimi için 172 günlük veri kullanılmıştır. Çizelge 5.7 de ise eğitilmiş ağ modelinin test edilebilmesi için bir haftalık veri kullanılmıştır.

Şekil 5.3 ile *SIS* modeli için önerilen çok katmanlı algılayıcı yapay sinir ağından elde edilen nümerik çözümler grafiksel olarak özetlenmiştir. Grafikte gözlemlenen gerçek değerler ile ağın ürettiği nümerik değerlerin oldukça uyumlu olduğu gözlenmiştir. Şekilde kesikli çizgili yeşil ve sarı oval simgeler ağın tahminlerini betimlemektedir. Çizelge 5.7'den elde edilen ortalama mutlak hata miktarı yaklaşık olarak 0.042 civarındadır. Bu sonuçtan da teyit edilebileceği gibi önerilen model hastalığın seyri hakkında bize oldukça yakın bilgi verebilir.

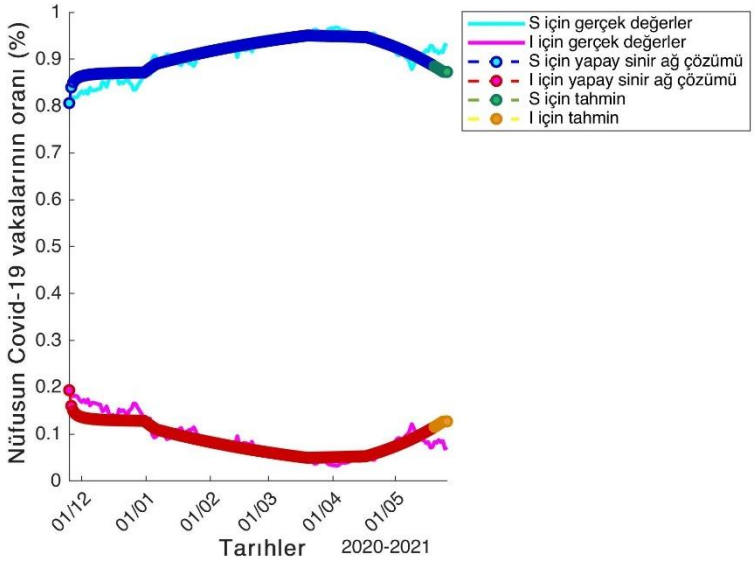
Çizelge 5.6 *SIS* Modeli için eğitim kümesinden elde edilen ileri beslemeli yapay sinir ağı çözümleri

Tarih	*t*	*S(t)*	*I(t)*	*E(S)*	*E(I)*
27.11.2020	1	0.8062	0.1938	0	0
06.12.2020	10	0.8664	0.1336	0.0406	0.0406
15.12.2020	19	0.8696	0.1304	0.0290	0.0290
24.12.2020	28	0.8707	0.1293	0.0174	0.0174
02.01.2021	37	0.8713	0.1287	0.0050	0.0050
11.01.2021	46	0.8935	0.1065	0.0028	0.0028
20.01.2021	55	0.9033	0.0967	0.0166	0.0166
29.01.2021	64	0.9123	0.0877	0.0051	0.0051
07.02.2021	73	0.9205	0.0795	0.0068	0.0068
16.02.2021	82	0.9279	0.0721	0.0237	0.0237
25.02.2021	91	0.9347	0.0653	0.0003	0.0003
06.03.2021	100	0.9409	0.0591	0.0020	0.0020
15.03.2021	109	0.9466	0.0534	0.0006	0.0006
24.03.2021	118	0.9502	0.0498	0.0075	0.0075

02.04.2021	127	0.9492	0.0508	0.0164	0.0164
11.04.2021	136	0.9481	0.0519	0.0105	0.0105
20.04.2021	145	0.9457	0.0543	0.0017	0.0017
29.04.2021	154	0.9326	0.0674	0.0087	0.0087
08.04.2021	163	0.9165	0.0835	0.0134	0.0134
17.04.2021	172	0.8970	0.1030	0.0167	0.0167

Çizelge 5.7 *SIS* Modeli için test kümesinden elde edilen ileri beslemeli yapay sinir ağı çözümleri

Tarih	*t*	*S(t)*	*I(t)*	*E(S)*	*E(I)*
19.05.2021	177	0.8844	0.1156	0.0320	0.0320
20.05.2021	178	0.8818	0.1182	0.0367	0.0367
21.05.2021	179	0.8790	0.1210	0.0334	0.0334
22.05.2021	180	0.8763	0.1237	0.0407	0.0407
23.05.2021	181	0.8734	0.1266	0.0412	0.0412
24.05.2021	182	0.8728	0.1272	0.0584	0.0584
25.05.2021	183	0.8728	0.1272	0.0549	0.0549



Şekil 5.3: *SIS* modelinin MADS ileri beslemeli yapay sinir ağı çözümleri

Aynı sistem için MADS algoritması ile eğitilen artık yapay sinir ağının ürettiği nümerik çözümler Çizelge 5.8 ve Çizelge 5.9 ile özetlenmiştir. Bu modelde ağda yine yedi adet ara katman var olup bu ara katmanlardaki nöron sayısı sırasıyla 10, 6, 3, 10, 6, 3 ve 1'dir. Artık yapay sinir ağı modelinde atlama indeksi 3 olarak alınmıştır. Yani her katman kendisinden sonraki üçüncü katmana bağlantılıdır. Bu durum ağdaki bilinmeyen parametre sayısının artmasına ve dolayısıyla uygun çözümün belirlenmesinde güçlüklere neden olur. Ağın istenilen düzeyde eğitilebilmesi için daha çok örneğe veya iterasyona bağlı eğitim süresinin arttırılmasına ihtiyaç duyulur. Ancak tezde ağ modellerinin sağlıklı karşılaştırılabilmesi için maksimum iterasyon sayısı çok katmanlı algılayıcı modellerindeki ile aynı, 1000 olarak alınmıştır.

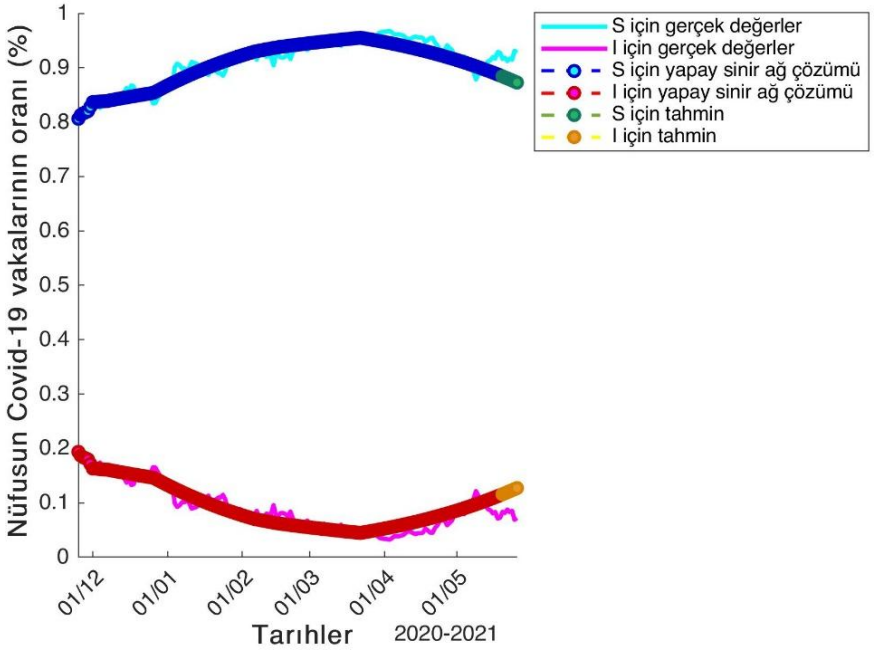
Çizelge 5.8 *SIS* Modeli için eğitim kümesinden elde edilen artık yapay sinir ağı çözümleri

Tarih	*t*	*S(t)*	*I(t)*	*E(S)*	*E(I)*
27.11.2020	1	0.8062	0.1938	0	0
06.12.2020	10	0.8379	0.1621	0.0121	0.0121
15.12.2020	19	0.8439	0.1561	0.0033	0.0033
24.12.2020	28	0.8508	0.1492	0.0026	0.0026
02.01.2021	37	0.8650	0.1350	0.0013	0.0013
11.01.2021	46	0.8842	0.1158	0.0121	0.0121
20.01.2021	55	0.9010	0.0990	0.0143	0.0143
29.01.2021	64	0.9156	0.0844	0.0017	0.0017
07.02.2021	73	0.9282	0.0718	0.0010	0.0010
16.02.2021	82	0.9362	0.0638	0.0320	0.0320

25.02.2021	91	0.9419	0.0581	0.0068	0.0068
06.03.2021	100	0.9467	0.0533	0.0037	0.0037
15.03.2021	109	0.9511	0.0489	0.0039	0.0039
24.03.2021	118	0.9552	0.0448	0.0025	0.0025
02.04.2021	127	0.9481	0.0519	0.0175	0.0175
11.04.2021	136	0.9399	0.0601	0.0187	0.0187
20.04.2021	145	0.9305	0.0695	0.0169	0.0169
29.04.2021	154	0.9196	0.0804	0.0217	0.0217
08.04.2021	163	0.9071	0.0929	0.0040	0.0040
17.04.2021	172	0.8929	0.1071	0.0208	0.0208

Çizelge 5.9 *SIS* Modeli için test kümesinden elde edilen artık yapay sinir ağı çözümleri

Tarih	*t*	*S(t)*	*I(t)*	*E(S)*	*E(I)*
19.05.2021	177	0.8841	0.1159	0.0323	0.0323
20.05.2021	178	0.8823	0.1177	0.0362	0.0362
21.05.2021	179	0.8805	0.1195	0.0319	0.0319
22.05.2021	180	0.8786	0.1214	0.0384	0.0384
23.05.2021	181	0.8767	0.1233	0.0380	0.0380
24.05.2021	182	0.8748	0.1252	0.0564	0.0564
25.05.2021	183	0.8728	0.1272	0.0549	0.0549



Şekil 5.4 *SIS* modelinin MADS artık yapay sinir ağı çözümleri

5.3 *SIR* Modeli Nümerik çözümleri

Son olarak (2.13), (2.14) ve (2.15) diferansiyel denklem sisteminin nümerik çözümlerine bakalım. *R(t), t*. günün iyileşen hasta sayısını göstermekte olup *E(R)* mutlak hata göstermektedir.

Çizelge 5.10 ile ağın eğitimi için kullanılan veriler ve ağın eğitimi sonunda elde edilen nümerik çözümlerde oluşan mutlak hata miktarları özetlenmiştir. Görüleceği üzere ağın eğitimi için 172 günlük veri kullanılmıştır. Çizelge 5.11 de ise eğitilmiş ağ modelinin test edilebilmesi için bir haftalık veri kullanılmıştır.

Şekil 5.5 ile *SIS* modeli için önerilen çok katmanlı algılayıcı yapay sinir ağından elde edilen nümerik çözümler grafiksel olarak özetlenmiştir. Grafikte gözlemlenen gerçek değerler ile ağın ürettiği nümerik değerlerin oldukça uyumlu olduğu gözlenmiştir. Şekilde kesikli çizgili yeşil ve sarı oval simgeler ağın tahminlerini betimlemektedir. Çizelge 5.11'den elde edilen

ortalama mutlak hata miktarı yaklaşık olarak 0.164 civarındadır. Bu sonuçtan da teyit edilebileceği gibi önerilen model hastalığın seyri hakkında bize oldukça yakın bilgi verebilir. Bunun temel nedeninin günlük vaka sayılarında zaman zaman görülen ani ve büyük miktardaki değişim oranları olduğu kanısına varılmıştır. Yapay sinir ağlarının bu değişimlere hızlı bir biçimde adapte olamadığı görülmüştür. Ağın eğitimi için, arama uzayının tamamını tarama yeteneğine sahip optimizasyon algoritmalarının tercih edilmesi gerektiği düşünülmektedir.

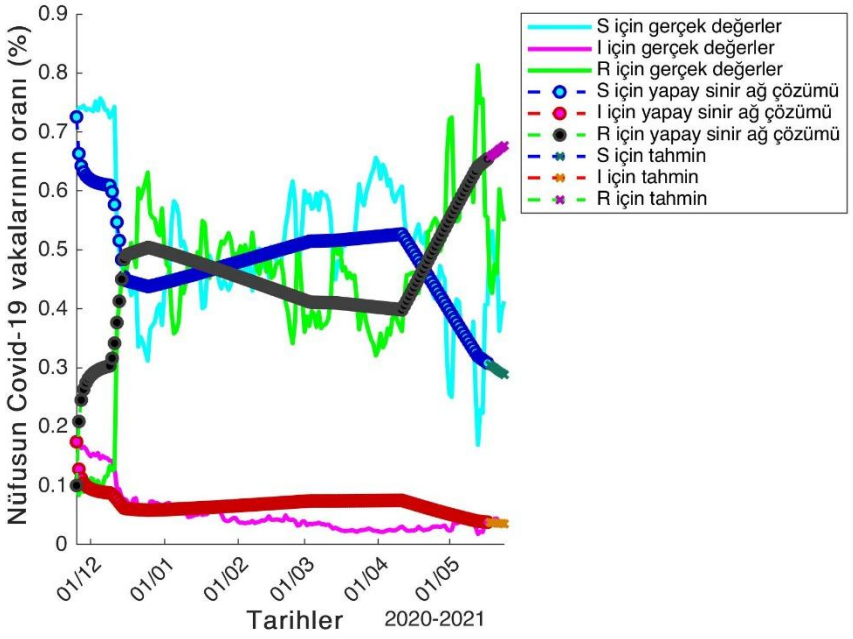
Çizelge 5.10 *SIR* Modeli için eğitim kümesinden elde edilen ileri beslemeli yapay sinir ağı çözümleri

Tarih	*t*	*S(t)*	*I(t)*	*R(t)*	*E(S)*	*E(I)*	*E(R)*
27.11.2020	1	0.7255	0.1744	0.1001	0	0	0
06.12.2020	10	0.6134	0.0906	0.2959	0.1230	0.0647	0.1877
15.12.2020	19	0.5157	0.0715	0.4128	0.0194	0.0226	0.0033
24.12.2020	28	0.4410	0.0590	0.5000	0.0888	0.0015	0.0873
02.01.2021	37	0.4433	0.0591	0.4976	0.0710	0.0016	0.0726
11.01.2021	46	0.4529	0.0607	0.4864	0.0614	0.0012	0.0602
20.01.2021	55	0.4630	0.0626	0.4743	0.0592	0.0110	0.0702
29.01.2021	64	0.4734	0.0647	0.4618	0.0320	0.0249	0.0570
07.02.2021	73	0.4839	0.0670	0.4491	0.0119	0.0299	0.0418
16.02.2021	82	0.4945	0.0693	0.4362	0.0410	0.0212	0.0622
25.02.2021	91	0.5050	0.0717	0.4233	0.0908	0.0303	0.0605
06.03.2021	100	0.5148	0.0740	0.4112	0.0548	0.0395	0.0152
15.03.2021	109	0.5158	0.0740	0.4102	0.0326	0.0471	0.0796
24.03.2021	118	0.5190	0.0743	0.4068	0.0014	0.0513	0.0499
02.04.2021	127	0.5225	0.0745	0.4030	0.1337	0.0511	0.0825

11.04.2021	136	0.5260	0.0748	0.3991	0.0676	0.0492	0.0184
20.04.2021	145	0.4822	0.0665	0.4514	0.0608	0.0363	0.0246
29.04.2021	154	0.4229	0.0560	0.5211	0.0498	0.0265	0.0233
08.04.2021	163	0.3638	0.0464	0.5898	0.0032	0.0077	0.0109
17.04.2021	172	0.3136	0.0387	0.6477	0.0903	0.0176	0.1079

Çizelge 5.11 SIR Modeli için test kümesinden elde edilen ileri beslemeli yapay sinir ağı çözümleri

Tarih	*t*	*S(t)*	*I(t)*	*R(t)*	*E(S)*	*E(I)*	*E(R)*
17.05.2021	175	0.3050	0.0375	0.6575	0.2145	0.0024	0.2169
18.05.2021	176	0.3021	0.0371	0.6608	0.2296	0.0044	0.2340
19.05.2021	177	0.2993	0.0367	0.6640	0.1817	0.0071	0.1888
20.05.2021	178	0.2964	0.0364	0.6672	0.1991	0.0077	0.2068
21.05.2021	179	0.2936	0.0360	0.6704	0.0687	0.0012	0.0675
22.05.2021	180	0.2908	0.0356	0.6736	0.1061	0.0004	0.1064
23.05.2021	181	0.2880	0.0352	0.6768	0.1247	0.0033	0.1280



Şekil 5.5 *SIR* modelinin MADS artık yapay sinir ağı çözümleri

Aynı sistem için MADS algoritması ile eğitilen artık yapay sinir ağının ürettiği nümerik çözümler Çizelge 5.8 ve Çizelge 5.9 ile özetlenmiştir. Bu modelde ağda yine yedi adet ara katman var olup bu ara katmanlardaki nöron sayısı sırasıyla 10, 6, 3, 10, 6, 3 ve 1'dir. Artık yapay sinir ağı modelinde atlama indeksi 3 olarak alınmıştır. Yani her katman kendisinden sonraki üçüncü katmana bağlantılıdır. Bu durum ağdaki bilinmeyen parametre sayısının artmasına ve dolayısıyla uygun çözümün belirlenmesinde güçlüklere neden olur. Ağın istenilen düzeyde eğitilebilmesi için daha çok örneğe veya iterasyona bağlı eğitim süresinin arttırılmasına ihtiyaç duyulur. Ancak tezde ağ modellerinin sağlıklı karşılaştırılabilmesi için maksimum iterasyon sayısı çok katmanlı algılayıcı modellerindeki ile aynı, 1000 olarak alınmıştır.

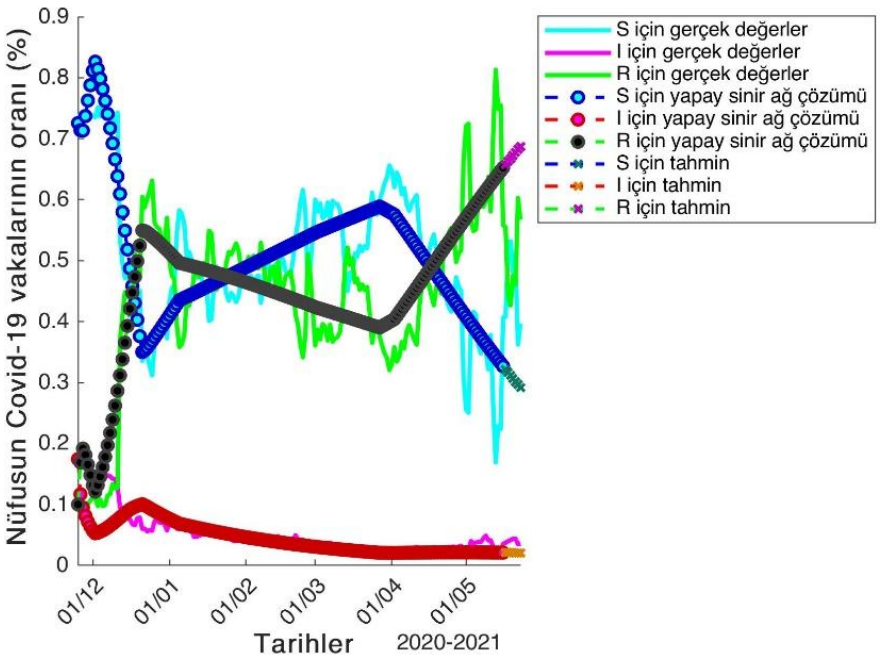
Çizelge 5.12 *SIR* Modeli için eğitim kümesinden elde edilen artık yapay sinir ağı çözümleri

Tarih	*t*	*S(t)*	*I(t)*	*R(t)*	*E(S)*	*E(I)*	*E(R)*

27.11.2020	1	0.7255	0.1744	0.1001	0	0	0
06.12.2020	10	0.7987	0.0555	0.1458	0.0622	0.0998	0.0375
15.12.2020	19	0.5797	0.0820	0.3382	0.0834	0.0121	0.0714
24.12.2020	28	0.3529	0.0985	0.5486	0.0007	0.0380	0.0387
02.01.2021	37	0.4041	0.0785	0.5174	0.0317	0.0210	0.0528
11.01.2021	46	0.4429	0.0648	0.4923	0.0715	0.0054	0.0661
20.01.2021	55	0.4608	0.0570	0.4823	0.0569	0.0054	0.0623
29.01.2021	64	0.4789	0.0500	0.4711	0.0376	0.0102	0.0478
07.02.2021	73	0.4972	0.0438	0.4590	0.0252	0.0068	0.0319
16.02.2021	82	0.5154	0.0383	0.4462	0.0619	0.0097	0.0522
25.02.2021	91	0.5335	0.0335	0.4330	0.0623	0.0079	0.0702
06.03.2021	100	0.5508	0.0293	0.4199	0.0187	0.0052	0.0239
15.03.2021	109	0.5659	0.0256	0.4085	0.0826	0.0013	0.0813
24.03.2021	118	0.5807	0.0224	0.3968	0.0603	0.0005	0.0598
02.04.2021	127	0.5799	0.0207	0.3994	0.0762	0.0027	0.0790
11.04.2021	136	0.5351	0.0210	0.4439	0.0585	0.0046	0.0632
20.04.2021	145	0.4827	0.0213	0.4959	0.0603	0.0088	0.0692
29.04.2021	154	0.4307	0.0215	0.5479	0.0421	0.0080	0.0501
08.04.2021	163	0.3801	0.0214	0.5985	0.0195	0.0173	0.0022
17.04.2021	172	0.3319	0.0211	0.6470	0.1086	0	0.1086

Çizelge 5.13 *SIR* Modeli için test kümesinden elde edilen artık yapay sinir ağı çözümleri

Tarih	*t*	*S(t)*	*I(t)*	*R(t)*	*E(S)*	*E(I)*	*E(R)*
17.05.2021	174	0.3217	0.0210	0.6574	0.0859	0.0160	0.1019
18.05.2021	175	0.3166	0.0209	0.6625	0.2029	0.0189	0.2218
19.05.2021	176	0.3115	0.0209	0.6676	0.2202	0.0206	0.2408
20.05.2021	177	0.3066	0.0208	0.6726	0.1744	0.0230	0.1975
21.05.2021	178	0.3016	0.0208	0.6776	0.1940	0.0232	0.2172
22.05.2021	179	0.2967	0.0207	0.6826	0.0656	0.0141	0.0797
23.05.2021	180	0.2919	0.0207	0.6875	0.1050	0.0153	0.1203



Şekil 5.6 *SIR* modelinin MADS artık yapay sinir ağı çözümleri

6. TARTIŞMA VE SONUÇ

Bu tezde epidemi problemlerinin çözümleri için literatürde tanıtılan SI, SIS ve SIR matematiksel modelleri kullanılarak COVID-19 pandemisinin gidişatını tahminlemek için yapay sinir ağı modelleri önerilmiştir. Çok katmanlı algılayıcı ve artık yapay sinir ağlarının kullanıldığı çalışmada ağın eğitimi için T.C. Sağlık Bakanlığı tarafından ilan edilen ve gerçekte gözlemlenen günlük vaka sayıları kullanılmıştır. Böylece SI, SIS ve SIR modelleri ile uyumlu ve gerçek gözlem değerlerine bağımlı modeller oluşturulmaya çalışılmıştır. Literatürde yapay sinir ağı kullanarak COVID-19 verilerini tahminlemeye çalışan yaklaşımların hepsi epidemik modellerden bağımsız olarak çözüm üretmektedir. Yani ağın maliyet fonksiyonu sadece gözlemlenen veri ile ağın ürettiği çıktının kareleri cinsinden ifade edilir. Bu da mevcut çalışmamızı diğerlerinden ayıran en temel unsurdur.

Önerilen ağların eğitimi için ise doğrusal olmayan optimizasyon problemlerinde kullanılan ve türev içermeyen bir optimizasyon yöntemi olan Örgü Uyarlamalı Doğrudan Arama yöntemi kullanılmıştır. Örgü Uyarlamalı Doğrudan Arama yönteminin tercih edilme nedeni problemin çok sayıda bilinmeyen parametre içermesidir. Kullanılan yöntemle bu parametrelere bağlı maliyet fonksiyonlarının kısmi türevlerinin hesaplanmasına gerek kalmamıştır.

Gerçek veriler üzerinde uygulanan deneysel çalışmalardan elde edilen temel bulgu, SI ve SIS modelleri ile gerçekleştirilen çalışmalarda önerilen ağ modelleri hem gözlemlenen verilerle hem de matematiksel epidemi modeliyle uyumludur. Ağın ürettiği çözümüm hem SI ve SIS modellerinin çözümlerine yakınsadığı hem de gerçek pozitif vaka oranı ve hasta birey oranına uyum sağladığı gözükmektedir. SI ve SIS modellerinde eğitim aşamasında kullanılan veriler üzerinden elde edilen mutlak hata oranı yaklaşık %1 civarında kalmaktayken, test aşamasında bu oran %3 ila %5 arasında değişmektedir. Elde edilen sonuçlar, maksimum iterasyon sayısı 1000 alınarak derlenmiştir. Pek çok tahminleme problemi için elde edilen hata miktarları oldukça düşük gözükse de COVID-19 gibi tüm insanlığın hayatını olumsuz etkileyen bir problemde bu hata payının daha çok düşürülmesi

gerekmektedir. Bu nedenle Örgü Uyarlamalı Doğrudan Arama yönteminin ağın eğitiminde etkisi yeterli düzeyde görülmemektedir. Çünkü ağın eğitimi için kullanılan maksimum sayısı değiştirilse bile maliyet fonksiyonun değerinde daha fazla bir düşüş sağlanamadığı gözlenmiştir. Bunun temel nedeni ise Örgü Uyarlamalı Doğrudan Arama yönteminin çözüm uzayının tamamını tarayacak bir yeteneğe sahip olmamasıdır. Algoritma sezgisel olarak üretilen bir örgüyü alt çerçevelere bölerek alt çerçeve üzerinde arama gerçekleştirir. Her iterasyonda elde ettiği çözümün içinde bulunduğu çerçeveyi daraltarak daha iyi bir çözüme ulaşmaya çalışır. Bu da başlangıçta rassal olarak örgü üzerinde konumlandırılan baz vektörlerinin belli bir doğrultuda hareket ettirilmesiyle gerçekleştirilir. Haliyle baz vektörlerinin başlangıçtaki konumlarının çözüme ulaşmadaki etkisi oldukça fazladır. Özellikle yüksek boyutlu problemlerde, başlangıçta kötü konumlandırılmış baz vektörleri ile en uygun çözüme ulaşmak oldukça güçtür. Dolayısıyla önerilen ağların eğitimi için daha farklı optimizasyon algoritmalarının problem çözümü üzerindeki etkisi incelenmelidir.

Çalışmada elde edilen bir diğer bulgu, SIR modeli ile entegre çalışacak şekilde önerilen yapay sinir ağı modeli, hasta birey oranlarını belirlemede başarılı iken popülasyondaki pozitif vaka oranları ile iyileşen birey oranlarını belirlemede yetersiz kalmıştır. T.C. Sağlık Bakanlığı tarafından ilan edilen verilerde anlık olarak büyük miktarda düşme ve yükselme olduğu gözlemlenen günlerde ağın bu ani değişimlere uyum sağlamada geciktiği söylenebilir. Bunun temel nedeninin ise ağın ağırlıklı olarak matematik modelin davranışına uyum sağlamaya çalıştığı olduğu düşünülmektedir.

Bir diğer önemli husus, problemde pozitif vaka sayısındaki artış oranı (beta) ve iyileşen birey oranı (gamma) günlük veriler baz alınarak En Küçük Kareler Yöntemiyle belirlenmiştir. Bu değerler ağın eğitimi sürecinde sabit olarak alınmıştır. Ancak gerçekte gözlemlenen değerler gösteriyor ki SI, SIS ve SIR modellerinde tanımlanan ve ağın maliyet fonksiyonun değerinin belirlenmesinde kullanılan bu oranlar anlık olarak değişebilmektedir. Bu da SIR modelinin başarı düzeyinin neden sınırlı kaldığını açıklamaktadır. Bu probleme çözüm önerisi olarak, beta ve gamma katsayılarının zamana bağlı olarak değişen birer fonksiyon olarak ifade edilmesi gerektiği düşünülmektedir. Bu amaçla, beta ve gamma katsayıları için günlük verileri baz olarak, örneğin bir haftalık belli zaman periyotları içinde değişen birer interpolasyon polinomu oluşturulabileceği düşünülmektedir. Doğal olarak, bu da epidemik modellerin çözümü için literatürde önerilen sabit katsayılı lineer olmayan adi diferansiyel denklem sistemlerinin değişken katsayılı hale dönüştürülmesi anlamına gelir.

Haliyle problemin çözümü de zorlaşır. Bununla birlikte gerçek veriye daha çok uyumlu bir model önerilmiş olur.

Çalışmada ağın sadece bir günlük veriyi girdi olarak aldığı modeller önerilmiştir. Eğer ağ belli bir zaman dilimindeki verileri girdi olarak kabul edip bu zaman dilimini takip eden gündeki tahmini verileri çıktı verecek şekilde modellenirse probleme daha başarılı bir çözüm getirilebileceği düşünülmektedir. Bu durumda zaman serisi tahminleme problemine dönüşecek olan problem için LSTM (Uzun-Kısa Vadeli Bellek, Long-Short Term Memory) ağları gibi daha farklı ağ modelleri kullanılabilir.

Eğer hem gerçekte gözlemlenen verilerle hem de SIR modeli ile uyumlu bir yapay sinir ağı modeli önerilebilirse $SEIR$ modeline geçiş sağlanabilir. Üstelik bu çözüm yaklaşımları aşılamanın popülasyon üzerindeki etkisinin belirlenmesi problemine genişletilebilir.

KAYNAKLAR

Aarts L. P. and Van Der Veer P. (2001), Neural Network Method for Solving Partial Differential Equations, Neural Process. Lett. 14, 3, 261-271.

Aminu, M., Ahmad, N. A., & Mohd Noor, M. H. (2021). Covid-19 detection via deep neural network and occlusion sensitivity maps. Alexandria Engineering Journal, 60(5), 4829–4855. https://doi.org/10.1016/j.aej.2021.03.052

Attaullah, & Sohaib, M. (2020). Mathematical modeling and numerical simulation of HIV infection model. Results in Applied Mathematics, 7, 100118. https://doi.org/10.1016/j.rinam.2020.100118

Audet, C., Hare, W. (2017). Derivative-Free and Blackbox Optimization. Springer Series in Operations Research and Financial Engineering, Springer International Publishing, doi: 10.1007/978-3-319-68913-5

Bernoulli, Daniel (1760). Essai d'une nouvelle analyse de la mortalité causée par la petite verole et des avantages de l'inoculation pour la prevenir. Mem. Math. Phys. Acad. Roy. Sci., Paris, 1–45. In Histoire de l'Academie Royale des Sciences, 1766.

Bohner, M., Streipert, S., & Torres, D. F. M. (2019). Exact solution to a dynamic SIR model. Nonlinear Analysis: Hybrid Systems, 32(December), 228–238. https://doi.org/10.1016/j.nahs.2018.12.005

Brauer, F. (2017). Mathematical epidemiology: Past, present, and future. Infectious Disease Modelling, 2(2), 113–127. doi: 10.1016/j.idm.2017.02.001

Brauer, F., Castillo-Chavez, C., & Feng, Z. (2019). Correction to: Mathematical Models in Epidemiology. https://doi.org/10.1007/978-1-4939-9828-9_17

Brauer, F., Castillo-Chavez, C., Feng, Z. (2019). Mathematical Models in Epidemiology, Texts in Applied Mathematics Series, Springer-Verlag New York. oi: 10.1007/978-1-4939-9828-9

Chandru, M., Das, P., & Ramos, H. (2018). Numerical treatment of two-parameter singularly perturbed parabolic convection diffusion problems with non-smooth data. Mathematical Methods in the Applied Sciences. https://doi.org/10.1002/mma.5067

Chimmula, V. K. R., & Zhang, L. (2020). Time series forecasting of COVID-19 transmission in Canada using LSTM networks. Chaos, Solitons and Fractals, 135, 109864. https://doi.org/10.1016/j.chaos.2020.109864

Cooper, I., Mondal, A., & Antonopoulos, C. G. (2020). A SIR model assumption for the spread of COVID-19 in different communities. Chaos, Solitons and Fractals, 139, 110057. https://doi.org/10.1016/j.chaos.2020.110057

Çetin, E., Kiremitci, B., Yurt, İ. D., (2009). Matematiksel Epidemiyoloji: Pandemik A/H1N1 Gribi Vakası, Istanbul University Journal of the School of Business Administration, 38(2), 197-209.

From百度文库. (2013). 済無No Title No Title. Journal of Chemical Information and Modeling, 53(9), 1689–1699.

Gao, S., Teng, Z., Nieto, J. J., & Torres, A. (2007). Analysis of an SIR epidemic model with pulse vaccination and distributed time delay. Journal of biomedicine & biotechnology, 2007, 64870. https://doi.org/10.1155/2007/64870

Goh, G. (2020). Epidemic Calculator, Erişim Adresi: https://gabgoh.github.io/COVID/index.html. Erişim Tarihi: 29.06.2020.

Gondim, J. A. M. (2021). Preventing epidemics by wearing masks: An application to COVID-19. Chaos, Solitons and Fractals, 143, 110599. https://doi.org/10.1016/j.chaos.2020.110599

Gölgeli, M. (2019). A mathematical model of Hepatitis B Transmission in Turkey. Commun. Fac. Sci. Univ. Ank. Ser. A1 Math.Stat. 68(2), 1586–1595. doi: 10.31801/cfsuasmas.544126

Gör, İ. (2020). Diferansiyel denklemlerin yapay sinir ağları ile nümerik çözümleri, Doktora Tezi, Aydın Adnan Menderes Üniversitesi, Fen Bilimleri Enstitüsü, Matematik Anabilim Dalı, Aydın, Danışman: Dr. Öğr. Üyesi Korhan Günel.

Güzel, S.C. (2012) SIR modelin kararlılık analizi, Yüksek Lisans Tezi, Selçuk Üniversitesi, Fen Bilimleri Enstitüsü, Matematik Anabilim Dalı, Konya, Danışman: Dr. Öğr. Üyesi Necati Taşkara.

Harko, T., Lobo, F. S. N., & Mak, M. K. (2014). Exact analytical solutions of the Susceptible-Infected-Recovered (SIR) epidemic model and of the SIR model with equal death and birth rates. Applied Mathematics and Computation, 236(March), 184–194. https://doi.org/10.1016/j.amc.2014.03.030

Harko, T., Lobo, F. S. N., & Mak, M. K. (2014). Exact analytical solutions of the Susceptible-Infected-Recovered (SIR) epidemic model and of the SIR model with equal death and birth rates. Applied Mathematics and Computation, 236, 184–194. https://doi.org/10.1016/j.amc.2014.03.030

Kröger, M., & Schlickeiser, R. (2020). Analytical solution of the SIR-model for the temporal evolution of epidemics. Part A: time-independent reproduction factor. Journal of Physics A: Mathematical and Theoretical, 53(50). https://doi.org/10.1088/1751-8121/abc65d

Kucharski, A.J. et. al (2020). Early dynamics of transmission and control of COVID-19: a mathematical modelling study, The Lancet Infectious Diseases, 20(5), 553-558, doi: 10.1016/S1473-3099(20)30144-4.

Lagaris, I. E., Likas, A. and Fotiadis, D. I. (1998), Artificial neural networks for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks, 9 (5) , 987-1000 .

Lakhmiri, D., Digabel, S. Le, & Tribes, C. (2019). HyperNOMAD: Hyperparameter optimization of deep neural networks using mesh adaptive direct search. ArXiv, 1–30.

Li-ying, X., Hui, W., Zhe-zhao, Z. (2007), The algorithm of neural networks on the initial value problems in ordinary differential equations, Industrial Electronics and Applications.

Malek, A. and Beidokhti, R.S. (2006), Numerical solution for high order differential equations using a hybrid neural network-optimization method, Applied Mathematics and Computation, 183 : 260-271.

Meade, A. J., Fernandez, A. A. (1994), The numerical solution of linear ordinary differential equations by feedforward neural networks, Mathematical and Computer Modelling, 20: 191-257.

Panwar, H., Gupta, P. K., Siddiqui, M. K., Morales-Menendez, R., & Singh, V. (2020). Application of deep learning for fast detection of COVID-19 in X-Rays using nCOVnet. Chaos, Solitons and Fractals, 138, 109944. https://doi.org/10.1016/j.chaos.2020.109944

Salgotra, R., Gandomi, M., & Gandomi, A. H. (2020). Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID- 19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information, (January).

Shabbir, G., Khan, H., & Sadiq, M. A. (2010). A note on Exact solution of SIR and SIS epidemic models, (January 2014). Retrieved from http://arxiv.org/abs/1012.5035

Umar, M., Sabir, Z., Raja, M. A. Z., Amin, F., Saeed, T., & Guerrero-Sanchez, Y. (2021). Integrated neuro-swarm heuristic with interior-point for nonlinear SITR model for dynamics of novel COVID-19. Alexandria Engineering Journal, 60(3), 2811–2824. https://doi.org/10.1016/j.aej.2021.01.043

Wu, J.T., Leung, K., Leung, G.M. (2020). Nowcasting and forecasting the potential domestic and international spread of the 2019-nCoV outbreak originating in Wuhan, China: a modelling study, The Lancet, 395(10225), 689-697, doi: 10.1016/S0140-6736(20)30260-9.

Zeroual, A., Harrou, F., Dairi, A., & Sun, Y. (2020). Deep learning methods for forecasting COVID-19 time-Series data: A Comparative study. Chaos, Solitons and Fractals, 140, 110121. https://doi.org/10.1016/j.chaos.2020.110121

EKLER

Ek- 1 T.C. Sağlık Bakanlığı tarafından ilan edilen Covid verisini çeken betik

T.C. Sağlık Bakanlığı'nın https://covid19.saglik.gov.tr/TR-66935/genel-koronavirus-tablosu.html web sayfasından günlük Covid verilerini çekmek için yazılan fonksiyon kodu aşağıda sunulmuştur. Fonksiyon çağrısı belirtilen web adresini url parametersi olarak alır ve geriye fonksiyon çağrısının yapıldığı tarih ile birlikte bir veri tablosu halinde covid verisini geri döndürür.

```
function [covidData_tr, tarih] = getCovidData(url)
tic
%% Grab the data
options =
weboptions('Timeout',Inf,'RequestMethod','get');
htmlFile = webread(url,options);
%% Extract Covid data from htmlFile
searchPattern ='geneldurumjson = [';
[~,endIndex] = regexp(htmlFile,searchPattern);
txtData = htmlFile(endIndex+1:end);
searchPattern =']];//]]>';
[startIndex,~]  = regexp(txtData,searchPattern);
txtData = txtData(1:startIndex-1);

% Parse data
expression = '\{*\}';
dailyData = regexp(txtData,expression,'split');
n = length(dailyData);

%% Generate a struct for Covid-19 Data
empty_covidData.tarih = [];
empty_covidData.gunluk_test_sayisi = [];
empty_covidData.gunluk_vaka_sayisi = [];
empty_covidData.gunluk_hasta_sayisi = [];
empty_covidData.gunluk_vefat_sayisi = [];
empty_covidData.gunluk_iyilesen_sayisi = [];
empty_covidData.toplam_test_sayisi = [];      % S
empty_covidData.toplam_vaka_sayisi = [];      % E
empty_covidData.toplam_hasta_sayisi = [];     % I
empty_covidData.toplam_vefat_sayisi = [];     % D
empty_covidData.toplam_iyilesen_sayisi = [];  % R
empty_covidData.toplam_yogun_bakim_sayisi = [];
empty_covidData.toplam_entube_sayisi = [];
empty_covidData.hastalarda_zaturre_orani = [];
empty_covidData.agir_hasta_sayisi = [];
empty_covidData.yatak_doluluk_orani = [];
empty_covidData.eriskin_yogun_bakim_doluluk_orani =
[];
```

```
36      empty_covidData.ventilator_doluluk_orani = [];
37      empty_covidData.ortalama_filyasyon_suresi = [];
38      empty_covidData.ortalama_temasli_tespit_suresi = [];
39      empty_covidData.filyasyon_orani = [];

40      covidData_tr = repmat(empty_covidData,n-1,1);

41      for i=1:n-1
42          str = regexp(dailyData{i},'"*"','split');
43          for j=1:length(str)-2
44              if strcmp(str{j},':') || strcmp(str{j},',')
45      || strcmp(str{j},'{')
46                  continue;
47              end
48              if strcmp(str{j+2},',') || isempty(str{j+2})
49                  continue;       % Missing data
50              end
51              if strcmp(str{j},'tarih')
                    covidData_tr(i).tarih =
        datetime(str{j+2},'TimeZone','local','Format','dd.MM
        .yyy');
52              else

53                  temp = regexprep(str{j+2},'[.]','');
54      % Decimal point format
55                  temp = regexprep(temp,'[,]','.');
56                  if contains(temp,'saat')
                        temp =temp(1:end-5);
57                  end
58                   if contains(temp,'\t')
                        temp =temp(1:end-2);
59                  end

60                  num = str2double(temp);
61                  switch str{j}
62                      case 'gunluk_test'

63      covidData_tr(i).gunluk_test_sayisi = num;
64                      case 'gunluk_vaka'

65      covidData_tr(i).gunluk_vaka_sayisi = num;
66                      case 'gunluk_hasta'

67      covidData_tr(i).gunluk_hasta_sayisi = num;
68                      case 'gunluk_vefat'

69      covidData_tr(i).gunluk_vefat_sayisi = num;
70                      case 'gunluk_iyilesen'

71      covidData_tr(i).gunluk_iyilesen_sayisi = num;
72                      case 'toplam_test'      % S

73      covidData_tr(i).toplam_test_sayisi = num;
74                      case 'toplam_hasta'   % I
```

```
75      covidData_tr(i).toplam_hasta_sayisi = num;
76                      case 'toplam_vefat'  % D

        covidData_tr(i).toplam_vefat_sayisi = num;
77                      case 'toplam_iyilesen'   % R
78
        covidData_tr(i).toplam_iyilesen_sayisi = num;
79                      case 'toplam_yogun_bakim'
80
        covidData_tr(i).toplam_yogun_bakim_sayisi = num;
81                      case 'toplam_entube'
82
        covidData_tr(i).toplam_entube_sayisi = num;
83                      case 'hastalarda_zaturre_orani'
84                          if num>100
85                              num=num/10;
                            end
86
        covidData_tr(i).hastalarda_zaturre_orani = num;
87                      case 'agir_hasta_sayisi'
88
        covidData_tr(i).agir_hasta_sayisi = num;
89                      case 'yatak_doluluk_orani'
90                          if num>100
91                              num=num/10;
                            end
92
        covidData_tr(i).yatak_doluluk_orani = num;
93                      case
94      'eriskin_yogun_bakim_doluluk_orani'
95                          if num>100
96                              num=num/10;
                            end
97
        covidData_tr(i).eriskin_yogun_bakim_doluluk_orani =
98      num;
                        case 'ventilator_doluluk_orani'
99                           if num>100
100                             num=num/10;
101                         end
102
        covidData_tr(i).ventilator_doluluk_orani = num;
103                     case 'ortalama_filyasyon_suresi'
104
        covidData_tr(i).ortalama_filyasyon_suresi = num;
105                     case
106     'ortalama_temasli_tespit_suresi'
107
        covidData_tr(i).ortalama_temasli_tespit_suresi =
108     num;
                        case 'filyasyon_orani'
```

```
109                         if num>100
110                             num=num/10;
111                         end
112     covidData_tr(i).filyasyon_orani = num;
113                 end
114             end
115         end
116     end
117
        %% Toplam Vaka Sayýlarýný Hesapla
118     X = [covidData_tr(:).gunluk_vaka_sayisi];
119     %X = X(end:-1:1);
120     Y = cumsum(X);
121     Y = Y(end:-1:1);
122     for i=1:length(Y)
123         covidData_tr(i).toplam_vaka_sayisi = Y(i);
124     end
125
        % Convert struct to data table
126     covidDataTable_tr = struct2table(covidData_tr);
127
        %% Report
128     dates = [covidData_tr(:).tarih];
129     S_gun = [covidData_tr(:).gunluk_test_sayisi];
130     fprintf('Daily records for S(Suspected):
131     %d\n',length(S_gun));      % Suspected
132     E_gun = [covidData_tr(:).gunluk_vaka_sayisi];
133     fprintf('Daily records for E(Exposed):
134     %d\n',length(E_gun));        % Exposed
135     I_gun = [covidData_tr(:).gunluk_hasta_sayisi];
136     fprintf('Daily records for I(Infected):
137     %d\n',length(I_gun));       % Infected
138     R_gun = [covidData_tr(:).gunluk_iyilesen_sayisi];
139     fprintf('Daily records for R(Recovered):
140     %d\n',length(R_gun));      % Recovered
141     D_gun = [covidData_tr(:).gunluk_vefat_sayisi];
142     fprintf('Daily records for D(Deceased):
143     %d\n',length(D_gun));       % Deceased
144
        fprintf('\n');
145     S = [covidData_tr(:).toplam_test_sayisi];
146     fprintf('Total records for S(Suspected):
147     %d\n',length(S));      % Suspected
148     E = [covidData_tr(:).toplam_vaka_sayisi];
149     fprintf('Total records for E(Exposed):
150     %d\n',length(E));        % Exposed
151     I = [covidData_tr(:).toplam_hasta_sayisi];
152     fprintf('Total records for I(Infected):
153     %d\n',length(I));       % Infected
154     R = [covidData_tr(:).toplam_iyilesen_sayisi];
155     fprintf('Total records for R(Recovered):
156     %d\n',length(R));      % Recovered
157     D = [covidData_tr(:).toplam_vefat_sayisi];
158     fprintf('Total records for D(Deceased):
```

```
159     %d\n',length(D));      % Deceased
160
        formatOut = 'dd mmmm yyyy';
161     %tarih = datestr(datetime('now'),formatOut,'local');
162     tarih = datestr(datetime('now'),formatOut);
163     save(['data\covidData_tr_' tarih
164     '.mat'],'covidData_tr','tarih')
165
        fprintf('\nThe confirmed Covid Data is gathered from
166     Turkey Ministery of Health on date : %s\n',tarih);
167     toc
168
```

Ek- 2 En Küçük Kareler Doğrusu yöntemine ilişkin betik

```
 1 function [A,B] = leastSquares(x,y)
 2 % (x,y) gözlem degerleri verildiginde y = f(x) = Ax + B
 3 % formundaki f fonksiyonun katsayılarını
 4 % tespit eder.
 5 % x ve y satır vektörü olarak girilmelidir.
 6   n = length(x); % Gözlem sayısı
 7   C = [sum(x.*x) sum(x); sum(x) n]\[sum(x.*y); sum(y)];
 8   A = C(1);
 9   B = C(2);
10 end
```

Ek- 3 MADS algoritması ile eğitilen Çok Katmanlı Algılayıcı ve Artık Yapay Sinir Ağı modellerini üreten kodlar

```
1  function y = Net(x,p)
2  m = floor((length(p)-1)/3);
3  n = length(x);
4  alpha = p(1:m);
5  w = p(m+1:2*m);
6  bias = p(2*m+1:3*m);
7  bias2 = p(3*m+1);

8  y = zeros(1,n);
9  z = zeros(m,n);
10 for j=1:n
11     for i=1:m
12         z(i,j) = w(i)*x(j)+bias(i);
13         y(j) = y(j) + alpha(i)*sigma(z(i,j)) + bias2;
```

```
    end
end
```

```
function y = dNet(x,p)
m = floor((length(p)-1)/3);
n = length(x);
alpha = p(1:m);
w = p(m+1:2*m);
bias = p(2*m+1:3*m);

y = zeros(1,n);
z = zeros(m,n);
for j=1:n
    for i=1:m
        z(i,j) = w(i)*x(j)+bias(i);
        y(j) = y(j) + alpha(i)*w(i)*sigma(z(i,j))*(1-sigma(z(i,j)));
    end
end
```

```
function [y] = trialSolution(net, inputs, a, A, B)
% y = A + (x-a).*Net(x,p);
  outs = simulateNet(net, inputs);
  y = [A + (inputs - a).*outs(1,:);
       B + (inputs - a).*outs(2,:)];
  y = y./sum(y,1);
end
```

```
function [y] = DtrialSolution(net, inputs, a)
%y = Net(x,p) + (x-a).*dNet(x,p);
outs = simulateNet(net, inputs);
z = outs./sum(outs,1);
% Complex Step Derivation
p = length(inputs);
h = 1e-150;
diffS = zeros(1,p);
diffI = zeros(1,p);
for j = 1:p
    z2 = simulateNet(net, inputs(j) + h*sqrt(-1));
    diffS(j) = z2(1);
    diffI(j) = z2(2);
end

fS = mean(real(diffS));
gS = imag(diffS)/h;

fI = mean(real(diffI));
gI = imag(diffI)/h;
```

```
y = [z(1,:) + (inputs - a).*gS;
     z(2,:) + (inputs - a).*gI];
end
```

```
function [S_exact,I_exact] = exactSolution(t, i0, N, beta)
   I_exact = i0*N./(i0 + (N - i0)*exp(-beta*(t-1)));
   S_exact = N - I_exact;
end
```

```
function out = f(S,I,N,beta)
  out = - beta*S.*I/N;
end
```

```
function out = g(S, I, N, beta)
  out =  beta*I.*(1-I/N);
end
```

```

        function E = CostFunction(net, inputs, S_train, I_train, a, s0,
        i0, N, beta, nDaysForRealData, layerIndex, p)

         veriSay = length(S_train);
         net = setLayerWeights(net, layerIndex, p);

        selectedIndexForRealData = net.selectedIndexForRealData;
        selectedIndexForSIModel = net.selectedIndexForSIModel;

        % Train set for Real Data
        inputsReal = inputs(selectedIndexForRealData);
        S_real = S_train(inputsReal);
        I_real = I_train(inputsReal);
        out = trialSolution(net,inputsReal,a,s0,i0);
        trialS_realData = out(1,:); trialI_realData = out(2,:);

        % Train Set for SI Model
        inputs = inputs(selectedIndexForSIModel);
        out = trialSolution(net, inputs, a, s0, i0);
        S = out(1,:); I = out(2,:);
        dout = DtrialSolution(net, inputs, a);
        dS = dout(1,:); dI = dout(2,:);

         F = f(S,I,N,beta);
         G = g(S,I,N,beta);
         E = 0.002*( sum((dS - F).^2)/length(inputs) + sum((dI -
        G).^2)/length(inputs));
         E = E + 0.5*(sum((trialI_realData -
        I_real).^2)/length(inputsReal) + sum((trialS_realData -
```

```
S_real).^2)/length(inputsReal));% + 0.5*sum(p.^2);
```

```
Function displayResults(x,y1,y2,y1_t,y2_t,caption,label, params)
  n = length(x);
  MSE_1 = sum((y1-y1_t).^2)/n;
  MSE_2 = sum((y2-y2_t).^2)/n;
  N = params.nPop; % # of individuals
  m = params.noNeurons; % # of neurons
  Low = params.low; % lower bound of search space
  Up = params.up;  % upper bound of search space
  max_it = params.Max_Iteration;  % Max number of iteration
  algo = params.optimAlgo;
  fileName='Results\table.tex';
  fileID=fopen(fileName,'w+');
  fprintf(fileID,'\\begin{table}[ht]\n');
  fprintf(fileID,['  \\caption{Parameters for ' algo '}\n']);
  fprintf(fileID,'  \\centering\n');
  fprintf(fileID,'  \\begin{tabular}{rr}\n');
  fprintf(fileID,'    \\hline\\hline\n');
  fprintf(fileID,'    %s & %s \n','Parameter','Value\\');
  fprintf(fileID,'    %s & %d\\ \n','Number of Agents',N);
  fprintf(fileID,'    %s & %d\\ \n','Number of neurons',m);
  fprintf(fileID,'    %s & [%d, %d]\\ \n','Range of search
space',Low,Up);
  fprintf(fileID,'    %s & %d\\ \n','Maximum Iteration',max_it);
  fprintf(fileID,'    \\hline\n');
  fprintf(fileID,'  \\end{tabular}\n');
  fprintf(fileID,'  \\label{%s}\n',label);
  fprintf(fileID,'\\end{table}\n');

  fprintf(fileID,'\\begin{figure}\n');
  fprintf(fileID,'  \\centering\n');
  fprintf(fileID,['  \\includegraphics[width=7cm]{figs\\Figure_' algo
'_trainSet.eps}\n'] );
  fprintf(fileID,['  \\caption{Numerical solution obtained via ' algo
'for training set}\\label{fig:' algo '_trainSet}\n']);
  fprintf(fileID,'\\end{figure}\n');

  fprintf(fileID,'\\begin{figure}\n');
  fprintf(fileID,'  \\centering\n');
  fprintf(fileID,['  \\includegraphics[width=7cm]{figs\\Figure_' algo
'_testSet.eps}\n']);
  fprintf(fileID,['  \\caption{Numerical solution obtained via ' algo
'for test set}\\label{fig:' algo '_testSet}\n']);
  fprintf(fileID,'\\end{figure}\n');

  fprintf(fileID,'\\begin{table}[ht]\n');
  fprintf(fileID,'  \\caption{%s}\n',caption);
  fprintf(fileID,'  \\centering\n');
  fprintf(fileID,'  \\begin{tabular}{rcccc}\n');
  fprintf(fileID,'    %s & %s & %s & %s & %s & %s
\n','$k$','$x_{k}$','$S(k)$','$I(k)$','$E_S$','$E_I$\\');
```

```
  fprintf(fileID,'    \\hline\\hline\n');
  fprintf('\n k   x(k)    S(k)      I(k)     S_test(k) I_test(k) E = |S(k) - S_test(k)| E = |I(k) - I_test(k)|\n');
  fprintf('-----------------------------------------\n');
  for k=1:n
      fprintf('%3d %3d %12.3e %6.3e %6.3e %9.3e  %12.3e %20.3e\n',k,x(k),y1(k), y2(k),y1_t(k),y2_t(k),abs(y1(k)-y1_t(k)),abs(y2(k)-y2_t(k)));
      fprintf(fileID,'    %3d & %3d & %12.3f & %9.3f &  %12.3e & %20.3e\\\\ \n',k,x(k),y1(k), y2(k),abs(y1(k)-y1_t(k)),abs(y2(k)-y2_t(k)));
  end
  fprintf('---------------------- ----------------\n\n');
  fprintf('Mean Squared Error for S : %1.3e\n',MSE_1);
  fprintf('Mean Squared Error for I : %1.3e\n',MSE_2);
  fprintf(fileID,'    \\hline\n');
    fprintf(fileID,'     \\multicolumn{6}{l}{\\textbf{Mean Squared Error for S :} %1.3e}\\\\ \n',MSE_1);
    fprintf(fileID,'     \\multicolumn{6}{l}{\\textbf{Mean Squared Error for I:} %1.3e}\\\\ \n',MSE_2);

  fprintf(fileID,'  \\end{tabular}\n');
  fprintf(fileID,'  \\label{%s}\n',label);
  fprintf(fileID,'\\end{table}\n');

    fclose(fileID);
end
```

```
function W = getLayerWeights(net,layerIndex)
% Returns the hidden layer weights combining the bias values as a row vector
W = net.Layers(layerIndex).Weights;
W = reshape(W,1,[]);
bias = [net.Layers(layerIndex).bias]';
W = [W bias];
```

```
function weights = getWeights(net)
weights=[];
L = length(net.Layers);

for k=1:L
    W = net.Layers(k).Weights;
    for j=1:size(W,1)
        weights = [weights  W(j,:)];
    end
    if isrow(net.Layers(k).bias)
        weights = [weights  net.Layers(k).bias];
    else
```

```
        weights = [weights  net.Layers(k).bias'];
    end
end
```

```
function weights = HeInitialization(m,n)
filterSize = [m n];
numChannels = 1;
numIn = filterSize(1) * filterSize(2) * numChannels;

varWeights = 2 / numIn;
weights = randn(m,n) * sqrt(varWeights);

end
```

```
function net = initializeResNet(inputs, params,CostFunction)
% This function returns a Residual Network
% inputs : 2-by-m input matrix
% M : Total # of neurons in each layer of resNet
% L : Number of hidden layers
% k : Skip index of the network
% VarMin : Lower bound for initial value of weights
% VarMax : Upper bound for initial value of weights
rng(0);
[nVar, n] = size(inputs);
VarMin =  params.low;
VarMax = params.up;
L = params.noHiddenLayers;
k = params.skipIndex;

emptyLayer.numberofNeurons = 0;      % Default
emptyLayer.Weights = [];             % Indicates the weights of the
  layer
emptyLayer.bias = [];
emptyLayer.activationFunction = @(z) relu(z);
emptyLayer.output = [];

net.inputs = inputs;
net.selectedIndexForRealData = params.selectedIndexForRealData;
net.selectedIndexForSIModel = params.selectedIndexForSIModel;
net.numberofHiddenLayer = L;
net.skipIndex = k;
net.VarMin = VarMin;
net.VarMax = VarMax;
net.CostFunction = CostFunction;
net.ErrorType = params.ErrorType;

net.Layers = repmat(emptyLayer,1,L+2);

%net.Layers(L+2).activationFunction = @(z) sigma(z);     % Sigmoid
%net.Layers(L+2).activationFunction = @(z) z.*sigma(z);   % Swish

% First Hidden Layer
```

```
net.Layers(1).numberofNeurons = params.noNeurons(1);
M = net.Layers(1).numberofNeurons;
%net.Layers(1).Weights = VarMin + (VarMax - VarMin)*rand(nVar,M); %
  Indicates the weights of the layer
%net.Layers(1).bias = VarMin + (VarMax - VarMin)*rand(M,1); %-
  1*ones(M,1);
net.Layers(1).Weights = HeInitialization(nVar,M);
net.Layers(1).bias = HeInitialization(M,1);

W = net.Layers(1).Weights;
for i=1:n
        X = inputs(:,i);
        z = W'*X + net.Layers(1).bias;
        net.Layers(1).output(1:M,i) =
  net.Layers(1).activationFunction(z);
end

% Hidden Layers
for ell = 2:L+1
    net.Layers(ell).numberofNeurons = params.noNeurons(ell);
    M1 = net.Layers(ell-1).numberofNeurons;
    M2 = net.Layers(ell).numberofNeurons;
    %net.Layers(ell).Weights = VarMin + (VarMax - VarMin)*rand(M1,M2);
    %net.Layers(ell).bias = VarMin + (VarMax - VarMin)*rand(M2,1);% -
  1*ones(M2,1);
    net.Layers(ell).Weights = HeInitialization(M1,M2);
    net.Layers(ell).bias = HeInitialization(M2,1);
    W = net.Layers(ell).Weights;
    for i=1:n
        X = net.Layers(ell-1).output(:,i);
        z = W'*X + net.Layers(ell).bias;
        if (ell>k) && mod(ell,k)==1
            net.Layers(ell).output(1:M2,i) =
net.Layers(ell).activationFunction(z + net.Layers(ell-
  k).output(1:M2,i));
        else
            net.Layers(ell).output(1:M2,i) =
net.Layers(ell).activationFunction(z);
        end
    end
end

% Output Layer
%net.Layers(L+2).activationFunction = @(x) softmax(x);
M1 = params.noOuts;     % # of outputs
net.Layers(L+2).numberofNeurons = M1;
M2 = net.Layers(L+1).numberofNeurons;
% net.Layers(L+2).Weights = VarMin + (VarMax - VarMin)*rand(M1,M2);
% net.Layers(L+2).bias = VarMin + (VarMax - VarMin)*rand(M1,1);% -
  1*ones(M1,1);
net.Layers(L+2).Weights = HeInitialization(M1,M2);
net.Layers(L+2).bias = HeInitialization(M1,1);% -1*ones(M1,1);
```

```
for i=1:n
    X = net.Layers(L+1).output(1:M2,i);
    W = net.Layers(L+2).Weights;
    z = W*X + net.Layers(L+2).bias;
    net.Layers(L+2).output(1:M1,i) =
net.Layers(L+2).activationFunction(z);
end
net.outputs =  net.Layers(L+2).output;
```

```
%% SI (Susceptible-Infected) modeli - Ana Program
% Problem Definition
% S hasta veya virüse bulaşmamış insan sayısı
% I ise virüse bulaşmı insan sayısı
% N = Popülasyondaki toplam birey sayısı
% S' = - beta*S*I/N
% I' = beta*I*(1-I/N)
% constraint : N = S + I

clear;clc;close;

exampleNo = 1;
url = 'https://covid19.saglik.gov.tr/TR-66935/genel-
koronavirus-tablosu.html';
global covidData_tr;
covidData_tr = getCovidData(url);

%% Prepare Data
veriSay = length([covidData_tr(:).toplam_vaka_sayisi]);
S = [covidData_tr(veriSay:-1:1).gunluk_vaka_sayisi];
% S
E = [covidData_tr(veriSay:-1:1).gunluk_test_sayisi];
% E
I = [covidData_tr(veriSay:-1:1).gunluk_hasta_sayisi];
% I
R = [covidData_tr(veriSay:-1:1).gunluk_iyilesen_sayisi];
% R
D = [covidData_tr(veriSay:-1:1).gunluk_vefat_sayisi];
% D
data = [S; E; I; R; D]';
dates = [covidData_tr(veriSay:-1:1).tarih];
t = 1:veriSay;

%% Normalize data
normalizedS = S./(S+I);
normalizedI = I./(S+I);
normalizedN = 1;

s0 = normalizedS(1);
i0 = normalizedI(1);
%beta = 1.0004;
%  beta = 1+getCovidDataByCities();
[days,ratios]=getCovidCasesRates();
[A,B]=leastSquares(days,ratios);
```

```
34 beta = 1 + A;

35 %% Prepare train and test data
36 nTestData = 7;   % for weekly prediction
37 nDaysForRealData = floor(veriSay-nTestData)/2;
38 inputs_train = 1:veriSay-nTestData;
39 inputs_test = veriSay-nTestData+1:veriSay;

40 countDays = length(inputs_train);
41 index = randperm(countDays);
42 selectedIndexForRealData =
43 index(2:floor(countDays*0.90));
44 selectedIndexForSIModel = index([1
45 floor(countDays*0.90)+1:end]);
46 selectedIndexForRealData =
   sort(selectedIndexForRealData);
47 selectedIndexForSIModel = sort(selectedIndexForSIModel);

48 S_train = normalizedS(inputs_train);
49 I_train = normalizedI(inputs_train);
50 S_real_train = normalizedS(inputs_train);
51 I_real_train = normalizedI(inputs_train);
52 S_real_test = normalizedS(inputs_test);
53 I_real_test = normalizedI(inputs_test);

54 %[S_exact, I_exact] = exactSolution(t, i0, normalizedN,
   beta);

55 %% Network and MADS Parameters
56 L = 7;                          % Number of hidden
57 layers
58 k = 3;                          % Skip index for resNet
59 M = [10 6 3];                   % Enter the number of
60 neurons for the first k hidden layers if k~=0
61 % k = 0;                        % If k=0, then Net is
   turned to MLP
62 % M = [10 6 6 4 4 3 3];         % If k = 0, enter the
   number of neurons in each layers
63 VarMin = -1;                    % Lower Bound of
   Variables
64 VarMax = 1;                     % Upper Bound of
   Variables
65 MaxIt = 1000;

66 params.nPop = normalizedN;
67 params.noOuts = 2;               % Number of outputs of
68 the ResNet
69 if k==0
70     M = [M params.noOuts];       % Insert # of neurons
   for the output layer
71 end
72 params.noNeurons = ones(1,L+1);
```

```
for i=1:L+1
    if k==0
       params.noNeurons(i) = M(i);
    else
        if mod(i,k)==0
            params.noNeurons(i) = M(k);
        else
            params.noNeurons(i) = M(mod(i,k));
        end
    end
end
params.noHiddenLayers = L;      % # of hidden layers
params.skipIndex = k;           % Skip index for
Residual Network
params.low = VarMin;            % lower bound of search
space
params.up =  VarMax;            % upper bound of search
space
params.dim = 10;                % dimension of search
space
params.Max_Iteration = MaxIt;   % Maximum number of
iteration

params.ErrorType = 1;           % Use 1 for L1 norm, 2
for L2 norm

params.selectedIndexForRealData =
selectedIndexForRealData;
params.selectedIndexForSIModel =
selectedIndexForSIModel;

params.optimAlgo = 'MADS';
params.noTours = 1;             % # of Tours in training
stage through output layer to input layer, and  input
layer to output layer

%% Stopping criterias
params.maxfeval = 10^4;         % Maximum number of
function evaluation
params.tolafun = 10^-12;        % The tolerance value
for absolute error
params.tolrfun = 1e-12;         % The tolerance value
for relative error
params.tolint = 1e-12;          % The tolerance value
for absolute errors between two successive  iterations
params.InitialMeshSize = 1;
params.MeshContractionFactor = 0.5;
params.MeshExpansionFactor = 1.5;
params.MeshTolerance = 10^-8;

maxTrials = 1;%25;               % Maximum # of trials
for cross validation

empty_solution.BestSol.Position.S = [];
```

```
106 empty_solution.BestSol.Position.I = [];
107 empty_solution.BestCost = [];
108 empty_solution.trainingTime = [];
109 empty_solution.testingTime = [];
110 empty_solution.trainMSE = [];
111 empty_solution.testMSE = [];
112 Solution=repmat(empty_solution,maxTrials,1);

113 CostFunction = @(p) Cost(net, inputs, S_train, I_train,
    a, s0, i0, N, beta, nDaysForRealData, layerIndex, p);

114 %% Main Loop
115 for trialID=1:maxTrials
116     T = tic;
117     %% Construct a residual network
118     net = initializeResNet(inputs_train, params,
    CostFunction);

119     fprintf('\n\nTrial Id : %d\n',trialID);
120     disp('-----------------------------------------------
    -----------------------------------------');
121     %% Training Neural Network
122     [net, E] = trainNet(net, inputs_train, S_train,
    I_train, t(1), s0, i0, normalizedN, beta,
    nDaysForRealData, params);

123     out = trialSolution(net, inputs_train, t(1), s0,
    i0);
124     S = out(1,:);       % Trial solutions for S using
    MADS optimization for training set
125     I = out (2,:);      % Trial solutions for I using
    MADS optimization for training set

126     disp('-----------------------------------------------
    -----------------------------------------');
127     Solution(trialID).trainingTime = toc(T);
128     Solution(trialID).Cost = E;
129     Solution(trialID).BestSol.net = net;

130     if trialID==1
131         BestCost = Solution(trialID).Cost;
132         BestSol.net = Solution(trialID).BestSol.net;
133     end
134     if BestCost > Solution(trialID).Cost
135         BestCost = Solution(trialID).Cost;
136         BestSol.net = Solution(trialID).BestSol.net;
137     end

138     % Trial solutions for test set
139     T = tic;
140     out = trialSolution(net, inputs_test, t(1), s0, i0);
        S_test = out(1,:); I_test = out (2,:);
```

```
141     Solution(trialID).testingTime = toc(T);

142     [S_exact, I_exact] = exactSolution(inputs_train, i0,
    normalizedN, beta);
143     Solution(trialID).trainMSE = [sum((S_exact -
    S_train).^2)/length(S_train);
144         sum((I_exact - I_train).^2)/length(I_train)];

145     [S_exact, I_exact] = exactSolution(inputs_test, i0,
    normalizedN, beta);
146     Solution(trialID).testMSE = [sum((S_exact -
    S_test).^2)/length(S_test);
147         sum((I_exact - I_test).^2)/length(I_test)];
148 end

149 %% Results
150 % Exact Solution
151 %[S_exact, I_exact] = exactSolution(t, i0, normalizedN,
    beta);

152 % Trial solutions using MADS optimization for training
    set
153 net = BestSol.net;
154 out = trialSolution(net, inputs_train, t(1), s0, i0);
155 S_trial_train = out(1,:); I_trial_train = out (2,:);

156 % Trial solutions using MADS optimization for test set
157 out = trialSolution(net, inputs_test, t(1), s0, i0);
158 S_trial_test = out(1,:); I_trial_test = out(2,:);

159 %% Reports
160 heading = ['The numerical solution of Residual Network
161 trained by ' params.optimAlgo ' for training set'];
162 label = ['tbl:' params.optimAlgo '_train'];
163 %[S_exact_train, I_exact_train] =
    exactSolution(inputs_train, i0, normalizedN, beta);
164 displayResults(inputs_train,S_real_train,I_real_train,S_
    trial_train,I_trial_train,heading,label,params);

165 heading = ['The numerical solution of Residual Neural
    Network trained by ' params.optimAlgo ' for test set'];
166 label = ['tbl:' params.optimAlgo '_test'];
167 %[S_exact_test, I_exact_test] =
    exactSolution(inputs_test, i0, normalizedN, beta);
168 displayResults(inputs_test,S_real_test,I_real_test,S_tri
    al_test,I_trial_test,heading,label,params);

169 M = [Solution.trainMSE];
170 mean_mse = mean(M,2);
171 mean_std(1) = std(M(1,:));
172 mean_std(2) = std(M(2,:));
173 fprintf('Mean of MSEs for training set :\n \t for S :
    %1.3e ± %1.3e\n \t for I : %1.3e ±
    %1.3e\n',mean_mse(1),mean_std(1),mean_mse(2),mean_std(2)
```

```
    );
174 M = [Solution.testMSE];
175 mean_mse = mean(M,2);
176 mean_std(1) = std(M(1,:));
177 mean_std(2) = std(M(2,:));
178 fprintf('Mean of MSEs for test set :\n \t for S : %1.3e
179 ± %1.3e\n \t for I : %1.3e ±
    %1.3e\n',mean_mse(1),mean_std(1),mean_mse(2),mean_std(2)
    );
180 fprintf('Mean of elapsed time for training set %1.3e ±
    %1.3e
    seconds\n',mean([Solution.trainingTime]),std([Solution.t
    rainingTime]));
181 fprintf('Mean of elapsed time for test set %1.3e ± %1.3e
    seconds\n',mean([Solution.testingTime]),std([Solution.te
    stingTime]));

182 save('Results\Results.mat');%,'BestSol','BestCost','Solu
    tion','dates','S_exact','I_exact','inputs_train','S_trai
    n','I_train','inputs_test','S_test','I_test');

183 %% Plots
184 figure;
185 hold on;
186 dates = [covidData_tr(veriSay:-1:1).tarih];
187 population = [data(:,1)+data(:,3)];

188 S = data(:,1); I = data(:,3);

189 plot(dates,normalizedS,'-
    c','LineWidth',2,'DatetimeTickFormat','dd/MM');
    plot(dates,normalizedI,'-
190 m','LineWidth',2,'DatetimeTickFormat','dd/MM');
    plot(dates(inputs_train),S_trial_train,'b--
191 o','LineWidth',1.5,'MarkerSize',5,'MarkerEdgeColor',[0,0
    ,0.8],'MarkerFaceColor','cyan');
    plot(dates(inputs_train),I_trial_train,'r--
192 o','LineWidth',1.5,'MarkerSize',5,'MarkerEdgeColor',[0.8
    ,0,0],'MarkerFaceColor','magenta');
    plot(dates(inputs_test),S_trial_test,'--
193 o','LineWidth',1.5,'MarkerSize',5,'MarkerEdgeColor',[0.0
    67, 0.47, 0.392],'MarkerFaceColor',[0.156,
    0.705,0.388]);
    plot(dates(inputs_test),I_trial_test,'y--
194 o','LineWidth',1.5,'MarkerSize',5,'MarkerEdgeColor',[0.8
    6, 0.506, 0],'MarkerFaceColor',[0.906, 0.6, 0.16]);
    xlabel('\fontsize{12}\bf Tarıhler');
195 ylabel('\fontsize{12}\bf Nüfusun Covid-19 vakalarının
196 oranı (%)');
    xlim([dates(1) dates(veriSay)])
197 xtickangle(45)
198 legend('S için gerçek değerler','I için gerçek
```

```
    değerler',...
        'S için yapay sinir ağ çözümü','I için yapay sinir
    ağ çözümü',...
        'S için tahmin','I için
    tahmin','Location','northeastoutside');
    hold off;
199
    fig=gcf;
200 fig.InvertHardcopy = 'on';
201 saveas(fig,['figs\Figure_' params.optimAlgo '.fig']);
202 print(gcf,['figs\Figure_' params.optimAlgo '.jpg'],'-
203 djpeg','-r300');
    print(gcf,['figs\Figure_' params.optimAlgo '.eps'],'-
204 depsc','-r300');
```

```
1  function y = relu(x)
2      m = length(x);
3      y = zeros(m,1);

4      for i=1:m
5          if x(i)>=0
6              y(i) = x(i);
7          else
8              y(i) = 0;
9          end
10     end
11 end
```

```
1   function [net,out] = resNet(net,x,t,weights)
2   % This function returns a Residual Network
3   % x : Spatial vector as input, and it is 1-by-m row vector
4   % t : Time vector as input, and it is 1-by-n row vector
5   % M : Total # of neurons in each layer of resNet
6   % L : Number of hidden layers
7   % k : Skip index of the network
8   M = net.numberofNeurons;
9   L = net.numberofHiddenLayer;
10  k = net.skipIndex;
11  m = length(x);
12  n = length(t);

13  %% Set network weights
14  net = setWeights(net, weights);

15  % First Hidden Layer
16  W = net.Layers(1).Weights;
17  for i=1:m
18      for j=1:n
```

```
19          X = [x(i); t(j)];
20          z = W'*X + net.Layers(1).bias;
21          net.Layers(1).output(1:M,i,j) = relu(z);
22      end
23  end

24  % Hidden Layers
25  for ell = 2:L+1
26      W = net.Layers(ell).Weights;
27      for i=1:m
28          for j=1:n
29              X = net.Layers(ell-1).output(1:M,i,j);
30              z = W'*X + net.Layers(ell).bias;
31              if (ell>k) && mod(ell,k)==1
                    net.Layers(ell).output(1:M,i,j) = relu(z +
    net.Layers(ell-k).output(1:M,i,j));
32              else
33                  net.Layers(ell).output(1:M,i,j) = relu(z);
34              end
35          end
36      end
37  end

38  % Output Layer
39  for i=1:m
40      for j=1:n
41          X = net.Layers(L+1).output(1:M,i,j);
42          W = net.Layers(L+2).Weights;
43          z = W*X + net.Layers(L+2).bias;
44          %net.Layers(L+2).output(i,j) = relu(z);
45          net.Layers(L+2).output(i,j) = z;
46      end
47  end

48  out.u = net.Layers(L+2).output;
49  [out.du_dx,out.du_dt] = gradient(out.u);
50  [out.d2u_dx2,out.d2u_dxdt] = gradient(out.du_dx);
    [~,out.d2u_dt2] = gradient(out.du_dt);
```

```
1   function net = setLayerWeights(net,layerIndex,layerWeights)
2   % n is the # of outputs of the previous layer
3   % M is the # of neurons of the layer specified by the layerIndex
4   [n, M] = size(net.Layers(layerIndex).Weights);
5   len = length(layerWeights);
6   if layerIndex==net.numberofHiddenLayer+2
7      W = layerWeights(1:len-n);
8      net.Layers(layerIndex).bias = layerWeights(len-n+1:end)';
9   else
10     W = layerWeights(1:len-M);
11     net.Layers(layerIndex).bias = layerWeights(len-M+1:end)';
```

```
12   end
13   net.Layers(layerIndex).Weights = reshape(W,n,M);
```

```
1  function outs = simulateNet(net, inputs)
2  net.inputs = inputs;
3  % each column of inputs matrix is a input of neural network
4  n = size(inputs,2);
5  L = net.numberofHiddenLayer;
6  k = net.skipIndex;

7  % First Hidden Layer
8  W = net.Layers(1).Weights;
9  M = net.Layers(1).numberofNeurons;
10 for i=1:n
11         X = inputs(:,i);
12         z = W'*X + net.Layers(1).bias;
13         net.Layers(1).output(1:M,i) =
   net.Layers(1).activationFunction(z);
14 end

15 % Hidden Layers
16 for ell = 2:L+1
17     W = net.Layers(ell).Weights;
18     M1 = net.Layers(ell-1).numberofNeurons;
19     M2 = net.Layers(ell).numberofNeurons;
20     for i=1:n
21         X = net.Layers(ell-1).output(1:M1,i);
22         z = W'*X + net.Layers(ell).bias;
23         if (ell>k) && mod(ell,k)==1
               M3 = net.Layers(ell-k).numberofNeurons;
24             net.Layers(ell).output(1:M2,i) =
   net.Layers(ell).activationFunction(z + net.Layers(ell-
   k).output(1:M3,i));
25         else
26             net.Layers(ell).output(1:M2,i) =
   net.Layers(ell).activationFunction(z);
27         end
28     end
29 end

30 % Output Layer
31 net.Layers(L+2).activationFunction = @(x) softmax(x);
32 M1 = net.Layers(L+2).numberofNeurons;
33 M2 = net.Layers(L+1).numberofNeurons;
34 for i=1:n
35     X = net.Layers(L+1).output(1:M2,i);
36     W = net.Layers(L+2).Weights;
37     z = W*X + net.Layers(L+2).bias;
38     net.Layers(L+2).output(1:M1,i) =
   net.Layers(L+2).activationFunction(z);
39 end
```

```
40 net.outputs =  net.Layers(L+2).output(:,1:n);
41 outs = net.outputs;
```

```
1 function z = softmax(x)
2   z = exp(x)./sum(exp(x));
3 end
```

```
1     function [net, E] = trainNet(net, inputs, S_train,
      I_train, a, s0, i0, N, beta, nDaysForRealData, params)
2     %inputs = net.inputs;

3     VarMin = params.low; VarMax = params.up;

4     options =
      optimoptions("patternsearch","AccelerateMesh",false,..
      .

5     "FunctionTolerance",params.tolafun,"InitialMeshSize",p
      arams.InitialMeshSize,"MaxFunctionEvaluations",params.
      maxfeval,...
          "MaxIterations", params.Max_Iteration
      ,"MeshContractionFactor",params.MeshContractionFactor,
      ...

6     "MeshExpansionFactor",params.MeshExpansionFactor,"Mesh
      Tolerance",params.MeshTolerance );

7     options = optimoptions(options,'PollMethod',
      'madspositivebasis2n');
8     %options = optimoptions(options,'PollMethod',
      'madspositivebasisnp1');
9     options = optimoptions(options,'PollOrderAlgorithm',
      'Random');
10    options = optimoptions(options,'UseCompletePoll',
      false);
11    options = optimoptions(options,'SearchFcn',
      @MADSPositiveBasis2N);
12    %options = optimoptions(options,'SearchFcn',
      @MADSPositiveBasisNp1);
13    options = optimoptions(options,'Display', 'iter');
      %options = optimoptions(options,'PlotFcn', {
14    @psplotbestf });
      options = optimoptions(options,'PlotFcn', { [] });


15    best.Cost = inf;
16    best.net = net;
```

```
17    trainingDirection = -1;          % -1 : Backward
      training,   1 :  Forward training
18    maxTurn = params.noTours;
19    count = 1;
20    for turnId =1:2*maxTurn
21        if trainingDirection == -1
              layers = net.numberofHiddenLayer+2:-1:2;
22            msg = 'Backward Training from input layer to
      output layer';
23        else
24            layers = 1:net.numberofHiddenLayer+1;
25            msg = 'Forward Training from output layer to
      input layer';
26        end
27        if mod(turnId,2)==0
              count=count+1;
              fprintf('%d. Turn\n',count);
28        end
29        disp('-----------------------------------------
      ----------------------------------------------------
      -------------------');
30        fprintf('\t%s\n',msg);

31        for layerIndex=layers
32            fprintf('\n\t\tUpdating Weights of Layer
      %d\n',layerIndex);
33            % Optimize the layer weigths of net_S adn
      net_I
34            Cost = @(p) CostFunction(net, inputs, S_train,
      I_train, a, s0, i0, N, beta, nDaysForRealData,
      layerIndex, p);
35            p0 = getLayerWeights(net,layerIndex);
36            dim = length(p0);
37            lb = VarMin*ones(1,dim); ub =
      VarMax*ones(1,dim);
38            % MADS
39            [p,E,exitflag,output] =
      patternsearch(@(p)Cost(p),p0,[],[],[],[],lb,ub,[],opti
      ons);
40            net = setLayerWeights(net, layerIndex, p);
41            if best.Cost > E
42               best.Cost = E;
43               best.net = net;
44            end
45        end
46        fprintf('\n\n');
47        disp('-----------------------------------------
      ----------------------------------------------------
      -------------------');
48        trainingDirection = -trainingDirection;
49    end

50    net = best.net;
51    E = best.Cost;
```

```
52      end
```

T.C.

AYDIN ADNAN MENDERES ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Bu tezde sunulan tüm bilgi ve sonuçların, bilimsel yöntemlerle yürütülen gerçek deney ve gözlemler çerçevesinde tarafımdan elde edildiğini, çalışmada bana ait olmayan tüm veri, düşünce, sonuç ve bilgilere bilimsel etik kuralların gereği olarak eksiksiz şekilde uygun atıf yaptığımı ve kaynak göstererek belirttiğimi beyan ederim.

13/07/2021

İmza

Muhammad Jalil Ahmad

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Muhammad Jalil AHMAD

EĞİTİM DURUMU

Lisans Öğrenimi : Aydın Adnan Menderes Üniversitesi Fen Edebiyat Fakültesi Matematik Bölümü

Yüksek Lisans Öğrenimi : -----

Yabancı Diller : İngilizce

BİLİMSEL FAALİYETLERİ

A) Bildiriler

Ahmad, M.J., Günel, K. (2021). Numerical Solutions of Dirichlet Boundry Value Problems using Mesh Adaptive Direct Search Algorithm, 5th International Students Science Congress, İzmir, Turkey.

Ahmad, M.J., Günel, K. (2020). Numerical Solutions of IVPs using Adaptive Nelder Mead Algorithm, 4th International Students Science Congress, İzmir, Turkey.