T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞLARININ YAPAY ARI KOLONİSİ ALGORİTMASI İLE EĞİTİLMESİ

Tezi Hazırlayan
Celal ÖZTÜRK

Tezi Yöneten
Prof. Dr. Derviş KARABOĞA

Bilgisayar Mühendisliği Anabilim Dalı
Doktora Tezi

Ocak 2011
KAYSERİ

T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞLARININ YAPAY ARI KOLONİSİ ALGORİTMASI İLE EĞİTİLMESİ

Tezi Hazırlayan
Celal ÖZTÜRK

Tezi Yöneten
Prof. Dr. Derviş KARABOĞA

Bilgisayar Mühendisliği Anabilim Dalı
Doktora Tezi

Bu çalışma, Erciyes Üniversitesi Bilimsel Araştırma Projeleri Birimi tarafından FBD-9-1004 kodlu proje ile desteklenmiştir.

Ocak 2011
KAYSERİ

Prof. Dr. Derviş KARABOĞA danışmanlığında **Celal ÖZTÜRK** tarafından hazırlanan **"Yapay Sinir Ağlarının Yapay Arı Kolonisi Algoritması ile Eğitilmesi"** adlı bu çalışma, jürimiz tarafından Erciyes Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında **Doktora** tezi olarak kabul edilmiştir.

27/01/2011

JÜRİ :

Başkan	:	Prof. Dr. H. Altay GÜVENİR
Üye	:	Prof. Dr. Derviş KARABOĞA
Üye	:	Doç. Dr. Veysel ASLANTAŞ
Üye	:	Doç. Dr. Erkan BEŞDOK
Üye	:	Doç. Dr. Coşkun ÖZKAN

ONAY :

Bu tezin kabülü Enstitü Yönetim Kurulunun 01/02/2011 tarih ve 2011/05-05 sayılı kararı ile onaylanmıştır.

01 / 02 / 2011

Prof.Dr. Necmettin MARAŞLI

Enstitü Müdürü

TEŞEKKÜR

"Yapay Sinir Ağlarının Yapay Arı Kolonisi Algoritması ile Eğitilmesi" konulu doktora tez çalışmalarımın yönlendirilmesinde, yürütülmesinde, değerlendirilmesinde ve sonuçlandırılmasında bilgi, tecrübe ve yardımlarını esirgemeyen değerli danışmanım Prof. Dr. Derviş KARABOĞA hocama teşekkür ederim.

Doktora tez çalışmalarımda yol gösteren, maddi ve manevi destek veren tez izleme komitesi üyeleri hocalarım Doç. Dr. Veysel ASLANTAŞ'a ve Doç. Dr. Erkan BEŞDOK'a katkılarından dolayı saygı ve teşekkürlerimi sunarım. Çalışmalarımın çeşitli aşamalarında değerli katkıları olan hocam Doç. Dr. Coşkun ÖZKAN'a ve çalışma arkadaşım Yrd. Doç. Dr. Bahriye AKAY'a teşekkürü bir borç bilirim.

Doktora tez çalışmamın her aşamasında beni teşvik eden, destekleyen, sabırla bekleyen ve bana anlayış gösteren eşim Vildan'a ve kızım Candan'a, ayrıca bugünlere gelmemde büyük emeği geçen aileme sevgi ve teşekkürlerimi sunarım.

Erciyes Üniversitesi Bilimsel Araştırma Projeleri Birimi'ne doktora tez projesi kapsamında yaptığı destekten dolayı ve tez çalışmalarım için yaptığım araştırmalarıma kısmen destek olan Erciyes Üniversitesi Vakfı'na teşekkür ederim. Türkiye Bilimsel ve Teknolojik Araştırma Kurumu'na (TUBİTAK) yurt içi doktora burs programı kapsamında doktora bursiyeri olarak çalışmalarıma olan destek ve katkılarından dolayı teşekkür ederim.

YAPAY SİNİR AĞLARININ YAPAY ARI KOLONİSİ ALGORİTMASI İLE EĞİTİLMESİ

Celal ÖZTÜRK
Erciyes Üniversitesi, Fen Bilimleri Enstitüsü
Doktora Tezi, Ocak 2011
Tez Danışman : Prof. Dr. Derviş KARABOĞA

ÖZET

Yapay Sinir Ağları (YSA) biyolojik beyinin öğrenme ve bilgi saklama yeteneklerinin benzetilmeye çalışıldığı yapay zeka tekniğidir. YSA'da bilgileri depolayan bağlantı ağırlıkları nöronları paralel ve ardışık bir biçimde birleştirirler. Oldukça zor bir optimizasyon problemi olan Yapay sinir ağlarının öğrenmesi süreci ağ yapısında ki ağırlıklara uygun değerlerin bulunması işlemidir. Bir çok çalışmanın yapıldığı bu alanda ilk çalışmalar yerel en iyileştirme (optimizasyon) tekniklerine dayalı iken genel en iyileştirici algoritmaların geliştirilmesi ile bu metotlar da yapay sinir ağlarının eğitiminde yaygın olarak kullanılmaya başlanmıştır. Bu doktora tez çalışmasında, yeni bir genel araştırma tekniği olan sürü zekasına dayalı Yapay Arı Kolonisi (ABC) algoritması YSA eğitimi için önerilmiş ve performansı geleneksel ve modern optimizasyon yöntemleriyle karşılaştırılarak analiz edilmiştir.

Tez çalışmasında ABC algoritması toplama-birimli ileri beslemeli YSA eğitiminde temel üç test problemi için kullanılmış ve başarısı yerel ve genel optimizasyon algoritmalarına karşı tartışılmıştır. Çarpım-birimli YSA modelleri girişlerin ağırlıklar kadar kuvvetlerinin birbirleriyle çarpılması ile nöronların çıkışlarının hesaplandığı ağlardır. ABC algoritması tahmin amaçlı çarpım ve toplama birimli ileri beslemeli ağların eğitimi için uygulanmış ve elde edilen sonuçlar tartışılmıştır.

Çok katmanlı algılayıcılar (MLP), vektör kuantalamalı öğrenme (LVQ) ve radyal temelli fonksiyonlar (RBF) sınıflandırma çalışmalarında yaygın kullanılan YSA modelleridir. ABC algoritması, UCI veritabanından alınan dokuz farklı test probleminin bu ağlarla sınıflandırılması amaçlı deneysel olarak değerlendirilmiş ve performansı literatürde yer alan makine öğrenme algoritmalarının performansları ile karşılaştırılmıştır. Elde edilen sonuçlar ABC ile eğitilen YSA modellerinin üstün genelleştirme yeteneğini ve sınıflandırma başarısını ortaya koymaktadır.

Anahtar Kelimeler: Yapay Arı Kolonisi Algoritması, Yapay Sinir Ağlarının Eğitilmesi, Sınıflandırma, Tahmin, Çok Katmanlı İleri Beslemeli Ağlar, Vektör Kuantalamalı Öğrenme, Radyal Temelli Fonksiyonlar

TRAINING ARTIFICIAL NEURAL NETWORKS WITH ARTIFICIAL BEE COLONY ALGORITHM

Celal ÖZTÜRK
Erciyes University, Graduate School of Natural and Applied Sciences
Ph.D. Thesis, January 2011
Thesis Supervisor: Prof. Dr. Derviş KARABOĞA

ABSTRACT

Artificial Neural Networks (ANN) are designed to emulate the storage and learning mechanisms within biological brains. The standard ANN model is based upon summation, calculating the net input as the weighted sum of the inputs. Finding the optimal weight set is the training phase and is considered as a difficult global optimization task, on which there is a strong research. Many training algorithms, most of which were local optimizers have been proposed so far to improve the performance of neural networks. Global optimization methods are under continuous development and lately, they have been studied on training ANN's. In this PhD thesis, a new swarm based on Artificial Bee Colony algorithm is proposed to training neural networks and analyzing the performance of the algorithm besides the well-known conventional and modern optimization techniques.

The performance of ABC algorithm is firstly tested for training summation-unit feedforward neural networks on three basic benchmark problems and the success of the algorithm is studied against local and global optimizers. Product-unit networks are based on multiplicative nodes instead of additive ones, where the nonlinear basis functions express the possible strong interactions between variables. ABC algorithm is then applied on two forecasting benchmark problems to determine the weights of the product-unit and summation-unit feedforward network models.

Multilayer Feedforward Perceptron (MLP), Learning Vector Quantization (LVQ), and Radial Basis Functions (RBF) are mostly preferred ANN models in classification problems. In this study, Artificial Bee Colony algorithm is used in training MLP, LVQ, RBF networks on nine classification benchmark problems from UCI database. The results indicate better performance of ABC algorithm in terms of generalization as well as the classification error than other machine-learning algorithms.

Keywords: Artificial Bee Colony Algorithm, Training Artificial Neural Networks, Classification, Forecasting, Multi-layer Perceptron, Learning Vector Quantization, Radial Basis Functions.

İÇİNDEKİLER

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

1. BÖLÜM

GİRİŞ

1.1. Çalışmanın Kapsamı ve Önemi

İnsan beyninin yapısı ile çalışma şeklinin sürekli merak edilmesiyle yapılan araştırmalar sonucunda yapay zeka teknikleri ortaya çıkmış ve zeki sistemler geliştirilmiştir. Günümüzde en çok kullanılan yapay zeka sistemlerine örnek olarak uzman sistemler, bulanık mantık, yapay sinir ağları, evrimsel hesaplama yöntemleri gibi teknikler verilebilir [1].

Yapay zekânın bir dalı olan Yapay Sinir Ağları (YSA), biyolojik beyinin çalışmasını modellemeye çalışan büyük paralel hesaplama mekanizmalarıdır. Bir yapay sinir ağı nöron adı verilen basit bilgi işleme elemanlarından oluşmaktadır. Nöronlar, bağlantı hatları üzerinden işaret göndererek birbirlerini etkilerler. Bu bağlantı hatlarının her birinin kendine özgü bir ağırlığı vardır. Bilgileri depolayan bağlantı ağırlıkları nöronları paralel ve ardışık bir biçimde birleştirirler. Uyarlamalı olarak hesaplanabilen bu ağırlıkların aldıkları değerler, bilginin ağ boyunca en doğru biçimde iletilmesini sağlarlar [2].

Yapay sinir ağlarının başarısı ağ yapısındaki ağırlıkların uygun değerleri almasına doğrudan bağımlıdır. Bu yüzden başarılı çalışabilmeleri amacıyla yapay sinir ağları eğitilirler; yani, verilen bir girdi vektör seti ile hesaplanan çıktı değerleri ile hedeflenen çıktı değerlerinin birbirine yakınlaşmasını sağlayacak nöronlar arası optimum bağlantı ağırlıklarına ulaşmak hedeflenir. Nöronlar arası bağlantı ağırlıkları ile hesaplanan çıktı ile beklenen çıktı arasındaki fark yeterli bir duyarlılığa yaklaştığında YSA'nın öğrendiği sonucuna varılabilir. Yapay sinir ağlarının en iyi şekilde öğrenmesini sağlayacak bağlantı ağırlıklarının belirlenmesi yani YSA'nın

eğitilmesi bilim insanlarının yıllardır çalıştığı optimizasyon problemlerinden bir tanesidir [3, 4].

Optimizasyon, belirlenmiş sınırlamalar dahilinde bir problemin mümkün olabilecek çözümleri arasından en iyi olanı seçme işlemi olarak tanımlanabilir. Optimizasyonun görevi belli kısıtları ya da sınırları olan parametre setlerinin değerlerini belirlemektir. Bu görev birçok uzmanlık alanı için çok büyük önem taşır. Örneğin fizikçiler, kimyacılar ve mühendisler belli kısıtlar altında üretimi maksimize edebilmek amacıyla en uygun tasarımın geliştirilmesiyle ilgilenirler. Ekonomistler ve yöneylem araştırmacıları endüstriyel ve sosyal kaynakların optimum paylaştırılmasını sağlamak için çalışırlar. Mühendislik uygulamalarında birçok problem optimizasyon problemi olarak modellenebilir. Mekanik tasarım, uçak, otomobil tasarımı, tesis organizasyonu, fabrikaların üretim ve dağıtım planlaması, süreç kontrolü, devre tasarımı, veritabanı tasarımı ve trafik planlaması örnek olarak verilebilecek uygulamalardan bir kısmıdır. Bu problemlerin bazıları sadece doğrusal modeller içerir, bu problemlerin çözümü için özel doğrusal optimizasyon teknikleri bulunmaktadır [5]. Diğer türdeki problemler doğrusal olmayan optimizasyon problemleridir ve çözümleri çok daha zordur. Bu sebeplerle, bilim insanları doğrusal olmayan eğriler ile çalışırken veya model uydururken güçlü optimizasyon tekniklerine ihtiyaç duymuşlardır [6].

Bu tez çalışmasında, ele alınan yapay sinir ağlarının eğitilmesi problemi de bir doğrusal olmayan optimizasyon problemidir. Optimizasyon terimi minimizasyon ve maksimizasyon terimleri yerine de kullanılır. Bir f fonksiyonunun maksimizasyon edilmesi, fonksiyonunun ters işaretlisinin $-f$ minimizasyon edilmesine denktir, bu nedenle minimizasyon, maksimizasyon ve optimizasyon ifadeleri birbirlerinin yerine kullanılabilmektedir [7]. Bazı problemler belirli kısıtları sağlayan parametreleri bulmayı gerektirir. Bu tür problemler "kısıtlamalı minimizasyon problemleri" olarak bilinir [8]. Yapay sinir ağlarının eğitilmesi problemi kısıtlamasız türde bir problem olduğundan kısıtlamasız minimizasyon işlemlerine temas edilmiştir ve kısıtlamalı minimizasyona değinilmeyecektir. Kısıtlamasız optimizasyon biçimsel olarak Eşitlik 1.1'de verilen şekilde ifade edilir:

$$\begin{aligned} &f : R^n - - - - R, \text{ 'de tanımlı olsun} \\ &\text{Bul, } x^* \in R^n \\ &\text{Öyle ki } \forall x \in R^n \text{ için } f(x^*) \leq f(x), \text{ şartını sağlasın.} \end{aligned} \tag{1.1}$$

Yapay sinir ağları örüntü sınıflandırması, fonksiyon yaklaşımları, optimizasyon, örüntü eşleştirmesi ve birleşmeli hafızalar gibi uygulamalarda çeşitli problemlerin çözümlerinde başarıyla kullanılmaktadır. Yapay sinir ağlarının başarısı büyük ölçüde mimarisine, eğitim algoritmasına ve eğitimde kullanılan özelliklerin seçimine bağlıdır. Tüm bunlar yapay sinir ağlarının tasarımını zor bir optimizasyon problemi haline getirir. Araştırmacılar, uygun mimari ve/veya eğitim algoritması ve/veya aktivasyon fonksiyonu ve/veya eğitim verileri tabanlı ağırlıkları bulmaya çalışarak ağ eğitimi üzerine çalışmaktadırlar. Birçok yaklaşımda topoloji ile transfer fonksiyonları sabit tutulur ve ağırlıkların olası değerleri ayarlanmaya çalışılarak ağ eğitilmeye çalışılır [9]. Prensipte sinir ağları, sınırsız ağ boyutu ve mevcut sonsuz veri olduğu müddetçe, verilen her problemi çözecek şekilde tasarlanabilirler. Fakat pratikte kaynaklar sınırlıdır ve ağın genelleme yeteneğine de güvenilmelidir. Aslında ağ yapısı sabitleştirildiğinde, örneğin saklı katman sayısı ve her bir saklı katmandaki nöron sayısı belirlendiğinde ağ optimal ağırlık setlerini elde edene kadar ağırlık ayarlamaları yapmaya devam eder. Ancak, birçok pratik uygulama probleminde ağlar uygun ağırlık setlerine yakınsama eğilimlerinde değildirler ve bu problemi kısmen gidermek için, bir dizi ağ optimizasyon yaklaşımı önerilmiştir [10, 11].

Yapay sinir ağlarının performansını direkt olarak etkileyen ağırlık setinin optimum değerinin bulunması, literatürde iyi bilinen başarılı yaklaşımların bulunmasına rağmen daha da iyileştirmelerin yapılabileceği bir araştırma alanıdır. Bu tez kapsamında yapay sinir ağlarının doğrusal olmayan karakterdeki ağırlık setinin en uygun şekilde hesaplanmasında yapay arı kolonisi (ABC) algoritmasının kullanılması ve algoritma performansının araştırılması hedeflenmiştir. 2005 yılında Karaboğa tarafından ortaya konan ABC algoritması bal arılarının zeki yiyecek (kaynak) arama davranışlarını modelleyen bir tasarım algoritmasıdır [12]. ABC algoritması öncelikle çok modlu çok parametreli nümerik fonksiyonların optimizasyonunda kullanılmış ve sınırlamasız problemler üzerinde literatürde iyi bilinen Genetik Algoritma, Diferansiyel Gelişim Algoritması ve Parçacık Sürü Optimizasyon gibi algoritmalara

karşı başarısı test edilmiş ve tartışılmıştır [13, 14].

Yapay arı kolonisi algoritması yeni bir optimizasyon yöntemi olmasına rağmen kısa sürede geniş bir uygulama alanına sahip olmaya başlamıştır [15]. Ancak, yapay sinir ağlarının eğitilmesinde kullanılması ilk olarak bu tez kapsamında yapılan çalışmalar ile başlamıştır. ABC algoritmasının yapay sinir ağlarının eğitilmesinde performansı araştırılırken algoritmanın performansı, dikkate alınan her farklı problemde ve yapay sinir ağı mimarisinde literatürde en yaygın kullanılan tekniklerin performansları ile karşılaştırılmıştır.

1.2. Çalışmanın Amacı ve Yöntemi

Tez çalışmasının temel amacı yapay sinir ağlarının eğitilmesinde yapay arı kolonisi algoritmasını bir alternatif yöntem olarak sunmak ve algoritmanın performansını diğer yaygın yöntemlerin performanslarıyla karşılatırarak tartışmaktır.

Yapay sinir ağları örneklerden öğrenebilme ve çevreden gelen olaylara karşı tepkiler üretebilmeleri yetenekleri ile esnek ve güçlü yapıları sayesinde makinalı öğrenme, sınıflandırma, genelleme ve optimizasyon gibi alanlarda başarıyla kullanılmaktadırlar. Bununla birlikte, çok çeşitli yapay sinir ağ yapıları (modelleri) geliştirilmiş ve bu modeller çeşitli alanlarda uygulanmıştır. Çeşitli yapay sinir ağ yapılarının bulunması öncelikle hangi yapının hangi tür problemlerde kullanılacağı sorusunu öne çıkarmaktadır. Ayrıca belki de daha önemlisi yapay sinir ağının başarısını doğrudan etkileyen ağın eğitimininde ne tür bir yöntemin kullanılacağı ve bu yöntemler içerisinde hangi tekniklerin tercih edileceği sorusu tam olarak cevap bulamamıştır. Bu noktalardan hareketle bu tez çalışmasında çeşitli yapay sinir ağ modellerinin eğitilmesinde yapay arı kolonisi algoritmasının kullanılabilirliği sunulmuş ve performansı detaylıca araştırılmıştır.

En yaygın kullanılan yapay sinir ağ mimarisi ileri beslemeli çok katmanlı algılayıcı (MLP) yapay sinir ağ modelleridir [16]. Yapay arı kolonisi algoritması çok katmanlı yapay sinir ağlarının eğitilmesinde temel test problemlerinden olan XOR, 3-Bit Parite ve Encoder-Decoder problemlerinin çözümünde kullanılmıştır.

Zaman serisi tahminleri yapay sinir ağlarının yaygın kullanım alanlarından olup bu amaçla kullanılan yapay sinir ağları genelde çarpım birimli yapay sinir ağlarıdır [17]. Çarpım birimli yapay sinir ağları öğrenme yetenekleri daha yüksek ve hafızaları daha kuvvetli olan ağlardır. Çarpım birimli yapay sinir ağlarının eğitilmesinde yapay arı kolonisi algoritmasının tahmin problemleri için kullanımı ve sonuçları bu tez çalışması kapsamında sunulmuştur.

Yapay sinir ağları için örüntü sınıflandırma en yaygın uyguluma alanı olarak karşımıza çıkmaktadır. Sınıflandırma amaçlı kullanılan yapay sinir ağlarının en çok tercih edilenleri ileri beslemeli yapay sinir ağları, vektör kuantalamalı öğrenme ağları ve radyal temelli fonksiyon ağlarıdır. Bu ağ çeşitlerinin hepsinin eğitiminde yapay arı kolonisi algoritmasının performansı araştırılmıştır. Yapay arı kolonisi algoritmasının etkinliğinin analiz edilebilmesi için UCI makine öğrenme deposundan alınan 9 farklı test sınıflandırma problemi üzerinde deneysel çalışmalar ve analizler gerçekleştirilmiştir.

Optimizasyon tekniklerinin birarada kullanılmasıyla hibrid yöntemler oluşturulabilir. Yapay sinir ağlarının eğitilmesi probleminde de genelde bölgesel ve küresel arama algoritmaları birleştirilerek oluşturulan hibrid algoritmaların performansları araştırılan konulardandır. Bu amaçla, MLP ağlarının eğitilmesinde ABC algoritması ile Levenberg-Marquardt algoritmasının melezleştirilmesi ile oluşturulan yeni algoritmanın performansı XOR, 3-Bit parite ve Encoder-Decoder test problemlerinde ve sınıflandırma test problemlerinde araştırılmıştır.

ABC algoritmasının yapay sinir ağlarının eğitimindeki başarısı araştırılırken algoritmanın performansı, kullanılan her farklı problemde ve yapay sinir ağı mimarisinde literatürde sık kullanılan tekniklerle karşılaştırılmıştır. Bu amaçla, türeve dayalı algoritmalardan eğim iniş (gradient descent) geri yayılım algoritması ve Levenberg-Marquardt geri yayılım algoritması; evrimsel algoritmalardan genetik algoritma, diferansiyel gelişim algoritması ve parçacık sürü optimizasyon algoritmaları ileri beslemeli yapay sinir ağlarının eğitiminde yapılan karşılaştırmalarda kullanılmıştır. Vektör kuantalamalı öğrenme ağları ve radyal temelli fonksiyon ağlarının eğitilmesinde ABC algoritmasının kullanılabilirliği, bu

ağların eğitimleri için özel geliştirilmiş öğrenme algoritmaları ile karşılaştırılarak tartışılmıştır.

1.3. Çalışmanın Bölümleri

Tez çalışmasının ikinci bölümünde çalışmanın çıkış noktası olan yapay sinir ağları ile ilgili bilgiler verilmektedir; bu ağların yapıları ve modelleri ile birlikte eğitilmesi ve eğitimlerinde kullanılan teknikler anlatılmıştır.

Üçüncü bölümde yapay arı kolonisi algoritması tanıtılmıştır; algoritmanın modeli ve çalışma prensipleri açıklanmıştır. Dördüncü bölümde yapay sinir ağlarının test problemleri ve tahmin problemleri için ABC algoritması ile eğitilmesi çalışmaları verilmiştir.

Beşinci bölümde sınıflandırma test problemleri için yapay sinir ağlarının ABC algoritması ile eğitilmesi çalışmaları anlatılmış, ABC algoritmasının performansı diğer tekniklerle karşılaştırılmış ve algoritmanın performansını etkileyen faktörler araştırılmıştır. Bunlarla birlikte, MLP ağlarının eğitilmesinde yapay arı kolonisi algoritması ile Levenberg-Marquardt algoritmasının birarada kullanıldığı hibrid model araştırılmıştır.

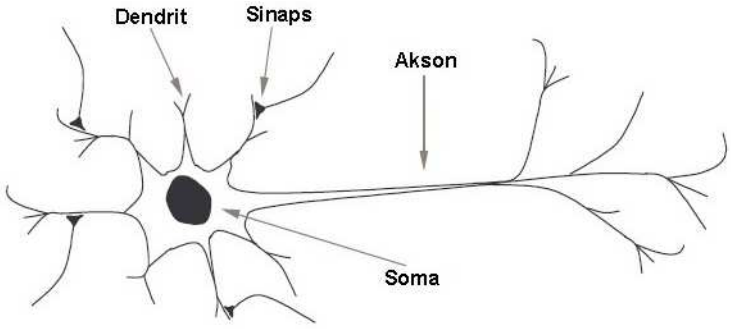
Altıncı bölümde vektör kuantalamalı öğrenme ve radyal temelli fonksiyon ağların sınıflandırma amaçlı eğitilmesinde yapay arı kolonisi algoritmasının kullanılması ile ilgili sonuçlar sunulmuştur. Son olarak sonuç ve öneriler bölümünde, tez çalışmasından elde edilen sonuçlar ve öneriler tartışılmıştır. Yapay sinir ağlarının eğitimi amaçlı geliştirilen yapay arı kolonisi algoritmasının kodları ile deneysel çalışma ve analizlerde kullanılan verilerin bazıları kullanım şekillerine örnek teşkil etmesi amacıyla Ekler bölümünde verilmiştir.

2. BÖLÜM

YAPAY SİNİR AĞLARI

2.1. Giriş

Yapay sinir ağları, biyolojik sinir ağlarının işlevlerinden esinlenilerek oluşturulmuş matematiksel modellerdir. Biyolojik sinir ağları, sinir hücrelerinden meydana gelmektedir ve insan beyninde yaklaşık on milyar sinir hücresi ve bunlarında altı yüz trilyondan fazla bağlantısının olduğu düşünülmektedir. Bir biyolojik sinir hücresinin yapısı Şekil 2.1'de verilmektedir ve şekildende görüldüğü gibi temel bir sinir hücresi; dendritler, hücre gövdesi, akson ve sinapslardan oluşmaktadır [18]. Çevreden alınan uyarıcılar elektrik sinyallerine dönüştürülerek hücre gövdesine ulaşır ve burada işlem uygulanarak başka bir sinyal oluşturulup akson aracılığıyla dendrite gönderilir. Dendritler bu bilgileri sinapslara ileterek diğer hücrelere gönderirler ve milyarlarca sinir hücresi biraraya gelerek sinir sistemini oluşturmaktadırlar. Bilim insanları biyolojik hücrelerin yapısal özelliklerinden yararlanarak yapay sinir ağlarını geliştirmişlerdirler [19].

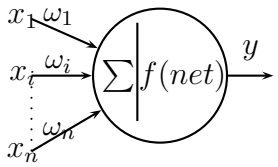


Şekil 2.1. Biyolojik sinir hücresi.

Yapay sinir ağları, insan beyninin özelliklerinden olan öğrenme yolu ile yeni bilgiler türetebilme, yeni bilgiler oluşturabilme ve keşfetme gibi yetenekleri herhangi bir yardım almadan otomatik olarak gerçekleştirmek amacı ile geliştirilen sistemlerdir. Bu sistemler, örnekleri kullanarak elde ettikleri bilgiler vasıtasıyla kendi deneyimlerini oluşturarak olayları öğrenmeye çalışırlar ve yeni gelecek olaylara nasıl tepkiler üretileceğini belirlerler [20]. Birbirlerine hiyerarşik olarak bağlı ve paralel olarak çalışan yapay hücrelerden oluşan yapay sinir ağları biyolojik sinir sisteminin modellenmesiyle oluşturulmuştur.

2.2. Yapay Sinir Hücresi

Yapay sinir ağları yapay sinir hücrelerinden oluşur. Genelde nöron şeklinde adlandırılan bu hücrelere düğüm, birim, işlemci eleman gibi farklı isimler de verilmektedir. Nöronlar basit fonksiyonlara sahip birimler olup temelde bağlantılar, ön işlem ve aktivasyon fonksiyonu olmak üzere 3 bölümden oluşurlar (Şekil 2.2). Nöronlar, bağlantı hatları üzerinden işaret göndererek birbirlerini etkilerler. Bağlantı hatlarının her birinin kendine özgü bir ağırlığı vardır. Bilgileri depolayan bağlantı ağırlıkları nöronları paralel ve ardışık bir biçimde birleştirirler. Uyarlamalı hesaplanabilen bu ağırlıkların aldıkları değerler; bilginin en doğru biçimde iletilmesini sağlar. Genellikle, bağlantı ağırlıkları bir öğrenme işlemi ile belirlenir [20, 21]. Yapay sinir ağlarının hafızaları olarak nitelendirilen bu ağırlıklar, sistemde modele ait bilginin taşındığı noktalardır. Yapay sinir ağlarının eğitilmesi, sisteme verilen örnek girdiler için beklenen çıktı değerlerini üretmesini sağlayacak ağırlık değerlerinin bulunması işlemidir. Dağınık biçimde bilgiyi taşıyan bu ağırlıkların alacağı değerler kendi başlarına bir anlam ifade etmezler. Ancak, sistemin performansı tamamıyla ağırlıkların aldığı değerlere yani ağırlık setine bağlıdır.



Şekil 2.2. Nöron yapısı.

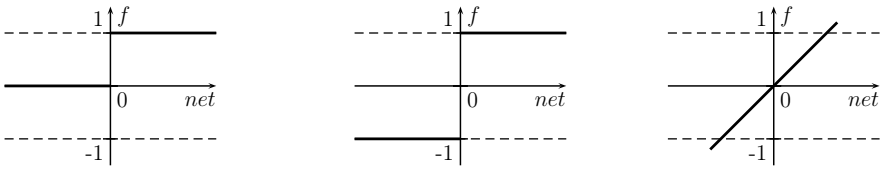
Yapay sinir hücresine dışarıdan verilen bilgiler girdi olarak adlandırılır. Ağırlık değerleri ise hücreye girdi olarak verilen bilgilerin önemini ve hücre üzerindeki etkisini gösterir. Yaygın olarak kullanılan ön işlem fonksiyonlarından biri toplama fonksiyonudur. Toplama fonksiyonu, hücreye gelen net girdiyi hesaplar ve genellikle gelen girdilerin kendi ağırlıklarıyla çarpımlarının toplamıdır. Bu fonksiyon Eşitlik 2.1'deki denklem ile ifade edilebilir [2].

$$\begin{aligned} net &= w_1x_1 + w_2x_2 + \ldots + w_nx_n + x_0b = \sum_{i=1}^{n} w_ix_i + \mathrm{x}_0b \\ y &= f(net) \end{aligned} \tag{2.1}$$

Girişlerde uygulanan değerler (x_i), bağlantılar üzerindeki ağırlıklar (w_i) ile çarpılır ve toplayıcı fonksiyona ($\sum$) bir eşik değeri (bias) $x_0 * b$ ile birlikte uygulanır. Eşik değeri nörona haricen uygulanan bir öndeğerdir. Toplayıcı fonksiyonun çıkışına ise f aktivasyon fonksiyonu uygulanarak çıkış elde edilir.

Her bir girdi bağlantılar üzerindeki ağırlıklar oranında çıkış üzerinde etkili olmaktadır ve eşik değeri sistemin girdilerinden bağımsızdır. Bütün girdilerin değerleri 0 olduğu durumda çıkış fonksiyonu $f(x_0 * b)$ olmaktadır.

Biyolojik nöronlarda giriş aktivasyon değerini aştığında nöron darbe göndermektedir. YSA'larda da bu özelliği uygulamak amacı ile toplama fonksiyonundan gelen net girdiyi işlemden geçirerek hücrenin çıktısını üreten ve genellikle doğrusal olmayan bir aktivasyon fonksiyonu kullanılır [22]. Kullanılan hücre modelinin çeşidine göre değişik aktivasyon fonksiyonları kullanılmaktadır. Bunlardan en yaygın kullanılanları basamak fonksiyonu (tek kutuplu ve çift kutuplu), doğrusal fonksiyon (tam doğrusal ve simetrik parçalı doğrusal) ve sigmoid fonksiyon (logaritmik sigmoid ve tanjant sigmoid), fonksiyonların çıkış aralıkları ve matematiksel tanımları Şekil 2.3'de verilmiştir [23]. Türevi alınabilir, sürekli ve doğrusal olmayan bir fonksiyon olması dolayısıyla, doğrusal olmayan problemlerin çözümünde yaygın olarak kullanılan logaritmik sigmoid fonksiyon ele alınan problemlerin çözümünde kullanılmıştır. Fonksiyonun matematiksel tanımı Eşitlik 2.2 de verilmektedir:



$$f(net) = \begin{cases} 1 & net \geq 0 \\ 0 & net < 0 \end{cases}$$

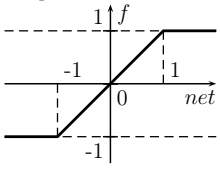
Tek kutuplu basamak fonksiyonu

$$f(net) = \begin{cases} 1 & net \geq 0 \\ -1 & net < 0 \end{cases}$$

Çift kutuplu basamak fonksiyonu

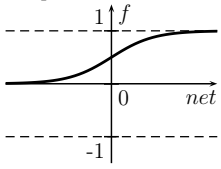
$$f(net) = \{\cdot net$$

Doğrusal fonksiyon



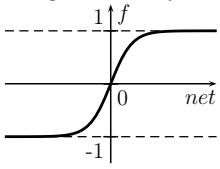
$$f(net) = \begin{cases} 1 & net > 1 \\ net & -1 \leq net \leq 1 \\ -1 & net < -1 \end{cases}$$

Simetrik parçalı doğrusal fonksiyon



$$f(net) = \frac{1}{1 + e^{-net}}$$

Sigmoid fonksiyonu



$$f(net) = \frac{1 - e^{-2net}}{1 + e^{-2net}}$$

Tanjant sigmoid fonksiyonu

Şekil 2.3. Yapay sinir ağları aktivasyon fonksiyonları.

$$y = f(net) = \tfrac{1}{1+e^{-net}} \tag{2.2}$$

2.3. Yapay Sinir Ağlarının Yapıları

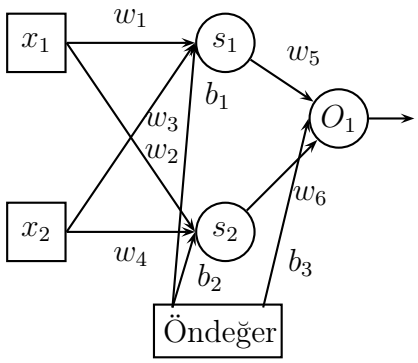
Yapay sinir nöronları birbirleriyle bağlanarak; üç veya daha fazla katman halinde ve her katman içinde paralel şekilde biraraya gelerek ağ oluştururlar [24]. Girdi katmanındaki nöronlar dış dünyadan bilgileri alarak ara katmanlara transfer etmekten sorumludur. Ara katmanlar kullanılarak girdi katmanından gelen bilgiler işlenip çıktı katmanına gönderilir. Çıktı katmanı süreç elemanları, ara katmandan gelen bilgileri işleyip ağın girdi katmanından sunulan girdi seti için üretilmesi gereken çıktıyı üretmeye çalışırlar.

YSA'larını bulundurdukları katman sayısına başka bir ifadeyle ağ mimarilerine göre tek katmanlı ve çok katmanlı ağlar olarak iki grupta incelenirler. Tek katmanlı bir sinir ağında, nöronlar çıktı katmanını ifade eder. Giriş değerlerini alan nöronlar girdi katmanı sayılmaz çünkü burada herhangi bir hesaplama gerçekleşmez. Giriş katmanından alınan bilgiler çıkış katmanında hesaplanarak ağ çıktısı elde edilir. Çok katmanlı ağlar ise saklı nöronlar veya gizli birimler olarak adlandırılan bir veya daha fazla gizli katmanın varlığıyla ve girdi katmanında da ağırlıklandırma olmasıyla diğerlerinden ayrılır. Çok katmanlı ağlarda çıktı katmanı ile girdi katmanı arasına

en az bir gizli katman yerleştirilmiştir. Bu katmanlardaki nöronların fonksiyonu dış girdi ile ağın çıktısı arasında aracılık yaparlar [25].

Yapay sinir ağları katmanları arasındaki bağlantı şekillerine göre de ileri beslemeli ve geri beslemeli yapay sinir ağları olmak üzere iki grupta incelenirler.

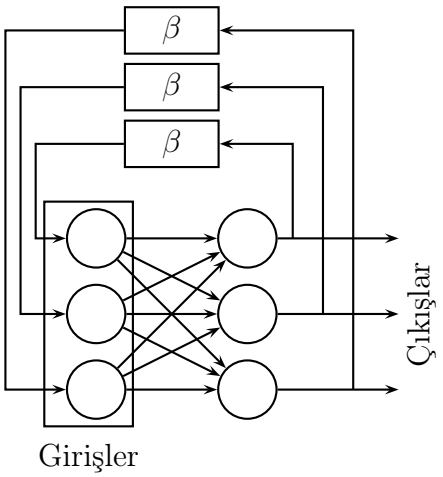
İleri beslemeli yapay sinir ağlarında her bir katmandaki nöronlar sadece bir önceki katmanın nöronları tarafından beslenirler. İleri beslemeli YSA'da, hücreler katmanlar şeklinde düzenlenir ve bir katmandaki hücrelerin çıkışları bir sonraki katmana ağırlıklar üzerinden giriş olarak verilir. Giriş katmanı, dış ortamlardan aldığı bilgileri hiçbir değişikliğe uğratmadan sonraki katmandaki nöronlara iletir [26]. Bilgi katmanlarda işlenerek ağ çıkışı belirlenir, örnek bir ileri beslemeli yapay sinir ağı Şekil 2.4'de verilmiştir.



Şekil 2.4. İleri beslemeli YSA ağ yapısı.

Geri beslemeli yapay sinir ağlarında ise en az bir hücre sonraki katmanlardaki hücrelerce de beslenir. Şekil 2.5'de verilen geri beslemeli YSA modelinde olduğu gibi geri beslemeli ağlarda çıkıştan girişe doğru ters yönlü ilişkiler mevcuttur [27]. Ağın girişleri verildikten sonra nöronların durumları belirlenir, sonra çıkış nöronları girişe bağlı olduğundan yeni girişler olarak ağı etkilerler. Bu çeşit sinir ağlarının dinamik hafızaları vardır ve bir andaki çıkış hem o andaki hem de önceki girişleri yansıtma yeteneğine sahiptir [28].

Yapay sinir ağlarında tipik bir ağın nöronları farklı katmanlı topolojilerde düzenlenebilir. Giriş katmanı aslında sinirsel bir yapıda değildir ve bu birim basitçe girdi değişken değerlerini alır. Saklı katman ve çıktı katman nöronlarının her biri bir önceki katmanın tüm birimleriyle bağlantılıdır. Bir önceki katmanın tüm birimleriyle



Şekil 2.5. Geri beslemeli YSA ağ yapısı.

bağlantı olmadığında ağlar, kısmi-bağlantılı olarak tanımlanırlar. Ancak, pek çok uygulama için tam-bağlantılı ağlar tercih edilmektedir [29]. Ağ çalıştırıldığında, girdi değişken değerleri giriş biriminde yerini alır ve sonra saklı katman ve çıktı katmanlarında ilerleyerek çalıştırılırlar. Her biri kendi aktivasyon değerini, önceki katmandaki birimin çıktılarının ağırlıklı toplam değerlerini ve öndeğerini alarak bulurlar. Aktivasyon değeri, aktivasyon fonksiyonunda işleme tabi tutulur ve nöronun çıktı değeri üretilir. Tüm ağ çalıştırıldığında çıkış katmanının çıkışları, tüm ağın çıktı değerleri gibi davranır [30].

Yapılarına göre temel anlamda ileri beslemeli ve geri beslemeli; tek katmanlı ve çok katmanlı yapay sinir ağları olmak üzere sınıflandırılan yapay sinir ağ yapıları ayrıca öğrenme yaklaşımlarına göre de sınıflandırılırlar [31]. Bu sınıflandırma çeşitleri bir sonraki bölümde yapay sinir ağlarının eğitilmesi ve eğitimde yaygın olarak kullanılan öğrenme algoritmalarıyla ile birlikte açıklanmıştır.

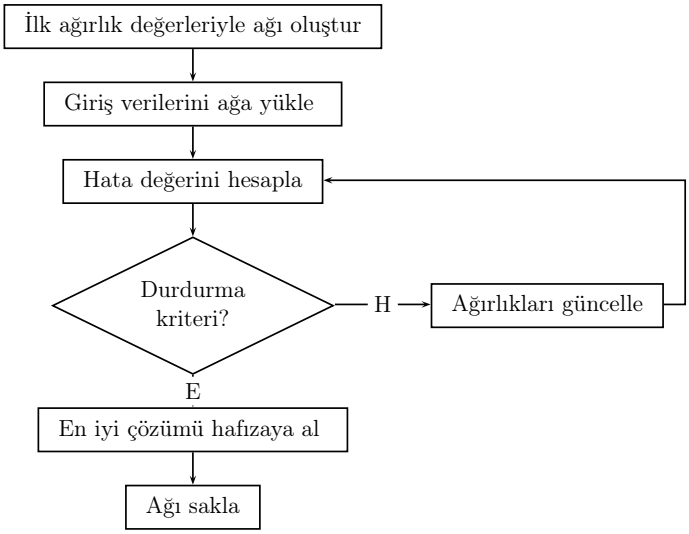
2.4. Yapay Sinir Ağlarının Eğitilmesi

Yapay sinir ağlarının en önemli özelliği öğrenebilme yeteneğine sahip olmalarıdır. Bir sinir ağında öğrenme, verilen bir girdi vektör seti ile onlara karşılık gelen çıktı değerlerine karşılık hesaplanan çıktı değerleri ve beklenen çıktı değerlerinin birbirine yakınlaşmasını sağlayacak olan nöronlar arası optimum ağırlık değerlerine ulaşmaktır. Bilgi, bağlantılarda ağırlıklar şeklinde dağıtılmakta ve tek bir bağlantı anlamlı bir bilgi ifade etmeyeceğinden anlamlı bir bilgi oluşturmak için işlem

elemanlarından oluşan bir bağlantı grubu gerekmektedir. Problemin en iyi çözümü için ağın bağlantıları doğru ağırlık değerlerine sahip olmaları gerekir. Öğrenme, ağırlık değerlerinin değişimini tanımlayan bir öğrenme kuralına dayanır. Geliştirilen bir çok öğrenme kuralı vardır [32]. Öğrenme kuralının temel ilkesi, öğrenme stratejisiyle tanımlanır ve literatürde üç tip öğrenme stratejisinden söz edilir; denetimli öğrenme, destekli öğrenme ve denetimsiz öğrenme.

Denetimli öğrenmede ağı eğitmek için bir öğretici gerekir. Öğretici, çıktı katmanında ağ kararının ne olması gerektiğini belirler [33]. Diğer yandan öğrenmede kullanılacak olan örneklerin seçimi de öğretici tarafından yapılır. Girdiler ve doğru çıktı örnekleri ağa verilir. Ağ, girdiyi işleyerek çıktıyı üretir ve üretilen çıktıyı dışarıdan belirlenen çıktı ile karşılaştırır. Ağ tarafından elde edilen çıkış, tecrübe ile elde edilen çıkış ile başka bir ifadeyle gerçek çıkış ile karşılaştırılarak hata değeri hesaplanır. Her defasında bağlantılardaki ağırlıklar, daha iyi çıktıyı üretmek amacıyla yeniden ayarlanarak işlem kabul edilebilir bir hata düzeyine erişinceye kadar devam eder.

Destekli öğrenmede de bir öğreticiye ihtiyaç vardır. Ancak, öğretici çıktının ne olması gerektiğini ağa vermez; ağın ürettiği çıktı değerlerinin doğru veya yanlış olduğunu bildirir ve ağdaki ağırlıklar bu şekilde ayarlanmaya çalışılır [34].



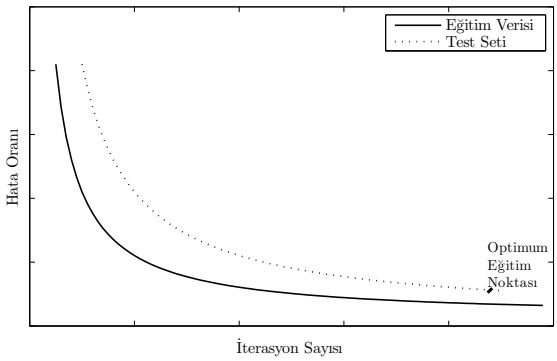
Şekil 2.6. Yapay sinir ağlarının eğitilmesi temel akış diyagramı.

Denetimsiz öğrenme diğerlerinin aksine bir öğreticiye gerek duymaz. Bu stratejide ağ, girdi ve çıktı arasındaki eşleştirmeyi düzenlemek için kendi kararlarını geliştirir. Bu nedenle, denetimsiz öğrenme stratejisini kullanan ağlar, kendi kendine organize olan ağlar olarak da tanımlanırlar [35]. Yapay sinir ağlarında bilgi temsil edilirken ağa verilen öğrenme seti, problemin öğretilmesinde kullanılan girdi ve çıktılardan oluşan bir settir. Denetimli öğrenmede, hedeflenen çıktılar ağa verilirken destekli öğrenme ve denetimsiz öğrenmede verilmezler.

Öğrenme işlemi için bir eğitim algoritması kullanılır ve ağırlıkların nasıl düzenleneceği bu algoritmalar tarafından belirlenir. Şekil 2.6'de verilen YSA eğitimi akış işlemlerinde hesaplanan hata değerine göre ağırlıkların güncellemesi adımında eğitim algoritmasına ihtiyaç duyulur. Kullanılan öğrenme algoritmaları ile hata azaltılıp gerçek çıkış ile uyumu en yüksek çıkış değerinin elde edilmesi öğrenme sürecinin amacıdır. Bu amaçla, sistem içerisinde ağırlıklar her bir çevrimde yenilenerek hata azaltılmaya çalışılır. Eğer yapay sinir ağı verilen giriş-çıkış çiftleriyle amaca ulaşmış ise ağırlık değerleri saklanır. Ağırlıkların sürekli yenilenip istenilen sonuca ulaşılana kadar geçen işleme "öğrenme" adı verilir [36]. Yapay sinir ağı öğrendikten sonra daha önce verilmeyen girişler verilip ağ çıkışıyla gerçek çıkışın yaklaşımı incelenir ki verilen bu girişlere "test seti" denir. Eğer yeni verilen örneklere de doğru yaklaşıyorsa sinir ağı başarılı bir şekilde öğrenmiş sayılır. Ağ tarafından öngörülen çıkış ile istenen çıkış arasındaki hata oranı, daha önce belirlenen bir sınıra erişmiş ise, ağın problemi yeterince kavradığı kabul edilir. Şekil 2.7'de eğitim seti ve test setinde jenerasyon arttıkça hatanın belirli bir noktaya kadar düştüğü, daha sonra da aynı düzeyde sabit kaldığı, artık hatada azalma olmadığı noktanın da optimum eğitim noktası olduğunu gösteren örnek eğitim eğrisi verilmiştir [37].

2.4.1. Eğitim Algoritmaları

Yapay sinir ağlarının eğitilmesinde kullanılmak üzere bir çok eğitim algoritması geliştirilmiştir. Öğrenme algoritmaları ile ilgili çalışmalarının ve kullanımlarının anlatımından önce geleneksel öğrenme algoritmalarının çoğunluğunun Hebb, Delta, Kohonen ve Hopfield ismindeki dört farklı öğrenme kuralından esinlenerek geliştirildiğini belirtmek gerekir [21]. **Hebb Kuralı:** Kuralın temelinde, bir



Şekil 2.7. Yapay sinir ağlarının eğitilmesi süreci için örnek optimum eğitim eğrisi.

nöron diğer bir nörondan giriş alıyorsa ve her iki nöron da aktif ise (matematiksel olarak aynı işarete sahip ise), nöronlar arasındaki ağırlıklar kuvvetlendirilir mantığı yatmaktadır. **Hopfield Kuralı:** Eğer istenilen çıkış ve girişin her ikisi de aktifse veya her ikisi de aktif değil ise, öğrenme oranı tarafından bağlantı ağırlığı arttırılır, diğer durumlarda ise azaltılır. **Delta Kuralı:** En çok kullanılan öğrenme kurallarından birisidir. Nöronun gerçek çıkışı ile, istenilen çıkış değeri arasındaki farkı azaltan bu öğrenme kuralı, giriş bağlantılarını güçlendiren ve sürekli olarak değiştiren bir düşünceye dayanmaktadır. Bu kural, bağlantı ağırlık değerlerinin değiştirilmesiyle ortalama karesel hatayı düşürme prensibine dayanır ve hata, çıkış katmanından giriş katmanına doğru geri yayılarak azaltılmaya çalışılır. Bu sebepten Delta kuralı, geri yayılım veya en küçük ortalama karesel öğrenme kuralı olarak da adlandırılmaktadır. **Kohonen Öğrenme Kuralı:** Biyolojik sistemlerdeki öğrenmeden esinlenen Kohonen tarafından geliştirilen bu kuralda, nöronlar öğrenmek için yarışırlar ve kazanan nöronun ağırlıkları güncellenir. Bu kuralın temelinde kazanan hepsini alır mantığı yatmakta ve en büyük çıkışa sahip nöron kazanmaktadır. Hedef çıkışa gereksinim duyulmadığı için danışmansız bir öğrenme metodudur [38].

Yapay sinir ağlarının eğitimi geleneksel olarak türeve dayalı algoritmalar ile yapılmaktadır. Ancak, bu tekniklerin hata yüzey şekline bağımlılığı, ağırlık bağlantılarının ve parametrelerin ilk değer alması gerekliliği gibi bir takım dezavantajları vardır. Eğer hata yüzeyi çok modlu ise türeve dayalı algoritmalar genellikle yerel minumuma takılabilmektedirler. Bu tür algoritmalarda bir

diğer sorunda farklılaşmış hata fonksiyonları içermek durumunda olmalarıdır. Yakınsamaya ulaşılmadan önce eğer çıktı bazı noktalarda en uç değerine itilirse saturasyon (aşırı doyma, doygunluk) oluşabilir; bunun için türev önemli ağırlık değişimleri yapmak için yetersiz kalabilmekte ve ağın yanlış yerel minimuma yerleşmesine sebep olabilmektedir. Türev tabanlı algoritmaların dezavantajlarının üstesinden gelebilmek için, bir çok küresel optimizasyon algoritması geliştirilmiş veya bu amaçla birçok optimizasyon algoritması kullanılmıştır. Genetik algoritmalar, parçacık sürü optimizasyon algoritması, diferansiyel gelişim algoritması ve evrimsel programlama algoritmaları yapay sinir ağlarının eğitiminde yaygın kullanılan küresel optimizasyon algoritmalarındandır [39]. Ayrıca, algoritmaların tek başlarına kullanıldıklarında karşılaşılan eksiklikleri gidermek amacıyla bazı karma teknikler, geleneksel teknikler ile optimizasyon tekniklerinin birleştirilmesiyle hazırlanan modellerde sinir ağlarının eğitiminde kullanılmaktadır [40].

Bu tez çalışmasında yapay sinir ağlarının eğitilmesinde kullanılan bütün teknikler veya yaklaşımlar ele alınmamış, her yapay sinir ağı yapısı için yaygınca kullanılan tekniklere yer verilmiştir. Bu amaçla, türeve dayalı algoritmalardan eğim iniş ve Levenberg-Marquardt geri yayılım algoritmaları; evrimsel algoritmalardan genetik algoritma ve diferansiyel gelişim algoritması; sürü zekasına dayalı algoritmalardan parçacık sürü optimizasyon algoritmaları anlatılmıştır.

2.4.2. Klasik Algoritmalar

2.4.2.1. Geri Yayılım Algoritması

İlk olarak Rumelhart ve arkadaşları (1986) tarafından tanıtılan hatayı geriye yayılım öğrenme modeli, yapay sinir ağı öğrenme modelleri arasında en çok kullanılanlardandır. Geri yayılım modelinin keşfi yapay sinir ağları için çok önemli tarihsel gelişmelerden biri olmuştur [41]. Hatayı geriye yayma ağları, temel olarak girdi örneği ile birlikte ağa sunulmak üzere sistemin istenen çıktısını gerektiren denetimli öğrenme stratejisini kullanan, birçok katmandan oluşan tamamıyla bağlantılı ileri beslemeli ağlardır. Başka bir deyişle, bir katmandaki her nöron alt katmandaki bütün nöronlara bağlanır. Hiçbir geri besleme bağlantısı ve aynı

katmandaki nöronlar arasında da hiçbir bağlantı yoktur.

Geriye yayılım öğrenme modeli çıkıştan alınan değerler ile beklenen değerlerin durumlarına göre hatayı çıkıştan geriye doğru azaltmaya çalıştığından dolayı bu ismi almıştır. Örnek değerler sisteme önceden öğretilmektedir ve yeni değerler için sonuç vermesi beklenilmektedir. Bir geri yayılım algoritmasında öğrenme şu şekilde gerçekleşmektedir:

1: **repeat**
2: Eğitim kümesinden rasgele bir sonraki örneği seçme ve ağ girişine giriş vektörü uygulama
3: Ağın çıkışını hesaplama
4: Ağın çıkışı ile hedef çıkış arasındaki hatayı hesaplama
5: Ağın ağırlıklarını geriye gönderilen hata yardımıyla güncelleme
6: **until** Durma kriteri

Geri yayılım algoritmasında (BP) öğrenme mekanizması, ağın gerçek ve istenen çıktıları arasındaki hatayı enazlayan iteratif eğim düşme yöntemine dayanır. Bu algoritmada hatalar çıkıştan geriye doğru gönderilerek ağırlıkların ayarlanmasıyla hata azaltılmaya çalışılmaktadır. Öğrenme kuralında ağ çıkışında hesaplanan hata, ağırlıkların yeni değerlerinin hesaplanmasında kullanılmaktadır. Eğitimin k'inci yinelemesinde YSA'nın çıkışındaki i'inci nöronunun çıkış değeri y_i, beklenen değer d_i ve i nöronunun hata işareti e_i olarak temsil edilir ise, hata değerinin hesaplanması Eşitlik 2.3 ile tanımlanır:

$$e_i = (d_i - y_i(k)) \tag{2.3}$$

Geri yayılımlı yapay sinir ağının genel yapısında bir katmandan bir başka katmana, aradaki katmanı atlayarak geçebilmek mümkün değildir. Bir giriş verisi ağın ilk katmanında yer alan nöronlara uygulandığında en üst katman olan çıkış katmanına erişinceye kadar, bu veri üzerinde çeşitli işlemler gerçekleştirilir. Bu işlemlerin sonucunda elde edilen çıktı, olması gereken çıktı ile karşılaştırılır ve uygunluk fonksiyonu Eşitlik 2.4 ile tanımlanır:

$$\mathrm{E} = \frac{1}{2}\sum_{\mathrm{i}} (\mathrm{e_i(k)})^2 = \frac{1}{2}\sum_{i} (d_i - y_i(k))^2 \tag{2.4}$$

Hesaplanan ve olması gereken değerler arasındaki fark, her çıktı düğümü için bir hata sinyali olarak hesaplanır. Hesaplanan hata sinyalleri, her çıktı düğümüne karşı gelen ara katmandaki düğümlere aktarılır. Böylece, ara katmandaki düğümlerin her biri toplam hatanın sadece hesaplanan bir kısmını içerir. Bu süreç her katmandaki düğümler toplam hatanın belirli bir kısmını içerecek şekilde giriş katmanına kadar tekrarlanır. Elde edilen hata sinyalleri temel alınarak, bağlantı ağırlıkları her nöronda yeniden düzenlenir. Bu düzenleme tüm verilerin kodlanabileceği bir duruma ağın yakınsamasını sağlar ve ağırlık değerlerinin değişme miktarı gradyan açılma yöntemi olarak adlandırılan Eşitlik 2.5 denkleminden yola çıkılarak gerçekleştirilir [42].

$$\Delta w_{ij} = -\eta \frac{\partial E(\vec{w})}{\partial w_{ij}} \tag{2.5}$$

Burada η öğrenme katsayısıdır. Geri yayılım algoritmasında her yineleme, ileri yayılım ve geri yayılım olmak üzere iki aşamadan oluşmaktadır. İleri yayılım aşamasında YSA'nın o andaki durumunda YSA'ya uygulanan giriş işaretlerine karşı YSA'nın çıkışlarında oluşan değerler bulunur. Geri yayılım aşamasında, çıkışlarda oluşan hatalardan yola çıkılarak devredeki ağırlıkların yeniden düzenlenmesi yapılmaktadır. YSA'daki her ağırlık değişimi Eşitlik 2.6 denklemi ile yapılmaktadır.

$$\Delta w_j = \eta \delta_j y_i \tag{2.6}$$

δ_j, çıkış katmanı sinirleri için

$$\delta_i = e_j(k) f_j` \tag{2.7}$$

gizli katmanlarda bulunan sinirler için ise

$$\delta_j = f_j^{'} \sum_m \delta_m w_{mj} \tag{2.8}$$

olarak tanımlanmır. f_j, j sinirinin aktivasyon fonksiyonudur. Bu tanımlar ile hata işaretlerinin çıkıştan girişe doğru akışı, ileri yayılma aşamasında işaretlerin ileri doğru akışına benzetilmiştir. İleri besleme safhasında, giriş tabakasındaki nöronlar veri değerlerini doğrudan gizli katmana iletirler. Gizli katmandaki her bir nöron kendi giriş değerlerini ağırlıklandırarak toplam değeri hesaplar ve bunları bir aktivasyon fonksiyonu ile işleyerek bir ileriki tabakaya veya doğrudan çıkış katmanına iletirler. Katmanlar arasındaki ağırlıklar başlangıçta rasgele küçük rakamlardan seçilir. Çıkış katmanındaki herbir nöron ağırlıklandırılmış değeri hesaplandıktan sonra bu değer yine aktivasyon fonksiyonu ile karşılaştırılarak mevcut hata minimize edilmeye çalışılır. Hata değeri belli bir mertebeye ininceye kadar iterasyon işlemine devam edilir ve böylece ağın eğitim aşaması tamamlanmış olur. Katmanlar arasındaki bağlantılardaki ağırlık değerleri eğitimi tamamlamış ağdan alınarak test safhasında kullanılmak üzere saklanır [43].

2.4.2.2. Levenberg-Marquardt Algoritması

Geri yayılım algoritmasının yakınsama hızı düşüktür ve yerel minumuma yakalanma riski oldukça yüksektir. Bu gibi durumların ortaya çıkabileceği problemler için genelde Levenberg-Marquardt (LM) algoritması tercih edilmektedir. BP algoritması birinci dereceden türev bilgisiyle hatayı azaltmaya çalışırken LM algoritması Newton metodu ile BP metodu arasında ara değerler hesaplar ve ikinci dereceden bir ifadeyle yani ikinci dereceden türev bilgisiyle hatayı azaltmaya çalışır [44, 45]. LM metodu, maksimum komşuluk fikri üzerine kurulmuş en küçük kareler hesaplama metodudur [46].

Algoritma temel olarak sistemin jakobyen matris denklemi ile tanımlanır. Jakobyen matris tanım itibari ile vektör değerli bir fonksiyonun tüm birinci dereceden kısmi türevlerinin matrisidir. Yapay sinir ağlarında bu matris N (sistemi eğitirken girdiğimiz örnek sayısı) x W (ağdaki toplam ağırlıkların ve öndeğerlerin toplamı) boyutlu bir matris olmaktadır ve her ağırlığa göre her bir çıkışın kısmi türevi alınarak

oluşturulmaktadır [47]:

$$\vec{h}(x_1, x_2, \ldots, x_N) = \begin{bmatrix} h_1(x_1, x_2, \ldots, x_N) \\ h_2(x_1, x_2, \ldots, x_N) \\ \vdots \\ h_M(x_1, x_2, \ldots, x_N) \end{bmatrix} \tag{2.9}$$

LM algoritmasının adımları:

1: **repeat**
2: Jakobyeni hesapla
3: Hata gradyanını ($g = J^T E$) hesapla
4: Hessian değerini ($H = J^T J$) bul
5: δ değerini bulmak için (H + λ I)δ = g denklemini çöz
6: δ değerini kullanarak ağdaki ağırlıkları güncelle
7: **until** Durma kriteri

$E(\vec{w})$'nin bir amaç hata fonksiyonu olduğu durumda $e_i^2(\vec{w})$ (Eşitlik 2.10) ile ifade edilebilir.

$$E(\vec{w}) = \sum_i e_i^2(\vec{w}) = \|f(\vec{w})\| \tag{2.10}$$

burada $(\vec{w})$ ağırlıkları ifade eder ve amaç fonksiyonu f'in ve onun jakobiyeni J'nin bir noktada $(\vec{w})$ bildiği kabul edilir. LM öğrenme algoritmasında hedef, parametre vektörü $(\vec{w})$'nin, $E(\vec{w})$ minimum noktasında iken bulunmasıdır. LM'nin kullanılmasıyla yeni vektör $(\vec{w}_{k+1})$, vektör $(\vec{w})$'dan Eşitlik 2.11 ile hesaplanır.

$$\vec{w}_{k+1} = \vec{w} + \delta\vec{w} \tag{2.11}$$

burada $\delta\vec{w}$ Eşitlik 2.12 ile verilen denklemden elde edilir.

$$(J_k^T J_k + \lambda I)\delta w_k = -J_k^T f(w_k) \tag{2.12}$$

burada J_k, f'in $(\vec{w})$ değerlendirilmiş jakobyeni; λ, Levenberg parametresi; δ

ağırlıklarda değişimi sağlayan ağırlık güncelleme vektörü; I, birim matrisi ve $J^T J$ matrisi Hessian matrisi olarak gösterilebilir. Hessian matrisini genelde hesaplamak yerine hesaplanacak değere çok yakın değerler veren $J^T J$ matrisiyle yakın değerler alınır.

2.4.3. Evrimsel Algoritmalar

Evrimsel algoritmalar, evrimsel hesaplama temelli doğal gelişimi simule eden algoritmalardır. Bu metodlar popülasyon tabanlı olup problemleri çözmek için rasgele varyasyon ve seçim kombinasyonunu kullanırlar. Evrimsel hesaplama paradigması çözülmesinin güç olduğu düşünülen karmaşık problemlerin çözülmesinde yaygınca kullanılmaktadır. Başarıyla uygulandığı birkaç kategoriye örnek olarak: planlama, tasarım, simulasyon ve tanımlama, kontrol, sınıflandırma gibi alanlar verilebilir [48–50].

Evrimsel hesaplamada işlemlerin yürütülmesinden mutasyon, üreme ve seçme işlemleri sorumludur [51]. Üreme, bireysel bir genetik programın kendi soyuna transferinden etkilenir. Bu yolla iyi bir organizmadan alınmış genetik özellikler muhafaza edilmeye calışılır. Genetik yapının transferi hatalara maruz kalabilir, daha başka bir ifadeyle değişimler meydana gelebilir. Bu hatalar ya da değişimler mutasyon olarak isimlendirilir ki bu organizmanın gelişimini artırır ya da engeller. Çevredeki kaynaklar sınırlandığında, organizma popülasyonları sıçrama olmadan genişleyemez ve sonuçlanmak zorunda kalır; ancak organizma hayatta kalabilmek amacıyla rakiplerinden daha iyi olmak için çalışmalıdır. Rekabet seçme işlemine öncülük eder ve dolayısıyla daha başarılı bireyler hayatta kalarak yeni nesiller üretirlerken başarısızların nesli ise tükenmeye başlar. Seçme işlemi sayesinde çok sayıda ki çözümün içinden iyileri seçilir ve kötüleri elenir. Evrimsel hesaplama tabanlı birçok farklı yaklaşım vardır ve bunlara en iyi örnekler: Evrimsel Algoritmalar (EAs), Evrimsel Stratejiler (ES), Evrimsel Programlama (EP), Genetik Algoritmalar (GAs) ve Genetik Programlama (GP) verilebilir [52] .

Yapay sinir ağlarında optimizasyonun amacı ağın mimarisini ya da ağın bağlantı ağırlıklarını optimize ederek hata fonksiyonunu minimize etmektir. Evrimsel

algoritmalar ile ağırlıkların optimizasyonun da altı çizilecek esas düşünce; YSA'ların ağırlık matrislerini yorumlamak, mutasyon ve çaprazlama gibi operasyonlar vasıtasıyla ağırlıkları değiştirmek ve seçme işleminde yol gösterici olan uygunluk ölçüsü olarak YSA'ların çıkışında üretilen hata değerini kullanmaktır. Bu olgu, evrimsel eğitim döngüsünün ortaya çıkmasına sebep olmuştur. Evrimsel eğitim döngüsü, ilk popülasyonun rasgele üretilmesinin ardından her çevrimde çaprazlama, mutasyon, uygunluk tabanlı seçme operatörleri vasıtasıyla YSA'ların gelecek popülasyonlarının düzenlenmesi işlemlerinin istenilen sonuç elde edilene ya da belirlenen çevrim sayısı tamamlana kadar devam ettirilmesi şeklinde tanımlanabilir.

2.4.3.1. Genetik Algoritma

Genetik algoritma, doğadaki evrim mekanizmasını örnek alan bir optimizasyon tekniğidir ve bir veri grubundan özel bir veriyi bulmak için kullanılır. Genetik algoritma, 1970'lerin başında John Holland tarafından evrimsel algoritmaların bir formu şeklinde tanımlanmıştır. Genetik operatörler ve doğal seçime dayalı olarak geliştirilmiş iteratif ve ihtimal hesaplamalarına dayanan bir arama metodudur [53].

Genetik algoritmalar doğada geçerli olan iyilerin yaşaması kuralına dayanarak sürekli iyileşen çözümler üretmeye çalışır. Bunun için iyilerin ne olduğunu belirleyen bir uygunluk fonksiyonu ve yeni çözümler üretmek için çaprazlama ve değiştirme (mutasyon) gibi operatörler kullanılır. Akış diagramı Şekil 2.8'de verilen genetik algoritmaların adımları algoritmik formda ifade edildiğinde şu şekilde olur:

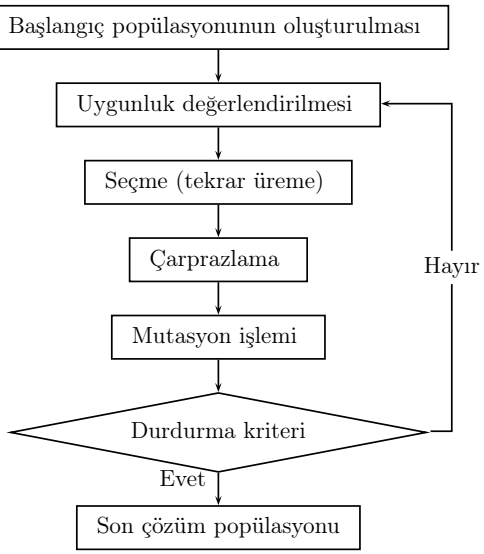
1: Başlangıç popülasyonunun oluşturulması
2: **repeat**
3: Değerlendirme
4: Seçim
5: Çaprazlama
6: Mutasyon
7: **until** (Durma kriteri)

Genetik algoritmaları (GA) diğer metodlardan ayıran noktalar şu şekilde sıralanabilir: GA, sadece bir arama noktası değil, bir grup arama noktası (adaylar)

üzerinde çalışır. Yani arama uzayında, bölgesel değil küresel arama yaparak sonuca ulaşmaya çalışır. GA, arama uzayında bireylerin uygunluk değerini bulmak için sadece uygunluk fonksiyonuna ihtiyaç duyar. Böylelikle sonuca ulaşmak için türev ve benzeri başka bilgi ve kabul kullanmaya gerek duymaz. Bireyleri seçme ve çaprazlama aşamalarında deterministik kurallar değil olasılık temelli kurallar kullanır. Çözümün uygunluğu onun seçilme şansını arttırır, fakat bunu garanti etmez. Seçim de ilk grubun oluşturulması gibi rasgeledir ancak bu rasgelelik çözümlerin uygunluğuna bağlıdır [54].

Genetik algoritma başlangıçta bilinmeyen bir arama uzayından toplanan bilgileri daha sonraki aramalarda alt arama uzaylarına yönlendirmek için kullanır ve yeni nesilleri oluşturabilmek için 3 aşamadan geçer:

- Eski nesildeki her bir bireyin uygunluk değerlerini hesaplama.
- Bireyleri, uygunluk değerini göz önüne alarak (uygunluk fonksiyonunu kullanarak) seçme.
- Seçilen bireyleri, çaprazlama (crossover), mutasyon (mutation) gibi genetik operatörler kullanarak uyuşturma.



Şekil 2.8. Genetik algoritmanın temel akış diyagramı.

2.4.3.2. Diferansiyel Gelişim Algoritması

Diferansiyel Gelişim Algoritması (DE), Price ve Storn tarafından 1995 yılında geliştirilmiş; özellikle sürekli verilerin söz konusu olduğu problemlerde etkin sonuçlar verebilen, işleyiş ve operatörleri itibariyle genetik algoritmaya dayanan ve popülasyon temelli sezgisel optimizasyon tekniğidir [55, 56]. DE algoritması da aynı anda birçok noktada araştırma yapmakta ve jenerasyonlar boyunca operatörler yardımıyla problemin çözümü için daha iyi sonuçlar araştırmaktadır. GA'daki çaprazlama, mutasyon ve seçim operatörleri DE'de de kullanılmaktadır. Farklı olarak her bir operatör tüm popülasyona sırayla uygulanmaktadır. Kromozomlar tek tek ele alınmakta, rasgele seçilen üç farklı kromozom kullanılarak yeni bir birey elde edilmektedir. Bu işlemler sırasında mutasyon ve çaprazlama operatörleri kullanılmaktadır. Mevcut kromozomla elde edilen yeni kromozomun uygunlukları karşılaştırılarak uygunluğu daha iyi olan birey, yeni birey olarak bir sonraki popülasyona aktarılmaktadır. Böylelikle seçim operatörü de kullanılmış olmaktadır. Üretilen çözümlerin kalitesi, amaç fonksiyonuna ürettikleri değer (uygunluk değeri) ile ölçülmektedir. DE'nin diger sezgisel algoritmalara önemli bir üstünlüğü de kolayca kodlanabilmesi ve anlaşılabilirliğidir [57]. Diğer algoritmalar için binlerle ifade edilen satırlardan oluşan kodlar söz konusu iken, DE için yaklasık 50 satırlık kod yeterli olmaktadır [58].

Basit bir DE algoritmasının temel kontrol parametreleri; popülasyon büyüklüğü, maksimum jenerasyon sayısı, çaprazlama oranı, ölçekleme faktörü ve kombinasyon katsayısıdır. Bunlarla birlikte mutasyon ve çaprazlama stratejilerine karar verilmelidir. Şekil 2.9'de temel akış diagramı verilen DE algoritmasının performansı üzerinde bu parametrelerin alacağı değerler oldukça etkilidir [58].

Algoritmanın temel adımları ve birimleri şu şekildedir:

1: Popülasyonun başlatılması
2: Değerlendirme
3: **repeat**

4: Mutasyon

5: Çaprazlama

6: Değerlendirme

7: Seçim

8: **until** (Durma kriteri)

Mutasyon: Çözüm vektörünün her bir elemanı mutasyona uğratılır. x_i çözümü mutasyona uğratıldığında elde edilen çözüm (2.13) ile ifade edilir:

$$\hat{x}_i = x_i + K(x_{r_1} - x_i) + F(x_{r_3} - x_{r_2}) \tag{2.13}$$

Burada, x_{r1}, x_{r2} ve x_{r3} Eşitlik 2.14'ü sağlamak kaydıyla rasgele seçilen çözüm vektörleridir. F ölçekleme faktörü [0,1] arasında değer alır ve x_{r3} ile x_{r2} farkından ileri gelen adım büyüklüğünün ölçeklenmesini gösterir, K katsayısı x_{r1} ile mevcut birey x_i arasında meydana gelen birleşmenin seviyesini temsil eder ve $\hat{x}_i$, x_i çözümünün mutasyona uğratılmasıyla elde edilen çözümdür.

$$x_{r_1}, x_{r_2}, x_{r_3} \,|r_1 \neq r_2 \neq r_3 \neq i \tag{2.14}$$

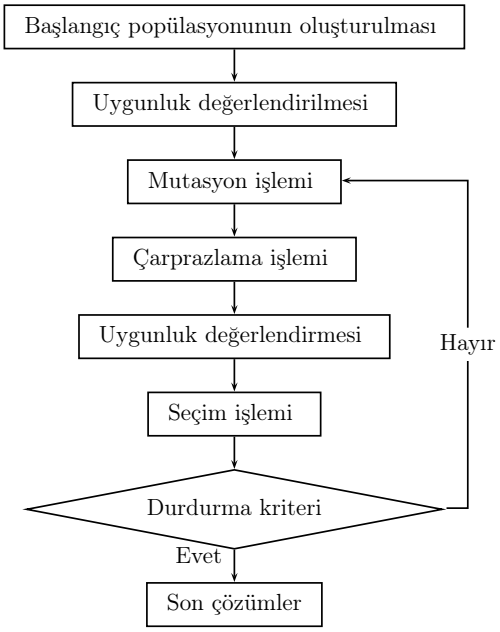
Çaprazlama: Ebeveyn çözüm x_i mutasyona uğramış çözüm $\hat{x}_i$ ile (2.15) kuralı kullanılarak çaprazlama işlemine tutulur ve deneme çözüm y_i elde edilir.

$$y_i^j = \begin{cases} \hat{x}_i^j & R_j \leq CR \\ x_i^j & R_j > CR \end{cases} \tag{2.15}$$

Burada, CR çaprazlama oranı, R_j [0,1] arasında rasgele bir reel sayı ve j çözümün j. parametresidir.

Seçim: Uygunluğu yüksek olan çözümün yeni jenerasyona atanmasıdır. Seçim operatörüne ait işlem Eşitlik 2.16'de verilmiştir:

$$x_{i+1} = \begin{cases} y_i & f(y_i) \leq f(x_i) \\ x_i & f(y_i) > f(x_i) \end{cases} \tag{2.16}$$



Şekil 2.9. Diferansiyel gelişim algoritmasının temel akış diyagramı.

2.4.4. Parçacık Sürü Optimizasyon Algoritması

Parçacık sürü optimizasyon algoritması (PSO) balık ve kuş sürülerinin yiyecek kaynaklarına doğru yaptıkları davranışlardan esinlenilerek 1995'de Kennedy ve Eberhart tarafından geliştirilmiştir [59]. PSO algoritması özellikle sürekli parametre optimizasyonunda ve çok boyutlu arama alanlarında yaygın olarak kullanılmaktadır. PSO, birbirleriyle etkileşim içinde olan büyük boyutlu canlı organizmaların davranışlarından esinlenilerek ortaya çıkan bir algoritmadır ve aramaları bir parçacık sürüsü kullanarak yönetir. Parçacık sürüsünü oluşturan parçacıklar evrimsel algoritmadaki bireylere karşılık gelir. PSO'da her bir aday çözüm parçacıkla, popülasyon ise sürü ile ifade edilir. İlk olarak, parçacık popülasyonu rasgele olarak üretilir. Her parçacık potansiyel çözümü temsil eder. Parçacıkların pozisyonu ise pozisyon vektörü ile gösterilir. Bir parçacık sürüsü rasgele belirlenen başlangıç pozisyonlarından problem uzayına doğru hareket eder ve hareketleri her bir parçacığın vektörel hareket hızı ile gösterilir. Bu hareketleri sırasında her parçacık kendi en iyi pozisyon bilgisini tutar, bunun yanında sürünün en iyi pozisyon bilgisi

en iyi uygunluk değerine karşılık gelir ve vektör değeriyle gösterilir ve saklanır. PSO modelinde parçacıklar arasındaki sosyal bilgi paylaşımını geliştirme amaç edinilmiştir ve parçacıklar aday çözümün uygunluğunu iteratif olarak hesaplar ve en iyi kaynağın yerini hafızalarına alırlar. Her bir aday çözüm (parçacık) bir sonraki pozisyonunu hız vektörünün değerine göre ayarlar (Eşitlik 2.17):

$$\vec{x}(t+1) = \vec{x}(t) + \vec{v}(t+1) \tag{2.17}$$

Hız değerinin güncellenmesinin hesaplanmasında, kendi tecrübesini (yerel en iyi değeri) ve sürü tecrübesini (küresel en iyi değeri) kullanır. Bu şekilde hız değişimi parçacığın, hem kendi en iyi pozisyonunun hem de küresel en iyi pozisyonun çevresinde arama yapmasını sağlar. Güncellenmiş bu hız vektörüne göre her bir parçacık kendi pozisyonunu Eşitlik 2.18'e göre değiştirir:

$$\begin{aligned} \vec{v}(t+1) = \quad & \omega\vec{v}(t) + \phi_1 \text{rand}(0,1)(\vec{p}(t) - \vec{x}(t)) \\ & + \phi_2 \text{rand}(0,1)(\vec{g}(t) - \vec{x}(t)) \end{aligned} \tag{2.18}$$

ω, yeni hız değerinin hesaplanmasında bir önceki hız değerinin $\vec{v}(t)$ etkisini belirleyen ağırlıktır. ϕ_1 ve ϕ_2 sırasıyla $\vec{p}(t)$ ve $\vec{g}(t)$'nin önem derecelerini belirlerler. $(\vec{p}(t))$ yerel en iyi ve $(\vec{g}(t))$ küresel en iyi çözümlerdir.

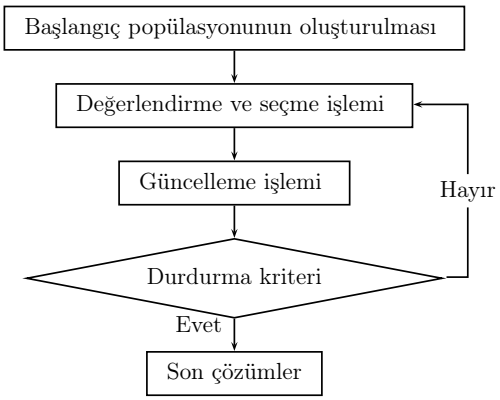
PSO algoritmasının temel blok şeması Şekil 2.10'de verilmiştir. Başlangıçta parametrelerin, uygunluk (amaç) fonksiyonunun ve durdurma kriterinin tanımlanması gerekir. Daha sonra, parametrelerin temsilinin, başlangıç popülasyonunun oluşturulmasının, uygunluk değerlendirilmesinin, seçme ve güncelleme işlemlerinin yapılması gerekir. Algoritmanın temel adımları aşağıdaki şekilde özetlenebilir:

1: Popülasyonun başlatılması
2: **repeat**
3: Parçacıkların uygunluk değerlerinin hesaplanması
4: Sürüdeki en iyi parçacıkların modifiye edilmesi
5: En iyi parçacığın belirlenmesi

6: Parçacıkların hızlarının hesaplanması

7: Parçacıkların pozisyonlarının güncellenmesi

8: **until** (Durma kriteri)



Şekil 2.10. Parçacık sürü optimizasyon algoritmasının temel akış diyagramı.

3. BÖLÜM

YAPAY ARI KOLONİSİ ALGORİTMASI (ARTIFICIAL BEE COLONY ALGORITHM)

3.1. Yapay Arı Kolonisi Algoritması

Kuş sürülerinin havada süzülmeleri ve farklı şekiller almaları, karıncaların yiyecek aramaları, balık sürülerinin beraberce yüzmeleri ve kaçışmaları, bal arılarının danslarla haberleşmeleri gibi sürü halinde gerçekleştirilen davranışları keşfedilmesiyle biyologlar ve bilgisayar uzmanları bu sürülerin davranışları ve davranışlarının modellenmesi arasındaki iletişim mantığınının üzerine çalışmalar yapmaya başlamışlardır. Bu çalışmalar, sürülerin davranışlarının sistem ve modellere uyarlandığı yaklaşımların gelişmesine yol açmıştır. Sürü zekası yaklaşımları adı verilen bu yaklaşımlar optimizasyon problemlerinde, robotik konularında ve askeri uygulamalarda gösterdikleri başarılar ile fazlaca ilgi çekmekte ve bu konu üzerindeki çalışmalar gün geçtikçe artmaktadır. Sürü zekası, kısaca özerk yapıdaki basit bireyler grubunun üst seviye de kolektif bir zeka geliştirmesi şeklinde tanımlanabilir [60].

Sürü zekası doğada karınca, arı, kuş ve balık gibi canlıların bireysel olarak gerçekleştiremediği ancak grup halinde hareket ederek gerçekleştirebildikleri faaliyetleri örnek alır. Sürü zekası temelli algoritmaların en popüler olanlarına Dorigo tarafından 1991'de karıncaların yiyecek arama da gösterdikleri zeki davranışı modelleyen karınca koloni optimizasyon (ant colony optimization) algoritması [61], kuş ve balık sürülerinin davranışlarını temel alan Kennedy ve Eberhart tarafından 1995'de geliştirilmiş parçacık sürü optimizasyon (particle swarm optimization) algoritması [59], doğal bağışıklık sisteminden esinlenerek Farmer ve arkadaşları tarafından 1986 yılında önerilen yapay bağışıklık algoritması (artificial immune

system) [62] ve bakterilerin davranışlarının Passino tarafından 2002 yılında modellendiği bakteri optimizasyon algoritması (bacterial foraging optimization) [63] gösterilebilir.

Arıların davranışlarına dayalı sürü zekası yaklaşımları 2000'li yılların başlarında başlamıştır. Bilim insanları arıların kraliçe arı benzetim modellerine, dans davranışlarına, görev paylaşımlarına, yuva yeri seçimlerine, üreme süreçlerine, navigasyon davranışlarına ve yiyecek kaynakları araştırma davranışlarına dayalı modeller geliştirmişlerdir [64]. Bunlardan arıların yiyecek kaynağı aramalarında ki zeki davranışları modelleyen algoritmalardan en yaygın kullanılanlarına Lucic ve Teodorovic'in arı sistemi (Bee System) adını verdikleri yaklaşım [65], Dell'Orco ve Teodorovic'in gezinti eşleme probleminin çözümü için önerdikleri arı koloni optimizasyonu (Bee Colony Optimization System) [66], Yang'ın iki boyutlu nümerik fonksiyonların çözümüne yönelik sunduğu sanal arı algoritması (virtual bee algorithm) [67] ve Pham ve arkadaşlarının arıların yiyecek arama davranışlarını simule ederek geliştirdikleri arılar algoritması (bees algorithm) [68] örnekleri verilebilir.

Karaboğa 2005 yılında bal arılarının yiyecek (kaynak) arama davranışlarını modelleyerek Yapay Arı Kolonisi (ABC) algoritmasını geliştirmiştir [12]. Bir arada yaşayan arıların, sürünün kalabalıklığına rağmen aralarındaki kusursuz iş bölümü ve disiplinleri sayesinde, kovandaki işlerde hiçbir aksama olmaz ve kovan içinde hiçbir kargaşa yaşanmaz. Belli bir ahenk ve düzen içerisinde kendilerine kaynak arayan bal arılarının davranışlarından esinlenilerek nümerik optimizasyon problemlerinin çözümü içim geliştirilen ABC algoritması çok parametreli nümerik fonksiyonların optimizasyonunda kullanılmış; sınırlamasız problemlerde literatürde yaygın olarak kullanılan genetik algoritma ve diferansiyel gelişim algoritmalarına karşı başarısı Karaboğa ve Akay tarafından ortaya konmuştur [13, 14].

Çok sayıda arının yaşadığı bir kovandaki hemen hemen tüm işlerden işçi arılar sorumludur ve kovandaki düzen işçi arıların üzerlerine düşen sorumlulukları tam olarak yerine getirmeleri ile sağlanır [69]. Arı sürüsünde yiyecek arama üç temel bileşen ile hedeflenebilir: yiyecek kaynakları, belirli yiyecek kaynaklarına gidip gelen

görevli arılar ve gideceği kaynaklar belli olmayan serbest arılar. Kaynağın değeri birçok parametreye bağlı olmakla birlikte basitçe kaynak miktarı yani kaynağın zenginliği ile ifade edilebilir. Görevli arılar, nektarın kovana getirilmesinden ve kovandaki diğer arılara kaynaklarla ilgili bilgilerin taşınmasından sorumludurlar. Serbest arıların bir grubu olan gözcü arılar, görevli arıları izleyerek aldıkları bilgiler ışığında seçtikleri kaynaklara yönelirler. Serbest arıların bir diğer grubu da kaşif arılardır ve kaşif arılar etrafı inceleyerek yeni kaynakların keşfedilmesini sağlarlar [70].

Arıların yukarıda anlatılan yiyecek arama davranışlarını model alarak geliştirilen ABC yaklaşımında üç tür arı bulunmaktadır: görevli arılar, kaşif arılar ve gözcü arılar. Basitlik amacıyla koloninin yarısı görevli, diğeri yarısı da gözcü arılardan oluşmaktadır. Her bir yiyecek kaynağı için bir görevli arı bulunmakta ve kaynağında çıkartılmaya değer yiyecek kalmayan görevli arı kaşif arıya dönüşerek yeni kaynaklar aramaya başlamaktadır.

3.2. Yapay Arı Kolonisi Algoritmasının Çalışması

ABC algoritmasında her bir çevrim (döngü) görevli ve gözcü arıların kaynaklara gönderilerek kaynaklarındaki yiyecek miktarlarının belirlenmesi, kaşif arıların çıkıp çıkmayacağının tespit edilerek çıkmaları durumunda bunların muhtemel yeni kaynaklara gönderilmesini içeren üç temel adımdan oluşur. Modelde, kaynakların yerleri (pozisyonları) çözülmek istenen problem için muhtemel çözümleri, kaynakların miktarları da bu çözümlerin kalitesini temsil etmektedir. Kaynak miktarının yüksek olması yani çözümün kaliteli olması, gözcü arıların bu kaynağa yönelme ihtimallerini arttırmaktadır. ABC algoritmasında ortaya çıkacak kaşif arı sayısı "limit parametresi" ile kontrol edilmektedir. Eğer bir kaynağı temsil eden çözüm belli bir sayıda denemenin ardından hala iyileştirilemiyorsa o kaynak görevli arısı tarafından bırakılır ve bu görevli arı kaşif arıya dönüşür. Çözümleri geliştirme esnasında müsade edilen başarısız denemelerin sayısı her çözüm için o çözümün geliştirememe sayacı ile kontrol edilir.

Algoritmanın temel adımları şu şekilde özetlenebilir;

Başlangıç yiyecek kaynaklarını oluştur (Başlangıç pozisyonları)

REPEAT

Görevli arıları kaynaklarına gönder ve kaynaklardaki yiyecek miktarlarını belirle

Kaynakların gözcü arılar tarafından tercih edilme ihtimallerini hesapla

Gözcü arıları seçtikleri kaynaklara gönder ve yiyecek miktarlarını belirle

Arıların gitmekten vazgeçtiği kaynağa gidişleri durdur

Kaşif arıları rasgele olarak yeni kaynak aramak üzere gönder

En iyi çözümü sakla

UNTIL (gereksinimler karşılanana kadar)

ABC algoritmasının temel aldığı model de her bir kaynağın nektarı sadece bir görevli arı tarafından alınır yani görevli arıların sayısı toplam yiyecek kaynağı sayısına eşittir. Gözcü arıların sayısı da görevli arı sayısına eşittir. Algoritma ilk adımda rasgele dağılmış çözüm popülasyonunu Eşitlik 3.1 aracılığıyla üretir.

$$x_{ij} = x^j_{\min} + \text{rand}(0,1)(x^j_{\max} - x^j_{\min}) \tag{3.1}$$

Burada, x_i popülasyonda bir çözümü göstermekte, $i = \{1, 2, ..., SN\}$ ve SN popülasyon boyutunu, $j = \{1, 2, ..., D\}$ ve D optimizasyon parametre sayısını temsil etmektedir. Parametreler için $x^j_{\min}$, j. parametrenin alt sınırı ve $x^j_{\max}$ ise j. parametrenin üst sınırıdır.

ABC algoritmasının başlangıç aşamasında çözümlerin rasgele oluşturulmasından sonra görevli arı, gözcü arı ve kaşif arı süreçlerinden geçirilerek çözümler iyileştirilmeye çalışılır.

3.2.1. Görevli Arıların Yiyecek Kaynağı Bölgelerine Gönderilmesi

ABC algoritmasında her bir kaynağın bir görevli arısı bulunduğundan yiyecek kaynaklarının sayısı görevli arıların sayısına eşittir. Görevli arı hafızasında yiyecek kaynağı ile ilgili bilgileri tutar. Bunlar kaynağın pozisyonu (çözüm), nektar miktarı (fitness değer) ve yeni kaynak (yeni çözüm) bigilerinden oluşur. Görevli arı hafızasında bulunan yiyecek kaynağı komşuluğunda yeni bir yiyecek kaynağı belirler

ve bunun kalitesini değerlendirir. Eğer yeni kaynaktaki nektar miktarı eskisinden daha çok ise arı hafızasından eski kaynak bilgilerini siler ve yeni kaynak bilgilerini hafızasına alır. Yeni kaynağın mevcut kaynak komşuluğunda belirlenmesi Eşitlik 3.2 ile yapılır:

$$\upsilon_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{3.2}$$

Burada x_{ij} her bir çözüm için çözümün tek bir parametresinin (rasgele seçilen parametresi, j) değiştirilerek x_i komşuluğunda υ_i çözümünün bulunmasını temsil eder. Eşitlikte j. parametre $[1, D]$ aralığında rasgele üretilen bir tamsayı ile belirlenir. Rasgele seçilen x_k komşu çözümünün ($k \in \{1, 2, \ldots SN\}$) j. parametresi ile mevcut çözümün j. parametresinin farkları alınıp $[-1, 1]$ arasında rasgele değer alan ϕ_{ij} sayısı ile ağırlıklandırılır ve mevcut çözümün j. parametresine eklenir. $\phi_{i,j}$, x_i etrafındaki komşu yiyecek kaynaklarının üretimini kontrol etmektedir. Komşu çözüm k rasgele seçilirken i den farklı olması gerekmekte ve bu çözüm ile fark alınması arının bölgede görsel bilgi ile yeni kaynak belirlemesini temsil etmektedir. Eşitlik 3.2'den anlaşılacağı üzere x_i ile x_k arasındaki fark azaldıkça x_i'nin pozisyonundaki değişim azalacak ve ince ayarlama daha iyi yapılacaktır.

Görevli arıların komşuluk bilgileri ile çözümler üretmesi (yeni kaynaklar araması) işleminde üretilen υ_{ij}'nin önceden belirlenen parametre sınırlarını aşması durumunda j. parametreye ait olan alt veya üst sınır değerlerine Eşitlik 3.3 ile yeniden ayarlama yapılır.

$$\upsilon_{ij} = \begin{cases} x_j^{\min}, & \upsilon_{ij} < x_j^{\min} \\ \upsilon_{ij}, & x_j^{\min} \leq \upsilon_{ij} \leq x_j^{\max} \\ x_j^{\max}, & \upsilon_{ij} > x_j^{\max} \end{cases} \tag{3.3}$$

Görevli arıların ürettikleri yeni çözümler (υ_i) yeni bir kaynağı temsil eder. Bu kaynakların kalitesi hesaplanır ve Eşitlik 3.4 ile uygunluk değerleri atanır.

$$fit_i = \begin{cases} 1/(1+f_i) & f_i \geq 0 \\ 1 + abs(f_i) & f_i < 0 \end{cases} \tag{3.4}$$

Burada f_i, υ_i kaynağının yani çözümünün maliyet değeridir. Görevli arılar mevcut x_i çözümleri ile muhtemel υ_i çözümleri arasında nektar miktarlarına yani uygunluk değerlerine göre bir aç gözlü seçme işlemi gerçekleştirler. Yeni bulunan υ_i çözümü daha iyi ise görevli arı hafızasından eski çözümü silerek υ_i çözümünü hafızasına alır ve geliştirememe sayacını sıfırlar. Mevcut x_i çözümü υ_i çözümünden daha iyi ise x_i çözümünü hafızasında tutmaya devam eder ve x_i çözümü ile ilgili geliştirememe sayacını bir artırır.

3.2.2. Gözcü Arıların Yiyecek Kaynağı Bölgesi Seçmeleri

Görevli arılar kovana döndüklerinde elde ettikleri bilgileri kovanda bekleyen gözcü arılara aktarırlar. Bu bilgi aktarma işlemi görevli arıların dans alanında sergiledikleri danslar aracılığıyla gerçekleşir. ABC modelinde görevli arıların danslarını izleyerek yiyecek kaynaklarının kalitesi hakkında bilgi sahibi olan ve bu bilgilere göre kaynak seçen serbest arılar gözcü arılar olarak adlandırılırlar. Gözcü arılar tüm görevli arıların verdiği bilgileri izledikten sonra kaynakları nektar miktarlarının oranlarına bağlı olarak Eşitlik 3.5 aracılığıyla hesaplanan olasılık değerlerine göre seçerler.

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{3.5}$$

Burada, fit_i, i. kaynağın nektar miktarı yani uygunluk değeri, SN yiyecek kaynağı sayısını göstermektedir. Bu olasılık hesaplama işlemine göre bir kaynağın uygunluk değeri arttıkça bu kaynak bölgesini seçecek gözcü arı sayısı da artacaktır. Temel ABC algoritmasında gözcü arılar kaynakları rulet tekerleği yöntemiyle seçerler: her bir kaynağın uygunluk değeri, tüm kaynakların uygunluk değerlerinin toplamına bölünerek her bir kaynağın seçilme ihtimali hesaplanır. Bu sayede uygunluk değeri nisbi olarak fazla olan kaynağın gözcü arılar tarafından seçilme ihtimali yüksek olmaktadır [71]. Gözcü arılar gidecekleri kaynakları belirledikten sonra aday yiyecek kaynağını görevli arıların komşuluk arama işlemine benzer şekilde Eşitlik 3.2 aracılığı ile üretirler.

Her bir aday kaynak pozisyonu v_i üretildikten sonra performansı eski pozisyon x_i ile karşılaştırılır. Eğer yeni kaynaktaki nektar miktarı öncekinden çok ya da öncekine

eşit ise eski kaynağın hafızadaki yerini belirlenen yeni kaynak alır. Aksi taktirde önceki kaynak pozisyonu hafızada tutulmaya devam eder. Başka bir ifadeyle, gözcü arılar da görevli arılar gibi eski ile aday kaynak arasında aç gözlü seçme mekanizmasını kullanırlar.

3.2.3. Kaşif Arıların Yeni Kaynakları Keşfi

ABC algoritmasında her çevrimde tüm görevli ve gözcü arılar arama işlemlerini tamamladıktan sonra iyileştiremedikleri çözümler için geliştirilememe sayacını bir artırırlar. Ancak herhangi bir aşamada çözümde iyileşme kaydedilirse sayaç başlangıç değerine getirilir yani sıfırlanır. Her bir çözüm için bir geliştirilememe sayacı tutulur ve her döngüde görevli arı da gözcü arı da o çözümü geliştiremezse sayacı 1 artırırlar. Eğer ki, bir çözümün sayaç değeri verilen eşik değeri (limit parametresi değeri) üzerine çıkarsa bu çözüm artık "tükenmiş" kabul edilir ve o çözümün görevli arısı tarfından terk edilir. Nektarı tükenen kaynağın görevli arısı kaynağını değiştirmek üzere kaşif arı olur ve terk edilecek kaynak x_i kabul edilirse, kaşif arı x_i ile değiştirmek üzere rasgele olarak Eşitlik 3.1 aracılığıyla üretilecek yeni kaynağı keşfetmektedir.

ABC algoritmasının sözde kodu:

Popülasyondaki çözümlere (x_i) ilk değerlerini Eşitlik 3.1'i kullanarak ata
Popülasyonu değerlendir
cycle=1
repeat
  FOR her görevli arı için**{**
    Eşitlik 3.2'i kullanarak yeni aday çözüm υ_i üret
    Üretilen yeni çözümün uygunluk değerini hesapla
    Aç gözlü seçme işlemini uygula**}**
  Görevli arıların hafızasındaki çözümler için p_i seçilme olasılıklarını Eşitlik 3.5'i kullanarak hesapla
  FOR her gözcü arı için**{**
    Olasılık değerlerine bağlı olarak bir çözüm seç
    Eşitlik 3.2 ile aday υ_i çözümü üret ve çözümü değerlendir

Aç gözlü seçme işlemini uygula**}**

If terk edilmiş çözüm var ise

then kaşif arı için Eşitlik 3.1 aracılığıyla yeni çözüm üret

Elde edilen en iyi çözümü hafızaya al

cycle=cycle+1

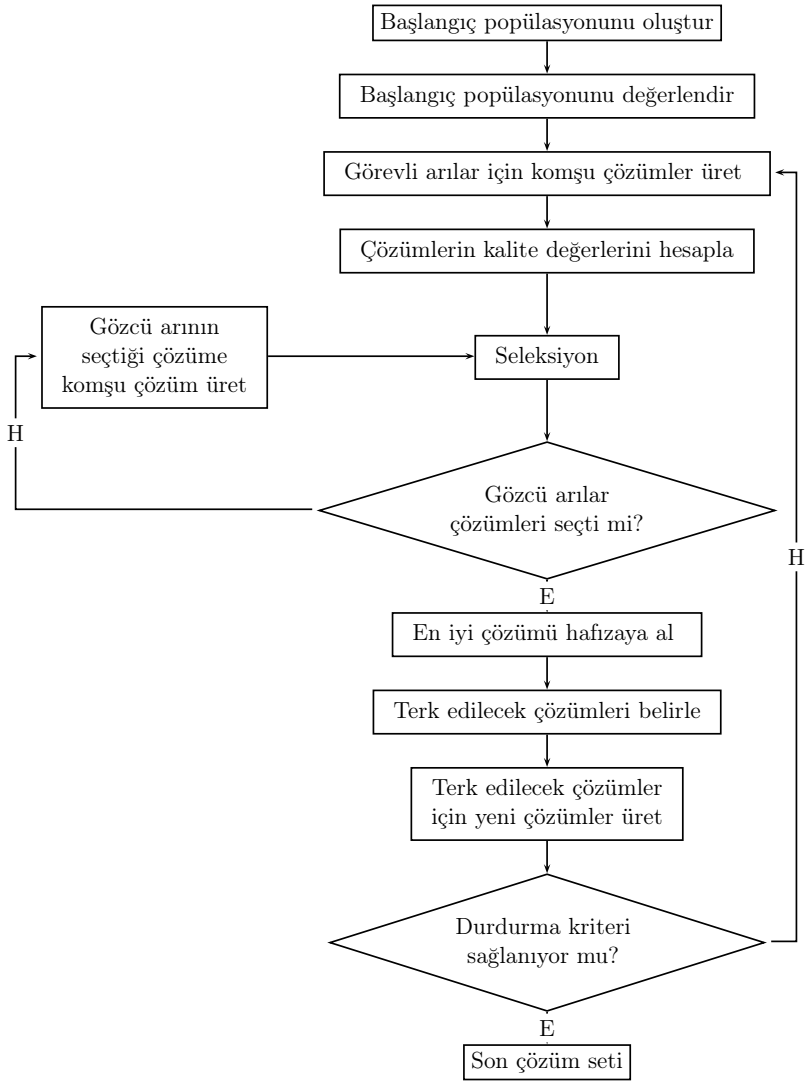
until cycle=MCN

Algoritmanın adımlarına da bakıldığında ABC algoritmasının 3 tane kontrol parametresi olduğu görünür: görevli veya gözcü arı sayısına eşit olan yiyecek kaynağı sayısı (SN), limit değeri ve maksimum döngü sayısı (MCN). Güçlü bir arama işleminde yeni keşif ve varolandan faydalanma işlemlerinin birlikte dengeli yürütülmesi gereklidir. ABC algoritmasında görevli ve gözcü arılar faydalanma işlemini yaparken, kaşif arılar da keşif işlemlerini yaparlar. ABC algoritmasının akış diagramı Şekil 3.1'de verilmiştir.

3.3. Yapay Arı Kolonisi Algoritmasının Uygulamaları

Yapay arı kolonisi yöntemi ilk olarak nümerik optimizasyon problemlerine uygulanmış ve başarısı bilim insanları tarafından fark edilerek farklı alanlardaki problemlerin çözümünde kullanılmaya başlanmıştır [15]. ABC algoritması Türk Üniversiteleri'nden Scholarpedia'da yayınlanan ilk çalışma olmuştur [72]. ABC algoritmasının uygulamalarının konuları ve uygulamalarla ilgili güncel literatür listesi aşağıda verilmiştir.

ABC algoritmasının literatüre önerilmiş sınırlamalı, sınırlamasız optimizasyon problem setleri, karma problemler ve mühendislik tasarım problemleri üzerindeki performansı kabul görmüş diğer yöntemlerle karşılaştırmalı çalışmalar [13, 14, 73–79] yayınlarında sunulmuştur. Yapay sinir ağları ve kümeleme yöntemleri hastalık sınıflandırılmasında ve tahmininde hekime yardımcı olan karar destek sistemi olarakta kullanılmaktadır. Ancak bu yöntemlerin başarımları tasarım süreçlerine ve süreçler içinde kullanılan optimizasyon tekniklerine oldukça bağımlıdır. Yapay arı



Şekil 3.1. ABC algoritmasının akış diyagramı.

kolonisi yöntemi sınıflandırma alanında yapay sinir ağlarının eğitiminde kullanılmış ve başarısı literatürde bilinen başka tekniklerle kıyaslanmıştır. Ayrıca kümeleme yönteminin kullanımında da yapay arı kolonisi yöntemi ile sınıflandırma çalışmaları yapılmıştır [80–87]. Yine, kümeleme çalışmalarında N adet nesnenin K adet kümeye optimal bir şekilde ayrılmasını sağlayan bir ABC kümeleme algoritması önerilmiş ve aday çözümlerin araştırma yönünün belirlenmesi için Deb'in kurallarının kullanıldığı çalışmada algoritma birçok veri seti üzerinde test edilmiş ve sonuçlar diğer algoritmaların sonuçları ile karşılaştırılmıştır [88]. Kümeleme amaçlı hibrid bir algoritma Marinakis ve arkadaşları tarafından [89]'de önerilmiştir: bu yöntemde ilk olarak özellik seçim probleminin çözümü için kesikli ABC algoritması, sonra ikinci aşamada ise kümeleme problemini çözmek amacıyla GRASP yöntemi tercih

edilmiştir.

Bilgisayar bilimleri alanında "en uygun ağaç yapısı bulunması" problemi yönlendirilmemiş, bağlantılı, ağırlıklandırılmış grafik yapısında en azından belli sayıda yaprağa sahip ve minimum ağırlıktaki ağacı bulmaya yönelik bir problemdir ve gerçek hayatta tesis yerleşimi, devre ve ağ tasarımı gibi alanlarda kullanılmaktadır [90]. Benzer bir problem olan "quadratic minimum spanning tree" (Q-MST) probleminin çözümü için de bir ABC algoritması Sundar ve Singh tarafından önerilmiştir [91]. Ayrık tipteki gezgin satıcı problemi literatürde yaygın kullanılan bir şehirden yola çıkan bir satıcının belirli şehirleri hangi sırada gezerse en az yolu kat edeceğini bulmaya yönelik bir test problemidir. Bu problemin çözümüne de ABC algoritması Fenglei ve arkadaşları tarafından uygulanmıştır [92].

Elektrik-Elektronik mühendisliği alanında, istenmeyen sinyalleri bastırabilmek amacıyla yaygın kullanılan dijital filtrenin en uygun şekilde tasarlanması, filtreyi oluşturan elemanların katsayılarının en iyi şekilde ayarlanması ile mümkündür. [93]'de yapay arı kolonisi yöntemi bu katsayıların bulunmasına yönelik olarak kullanılmıştır. Görüntü işleme alanında görüntüden hassas bilgilerin çıkarılması gerektiği çalışmalarda kamera kalibrasyonu önemli bir işlem olarak ortaya çıkmaktadır. Kamera kalibrasyonunda kullanılan yönteme ait parametrelerin bulunmasında yapay arı kolonisi yöntemi [94]'de kullanılmıştır. Elektrik mühendisliği alanında dağıtım şebekelerindeki hat kayıplarının azaltılması amacıyla yapay arı kolonisi yöntemi uygulanmıştır [95]. Endüstri mühendisliği alanında iş planlama problemlerinin çözümünde kullanılmıştır [96].

Makine mühendisliği alanında yapısal optimizasyon problemlerinde uygun tasarım parametrelerinin bulunması amacıyla [97], inşaat mühendisliği alanında baraj sistemlerinin elastiklik modülü veya sertlik modülü gibi mekanik parametrelerinin belirlenmesi için [98], elektro-kimyasal işleme de aşındırmalı akış işleme sürecinin en uygun parametrelerinin bulunmasında ve frezeleme süreci parametrelerinin en iyilenmesinde [99–101], valf noktası etkili ekonomik yük dağıtımı uygulamasında [102], tabakalı kompozit bileşenlerin çok amaçlı tasarım optimizasyonunda kesikli değişkenlerin bulunması amacıyla [103]'de ABC algoritması kullanılmıştır. Atıl ve

atıl olmama durumları altında, toplam ağırlıklı erken bitirme ve gecikme cezaları kriterleri ile parti bölmeli akış tipi çizelgeleme probleminin çözümüne kesikli ABC (DABC) algoritması [104]'de, ters analiz problemleri için Nelder-Mead simplex metodunu ABC algortiması ile birleştiren bir karma algoritma olan hybrid ABC (HSABCA) algoritması [105]'de geliştirilmiştir .

[106]'de yapılan çalışma da uçaklarda hedef tanıma işini gerçekleştirmek amacıyla kenar potansiyel fonksiyonu ile ABC algoritması çalıştırılmış ve deney çalışmalarının sonuçları GA algoritması ile elde edilen sonuçlar ile karşılaştırılmıştır. MESFET için küçük sinyal eşdeğer devre modeli parametrelerinin ortaya çıkarılmasında ABC algoritması kullanılmış ve sonuçlar PSO algoritmasının sonuçları ile kıyaslanarak tartışılmıştır [107]. Çeşitli savaş alanı ortamlarında (UCAV) yol planlama probleminin çözümü için kaos teorisine dayalı bir ABC optimizasyon algoritması [108]'de ve yakınsama karakteristiklerini geliştirmek ve ABC'nin yerel çözümlere takılmasını önlemek amacıyla parametre ayarlamalarını yaparken kaotik haritalar kullanan yeni bir kaotik ABC algoritması [109]'de geliştirilmiştir. Ters ısı iletimi probleminin çözümü amacıyla ABC algoritmasına dayalı sayısal bir metod geliştirilmiştir [110]. GA ve ABC algoritmaları birleştirilerek sürü zekasına dayalı yeni bir karma yaklaşım [111]'de önerilmiş, dört tane test fonksiyonu kullanarak farklı boyutlar için test edilmiş ve elde edilen sonuçlar standard GA ve ABC algoritmalarının sonuçları ile kıyaslanmıştır.

ABC algoritmasında popülasyon büyüklüğünün optimizasyon davranışına etkisi incelenirken gözcü arıların kullanımının ne zaman avantajlı olduğunun ele alındığı ve yapay arıların konumlarının güncellenmesinde yeni metodların kullanıldığı iki yeni ABC türevi [112]'de ortaya konmuş ve test problemleri üzerinde performansları değerlendirilmiştir. Kablosuz sensör ağları üzerinde sensörlerin en uygun şekilde yerleşim noktalarının tayini ve etkin ağ dolaşımının sağlanabilmesi için ABC algoritmasına dayalı yaklaşımlar önerilmiştir [113–115].

ABC algoritması ile aç gözlü sezgisel ve yerel araştırma mekanizmaları birleştirilerek yeni bir hibrid yaklaşım Quadratik Knapsack probleminin çözümü için önerilmiş ve önerilen algoritmanın performansı bilinen diğer sezgisel tekniklerle kıyaslanmıştır

[116]. Tamsayı programlama problemlerine ABC uygulanmış ve sonuçlar, PSO türevleri ve dal sınır tekniği ile karşılaştırılmıştır [117]. Kontrol parametrelerinin ABC algoritmasının performansı üzerindeki etkileri [118]'de analiz edilmiştir. Kuantum gelişim algoritması (QEA) ve ABC algoritması birleştirilerek geliştirilen hibrid algoritmanın performansı test problemleri üzerinde değerlendirilmiştir [119]. Paylaşımlı hafıza mimarileri için ABC algoritmasının paralel türevi PABC algoritması önerilmiştir [120]. ABC algoritması, en düşük serbest enerji yapısını bulmak için protein yapısal araştırma uzayını araştırmak amaçlı [121]'de uyarlanmıştır. ABC görüntülerde nesne tanıma problemine uygulanmış ve sonuçlar diğer evrimsel algoritmalarla karşılaştırılmıştır [122]. ABC algoritmasının komşuluk arama mekanizmalarında yapılan değişiklikleri içeren yeni versiyonu dağınık sistemlerde optimum dağıtım yerleşiminin bulunması için öne sürülmüştür [123].

Standart ABC algoritmasının bir türevi olan ABC programlama (ABCP) [124]'de sunulmuş, 15 test problemine uygulanmış ve sonuçlar standart ABC ile karşılaştırılmıştır. ABC algoritmasının başka yeni bir versiyonu popülasyon çeşitliliğini artırmak ve erken yakınsamayı önlemek amacıyla birçok seleksiyon stratejisi kullanılarak [125]'de karşılaştırılmış ve analiz edilmiştir. Görüntü kenarlarının kalitesini artırma ve segmentasyon, özellik çıkarımı, sınıflandırma ve görüntü üretimi gibi karmaşık görüntü işleme problemlerini çözme tekniklerinin ele alındığı [126] nolu çalışma da ABC optimizasyonu algoritması ile hibridlenmiş düzeltme filtreleri kullanılarak kenar iyileştirme yapılmış ve sonuçlar genetik algoritma ile karşılaştırılmıştır. ABC optimizasyon algoritması, sınırlandırılmış ekonomik yük dağıtım problemine uygulanmış ve sonuçları benzer yaklaşımlarla kıyaslanmıştır [127]. Boyut bakımından geniş aralıklara sahip yedi test fonksiyonunda Hooke Jeeves örüntü araştırması ve ABC yöntemleri ile yeni hibrid Hooke Jeeves ABC (HJABC) algoritması ortaya konmuştur [128].

ABC algoritması [129]'de tam zamanında üretim felsefesine dayalı olarak erken üretim ve gecikme penaltıları maliyetinin kombinasyonunu minimize edecek şekilde çizelgeleme problemi için tasarlanmış ve parametre konfigürasyonunun algoritma performansı üzerindeki etkileri incelenmiştir. ABC algoritmasının gelişmiş

versiyonunun (IABC) Sudoku Puzzle problemlerini çözme ihtimali araştırılmıştır [130]. Yapısal yazılım testlerinin otomatik üretimi için ABC temelli yeni bir araştırma tekniği sunulmuş ve geliştirilen tekniğin performansı on gerçek programda değerlendirilmiştir [131]. Standart ABC ve geliştirilmiş bir türevi, Loney's selonid problemine uygulanmış ve sonuçlar ile algoritmanın elektromagnetik alanda optimizasyona uygunluğu gösterilmiştir [132]. Çalışma [133]'de GA, PSO, ABC, Yayılan Otlar Optimizasyonu (IWO) ve Yapay Bağışıklık (AI) evrimsel algoritmaları bazı çok değişkenli standart test fonksiyonları üzerinde kıyaslanmıştır.

4. BÖLÜM

YAPAY SİNİR AĞLARININ YAPAY ARI KOLONİSİ ALGORİTMASI KULLANILARAK EĞİTİLMESİ

4.1. Giriş

Yapay sinir ağlarının eğitiminde nöronlar arası bağlantıları sağlayan ağırlıklar, ayarlanabilen bir dizi parametredir. Başarılı bir öğrenme süreci sonunda eşleştirme hatalarını minimize edecek ağırlık seti bulunması hedeflenir. Eşleştirme hatası, veri setinde önceden belirlenen çıktı değeri ile belli bir giriş değerine karşılık üretilen çıktı değeri arasındaki farktır. Eğitim sürecinde girdilerin bulunduğu girdi veri seti ile bu girdilere karşılık gelen çıktılardan oluşan çıktı veri seti arasında eşleme yapılması beklenir. Bu veri setleri gerçek değerli, ayrık veya vektörel değerli olabilir, doğru olmayan örnekler içerebilir veya gürültülü veriler olabilir [134].

Gerçek dünya problemlerinin çoğunda girdi veri setinin tamamını ve kendi çıktı değerlerini elde etmek mümkün değildir. Dolayısıyla ağı eğitmek için kullanılan veri setlerinin doğru popülasyonlardan alınan örnekler olmasına dikkat edilmelidir. Ağı eğitmekteki amaç gerçek dünya senaryolarındaki eşleştirme hatalarını minimize etmektir. Ağın test setindeki verilerin kullanılması durumunda oluşan hata, eğitim setindeki veriler ile elde edilen hatadan daha önemli bir performans göstergesidir. Eğer girdi verisinin kopyası ya da çok benzeri bir örneği mevcut ve bu girdilerin çıktıları da biliniyor ise yaklaşık olarak ağ hatasını tespit etmek hiç görülmemiş veriye göre daha kolay olacaktır. Bu sebeple, test aşamasında eğitilen ağa önceden görmediği ve içerisinde çıkış verisi olmayan veri seti verilir, bu veri seti test seti olarak isimlendirilir.

Ağdaki ağırlık sayısı ağın öğrenme yeteneğini belirleyen bir faktördür [135]. Sabit tipte bir ağda saklı birimler ağdaki ağırlık sayısını belirler çünkü girdi ve çıktı birim

sayısı problem tarafından belirlenir. Buradan anlaşıldığı gibi saklı birimi çok olan ağ uygun bir eğitim algoritması varsa daha karmaşık eşleşmeleri öğrenebilir. Ancak eğitim sürecinde bir ağ, eğitim veri seti için oldukça başarılı şekilde eğitilebilirken test setinde kötü bir performans gösterebilir ki bu nedenle de gerçek dünya problemlerinde de düşük performans gösterme ihtimali vardır. Bu olgu ezberleme (overfitting) olarak adlandırılır ve bu durum çoğunlukla ayarlanabilen ağırlık sayısı çok fazla olduğunda oluşur [136]. Ezberlemenin oluşmasını engellemenin birkaç yolu vardır; örneğin eğer eğitim seti gürültü içeriyorsa, çok sayıda saklı birimi olan bir ağ gürültüyü öğrenebilir, bunun yanında ağın genelleştirme yeteneğini azaltır. Ağda, saklı katmanda bulunan birim sayısı sınırlanırsa, ağ gürültüyü öğrenemez hale gelir ve genelleştirme performansı artar. Bu yaklaşım, minimal beyin hasarlı algoritmasından alınan bir yaklaşımdır ve minimum sayıda saklı birim kalana kadar dıştan gelen saklı birimleri etkisiz hale getirmeye çalışır [137]. Ayrıca ağ dışarıdan gelen girdi birimlerine sahip olabilir ve bunlar ağın öğrenmesinde zararlı bir etkiye yol açabilir. Gereksiz katmanları yok etmek amacıyla Engelbrect [138]'de bir metod geliştirilmiştir. Diğer bir yaklaşım düzenleme (regularisazyon), düzgün ağ eşlemesiyle verilerin overfitting olma durumlarının önlenebileceği varsayılır ki bu, ağ ağırlıklarının küçük olacağı anlamına gelir ve düzenlemenin bu formuna ağırlık bozulması denir. Alternatif olarak, ağ oldukça basit bir mimari modelle başlayabilir ve hata yeterince küçülene kadar saklı birim eklenebilir. Bu yaklaşım da büyüme (growing) olarak isimlendirilir ve cascade-correlation ağı bu yaklaşımı kullanan bir mimaridir [139]. Bu tezin amacı ABC algoritmasının yapay sinir ağlarının eğitiminde kullanımını sunmak ve performansını bu amaçla kullanılan çeşitli algoritmalar ile karşılaştırmaktır. Bu sebeple, bu tezde bir test problemi için en uygun mimariyi bulmak ya da ulaşılabilecek en düşük hatayı bulmak ana hedef olarak gösterilmemiştir. Bunun yanında amaç ABC algoritmasının ağ eğitiminde diğer algoritmalar kadar rekabetçi olduğunu göstermek ve başarısını tartışmaktır.

4.2. Test Problemleri için Yapay Sinir Ağlarının Eğitilmesi

Tez çalışmasında ilk olarak çok katmanlı ileri beslemeli algılayıcı (MLP) yapay sinir ağlarının eğitiminde sıkça kullanılan test problemleri XOR, 3-Bit Parite ve 4-Bit

Tablo 4.1. XOR test problemi doğruluk tablosu.

Giriş1	**Giriş2**	**Çıkış**
0	0	0
0	1	1
1	0	1
1	1	0

Encoder/Decoder problemleri ele alınmıştır. Bu problemlerin en belirgin özellikleri doğrusal olmayan ilişkilere sahip olmalarıdır. Yani çıktıların arasına bir doğru veya doğrular çizerek onları iki ya da daha fazla sınıfa ayırmak mümkün değildir.

Genellikle uyumluluk, hata fonksiyonunun minimize edilmesiyle yürütülür. Hata fonksiyonunu, E kabul edersek, formülü Eşitlik 4.1 ile ifade edilir:

$$E(\vec{w}(t)) = \frac{1}{N} \sum_{j=1}^{N} \sum_{k=1}^{P} (d_k - o_k)^2 \tag{4.1}$$

Burada, E, t. iterasyondaki hata; $\vec{w}(t)$ t. iterasyondaki bağlantı ağırlıkları; d_k istenen çıktı değeri ve o_k üretilen çıktı değerini; P çıktı nöron sayısını ve N örnek sayısını göstermektedir.

4.2.1. Exclusive-OR (XOR) Problemi

Bu problemin çok yaygın olarak kullanılmasının sebebi şu şekilde açıklanabilir; Minsky özellikle perseptron modelinin bu probleme çözüm üretemediğini göstermiş ve yapay sinir ağlarının doğrusal olmayan problemlere çözüm üretemediğini ileri sürmüştür. Günlük olayların hemen hemen hepsinin doğrusal olmayan nitelikler taşımasından dolayı da YSA'lar ile ilgili bilimsel araştırmaların, çalışmaların neredeyse hepsini durma noktasına getirmiştir. Bu iddaların ardından sadece birkaç araştırmacı çalışmalara devam etmiş ve bu problemi çözerek yapay sinir ağlarına tekrar dikkatleri çekmeyi başarmışlardır. O nedenle bu problem yapay sinir ağı araştırmalarında bir kilometre taşı olarak nitelendirilmektedir [140].

XOR problemi (0 0; 0 1; 1 0; 1 1) ikili girişlerine karşılık (0; 1; 1; 0) tekli çıkış şeklinde

Tablo 4.2. 3-Bit Parite test problemi doğruluk tablosu.

Giriş1	**Giriş2**	**Giriş3**	**Çıkış**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

haritalanan XOR Boolean fonksiyonundan oluşur (Tablo 4.1). Simulasyonda kullanılan MLP yapıları: 2-2-1 ileri beslemeli sinir ağı 6 bağlantı ağırlıklı ve öndeğersiz (6 parametreli XOR6); 2-2-1 ileri beslemeli sinir ağı 6 bağlantı ağırlıklı, 3 öndeğerli (9 parametreli XOR9); 2-3-1 ileri beslemeli sinir ağı 9 bağlantı ağırlıklı, 4 öndeğerle (13 parametreli XOR13) modelleridir.

4.2.2. 3-Bit Parite Problemi

Üç girişli tek çıkışlı 3-Bit parite probleminde girişlerdeki 1'lerin sayısı tek ise çıktı 1; çift ise çıktı sıfır şeklindedir. Bu problemin giriş ve çıkış setleri Tablo 4.2 verildiği gibi sırasıyla: (000; 001; 010; 011; 100; 101; 110; 111), (0; 1; 1; 0; 1; 0; 0; 1) şeklindedir [141, 142]. 3-Bit Parite problemi için 3-3-1 ileri beslemeli sinir ağı yapısı kullanılmıştır. Bu yapı 12 bağlantı ağırlıklı ve 4 öndeğer olmak üzere toplam 16 parametrelidir ve parametre aralığı [-10, 10] şeklinde alınmıştır.

4.2.3. 4-Bit Encoder/Decoder Problemi

4-Bit Encoder/Decoder probleminde her biri yalnız tek bir 1 değeri içeren 4 farklı örneğe göre hazırlanır ve çıktılar girdinin tekrarıdır. Problemin doğruluk tablosu (0001; 0010; 0100; 1000) girişlerine karşılık aynen çıkışlarda da (0001; 0010; 0100; 1000) üretilmesi ile elde edilir (Tablo 4.3) [143]. Bu durum gerçek hayattaki sınıflandırma işlemlerine oldukça yakındır [144]. Girişteki küçük değişiklikler çıktıda rahatlıkla gözlemlenebilir. Bu problemde 4-2-4 ileri beslemeli sinir ağı yapısı

Tablo 4.3. 4-Bit Encoder-Decoder test problemi doğruluk tablosu.

Giriş1	Giriş2	Giriş3	Giriş4	Çıkış1	Çıkış2	Çıkış3	Çıkış4
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
1	0	0	0	1	0	0	0

Tablo 4.4. Test problemlerin özellikleri; parametre aralığı, kullanılan ağ yapıları, problem boyutu.

	Aralık	Yapı	Boyut
XOR6	[-100,100]	2-2-1	6
XOR9	[-10,10]	2-2-1	9
XOR13	[-10,10]	2-3-1	13
3-Bit Parite	[-10,10]	3-3-1	16
4-Bit Enc.-Dec.	[-10,10]	4-2-4	22

kullanılmış ve 16'sı bağlantı ağırlığı, 6'sı öndeğer olmak üzere toplam 22 parametre içermektedir. Parametre aralığı [-10,10] aralığında alınmıştır.

4.2.4. Test Problemleri için YSA'ların Eğitilmesinde Yapılan Ayarlamalar

Deneyler her durum için 30 kez tekrarlanmış ve her çalışma rasgele popülasyonlar ile başlatılmıştır. Problemlerin her biri için seçilen her bir ağda transfer fonksiyonu olarak logaritma sigmoid fonksiyonu seçilmiştir. Ağ eğitimleri, girdiyle çıktı arasındaki ortalama kare hata oranı (MSE) (Eşitlik 4.1) 0.01'den küçük olduğunda veya eşit olduğunda ya da maksimum döngü sayısına ulaşıldığında sona erdirilmiştir. Her bir problemin parametre ayarları literatürde en yaygın tercih edilen değerler baz alınarak yapılmıştır. Tüm problemler arasında XOR6 en zor olanıdır, çünkü nöronlara öndeğerler uygulanmamıştır. Bu nedenle, bu problem için tüm algoritmalar daha çok döngü ile çalıştırılmıştır. Ağlarda ayarlanan ağırlıklar için kullanılan parametre aralıkları ve her yapıda ayarlanacak ağırlık sayısı Tablo 4.4'de verilmiştir.

Ele alınan problemlerden XOR9 problemi için oluşturulan ağ, Bölüm 2.3'de Şekil 2.4 ile gösterilen ileri beslmeli ağ yapısındadır. 2 girişli, 1 çıkışlı, saklı katmanında

Tablo 4.5. XOR9 Probleminin çözümü için ağ eğitimi sırasında ayarlanacak ağırlık ve eşik değerlerinin örnek gösterimi.

i. çözüm	x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}	x_{i6}	x_{i7}	x_{i8}	x_{i9}
Ağırlıklar	w_1	w_2	w_3	w_4	b_1	b_2	w_5	w_6	b_3

2 nöron bulunduran ve nöronlara eşik değerlerinin uygulandığı ileri beslemeli ağ yapısının eğitiminde ABC algoritması kullanılırken olası bir i çözümünün karşılık geldiği ağırlık ve eşik parametreleri Tablo 4.5'de verilen şekildedir.

ABC için kontrol parametreleri: "limit" değeri SN*D ifadesine eşittir. D problemin boyutunu gösterir. Tüm problemler için koloni büyüklüğü (2*SN) 50 olarak alınmıştır. GA için parametre değerleri: deneylerde rulet tekerleği seçim stratejisi seçilmiş, tek nokta çaprazlama ve çaprazlama oranı 0.8, mutasyon oranı 0.05 alınmış, jenerasyon boşluğu 0.9 ve her problem için popülasyon büyüklüğü 50 olarak ayarlanmıştır. BP için ayarlar: Geri yayılım deneylerinde, sinir ağları LM ve BP eğitim algoritmaları kullanılarak eğitilmiş ve öğrenme oranı 0.8 olarak seçilmiştir.

ABC için maksimum döngü sayısı ve GA için maksimum jenerasyon sayısı XOR6, XOR9, XOR13, 3-Bit Parite ve 4-Bit Encoder/Decoder problemleri için sırasıyla 7500, 100, 75, 1000 ve 1000 olarak ayarlanmıştır. Bu sebeple, toplam amaç fonksiyon hesaplama sayıları problemler için sırasıyla 375000, 5000, 3750, 50000, 50000 şeklinde alınmıştır. BP algoritması uygulandığında problemler için epok sayıları 32000, 500, 250, 1600, 2100 olarak alınmıştır.

PSO için, bilişsel ve sosyal parametrelerin değerleri 1.8, parametrenin daha önceki değerinin o anki değeri etkilemesinin oranı olan ivme (inertia) parametresinin değeri [145] çalışmasında önerildiği üzere 0.6 olarak seçilmiştir ve sürü 50 bireyden oluşmaktadır. Maksimum amaç fonksiyonu değerlendirme sayısı ABC algoritmasında alındığı gibi PSO ve DE algoritmaları için de XOR6, XOR9, XOR13, 3-Bit Parite ve 4-Bit Encoder-Decoder problemlerinde sırasıyla 375000, 5000, 3750, 50000 ve 50000 olarak seçilmiştir. DE algoritmasında iki çözüm arasındaki farkı ölçekleyen F değeri 0.5, çaprazlama operatörünün değeri [57]'de önerildiği şekilde 0.9 olarak seçilmiştir. Popülasyon büyüklüğü de ABC, GA ve PSO algoritmalarıyla

Tablo 4.6. Test problemleri için YSA eğitiminden elde edilen sonuçlar, ortalama MSE hata ve standart sapma değerleri.

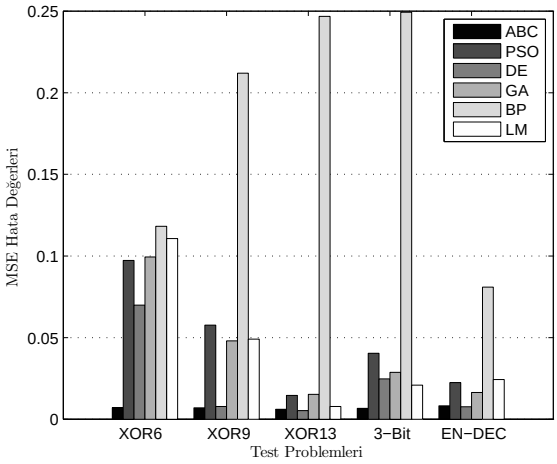
Problem		**ABC**	**PSO**	**DE**	**GA**	**BP**	**LM**
XOR6	**MSE**	0.007051	0.097255	0.069908	0.099375	0.118200	0.110700
	sd	0.00223	0.027367	0.000000	0.027850	0.076300	0.063700
XOR9	**MSE**	0.006956	0.057676	0.007789	0.047968	0.212000	0.049100
	sd	0.002402	0.061530	0.009581	0.052000	0.036900	0.064600
XOR13	**MSE**	0.006079	0.014549	0.005293	0.015200	0.246800	0.007800
	sd	0.003182	0.024627	0.002651	0.022850	0.008000	0.022300
3-Bit Par.	**MSE**	0.006679	0.040365	0.024716	0.028725	0.249300	0.020900
	sd	0.002820	0.049018	0.031960	0.032900	0.002500	0.043000
Enc. Dec.	**MSE**	0.008191	0.022389	0.007677	0.016400	0.080900	0.024300
	sd	0.001864	0.031639	0.001708	0.031300	0.075600	0.042400

aynı olacak şekilde 50 değerine ayarlanmıştır.

4.2.5. Test Problemleri için YSA'ların Eğitilmesinden Elde Edilen Sonuçlar

Problemler için 30'ar koşmalar sonucunda elde edilen ortalama karesel hata ve standart sapma değerleri Tablo 4.6'de, ortalama karesel hata değerlerinin grafikleri Şekil 4.1'de verilmiştir. E-6'dan daha küçük değerler 0 olarak kaydedilmiştir. Sonuçlar incelendiğinde ortalama MSE değerlerine göre ABC algoritması problemlerin geneli üzerinde daha iyi sonuçlar vermektedir. Algoritmaların başarı oranları Tablo 4.7'da gösterilmiştir. XOR6 probleminde PSO, GA ve DE algoritmalarının başarısız, Geri yayılım algoritmasının %3 ve LM algoritmasının %6 oranlarında başarı sergilerken ABC algoritmasının %100 başarılı olduğu görülmektedir. Tablo 4.8'deki ortalama jenerasyon sayılarından da PSO, GA ve DE algoritmalarının XOR6 probleminde tüm jenerasyonları tamamladıkları halde hatayı 0.01 değerinin altına düşüremezken ABC algoritmasının ortalama döngü sayısı 2717.4 olmuştur. ABC' nin ortalama döngü sayısı LM'nin ortalama epok sayısından daha fazla olmasına rağmen LM algoritmasının yerel minumum noktasına takılı kaldığı açıkça görülmektedir.

XOR9 probleminde PSO %60, GA %40, DE %93, BP %0 ve LM %66.66 oranında başarılı olurken ABC algoritması ortalama 32 döngüde %100 başarılı olmuştur. LM



Şekil 4.1. Test problemleri için YSA eğitimi, ortalama MSE hata değerleri grafikleri.

algoritması başarılı olduğu koşmlarda ortalama 13 epokta başarı kaydedebilmiş, yani yakınsama hızı yüksek (hızlı) olmasına rağmen ortalama her üç koşmadan birinde küresel minimum noktasını bulmakta başarısız olmuştur.

Başarı oranlarına bakıldığında XOR13 probleminde DE algoritmasının ve ABC algoritmasının %100, PSO algoritmasının %93, Genetik Algoritmanın %76.67, LM algoritmasının %96.66 başarılı olduğu ve BP algoritmasının her koşmada başarısız olduğu görülmektedir. XOR13 probleminde algoritmalar bu başarılarını ortalama LM 9 epokta, DE 22.5 jenerasyonda, PSO 23.4 jenerasyonda, ABC 28.2 döngüde ve GA 38.6 jenerasyonda elde etmişlerdir. Her ne kadar XOR6, XOR9 ve XOR13 yapılarında ele alınan problem aynı olsa da ağ yapıları farklıdır. Sinir ağı XOR13 yapısında 3 saklı nöron kullanılırken, XOR6 ve XOR9 için 2 saklı nöron ile ağ eğitilmeye çalışılmıştır. Karmaşık yapılı ağ yapıları problemin çözümü için her zaman uygun değildir ya da çok küçük boyutlardaki ağlar problemler için yeterli sonuçları üretemeyebilirler. Bu nedenle uygun ağ yapısının seçilmesi, ağ eğitiminde uygun bağlantı ağırlıklarının bulunması gibi ayrı bir tasarım problemidir.

3-Bit parite probleminde BP hiçbir çalışmasında küresel optimum noktasını bulamamıştır. ABC algoritması ortalama 179.06 döngüde her çalışmasında istenen başarı kriterini sağlarken LM %86.66, DE %77, GA %63.33, PSO %60 oranında başarılı olabilmişler yani bu oranlarda istenilen ağ çıktı ortalama hata değerlerine ulaşabilmişlerdir. Ele alınan diğer test problemi 4-bit encoder/decoder probleminde

Tablo 4.7. Test problemleri için yapılan eğitimlerde algoritmaların başarı yüzdeleri.

Problem	**ABC**	**PSO**	**DE**	**GA**	**BP**	**LM**
XOR6	100	0	0	0	3	6
XOR9	100	60	93	40	0	66.66
XOR13	100	93	100	76.67	0	96.66
3-Bit Par.	100	60	77	63.33	0	86.66
Enc. Dec.	100	80	100	86.66	2	73.33

Tablo 4.8. Test problemleri için yapılan eğitimlerde ortalama jenerasyon (MGC) ve standart sapma değerleri (SD).

Problem		**ABC**	**PSO**	**DE**	**GA**	**BP**	**LM**
XOR6	**MGC**	2717.4	7500	7500	7500	31603	67.53
	SD	3.4	0	0	0	2176	59.38
XOR9	**MGC**	32	53	31.9	77.07	500	13
	SD	0.2	39.6	23.2	33.39	0	6.83
XOR13	**MGC**	28.2	23.4	22.5	38.6	250	9
	SD	1.2	15.3	12.7	25.02	0	3.21
3-Bit Par.	**MGC**	179	449.9	327.1	501.13	1600	21.33
	SD	12.8	456.2	391.4	415.87	0	10.01
Enc. Dec.	**MGC**	185	263.7	106.4	340.48	2020	83.37
	SD	5.9	370.2	64.56	185	236.80	174.17

BP %2, LM %73.33, PSO %80, GA %86.66 ve ABC ile DE algoritmaları %100 başarılı olmuşlardır.

ABC algoritması, bu çalışmada kullanılan tüm problemlerin çözümünde performans bakımından diğer algoritmaları geride bırakmış ve ele alınan ağlar için en uygun bağlantı ağırlıklarını bulabilmiştir. ABC nin tüm problemler için başarı oranı %100 dür. Çok modlu problemlerde iyi sonuçlar elde edebilmek için arama stratejisi, arama (keşif) ve faydalanma parçalarını etkili bir şekilde birleştirmelidir. ABC algoritması küresel araştırmayı ve bulduğu çözümleri geliştirmeyi yani bölgesel arştırmayı başarıyla birleştirerek, bu bölümde ele alınan problemlerde ileri beslemeli MLP ağlarının eğitiminde yüksek performans göstermiştir. Hem yaygın olarak kullanılan geleneksel BP ve LM algoritmalarına hem de GA, DE ve PSO evrimsel algoritmalarına karşın üstün bir performans sergilemiştir. Tablo 4.6'de algoritmaların ürettikleri ortalama karesel hata değerleriyle (MSE) birlikte verilen bu

hata değerlerinin yayılımını gösteren standart sapma (sd) değerlerine bakıldığında ABC algoritmasının DE algoritmasıyla birlikte en az yayılım çizdiğini ve oldukça kararlı bir davranış sergilediğini vurgulayabiliriz.

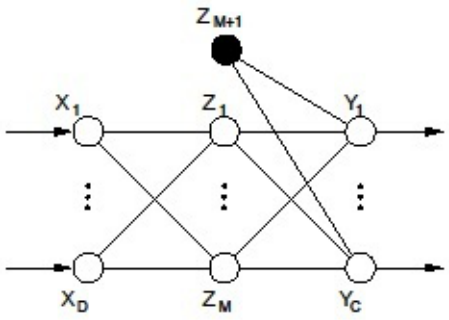
4.3. Zaman Serilerinin Tahmini Amacıyla Yapay Sinir Ağlarının Eğitilmesi

Zaman serisi tahminleri günlük yaşamın tamamlayıcı parçalarındandır. Zaman serilerinin analizi, birçok istatiksel metodlar ile probleme modeller oluşturarak verilerin anlaşılmasını sağlamaktır. Problemin modellenmesindeki amaç, dinamik sistemin davranışını kesin olarak açıklayabilen uygun matematiksel modelin bulunmasının istenmesidir. Bu bölümde yapılan çalışmanın amacı zaman serilerinin tahmin işlemlerinde kullanılan yapay sinir ağları modellerini geliştirmektir. Bu tür tahmin işlemlerinde en yaygın kullanılan yapay sinir ağ modelleri çarpım birimli yapay sinir hücreleri temellidir [146]. Çarpım birimli yapay sinir ağları, bazı problemlerde öğrenme yetenekleri daha yüksek ve hafızaları daha kuvvetli olarak kabul edilen yapay sinir hücrelerinde çıkışın hesaplanmasında farklılıklar gösteren ağlar olarak tanımlanmışlardır [147].

4.3.1. Çarpım Birimli Yapay Sinir Ağları

Çarpım Birimli Yapay Sinir Ağları ilk kez Durbin ve Rumelhart tarafından bulunmuş ve çok iyi bilinen toplama birimli geri yayılım ağlarına karşı kullanılabilirlikleri tartışılmıştır [148]. Şekil 4.2'de D girişli, M saklı birimli ve C çıkışlı bir ağ gösterilmektedir. Sadece çarpım birimlerinin saklı katmanda kullanıldığı, çıkış katmanında toplama birimleri ile devam ettiği, baştan başa lineer aktivasyon fonksiyonlu olduğu kabul edilmektedir [149]. Çarpım birimli yapay sinir ağları Bölüm 2.2'de anlatılan yapay sinir hücrelerinden toplama fonksiyonu yerine hücre girdilerinin bağlantılar kadar kuvvetlendirilmesi şeklinde çalışmalarıyla farklılık gösterirler. Bu ağlarda, işlem elemanı için birim kavramı nöron kavramına göre daha yaygın olarak kullanılmaktadır.

Örnek p için y_k çıkış değerinin hesaplanması şu şekilde yapılır:



Şekil 4.2. Çarpım birimli ileri beslemeli çok katmanlı yapay sinir ağları yapısı.

$$y_k = \sum_{j=0}^{M+1} w_{kj} \prod_{\mathrm{i=1}}^{\mathrm{D}} \mathrm{x}_{\mathrm{i,p}}^{\mathrm{w_{ji}}} \tag{4.2}$$

Burada, w_{kj} çıkış birimi y_k'dan saklı birim z_j'ye kadar olan değerler, w_{ji} saklı birim z_j'den giriş birimi x_i'ye kadar olan değerler, D girdi katmanındaki girişler sayısı ve Z_{M+1} saklı katman ile çıkış katmanı arasındaki eşik birimidir.

Quadratik fonksiyon $ax^2 + c$ formunun en az iki saklı birime ihtiyacı olan toplama ağı yerine sadece 1 giriş birimli ve 1 çarpım birimli ağ ile gösterilebileceğine [150]'de dikkat çekilmiş ve çarpım birimli sinir ağlarının toplama birimli sinir ağlarına göre daha fazla bilgi saklayabilme kapasitesinin göstergesi olarak belirtilmiştir. Ancak, geri yayılım algoritması gibi eğim azalmasına dayalı genel optimizasyon algoritmalarının toplama birimli ağlarda gösterdikleri etkinliği çarpım birimli ağları çalıştırmakta gösteremedikleri de ortaya konmuştur [151]. Bunun sebebi olarakta çarpım birimli nöronların çıkışlarında oluşan hata yüzeylerinin daha fazla türbülanslı olması belirtilmiştir. Çarpım birimli yapay sinir ağlarının performansı bilim insanları tarafından çeşitli problemler üzerinde araştırılmıştır [152–155].

4.3.2. Tahmin Problemleri

Çarpım birimli yapay sinir ağlarının performans analizi için yaygın olarak kullanılan iki zaman serisi tahmin problemi (Box-Jenkins ve Mackey-Glass kaotik zaman serileri) ele alınmıştır. Bu örneklerle ilgili veriler [156]'de Working Group on Data

Modelling Benchmark - IEEE Neural Network Council grubu tarafından derlenmiş ve kullanıma sunulmuştur. Veri setlerinin kullanılan şekilleri EK -1 kısmında verilmiştir.

4.3.3. Mackey-Glass Zaman Serileri

Kaotik Mackey-Glass diferansiyel gecikme eşitliği [157] test problem olarak tanımlanmaktadır. Farklı modellerin genelleştirme yeteneklerinin ve öğrenmelerinin karşılaştırılmasında kullanılabileceği araştırmacılar tarafından rapor edilmiştir. Mackey-Glass kaotik zaman serisi, Eşitlik 4.3 ile elde edilir:

$$\frac{dy(t)}{dt} = \frac{ay(t-\tau)}{1+y^{10}(t-\tau)} - by(t) \tag{4.3}$$

Burada, $\tau > 17$ olduğunda Eşitlik 4.3 kaotik davranış sergiler. Bu zaman serisi için yaygın olarak kullanılan yapay sinir ağında girişe 4 giriş verilir (t-18, t-12, t-6, t) ve çıkışta da t+6 adımı tahmin edilmeye çalışılır. Bu şekilde 750 veri (t=118 - 867) eğitim aşamasında 250 veri de (t=868 - 1117) test aşamasında kullanılmıştır.

4.3.4. Box-Jenkins Zaman Serileri

Metan-hava karışımının yanma işleminde doğal gaz ile çalışan ısıtma sisteminin verileri Box ve Jenkins tarafından 1970'de kaydedilmiştir [158]. Tanımlama ve tahmin algoritmalarının test edilmesinde bu yöntem sıklıkla referans olarak kullanılmaktadır. Box-Jenkins yöntemi tek değişkenli zaman serilerinin ileriye dönük tahmin ve kontrolünde kullanılan istatistiksel öngörü yöntemlerinden biridir. Zamana bağlı olayların rassal karakterde olaylar, bu olaylarla ilgili zaman serilerinin ise stokastik süreçler olduğu varsayımına dayanarak geliştirilmiş olan bu yöntemin uygulandığı zaman serisinin eşit aralıklı gözlem değerlerinden oluşan kesikli ve durağan bir seri olduğu varsayılmaktadır [159]. Veri seti 290 çift girdi-çıktıdan oluşur. Girdi $u(t)$ sisteme gaz akışı; çıktı $y(t)$ çıkan gazdaki CO_2 konsantrasyonudur. Veri setinden $u(t)$ ve $y(t-1)$ sistem girişleri olarak alınmış ve $y(t)$ çıktısı tahmin ettirilmeye çalışılmıştır.

Tablo 4.9. Tahmin problemlerinin parametreleri, N - ara katmanda kullanılan nöron sayısı.

Zaman Serileri	**Mackey-Glass**	**Box-Jenkins**
YSA Yapısı	4-N-1	2-N-1
Parametre Aralığı	[-1,1]	[-1,1]
Örnek Sayısı	1000	290
Eğitim Verisi	750	200
Test Verisi	250	90

4.3.5. Tahmin Problemleri için YSA'ların Eğitilmesinde Yapılan Ayarlamalar

Yapay sinir ağlarının tahmin amaçlı eğitiminde yapay arı kolonisi algoritmasının performansı incelenirken sonuçlar PSO algoritması ve DE algoritması yöntemleri ile karşılaştırılmıştır. Problemlerin özellikleri Tablo 4.9'de özetlenmiştir. Ağ yapıları kurulurken saklı katmanlarında kullanılan nöronlar 1'den 6'ya (1:6) kadar alınarak her model için ayrı ayrı test edilmiştir. Algoritmalar her bir problem için 30'ar kez koşulmuş ve eğitim ile test verilerinin yakınsamaları ortalama karesel hata hesaplanarak test edilmiştir. Ortalama karesel hata,

$$E = \frac{1}{N} \sum_{k=1}^{N} (d_k - o_k)^2 \tag{4.4}$$

şeklinde tanımlıdır. Burada d_k istenen çıktı değeri, o_k üretilen çıktı değeri ve N ise girdi veri sayısını göstermektedir.

Eğitim aşamasında kullanılan algoritmaların parametreleri ayarlanırken maksimum koşma sayısı 1000 olarak ayarlanmış, diğer parametrelerinin ayarları ise ABC algoritması için koloni büyüklüğü 20 ve limit değeri 200 şeklindedir. PSO algoritması için sürü büyüklüğü 20, eylemsizlik faktörü 0.7298 ve hızlandırma faktörü 1.4960 değerlerinde; DE algoritması için popülasyon boyutu 20, çaprazlama oranı 0.9 ve F'in (ölçekleme faktörü) değeri de 0.5 şeklinde ayarlanmıştır.

İki problemde hem çarpım birimli ağlarda hem de toplama birimli ağlarda modellenerek tahmin edilmeye çalışılmıştır. Bu modellerde gizli katmanda kullanılan nöron sayıları (1:6) 1'den 6'ya kadar farklı farklı denenmiştir. 30'ar koşma sonucunda

Tablo 4.10. Mackey-Glass problemi için çarpım birimli YSA eğitimi, eğitim MSE hata ve standart sapma değerleri.

Yapı		**ABC**	**PSO**	**DE**
4-1-1	**MSE**	0.014371	0.015009	0.014638
	SD	0.000051	0.001250	0.000694
4-2-1	**MSE**	0.006719	0.009799	0.008093
	SD	0.000083	0.003138	0.002087
4-3-1	**MSE**	0.004898	0.006829	0.005378
	SD	0.000348	0.002092	0.000828
4-4-1	**MSE**	0.004685	0.005840	0.004387
	SD	0.000279	0.001825	0.000829
4-5-1	**MSE**	0.004139	0.003956	0.003662
	SD	0.000525	0.001282	0.000716
4-6-1	**MSE**	0.003549	0.003786	0.002937
	SD	0.000072	0.000025	0.000015

elde edilen ortalama MSE ve standart sapma değerleri Mackey-Glass zaman serisinin eğitim verilerinde çarpım birimli ağlar ile Tablo 4.10'de, toplama birimli ağlar ile elde edilen sonuçlar Tablo 4.12'de verilmiştir. Mackey-Glass zaman serisinin test verilerinden elde edilen sonuçlar çarpım birimli MLP ağları ile Tablo 4.11'de, toplama birimli MLP ağlar ile Tablo 4.13'de kaydedilmiştir. Box-jenkins zaman serisi için ise çarpım birimli ağlar ile eğitim hatası Tablo 4.14'de, toplama birimli ağlar ile eğitim hatasıda Tablo 4.16'de sunulmuştur. Box-jenkins zaman serisinin test hataları çarpım birimli ağlar ile Tablo 4.15'de, toplama birimli ağlar ile Tablo 4.17'de verilmiştir. Tablolarda her bir algoritma için test verilerinde oluşan ortalama MSE hata değerleri saklı katmanlarda kullanılan farklı nöron sayıları için ayrı ayrı gösterilmiştir. Bunlarla birlikte iki problem için çarpım birimli ve toplama birimli YSA yapılarıyla yapılan eğitimlerden elde edilen sonuçların grafikleri, Mackey-Glass probleminin çözümü için çarpım birimli YSA'lar ile yapılan eğitimlerin eğitim verisi için Şekil 4.3(a) ve test verisi için Şekil 4.3(b), toplama birimli YSA'lar ile yapılan eğitimlerin eğitim verisi için Şekil 4.4(a) ve test verisi için Şekil 4.4(b); Box-Jenkins probleminin çözümü için çarpım birimli YSA'lar ile yapılan eğitimlerin eğitim verisi için Şekil 4.5(a) ve test verisi için Şekil 4.5(b), toplama birimli YSA'lar ile yapılan eğitimlerin eğitim verisi için Şekil 4.6(a) ve test verisi için Şekil 4.6(b) verilmiştir.

Tablo 4.11. Mackey-Glass problemi için çarpım birimli YSA eğitimi, test MSE hata ve standart sapma değerleri.

Yapı		**ABC**	**PSO**	**DE**
4-1-1	**MSE**	0.015340	0.016042	0.015722
	SD	0.000615	0.001119	0.000551
4-2-1	**MSE**	0.006564	0.010162	0.008100
	SD	0.000085	0.003628	0.002273
4-3-1	**MSE**	0.004735	0.006904	0.005239
	SD	0.000344	0.002512	0.000833
4-4-1	**MSE**	0.004536	0.005886	0.004210
	SD	0.000296	0.002203	0.000841
4-5-1	**MSE**	0.003987	0.003837	0.003488
	SD	0.000535	0.001442	0.000710
4-6-1	**MSE**	0.003404	0.003626	0.002750
	SD	0.000697	0.001332	0.001023

4.3.6. Tahmin Problemleri için YSA'ların Eğitilmesinden Elde Edilen Sonuçlar

Bu bölümde elde edilen sonuçları incelemeden önce eğitim hata sonuçlarının nasıl yorumlanması gerektiğini tartışacak olursak; aynı zaman diliminde (aynı sayıda çevrimde) çalışmalarına izin verildiğinde A algoritmasının B algoritmasından daha az hata üretiyor olması durumunda çıkarılacak iki sonuç vardır. Birincisi, belli bir hata değerine ulaşıldığında algoritmalar durduruluyorsa A algoritması B algoritmasından daha az zaman (veya işlem gücü) gerektirmektedir. İkincisi, A algoritması, B için oldukça karmaşık olan ağları aynı zaman zarfında başarıyla eğitebilecektir [151]. Bu bölümde yapılan çalışmalarda hem farklı yapıda yapay sinir ağları (toplama birimli ve çarpım birimli) hem de farklı mimarilerde yapay sinir ağları (saklı katmanlarda kullanılan nöron sayıları bakımından) ele alındığından algoritmaların performanslarındaki farklılıklar ve değişik yapıların performanslarındaki farklılıklar daha da göze çarpmaktadır.

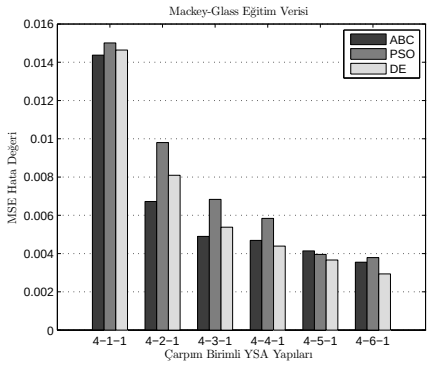
Her problem için optimum sayıdaki saklı katman sayısı farklı olmasına rağmen bu bölümde kullanılan problemlerde toplama birimli ağlar ve çarpım birimli ağlar için çeşitli varyasyonlar denenmiş ve bu yapıların öğrenme kapasiteleri karşılaştırılmıştır. Mackey-Glass verisinin eğitiminde çarpım birimli ağlarda ağ mimarisi değiştiğinde saklı katmandaki nöron sayısı artırıldıkça eğitim başarısı kısmen artmakta ancak

Tablo 4.12. Mackey-Glass problemi için toplama birimli YSA eğitimi, eğitim MSE hata ve standart sapma değerleri.

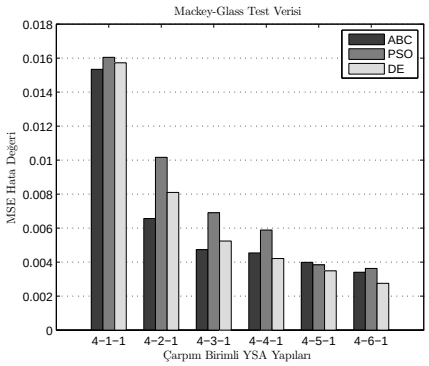
Yapı		**ABC**	**PSO**	**DE**
4-1-1	**MSE**	0.033623	0.033879	0.033668
	SD	0.000001	0.000880	0.000828
4-2-1	**MSE**	0.026016	0.027023	0.026262
	SD	0.000063	0.001287	0.000809
4-3-1	**MSE**	0.019812	0.022061	0.019950
	SD	0.000023	0.001775	0.000366
4-4-1	**MSE**	0.014979	0.019001	0.015914
	SD	0.000053	0.003285	0.000691
4-5-1	**MSE**	0.011444	0.016066	0.012124
	SD	0.000118	0.002820	0.001010
4-6-1	**MSE**	0.009075	0.014939	0.010005
	SD	0.000132	0.002562	0.001277

Tablo 4.13. Mackey-Glass problemi için toplama birimli YSA eğitimi, test MSE hata ve standart sapma değerleri.

Yapı		**ABC**	**PSO**	**DE**
4-1-1	**MSE**	0.033644	0.033864	0.033717
	SD	0.000059	0.000888	0.000079
4-2-1	**MSE**	0.025886	0.027010	0.026135
	SD	0.000013	0.001427	0.000744
4-3-1	**MSE**	0.019617	0.021870	0.019782
	SD	0.000031	0.001789	0.000407
4-4-1	**MSE**	0.014801	0.018891	0.015779
	SD	0.000079	0.003317	0.000680
4-5-1	**MSE**	0.011355	0.015959	0.012052
	SD	0.000141	0.002848	0.001050
4-6-1	**MSE**	0.009124	0.014868	0.010028
	SD	0.000166	0.002527	0.001245

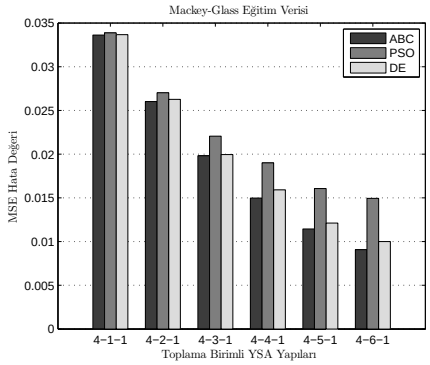


(a) Eğitim verisi.

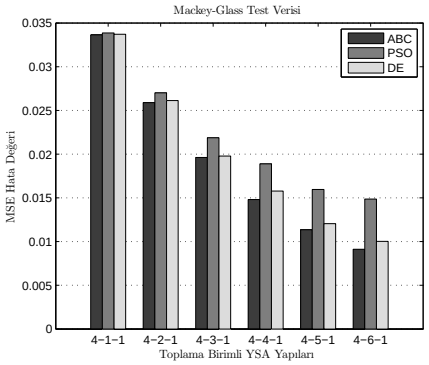


(b) Test verisi.

Şekil 4.3. Mackey-Glass problemi için çarpım birimli YSA eğitimi, eğitim ve test verileri MSE hata değerleri grafikleri.



(a) Eğitim verisi.



(b) Test verisi.

Şekil 4.4. Mackey-Glass problemi için toplama birimli YSA eğitimi, eğitim ve test verileri MSE hata değerleri grafikleri.

saklı katmanda kullanılan nöron sayılarının 3'den 6'ya kadar artışlarında birbirlerine yakın performans sergiledikleri gözlemlenmektedir. Aynı problemde toplama birimli ağ mimarisi kullanıldığında saklı katmandaki nöron sayısı arttıkça ağın eğitim başarısı daha çok artmaktadır. Algoritmaların başarıları incelendiğinde Mackey-Glass verisi için çarpım birimli ve toplama birimli yapay sinir ağlarının ele alınan 12 farklı modelinde ABC algoritması 9 modelde hem PSO algoritmasından hemde DE algoritmasından daha başarılı olmuştur. Test verileri incelendiğinde de yine toplama birimli ele alınan 6 farklı yapının hepsinde ve çarpım birimli 6 ağ yapılarının 3'ünde en başarılı algoritmanın ABC algoritması olduğu sonucu elde edilmiştir.

Tablo 4.14. Box-Jenkins problemi için çarpım birimli YSA eğitimi, eğitim MSE hata ve standart sapma değerleri.

Yapı		**ABC**	**PSO**	**DE**
2-1-1	**MSE**	0.005606	0.005731	0.005684
	SD	0.000144	0.000101	0.000079
2-2-1	**MSE**	0.000662	0.001522	0.001021
	SD	0.000094	0.001692	0.000967
2-3-1	**MSE**	0.000649	0.000659	0.000556
	SD	0.000083	0.000236	0.000137
2-4-1	**MSE**	0.000662	0.000550	0.000483
	SD	0.000094	0.000251	0.000029
2-5-1	**MSE**	0.000614	0.000450	0.000460
	SD	0.000078	0.000034	0.000020
2-6-1	**MSE**	0.000637	0.000436	0.000438
	SD	0.000072	0.000025	0.000015

Box-Jenkins verisi için elde edilen sonuçlar incelendiğinde ağ yapılarındaki farklılıkların belirgin şekilde sonuçlara yansıdığı dikkat çekmektedir. Burada, çarpım birimli yapay sinir ağlarının öğrenme kapasitelerinin toplama birimli yapay sinir ağlarının öğrenme kapasitelerinden çok daha yüksek olduğu göze çarpmaktadır. Şöyle ki, eğitim aşamasında çarpım birimli ağların ortalama kare hataları saklı katmanlarında 2 nörondan 6 nörona kadar değiştiğinde ortalama 0.000644 dolaylarında iken yine aynı sayıda nöronlarla toplama birimli yapay sinir ağlarının eğitim ortalama kare hataları 0.014922 civarlarındadır. Özellikle, saklı katmanında 2 nöron olduğunda çarpım birimli ağ mimarisinin eğitim hata değeri 0.000662'dir. Ancak toplama birimli ağ mimarisinin saklı katmanında nöron sayısı artırıldığında başarısı artarken bu katmanda 6 nöron bulundurulduğunda dahi eğitim hatası 0.006626'dır ki bu değer çarpım birimli 2 nöronlu ağdan elde edilen hata değerinin neredeyse 10 katıdır. Bu sonuç, toplama birimli yapay sinir ağlarında yüksek başarı elde edebilmek için ağın büyütülmesi gerektiği yani karmaşıklığın artmasına yol açacağı anlamına gelmektedir. Çarpım birimli ağların ise çok daha basit halleriyle oldukça başarılı sonuçlar verebilme yeteneklerinin olduğunu göstermektedir.

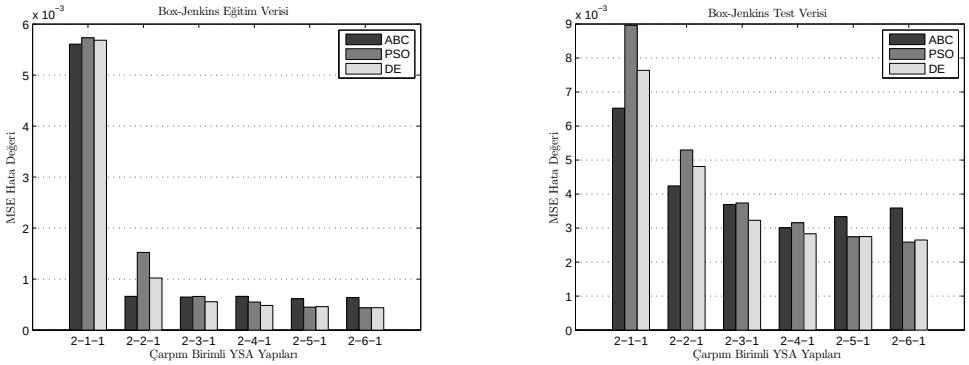
Box-Jenkins zaman serisinin eğitilmesinde algoritmaların performansları incelendiğinde çarpım birimli ağlar için eğitim aşamasında elde edilen sonuçlara

Tablo 4.15. Box-Jenkins problemi için çarpım birimli YSA eğitimi, test MSE hata ve standart sapma değerleri.

Yapı		**ABC**	**PSO**	**DE**
2-1-1	**MSE**	0.006525	0.008959	0.007637
	SD	0.000384	0.002811	0.002273
2-2-1	**MSE**	0.004240	0.005295	0.004809
	SD	0.000699	0.002683	0.002193
2-3-1	**MSE**	0.003690	0.003736	0.003229
	SD	0.000721	0.001003	0.000644
2-4-1	**MSE**	0.003008	0.003153	0.002832
	SD	0.000655	0.001330	0.000358
2-5-1	**MSE**	0.003335	0.002742	0.002749
	SD	0.000645	0.00258	0.000251
2-6-1	**MSE**	0.003589	0.002586	0.002650
	SD	0.000711	0.000142	0.000150

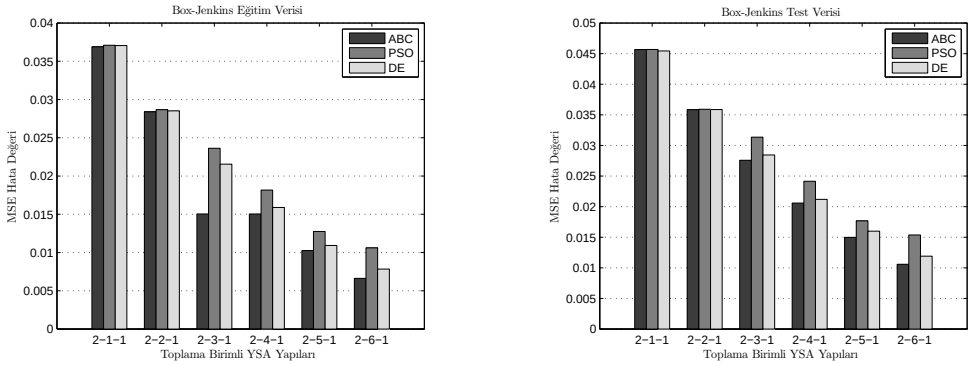
göre 3 algoritma da birbirlerine oldukça yakın sonuçlar üretmişlerdir: 6 farklı mimarinin ikisinde ABC, ikisinde PSO, ikisinde de DE algoritması en başarılı sonuçları vermiştir. Toplama birimli yapay sinir ağlarının eğitiminde ise 6 farklı mimaride ABC algoritması diğer iki algoritmadan daha başarılı olmuş, ABC algoritmasından sonra da PSO algoritması DE algoritmasından daha iyi sonuçlar üretmiştir. Test verilerine bakıldığında çarpım birimli ağlarda kullanılan 6 farklı mimarinin dördünde PSO algoritması en başarılı algoritma, ikisinde ABC algoritması en başarılı algoritma olmuştur. Toplama birimli ağlarda saklı katmanda 1 nöron bulunduran ağ mimarisinde PSO algoritması en az hata üreten algoritma olurken 5 farklı mimaride (saklı katmanlarda 2 ile 6 arası nörona sahip ağlar) test aşamasında ABC algoritması iki algoritmadan daha az hata değeri üretmiştir. Bu hata değerleriyle birlikte hata değerlerinin yayılımlarını gösteren standart sapma değerleri incelendiğinde ele alınan 48 farklı ağ kombinasyonunun (toplama-çarpım birimli ağlar, saklı katmanlarında farklı nöron sayılarıyla) 34'ünde ABC algoritması en düşük sapma değerine sahip olarak oldukça kararlı bir çalışma performansı sergilemiştir.

Sonuçlar incelendiğinde ABC algoritmasının ele alınan test problemleri için yapay sinir ağlarının eğitilmesinde PSO ve DE algoritmalarından daha başarılı ya da en az onlar kadar başarılı bir performans sergilediği, bunun yanısıra çok daha kararlı



(a) Eğitim verisi. (b) Test verisi.

Şekil 4.5. Box-Jenkins problemi için çarpım birimli YSA eğitimi, eğitim ve test verileri MSE hata değerleri grafikleri.



(a) Eğitim verisi. (b) Test verisi.

Şekil 4.6. Box-Jenkins problemi için toplama birimli YSA eğitimi, eğitim ve test verileri MSE hata değerleri grafikleri.

bir yapıya sahip olduğu açıkça görülmektedir. ABC Algoritmasının bu performansı dikkate alındığında çarpım birimli yapay sinir ağları yapılarının ve tahmin amaçlı yapay sinir ağlarının eğitilmesinde rahatlıkla kullanılabileceği sonucuna varılabilir. Ayrıca, çarpım birimli ağların toplama birimli ağlar ile aynı nöron sayılı yapıları karşılaştırıldığında bilgiyi daha iyi temsil edebildiği ve daha düzgün bir ağ eşleştirmesi yapabildiği sonucuna varılabilir.

Tablo 4.16. Box-Jenkins problemi için toplama birimli YSA eğitimi, eğitim MSE hata ve standart sapma değerleri.

Yapı		**ABC**	**PSO**	**DE**
2-1-1	**MSE**	0.036878	0.037095	0.037046
	SD	0.000878	0.000326	0.000035
2-2-1	**MSE**	0.028384	0.028673	0.028516
	SD	0.000012	0.000850	0.000312
2-3-1	**MSE**	0.015033	0.023628	0.021549
	SD	0.000010	0.002703	0.000804
2-4-1	**MSE**	0.015033	0.018165	0.015883
	SD	0.000011	0.003000	0.001408
2-5-1	**MSE**	0.010256	0.012752	0.010925
	SD	0.000017	0.001777	0.000908
2-6-1	**MSE**	0.006626	0.010611	0.007835
	SD	0.000029	0.001932	0.001395

Tablo 4.17. Box-Jenkins problemi için toplama birimli YSA eğitimi, test MSE hata ve standart sapma değerleri.

Yapı		**ABC**	**PSO**	**DE**
2-1-1	**MSE**	0.045661	0.045684	0.045436
	SD	0.000384	0.002811	0.002273
2-2-1	**MSE**	0.035840	0.035914	0.035858
	SD	0.000699	0.002683	0.002193
2-3-1	**MSE**	0.027579	0.031351	0.028441
	SD	0.000721	0.001003	0.000644
2-4-1	**MSE**	0.020583	0.024146	0.021181
	SD	0.000655	0.001330	0.000358
2-5-1	**MSE**	0.014968	0.017672	0.015988
	SD	0.000645	0.00258	0.000251
2-6-1	**MSE**	0.010572	0.015363	0.011906
	SD	0.000711	0.000142	0.000150

5. BÖLÜM

ÇOK KATMANLI ALGILAYICI (MLP) AĞLARININ SINIFLANDIRMA AMAÇLI YAPAY ARI KOLONİSİ ALGORİTMASI KULLANILARAK EĞİTİLMESİ

5.1. MLP Ağlarının Sınıflandırma Test Problemleri için Eğitilmesi

Yapay sinir ağları örüntü sınıflandırması, fonksiyon yaklaşımları, optimizasyon, örüntü algılama ve birleşmeli hafızalar gibi uygulamalardaki problem çözümlerinde başarıyla uygulanmaktadırlar. Yapay sinir ağlarının geniş uygulama sahasında örüntü algılama; sınıflandırma, konuşma tanıma, teşhis problemleri, tıp, finans, robotik kontrol, sinyal işleme ve bilgisayar görüş (vision) alanlarındaki çalışmalar ile en yaygın uyguluma alanı olarak karşımıza çıkmaktadır [160, 161]. Birçok farklı sinir ağı sınıflandırıcısı arasında, çok katmanlı ileri beslemeli algılayıcılar (MLP), evrensel yaklaşım yetenekleri ile sınıflandırma amaçlı kullanılan başlıca yapılardandır [162]. Sinir ağlarının başarısı büyük ölçüde mimari, eğitim algoritması ve eğitimde kullanılan parametrelerin seçimine bağlıdır. Tüm bunlar yapay sinir ağlarının tasarımını zor bir optimizasyon problemi haline getirir. Araştırmacılar, uygun mimari ve/veya eğitim algoritması ve/veya aktivasyon fonksiyonu ve/veya eğitim verileri tabanlı ağırlık ve eşik değerleri bulmaya yönelik ağ optimizasyonu üzerine çalışmaktadırlar. Bu yaklaşımların çoğunda topoloji ve aktivasyon fonksiyonları sabit tutulur ve ağırlıkların olası değerleri ayarlanmaya çalışılarak ağ eğitilir [163]. Teorik olarak, sinir ağları, sınırsız ağ boyutu ve sonsuz mevcut veri olması durumunda verilen her problemi çözecek şekilde tasarlanabilirler. Ancak, pratikte kaynaklar sınırlıdır ve ağın genelleme yeteneğine güvenilmelidir. Aslında ağ yapısı sabitleştirildiğinde, örneğin saklı katman sayısı ve her bir saklı katmandaki nöron sayısı belirlendiğinde, ağ, optimal ağırlık setlerini elde edene kadar ağırlık ayarlamaları yapmaya devam eder. Birçok pratik örüntü tanıma

probleminde, sinirsel ağlar uygun ağırlık setlerine yakınsama eğiliminde değildir. Bu problemi kısmen gidermek için bir dizi ağ optimizasyon yaklaşımı önerilmiştir [164–168]. Ağırlıkları optimize etmeye ve ağların yapı ayarlamalarına ek olarak, veri madenciliği alanında diğer yaklaşımlar ve teknikler üzerinde son yıllarda yoğun olarak çalışılmaktadır [169, 170].

İleri beslemeli çok katmanlı algılayıcı ağlar doğrusal olmayan modellemedeki başarısı ve genelleştirme yeteneği, adaptasyon ile kendini yönetebilme, gerçek zamanlı işlem yapabilme, hata toleransı gibi karakteristik özellikleri sayesinde araştırma sahasında yer alan birçok uygulamada tercih edilmektedirler. Uygun ağ yapısının ve optimum ağırlık değerlerinin bulunması yapay sinir ağ tasarımını zorlaştıran faktörlerdir. Başka bir ifadeyle YSA'ların başarısı, mimarisine, eğitim algoritmasına ve eğitimde kullanılan özellik seçeneklerine bağlıdır [171].

Tezin bu bölümünde, yapay arı kolonisi algoritmasının performansını değerlendirmek için 9 tane sınıflandırma problemi kullanılmıştır. Tüm veri setleri test problemlerinin verilerini düzenleyen ve saklayan UCI veri-havuzundan seçilmiştir [172]. ABC algoritmasının sınıflandırma performansı Geri Yayılım (BP), Levenberg-Marquardt (LM), Parçacık Sürü Optimizasyonu (PSO), Genetik Algoritma (GA) ve Diferansiyel Gelişim (DE) algoritmalarının performansları ile karşılaştırılmıştır. Veritabanındaki her problemin %75'lik ilk veri kısmı eğitim seti olarak geri kalan %25'lik kısmı ise test verisi olarak kullanılmıştır.

Bu çalışmada, ağırlıkların ayarlanması esnasında uyumluluk Standart Hata Yüzdesi (SEP) olarak isimlendirilen ağ hata fonksiyonu minimize edilerek çalışılmıştır. SEP hata fonksiyonu Eşitlik 5.1'de verilmiştir.

$$SEP = 100 \times \frac{O_{max} - O_{min}}{N.P} \sum_{j=1}^{N} \sum_{k=1}^{P} (d_k - o_k)^2 \tag{5.1}$$

Eşitlikte O_{max} ve O_{min} sırasıyla çıktı nöronun çıktı sinyalinin maksimum ve minimum değerlerini göstermektedir. N örnek sayısını, P sınıf sayısını, d_k çıktı nöronun istenen değerini ve o_k ise çıktı nöronunun o anki üretilen değerini temsil etmektedir. Veri setlerinde çıkışlar ait oldukları sınıfı temsil ederken 1 değerini, diğer durumlarda

ise 0 değerini aldıklarından dolayı sırasıyla O_{max} 1 değerine ve O_{min} 0 değerine eşitlenmiştir.

5.1.1. Sınıflandırma Test Problemleri

Kanser Problemi: Kanser verileri göğüs kanseri teşhis verisidir (tümörü iyi huylu veya kötü huylu olarak sınıflandırır). Veri seti 699 örnek içerir, her biri 9 girdi, 2 sınıfa sahiptir. İlk 525 örnek sinir ağlarının eğitiminde, geriye kalan 174 örnek test işleminde kullanılmıştır.

Diyabet Problemi: Diyabet verisi şeker hastalığı teşhis verisidir, bireydeki diyabet pozitif veya negatif şeklinde sınıflandırılır. Veri seti 768 örnek içerir. Her bir örnek 8 girdi ve 2 çıktıdan oluşur. İlk 576 örnek sinir ağlarının eğitiminde, geriye kalan 192 örnek test işleminde kullanılmıştır.

Kalp Problemi: Kalp verisi, kalp durumunun teşhisi amacıyla kullanılır (dört ana damardan en az birinin çapının %50 oranında azalıp azalmadığına karar verir). Veri seti 920 örnek içerir ve her bir örnek 35 girdi ve 2 çıktı içerir. İlk 690 örnek sinir ağlarının eğitiminde, geriye kalan 230 örnek test işleminde kullanılmıştır.

Kart Problemi: Kart veri seti müşteriye kredi kartı verilip verilmeyeceğine karar verilmesini sağlayan sınıflandırma verisidir. Veri seti 51 girdi ve 2 çıktı içerir. Toplamda 690 örneğin 518'i eğitim 172'si test için kullanılmıştır.

Gen Problemi: Gen veri seti nükleotid dizisindeki intron/exon sınırını ortaya çıkarmak için kullanılır. 3 sınıf çıktının içerdikleri; bir adet intron/exon sınırı, bir adet exon/intron sınırı ya da bunların hiçbiri şeklindedir. 120 giriş 60 DNA sıralı elemanın bir araya getirilmesiyle oluşur ve ikili girdi kullanarak ikiliye çevrilen dört değerli nominal değer bulunur. 120 giriş ile en fazla giriş verisi içeren veri seti gen veri setidir. 3175 örnekten oluşur ve bunların 2382 tanesi sinir ağlarının eğitiminde, geriye kalan 793 tanesi ise test işleminde kullanılmıştır.

Cam Problemi: Cam veri seti ele alınan problemlerin içinde ikinci en büyük sınıfa (6 sınıflı) sahiptir. Cam tiplerini sınıflandırmak için kullanılır; düz işlenmiş bina pencereleri, düz olmayan işlenmiş bina pencereleri, taşıt pencereleri, yiyecek kapları,

masa takımları gibi. Bu veri seti 9 girdi setinden meydana getirilmiştir; her bir sınıfa ait sırasıyla 70, 76, 17, 13, 9, 29 örnekleri bulunmaktadır. Toplamda 214 örneğin 161'i eğitim 53'ü de test için kullanılmıştır.

At Problemi: At veri seti, mide ağrısı çeken bir atın geleceğinin önceden tahmin edilmesinde kullanılır. At ölecek mi, kurtulacak mı yoksa ata ötenazi yapılmalı mı durumlarına göre üç tür sınıflandırma yapılır. Data seti her biri 58 girdi, 3 çıktısı olan 314 örnek içerir. Veri setindeki ilk 273 örnek eğitim için, geriye kalan 91 örnek de test işlemleri için kullanılmıştır.

Soya fasulyesi Problemi: Soya fasulyesi veri seti soya fasulyelerindeki 19 farklı hastalığı tanımlayan en fazla sınıf sayısına sahip veri setidir. Sınıflandırma, fasulyelerin özellikleri (boyutlarının ve renklerinin normal olup olmadığı gibi) ve bitki yapısının özellikleri (yapraklar üzerindeki benekler, bu beneklerin haleli olup olmadıkları, bitkinin büyümesinin normal olup olmadığı, köklerinin çürük mü sağlam mı olduğu gibi) temel alınarak yapılmaya çalışılır. Buna ek olarak bitkinin yaşamının tarihçesi hakkında bilgiler (geçen yıl ve son iki yılda ürün oluşumundaki değişiklikler, tohumların kimyevi maddelere maruz kalıp kalmadıkları, çevrenin sıcaklığı gibi) de sınıflandırmada göz önünde bulundurulan etkenlerdir. Veri seti 82 girdi ile 19 çıktı içerir ve 683 örneğin 513'ü eğitim 170'i test için kullanılmıştır.

Tiroit Problemi: Tiroit hastalığı veri seti tiroitin fizyolojik fonksiyonunun normalden az oluşu veya çok oluşuna göre teşhis edilir. 21 adet girdi, hastanın tiroit fonksiyonunun aşırı, normal veya normalin altında oluşuna göre 3 sınıf şeklinde sınıflandırma da kullanılır. Bu veri seti, 7200 örnekle en çok örnek içeren veri setidir. 5400 örnek eğitim seti, 1800 örnek seti test için ayrılmıştır.

Veri setlerine ait özellikler Tablo 5.1'de örnek sayısı, yapı içerisinde problemdeki giriş ve çıkış sayılarıyla birlikte verilmiştir. Problemlere ait veriler [172]'de (UCI) yer alan veri setlerinin [173]'de (Proben1) düzenlenmesiyle elde edilmiş ve kullanıma sunulmuş halleri ile kullanılmıştır. Bu veri setlerinden en yaygın olarak kullanılan ve en kısalarından iki tanesi kanser ve diyabet verileri kullanım şekillerini göstermek amacıyla Bölüm EK -2'de verilmiştir.

Tablo 5.1. Sınıflandırma test problemlerinin ve kullanılan MLP ağlarının parametreleri.

No	Problem	YSA Yapısı	Büyüklük	Örnek Sayısı
1	**Kanser**	9-6-2	74	699
2	**Diyabet**	8-6-2	68	768
3	**Kalp**	35-6-2	230	920
4	**Kart**	51-6-2	326	690
5	**Gen**	120-6-3	747	3175
6	**Cam**	9-6-6	102	214
7	**At**	58-6-3	375	364
8	**Soya**	82-6-19	631	683
9	**Tiroit**	21-6-3	153	7200

5.1.2. Sınıflandırma Test Problemleri için MLP Ağlarının Eğitilmesinde Yapılan Ayarlamalar

ABC algoritmasının performansını karşılaştırmak için gradyan tabanlı ve popülasyon tabanlı optimizasyon algoritmaları seçilmiştir. İlk sınıfta geri yayılım ve Levenberg-Marquart algoritmaları, sonraki sınıfta genetik algoritma, diferansiyel gelişim ve parçacık sürü optimizasyon algoritmaları ele alınmıştır. Bu algoritmalar Bölüm 2.4.1'de detaylı olarak analtılmıştı. Geri-Yayılım algoritması halen sinir ağları eğitiminde en yaygın olarak kullanılan tekniktir ve ağ çıktılarının hatalarını ve hata eğilim yönünü dikkate alarak ağ ağırlıklarını düzenler. Levenberg-Marquart algoritması Newton metodu ile eğim azalan metodu arasında ara değerler hesaplar. Levenberg-Marquart algoritması ikinci dereceden bir ifadeyle ağdaki hatayı azaltmaya çalışırken BP algoritması birinci dereceden bir ifadeyle hata değerini düşürmeye çalışır. Genetik algoritma bir popülasyondan alınan sonuçlar bir öncekinden daha iyi olacağı beklenen yeni bir popülasyon oluşturmak için kullanan evrensel araştırma tekniklerindendir. Yeni popülasyon oluşturmak için tahmini seçim, çaprazlama ve mutasyon gibi işlemler kullanılır. Diferansiyel gelişim algoritması genetik algoritmalar gibi popülasyon tabanlıdır; ayrıca çaprazlama, mutasyon, seçme gibi benzer operatörler kullanır. Genetik algoritma problem çözümlerinde daha iyi sonuçlar için çaprazlamayı kullanırken, DE algoritması mutasyon operatörünü kullanır. Parçacık sürü optimizasyon algoritması kuş sürülerinin yiyecek kaynaklarına uçuş davranışlarını modeller, parçacıklar muhtemel

çözümün uygunluğunu iteratif olarak hesaplar ve en iyi kaynağın yerini hafızasına alır.

Giriş katmanı, saklı katman ve çıkış katmanından oluşan 3 katmanlı MLP yapay sinir ağ yapısı her problem için kullanılmıştır. Giriş ve çıkış katmanlarındaki nöron sayısı, problemlerin özelliklerinde bahsedilen eşleştirilmiş örneklere bağlıdır. Saklı katmana ise deneme yanılma yoluyla yapılan çok sayıda deneyden sonra bu problemler için literatürde denemeleri yer alan ve tavsiye edilen bir yapı seçilmiş ve 6 nöron yerleştirilmiştir [174–178]. Ağ yapılarında ara ve çıkış katmanlardaki nöronlara biaseşik değerleri de uygulanmış, logaritma sigmoid fonksiyonu ise nöronların aktivasyon fonksiyonu olarak seçilmiştir. Ağ mimarisi Tablo 5.1'de verilmiştir, bu tabloda yapı içerisinde verilen ikinci değerler ara katmanlarda kullanılan nöron sayısını belirtmektedir.

Problemlerin giriş ve çıkış katmanlarındaki sayılara göre de problemin boyutunu temsil eden büyüklük değerleri eğitim aşamasında ayarlanacak ağırlıkların ve öndeğerlerin sayılarıdır. Tablo 5.1'de toplam örnek sayısı örnek sayısı sütununda gösterilmiştir. Uygulamalarda bu veri seti ağa veri eğitim seti ve test seti olmak üzere iki parça halinde uygulanır. Eğitim verileriyle sinir ağı eğitildikten sonra, YSA'nin görmediği test verileri ağa verilir ve ağın bu verilere vereceği tepki hesaplanır. Deneyler her bir problem için farklı tohum (seed) sayılarıyla oluşturulan rasgele popülasyonlar ile 30 kez tekrarlanmıştır. Ağın eğitim işlemi, SEP değeri 0.01 olduğunda veya maksimum jenerasyona ulaşıldığında durdurulmuştur. Maksimum jenarasyon/döngü sayısı 1000 çevrim değerine ve tüm parametreler için arama alanının sınırlarına [-2.0, 2.0] değerleri atanmıştır. ABC algoritması için koloni büyüklüğü 30 ve limit değeri 1000 olarak seçilmiştir. Genetik Algoritmada rulet tekerleği seçim seleksiyonu, bit-flip mutasyon, elitizm seçim metodları çalıştırılmış ve popülasyon boyutu 30 alınmıştır. Elitizm, yerine geçme ya da en iyi kromozomu veya birkaç en iyi kromozomu yeni nesile kopyalama metodunun adıdır. Bu yöntemin kullanılması mutasyon ve çaprazlamalarla yeni nesil oluştururken en iyi kromozomu seçmek için büyük bir şansa sahip olmayı sağlar. Elitizm çok hızlı bir şekilde GA'nın performansını artırır. Çünkü bulunan en iyi çözümü kaybetme ihtimalini önler [179]. BP için öğrenme oranı 0.01, LM için mutasyon oranı 0.001 ve mu_inc=10 olarak

alınmıştır. PSO algoritması için sürü büyüklüğü 30, eylemsizlik faktörü 0.7298 ve hızlandırma faktörü 1.4960 seçilmiştir. DE için popülasyon boyutu 30, çaprazlama oranı 0.9 ve F (ölçekleme faktörü) 0.5 olarak ayarlanmıştır. Bu seçilen değerler algoritmalar için tavsiye edilen değerler olup benzer çalışmalarda bilim insanlarının çoğunlukla tercih ettikleri değerlerden derlenmişlerdir [180–182].

5.1.3. Sınıflandırma Test Problemleri için MLP Ağlarının Eğitilmesinden Elde Edilen Sonuçlar

Her veri seti için tüm algoritmalar 30 kez koşulmuş ve her koşma aynı ayarlarla ancak farklı rasgele popülasyonlar ile başlatılmıştır. Sonuçta algoritmaların sınıflandırma başarılarının analizi için ortalama sınıflandırma hata yüzde (Classification Error Percentage, CEP) ve standart sapma değerleri raporlanmıştır. CEP, hatalı olarak sınıflandırılmış örneklerin yüzdesidir ve her problem için belirlenen test örneklerinden elde edilmektedir. Yapay sinir ağlarında kullanılan kazanan hepsini alır (winner takes all, WTA) metodu, eğitim sırasında bir nöronun diğer nöronları engellemesi durumudur. Buna göre belirli bir zamandan sonra tek bir nöron aktif olacak ve diğer nöronlar bu baskın nörondan dolayı görevini yitirecektir. WTA metodu kullanılarak, çıktı nörona bir sınıf tayin edip örnekler sınıflandırılmıştır. Elde edilen yeni çıktı istenen çıktıyla karşılaştırıldıktan sonra eğer tamamen aynı değillerse örnek yanlış sınıflandırılmış sayılmaktadır. Bu, tüm test verileri için hesaplanmakta ve toplamda hatalı sınıflandırılmış örnek sayısı, test veri setine bölünerek hatanın yüzdesi hesaplanmaktadır. CEP'in hesaplanması Eşitlik 5.2 kullanılarak tanımlanmaktadır:

$$CEP = 100 \times \frac{\text{hatalı sınıflandırılan test örnek sayısı}}{\text{toplam test verisi sayısı}} \tag{5.2}$$

Algoritmaların 30'ar kez çalıştırılmaları sonucu eğitim aşamasında her bir problem için elde edilen ortalama SEP hata değerleri ve bunların standart sapma değerleri Tablo 5.2'de verilmektedir. Aynı zamanda ABC algoritmasının her problem için eğitim aşamasındaki yakınsama grafikleri Şekil 5.2'de verilmiştir. Eğitim aşamasının ardından her problemin test verisinden ürettiği sınıflandırma başarıları ortalama

CEP hata ve standart sapma değerleri ile Tablo 5.3'de gösterilmiştir. Eğitim ve test aşamalarında elde edilen sonuçların grafiksel gösterimleri eğitim verisi için SEP değerleri Şekil 5.1(a)'de ve test verisi için CEP değerleri Şekil 5.1(b)'de sunulmuştur. Ortalama CEP değerine göre yapılan her veri seti için algoritmaların derecelendirme işlemi, dengeli olarak tüm algoritmaların performansını göstermek amacıyla Tablo 5.4'de sıralanmıştır. İlk bakışta Tablo 5.4 incelendiğinde ABC algoritmasının 3 problemde en düşük ortalama CEP hata değerlerini, 4 problemde en düşük ikinci CEP değerlerini ve 2 problemde ise üçüncü en uygun hata değerlerini ürettiği görülmektedir.

Veri setlerine bakıldığında, Kanser veri setinin uğraşılan diğer problemler içinde en basiti olduğu görülür. Bu probleminin eğitiminde tüm algoritmalar %10'nun altında hata sonuçları vermişlerdir. ABC %1.14 CEP değeriyle en iyi performans gösteren algoritmadır. DE %1.19 CEP değeriyle ikinci, GA ise %1.35 CEP değeri ile üçüncüdür. BP %1.89, PSO %2.01 ve LM %8.18 oranında ortalama sınıflandırma hata değerlerine sahip olmuşlardır.

Diyabet, girdi ve parametre boyutlarına göre Kanser'e benzer bir problem gibi görünmesine rağmen sonuçlara bakıldığında Kanser'den daha zor bir problem olduğu gözlemlenmektedir. Test CEP değerlerine bağlı olarak algoritmaların göstermiş oldukları performanslar Kanser'deki kadar iyi değildir. Diyabet probleminde en iyi sonucu %24.84 CEP değeriyle DE algoritması vermiştir. ABC %25.22, GA %26.51, PSO %27.50, BP %28.27, LM %29.80 oranında CEP değerlerini üretmişlerdir.

Kalp tanısı probleminde ABC %19.48 CEP değeri ve 0.97 standart sapma değeri ile tüm algoritmaları geride bırakmıştır. Diğer algoritmaların CEP değerleri birbirine yakın olup ve şu şekilde sıralanmaktadırlar: DE %20.33, GA %20.60, BP %21.44, PSO %21.87 ve LM % 24.96.

Kart probleminde bir müşteriye kredi kartı verilmesine karar verilmesi durumuna göre sınıflandırma yapılmakta ve iki çıktı üretilmektedir. Algoritmaların CEP değerleri ile performans sıralaması ABC %13.53, GA %13.56, BP %13.86, DE %13.90, PSO %15.58 ve LM %21.59 şeklindedir.

Gen problemi 120 girdi ve 747 ağırlık ile ele alınan problemlerden en fazla girdi ve parametreye sahip olanıdır. Ayrıca bu problem algoritmaların performanslarının farklılaşmasına sebep olmuştur. Popülasyon tabanlı algoritmaların performansları düşerken, gradyan tabanlı algoritmaların performanslarında yükselme görülmüştür. BP %11.37, LM %14.35, ABC %29.50, GA %31.02, DE %31.18 ve PSO %36.30 ortalama CEP değerleri üretmişlerdir.

Cam problemi 6 çıktı sınıfıyla ve 214 örnekle veri setindeki en zor problemlerden birisidir. Bu problemde görülüyor ki tüm algoritmalar test verilerinin neredeyse sadece yarısını sınıflandırabilmişlerdir ve CEP değerleri %43-59 arasında değişmektedir. LM %42.99, DE %44.90, ABC %45.62, GA %50.18, PSO %52.49 ve BP %59.09 CEP değerine sahiptirler. Algoritmaların bu denli düşük performans göstermelerinin sebebi olarak yetersiz eğitim seti gösterilebilir. Çünkü veri seti 6 sınıftan oluşurken sadece 153 örnek eğitim verisi olarak bu sınıflara yerleştirilmeye çalışılmaktadır. Eğitim setindeki örnek sayısının az olması yapay sinir ağının öğrenmesini güçleştirmektedir.

At verisi için algoritmaların CEP değerleri %28-34 arasında değişmektedir. CEP değerlerine göre algoritmaları sıralayacak olursak: BP %27.84, ABC %28.63, GA%29.12, DE %29.29, PSO %31.24, LM %33.79 şeklindedir.

Soya fasulyesi problemi 82 girdi ve 19 çıktıyla en fazla sınıfa sahip olan problemdir. Ancak bu problemde algoritmaların performansları diğer problemlerde gösterilen performanslarına pek benzememektedir. BP %61.16 ve DE %74.90 CEP değerleri ile oldukça kötü performans sergilemişlerdir. LM algoritması %25.51 CEP değeriyle en başarılı algoritma iken, ABC %38.63 ile ikinci, GA %40.39 ile üçüncü PSO %60.50 ile dördüncü sıradadır.

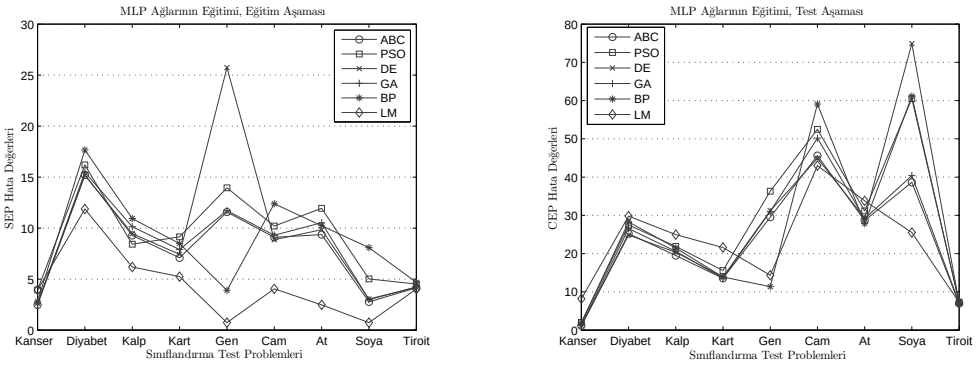
Tiroit problemi en fazla örnek sayısına sahiptir ve bunların 1800 tanesi test işleminde kullanılmıştır. Sonuçlar gösteriyor ki, LM dışındaki diğer algoritmaların CEP değerleri birbirine çok yakın ve %7 civarındadır. LM algoritması, bu problemde %2.61 CEP oranıyla en iyi performansı göstermiştir. Tiroit probleminde performans değerlerine göre algoritmaların sıralaması şu şekildedir: LM, ABC, DE, GA, BP, PSO.

Tablo 5.2. MLP ağlarının sınıflandırma test problemleri için eğitilmesinden elde edilen eğitim aşaması ortalama SEP hata ve standart sapma değerleri.

No	**Problem**		**ABC**	**PSO**	**DE**	**GA**	**BP**	**LM**
1	**Kanser**	**SEP**	2.47	3.94	2.63	2.57	2.83	3.93
		sd	0.01	0.87	0.22	0.05	0.08	9.70
2	**Diyabet**	**SEP**	15.21	16.19	15.17	15.48	17.66	11.87
		sd	0.07	0.78	0.42	0.13	4.45	6.42
3	**Kalp**	**SEP**	9.28	8.42	9.46	10.13	10.94	6.18
		sd	0.17	0.53	0.32	0.16	0.12	10.55
4	**Kart**	**SEP**	7.08	9.14	7.43	7.90	8.48	5.24
		sd	0.23	1.07	0.46	0.29	0.31	10.02
5	**Gen**	**SEP**	11.57	13.95	25.73	11.71	3.90	0.73
		sd	0.52	0.93	1.34	0.34	0.69	0.34
6	**Cam**	**SEP**	9.09	10.20	8.86	9.30	12.39	4.04
		sd	0.07	0.53	0.14	0.09	3.29	3.69
7	**At**	**SEP**	9.36	11.94	9.84	10.52	10.23	2.48
		sd	0.32	0.86	0.46	0.65	1.38	5.22
8	**Soya**	**SEP**	2.77	5.01	3.04	2.99	8.08	0.73
		sd	0.13	0.70	0.19	0.08	4.10	0.64
9	**Tiroit**	**SEP**	4.14	4.51	4.19	4.23	4.68	4.02
		sd	0.02	0.06	0.05	0.02	0.04	1.48

Tablo 5.3. MLP ağlarının sınıflandırma test problemleri için eğitilmesinden elde edilen test aşaması ortalama CEP hata ve standart sapma değerleri.

No	**Problem**		**ABC**	**PSO**	**DE**	**GA**	**BP**	**LM**
1	**Kanser**	**CEP**	1.14	2.01	1.19	1.35	1.89	8.18
		sd	0.00	0.86	0.28	0.28	0.43	10.29
2	**Diyabet**	**CEP**	25.22	27.50	24.84	26.51	28.27	29.80
		sd	0.97	2.05	1.32	1.21	6.30	3.77
3	**Kalp**	**CEP**	19.48	21.87	20.33	20.60	21.44	24.96
		sd	1.41	1.45	1.29	0.68	0.55	7.40
4	**Kart**	**CEP**	13.53	15.58	13.90	13.56	13.86	21.59
		sd	1.17	1.59	1.26	1.21	0.47	7.98
5	**Gen**	**CEP**	29.50	36.30	31.18	31.02	11.37	14.35
		sd	1.88	2.10	2.18	0.71	1.15	2.48
6	**Cam**	**CEP**	45.62	52.49	44.90	50.18	59.09	42.99
		sd	3.11	7.14	2.84	3.15	9.52	11.55
7	**At**	**CEP**	28.63	31.14	29.29	29.12	27.84	33.79
		sd	2.61	3.94	2.72	2.84	2.12	4.50
8	**Soya**	**CEP**	38.63	60.50	74.90	40.39	61.16	25.51
		sd	3.18	8.50	2.10	4.93	19.18	9.89
9	**Tiroit**	**CEP**	6.95	7.27	7.08	7.09	7.26	7.22
		sd	0.01	0.02	0.95	0.03	0.00	0.07



(a) Eğitim verisi. (b) Test verisi.

Şekil 5.1. MLP ağlarının eğitimi, eğitim (SEP) ve test (CEP) hata değerleri grafikleri.

Tablo 5.3'deki sınıflandırma hatalarında görüldüğü üzere ABC algoritması yapay sinir ağlarının eğitilmesinde üstün bir performansa sahiptir. Buna rağmen ABC algoritmasını tüm problemlerde en iyi performansı gösterdiği söylenemez. Problemlerde özellikle sinir ağı yapısı seçme veya ayarlama işlemi yapılmamıştır. Bunun yerine literatürde en fazla kullanılan ağ yapıları referans alınmış ve deneme yanılma yöntemiyle kullanılacak yapılara karar verilmiştir. Farklı yapay sinir ağı mimarileri her bir veri setine uygulandığında daha iyi sınıflandırma sonucu elde etmek mümkün olabilir. Algoritmaların genel eğitim hata değerleri sınıflandırma başarılarıyla birlikte Tablo 5.4'de algoritmaların her bir problem için başarı sıraları verilmiştir. En iyi performans gösterenden, en düşük performans sergileyene göre sıralandıklarında algoritmalar; ABC, DE, GA, BP, LM, PSO şeklinde sıralanmaktadırlar. ABC, ele alınan sınıflandırma test problemlerinde düşük sınıflandırma hataları sergilemesinin yanında standart sapma değerlerinin düşük olması da dikkat çekmektedir. ABC algoritmasının tüm problemlerde standard sapma değerleri oldukça düşüktür yani algoritmalar her koşmada farklı başlangıç popülasyonları ile başlamışlar ve ABC algoritması diğer algoritmalara göre daha kararlı davranışlar sergilediği söylenebilir.

Sonuç olarak, bu bölümdeki çalışmada yapay arı kolonisi algoritması makine öğrenmesi alanında yaygınca kullanılan veri kümelerini sınıflandırmak amacıyla MLP yapay sinir ağlarının eğitiminde kullanılmıştır. Algoritmanın performansı, iki gradyan tabanlı algoritma (BP, LM) ve üç popülasyon tabanlı algoritma (DE, GA,

Tablo 5.4. Sınıflandırma başarılarına (CEP) göre algoritmaların problemler için sıralaması.

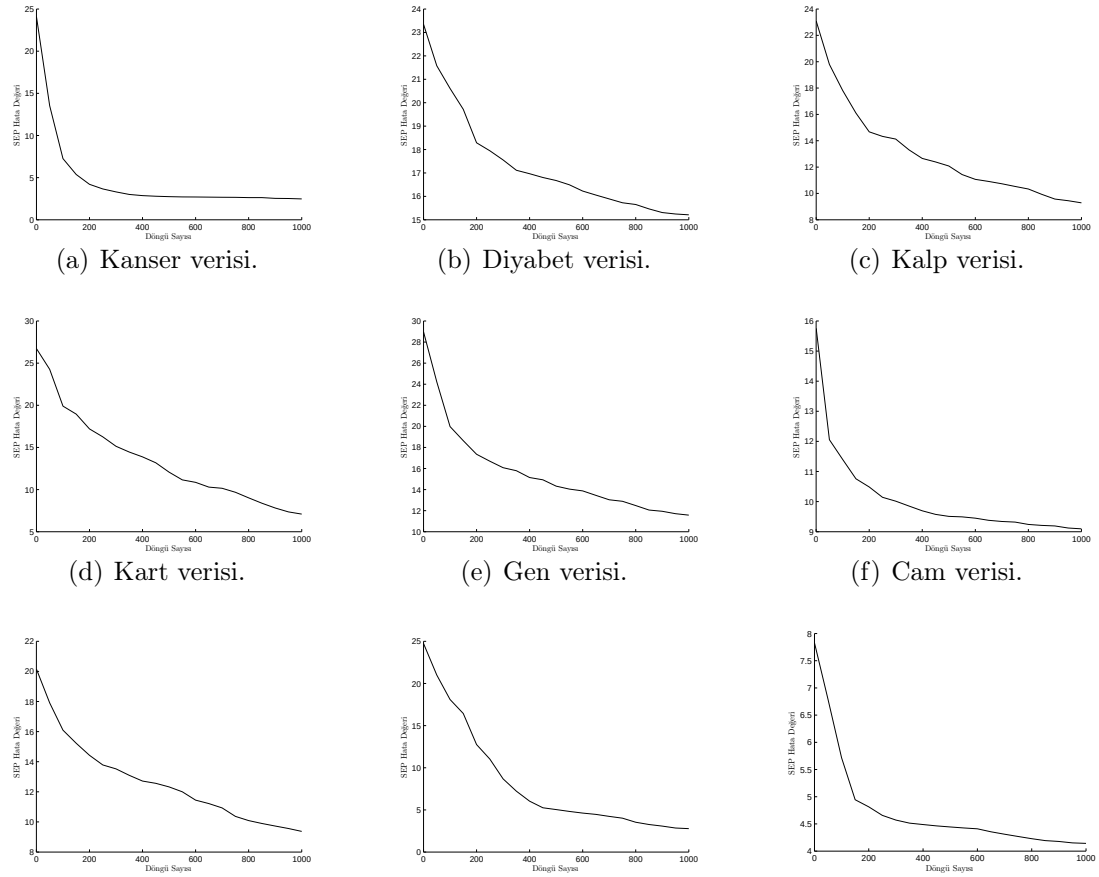
No	Problem	ABC	PSO	DE	GA	BP	LM
1	**Kanser**	1	5	2	3	4	6
2	**Diyabet**	2	4	1	3	5	6
3	**Kalp**	1	5	2	3	4	6
4	**Kart**	1	5	4	2	3	6
5	**Gen**	3	6	5	4	1	2
6	**Cam**	3	5	2	4	6	1
7	**At**	2	5	4	3	1	6
8	**Soya**	2	4	6	3	5	1
9	**Tiroit**	2	6	3	4	5	1
Toplam		17	45	29	29	34	35

PSO) ile karşılaştırılmıştır. Deneylerin sonuçları ABC algoritmasının sınıflandırma problemlerinde başarıyla ileri beslemeli çok katmanlı algılayıcı yapay sinir ağlarının eğitiminde kullanılabilirliğini göstermiştir.

5.2. MLP Ağlarının ABC ile Eğitilmesini Etkileyen Faktörlerin Etkilerinin İncelenmesi

Yapay arı kolonisi algoritmasında koloni büyüklüğü, çevrim sayısı ve limit değeri olmak üzere üç temel kontrol parametresi bulunmaktadır. Bunlardan koloni büyüklüğü ve çevrim sayısı bütün popülasyon tabanlı algoritmalarda farklı isimlerde bile olsa popülasyon boyutu ve her bir popülasyonun kaç sefer yenilenip değerlendirileceğini belirleyen jenerasyon sayısı olarak yer almaktadır. ABC algoritmasında bir çözümün iyileştirme çalışmalarının ne zaman sonlandırılacağına karar veren limit parametresi algoritmanın performansını direkt etkileyen parametredir. Bu sebeple, limit parametresinin ele alınan sınıflandırma test problemlerinden araştırmalarda en çok tercih edilen kanser, diyabet ve kalp problemleri için MLP ağlarının eğitilmelerindeki etkileri bu bölümde incelenmiştir.

Yapay sinir ağlarının eğitilmesinde standart hata yüzdesi kullanılmasının yerine eğitim aşamasında sınıflandırma hatasının kullanılması sınıflandırma çalışmalarında bilim insanları tarafından çok sık olmasa da tercih edilen yaklaşımlardandır [183]. Kanser, diyabet ve kalp verilerinin eğitim aşamasında standart hata yüzdesi



(a) Kanser verisi. (b) Diyabet verisi. (c) Kalp verisi.

(d) Kart verisi. (e) Gen verisi. (f) Cam verisi.

(g) At verisi. (h) Soya verisi. (i) Tiroit verisi.

Şekil 5.2. MLP ağlarının eğitilmesinde ABC algoritmasının ortalama eğitim hata yakınsama grafikleri.

kullanılması yerine sınıflandırma hatasının kullanılması ile ilgili yapılan eğitimlerin sonuçları da bu bölümde araştırılmıştır. Yapay sinir ağlarının ağırlık ve eşik değerlerinden oluşan ağırlık setinin alacağı değer aralıklarının değiştirilmesinin eğitim başarısına etkisi de bu bölümde ele alınan çalışmalardandır.

Toplama-birimli YSA yapıları yerine çarpım-birimli YSA yapılarının sınıflandırma perfromansları araştırılmıştır. Bunlarla birlikte problemlerin verileri düzenlenirken çapraz-geçerlilik yöntemleri kullanılarak eğitim ve test verilerinin değiştirilmesinin eğitim ve test başarılarına etkileri incelenmiştir.

5.2.1. Limit Parametresinin Değiştirilmesinin Etkisi

Limit parametresinin etkisinin analizi için yapılan çalışmalarda koloni büyüklüğü

Tablo 5.5. Limit parametresinin test verisine etkisi, test hata ve standart sapma değerleri.

		LİMİT DEĞERLERİ			
Problem		**2000**	**1000**	**500**	**100**
Kanser	**CEP**	1.17	1.14	1.18	1.19
	sd	0.09	0.00	0.06	0.13
Diyabet	**CEP**	25.34	25.22	25.37	26.20
	sd	0.92	0.97	1.01	1.22
Kalp	**CEP**	19.40	19.48	19.81	20.66
	sd	1.07	1.41	1.34	1.89

30 ve maksimum çevrim sayısı 1000 olarak ayarlanmıştır. Bölüm 5.1.3'de ABC algoritmasının sonuçları limit parametresi 1000 alındığı durum için verilmişti. Bu bölümde bu değerle elde edilen sonuçlarla birlikte limit değerleri 2000, 500 ve 100 seçildiğinde ABC algoritmasının performansındaki değişiklikler 30'ar koşma sonucunda elde edilen ortalama hata ve standart sapma değerleri kaydedilerek incelenmiştir. Tablo 5.5'de test verisi üzerindeki sınıflandırma başarısı farklı limit parametre değerleriyle 30'ar koşma sonucu ortalama CEP hata ve standart sapma değerleriyle sunulmuştur.

Tablo 5.5 incelendiğinde limit parametresinin küçük değerler alması (100 ve 500 değerleri) özellikle kansere göre daha zor problemler olan diyabet ve kalp problemlerinde başarıyı düşürmüş ve daha yüksek sınıflandırma hatasının elde edilmesine yol açmıştır. Bunun sebebi, küçük limit değerleriyle algoritma çalıştırıldığında algoritmanın bazı çözümleri, çözümlerin etrafında yeterince arama gerçekleştirmeden o çözümleri terk etme ihtimali olmasındandır.

Limit değerinin 2000 alınması ise algoritmanın neredeyse kaşif arılarsız çalıştırılması anlamına gelmektedir. Bu durum, algoritmanın keşif yeteneğinin azaltılması anlamına gelmektedir ancak, bunun yanında yüksek limit değerleri seçildiğinde algoritmanın kararlılığının arttığı da Tablo 5.5'den limit değerleri 1000 veya 2000 ayarlandığında oluşan standart sapma değerlerinin düşüklüğüne bakılarak söylenebilir.

5.2.2. Sınıflandırma Hatasının Kullanılmasının Etkisi

Sınıflandırma amaçlı yapay sinir ağlarının eğitiminde optimizasyonun sınıflandırma hatasına dayanarak yapıldığına literatürde rastlanmaktadır. Sınıflandırma hatasının test verileri için hesaplanması Eşitlik 5.2 ile tanımlanmıştı. Eğitim verileri için hesaplanmasında ise kullanılan veri eğitim verisi olduğundan hatalı sınıflandırılan eğitim veri sayısı, toplam eğitim veri sayısına bölünerek hesaplanmaktadır. Eşitlik 5.3'de eğitim sınıflandırma hatası (ECEP) tanımlanmaktadır:

$$ECEP = 100 \times \frac{\text{hatalı sınıflandırılan eğitim örnek sayısı}}{\text{toplam eğitim verisi sayısı}} \tag{5.3}$$

Eğitim aşamasında sınıflandırma hatasının kullanılmasıyla elde edilen sonuçlar yine 30'ar koşma sonucunda hem eğitim verisi için hem de test verisi içi ayrı ayrı kaydedilmiş ve Tablo 5.6'de sunulmuştur. Bu sonuçlar alınırken ABC algoritması yine 30 koloni büyüklüğü ile 1000 çevrim ve 1000 limit değerleriyle koşulmuştur. Eğitim verisi için elde edilen sınıflandırma hatası (ECEP) ve sapma değerleri parantez içerisinde, test verisi için elde edilen sınıflandırma hatası (CEP) ve bunların sapma değerleri parantez içerilerinde gösterilmiştir.

Standart hata yüzdesi ile eğitim yapıldığında elde edilen, Bölüm 5.1.3'de Tablo 5.2'de ve Tablo 5.3'de verilen sonuçlarla karşılaştırıldığında eğitim aşamasında sınıflandırma hatasının kullanılmasının üç problemde de hem test sınıflandırma başarılarını düşürdüğü hem de standart sapma değerlerini yükselttiği gözlemlenmektedir. Kanser problemi için CEP hata değeri %1.14 iken %2.53; diyabet problemi için %25.22 iken %27.20; kalp problemi için ise %19.48 olan CEP hata değeri %23.91 olmuştur. Bu sonuçlar dikkate alındığında yaygınca kullanılan standart hata yüzdesinin problemlerin eğitiminde oldukça başarılı genelleştirme yeteneğine sahip olduğu ancak bununla birlikte sınıflandırma hatasının eğitim aşamasında kullanılmasının çokta uygun olmadığı rahatlıkla vurgulanabilir.

5.2.3. Koloni Büyüklüğünün ve Parametre Aralık Değerlerinin Etkisi

Bu bölümde şu ana kadar yapılan çalışmalarda kullanılan yapay sinir ağlarının

Tablo 5.6. Eğitim aşamasında sınıflandırma hatasının (ECEP) kullanılması sonucu elde edilen eğitim ve test sınıflandırma hata (CEP) ile standart sapma değerleri.

Problem	**ECEP**	**CEP**
Kanser	2.84 (0.19)	2.53 (0.98)
Diyabet	22.91 (1.15)	27.20 (3.06)
Kalp	16.26 (0.97)	23.91 (2.42)

Tablo 5.7. Parametre aralığının [-1,1], koloni büyüklüğünün 20 seçilmesi sonucu eğitim ve test hata ile standandard sapma değerleri.

Problem	**SEP**	**CEP**
Kanser	2.68 (0.17)	1.73 (0.75)
Diyabet	16.61 (0.47)	27.95 (2.96)
Kalp	9.92 (0.81)	22.43 (2.30)

eğitilmesinde ağırlık ve eşik değerlerinin alabileceği değer aralığı [-2,2] olarak ayarlanmıştı. Ağırlık değerlerinin alabileceği aralığın değiştirilmesinin etkisinin araştırılması amacıyla ağırlıkların alabileceği değerler [-1,1] ve [-5,5] şeklinde ayrı ayrı seçilerek sonuçlar alınmıştır. Parametre aralığı [-5,5] seçildiğinde kullanılan eğitim algoritmasının daha büyük bir arama uzayında arama yapacak olmasından dolayı daha fazla değerlendirme sayısına ihtiyaç duyulabileceği göz önünde bulundurularak koloni büyüklüğü de 30 yerine 60 olarak ayarlanmıştır. Yapılan değişikliklerin diğer değişikliler ile tutarlı olması amacıyla parametre aralığı [-1,1] seçildiğinde de koloni büyüklüğü 20 olarak alınmıştır. Bu ayarlamalar ile 30'ar koşma sonucu yapılan eğitimler ile elde edilen eğitim verisi için ortalama SEP hata ve test verisi için ortalama CEP hata değerleriyle bunların standart sapma değerleri sırayla Tablo 5.7 ve 5.8'de sunulmuştur.

Parametre aralığı [-1,1] alındığında elde edilen ve Tablo 5.7'de verilen sonuçlara göre YSA'ların sınıflandırma başarıları oldukça düşmektedir. Kanser için %1.73, diyabet için %27.95 ve kalp için %22.43 ortalama sınıflandırma hata değerleri elde edilmiştir. Parametre aralığı [-5,5] olarak ayarlandığında elde edilen Tablo 5.8'de sunulan sonuçlar değerlendirildiğinde üç problem için de sınıflandırma başarısının [-1,1] aralığına göre daha iyi olduğu ancak [-2,2] aralığı seçildiğinde elde edilen

Tablo 5.8. Parametre aralığının [-5,5], koloni büyüklüğünün 60 seçilmesi sonucu eğitim ve test hata ile standandard sapma değerleri.

Problem	**SEP**	**CEP**
Kanser	2.11 (0.07)	1.61 (0.66)
Diyabet	15.52 (0.18)	26.34 (1.62)
Kalp	9.32 (0.29)	20.49 (1.71)

sonuçlara göre daha başarısız olduğunu görülmektedir. Bu sonuçlara ek olarak birlikte her problem için farklı koloni büyüklüğü, parametre aralık değerleri ve limit değerleri içi sonuçlar alınmasına rağmen birbirlerine benzer sonuçlar olduğundan dolayı her problem için aynı ifadelerin tekrarına yer vermemek adına hepsi ayrı ayrı sunulmamıştır.

5.2.4. Çarpım Birimli Yapay Sinir Ağlarının Kullanılmasının Etkisi

Çarpım birimli yapay sinir ağlarının yapıları ve özellikleri Bölüm 4.3'de anlatılmış ve genelikle dinamik sistemlerin modellenmesinde tercih edilen ağlar oldukları vurgulanmıştı. Literatürde az da olsa sınıflandırma amaçlı kullanıldıklarına rastlanmaktadır [155, 184–187]. [184]'de kanser, diyabet, balans, hipotiroit ve dalgaformu problemlerinin sınıflandırılması amacıyla çarpım-birimli ağlar gelişim algoritmalarıyla eğitilmiştir. [185]'de PSO ile eğitilen çarpım-birimli ağlar iris, şarap ve tiroit problemlerinin sınıflandırılmasında kullanılmıştır. Kanser, diyabet, balans ve cam problemlerinin sınıflandırılmasında en uygun çarpım-birimli ağ yapısı [187]'deki çalışmada araştırılmıştır. [155, 186]'deki çalışmalarda sınıflandırma amaçlı çarpım-birimli YSA yapılarına dayalı hibrid ağlar oluşturulmuş ve UCI veritabanından alınan 4'er problemle test edilmiştir.

Tez çalışmasında ele alınan sınıflandırma test problemlerinden kanser, diyabet ve kalp problemlerinin sınıflandırılması amacıyla kullanılan çok katmanlı ağlarda toplama birimleri yerine çarpım birimleri kullanılmasının eğitim hatasına ve sınıflandırma başarısına etkisini incelemek amacıyla Bölüm 5.1.2'de oluşturulan MLP ağları aynı özelliklerle çarpım birimleriyle kurulmuştur. Eğitim verisinden elde edilen hata SEP değerleriyle, test verisinin başarısıda CEP değerleri ile

Tablo 5.9. Sınıflandırma test problemleri için çarpım birimli ağların eğitilmesinden elde edilen eğitim ve test sınıflandırma hata ile standart sapma değerleri.

	Çarpım Birimli Ağlar		**Toplama Birimli Ağlar**	
Problem	**SEP**	**CEP**	**SEP**	**CEP**
Kanser	2.83 (0.11)	2.72 (0.81)	2.47 (0.01)	1.14 (0.00)
Diyabet	15.83 (0.16)	25.73 (1.20)	15.21 (0.07)	25.22 (0.97)
Kalp	17.42 (0.23)	25.79 (1.43)	9.28 (0.17)	19.48 (1.41)

raporlanmıştır. Bu hata değerleri ve bunların standard sapma değerleri Tablo 5.9'da verilmiştir.

Bölüm 4.3'de verilen yapıların eğitilmesinde performanslarının çok farklılık göstermeleri üzerine sınıflandırma amaçlı kullanılan çarpım-birimli MLP ağlarda saklı katmanlarında 2, 4 ve 6 nöron bulunduran yapılar ile test edilmiştir. Bunlardan en başarılıları her üç problemde de saklı katmanında 6 nöron bulunduran ağ yapısı olduğundan bu ağların sonuçları verilmiştir. Çarpım-birimli MLP ağlarının sınıflandırma başarıları incelendiğinde her üç problemde de toplama-birimli ağlara göre hem eğitim aşamasında hem de test aşmasında daha başarısız olduğu açıkça görülür. Sadece diyabet probleminde toplama-birimli MLP ağların sonuçlarına yakın sonuçlar üretebilirken diğer iki problemde oldukça kötü performans sergilemiştir. Aynı zamanda sadece kalp probleminin sınıflandırma yüzde hatasının standard sapma değeri çarpım-birimli ağlar ve toplama-birimli ağlarda birbirine yakındır ve üç problemde de hem standard yüzde hata değerlerinin hem sınıflandırma yüzde hata değerlerinin dağılımında çarpım-birimli ağlar toplama-birimli ağlara göre daha kararsız bir yaklaşım sergilemişlerdir.

5.2.5. Eğitim Verilerinin Seçiminin Etkisi

Sınıflandırma çalışmalarında sınıflandırıcının eğitimi için ayrılan veri sayısının az olması test başarımını olumsuz yönde etkileyebilmektedir. Sınıflandırma yaklaşımının sınıflandırmadaki güvenilirliğini ve geçerliliğini test etmek amacıyla yaygın olarak kullanılan yöntem çapraz geçerlilik (cross-validation) testidir. Çapraz geçerlilik tüm veriyi eğitim ve test aşamalarında kullanmayı sağlar. Veri iki kısma

ayrılır, önce ilk veri setiyle eğitilir ve sonra diğer kısmıyla test edilir. Ardından ikinci veri setiyle eğitilir ve birinci veri setiyle test edilerek ortalamaları alınır. Veri setini ikiye bölmek yerine veriler rasgele ya da K-Katlı çapraz geçerlilik yaklaşımlarıyla da düzenlenebilirler. Rasgele çapraz geçerlilik (random holdout cross-validation) modelinde verinin belli bir kısmı rasgele seçilen örneklerle test için ayrılırken kalan kısım eğitimde kullanılır. Bu işlem her eğitim aşamasının başlangıcında tekrarlanır, yani her yeni eğitim işleminde eğitim ve test verileri rasgele belirlenir. K-katlı çapraz geçerlilik (K-fold cross-validation) yönteminde ise ilk olarak veri seti K sayısı kadar alt sete bölünür [188]. K-1 adet alt veri seti eğitim için kullanılırken kalan 1 adet alt veri seti eğitilen ağın testi için kullanılır. Bu işleme bütün alt veri setleri eğitim dışında bırakılıp test edilinceye kadar devam edilir. K adet veri setinin test başarısının ortalaması alınarak tek bir geçerlilik değeri elde edilebilir.

Tez çalışmasında, sınıflandırma problemlerinin eğitim ve test verileri seçilirken rasgele çapraz geçerlilik (RCV) ve 4-katlı çapraz geçerlilik (KCV) yöntemleri uygulanmıştır. RCV'de üç problem için yapılan 30'ar koşma da her koşmanın başlangıç aşamasında veri setlerinden %25'lik kısım rasgele seçilmiş ve kalan %75'lik kısım eğitim veri seti olarak kullanılmıştır. 4-katlı çapraz geçerlilik yöntemi uygulanırken veri setleri eşit şekilde %25'lik dört parçaya bölünmüştür. Her parça test veri setini oluşturacak şekilde 10'ar kez koşulmuş ve toplam 40 koşmanın sonucunda elde edilen hata oranlarının ortalaması kaydedilmiştir. Eğitim hata ve standard sapma değerleri Tablo 5.10'de, eğitimlerin tamamlanmasının ardından test aşamasında elde edilen SEP ve standard sapma değerleri Tablo 5.11'da gösterilmiştir. Tablolarda Bölüm 5.1.3'de verilen sonuçlar standard çapraz geçerlilik (SCV) başlığı altında verilmiştir.

Kullanılan eğitim verilerinin ve test verilerinin Proben1 veritabanında [173] verilen sıralarının dışında kullanılması eğitim standard yüzde hatasına üç problem için de önemli bir etki oluşturmamıştır. Üç problemde de birbirlerine yakın SEP değerleri elde edilmiş olup rasgele veri seçtiren RCV yönteminde eğitim hata değerlerinin standard sapma değerleri daha yüksektir. Ancak, test performansları incelendiğinde kanser probleminde KCV ve RCV yöntemleri başarıyı düşürmüş, diyabet ve kalp problemlerinde ise sınıflandırma başarısının artmasına sebep olmuştur. Rasgele

Tablo 5.10. Çapraz geçerlilik yöntemleriyle eğitim verilerinin oluşturulması, SEP hata ve standart sapma değerleri.

Problem		**SCV**	**KCV**	**RCV**
Kanser	**SEP**	2.47	2.05	2.12
	sd	0.01	0.02	0.25
Diyabet	**SEP**	15.21	15.32	15.35
	sd	0.07	0.09	0.39
Kalp	**SEP**	9.28	9.19	9.53
	sd	0.17	0.15	0.39

Tablo 5.11. Çapraz geçerlilik yöntemleriyle eğitim verilerinin oluşturulması, CEP hata ve standart sapma değerleri.

		Çapraz Geçerlilik		
Problem		**SCV**	**KCV**	**RCV**
Kanser	**CEP**	1.14	3.26	2.91
	sd	0.00	0.02	1.21
Diyabet	**CEP**	25.22	22.87	22.48
	sd	0.97	0.55	2.69
Kalp	**CEP**	19.48	17.28	17.22
	sd	1.41	1.22	2.24

çapraz geçerlilik yöntemiyle düzenlenen veriler ile yapılan eğitimlerde oluşturulan ağlar daha kararsız olmalarına rağmen diyabet ve kalp problemlerinde sınıflandırma performansı açısından daha başarılı sınıflandırma performansı sergileyen ağlar olmuşlardır.

5.3. Yapay Sinir Ağlarının Hibrid Yapay Arı Kolonisi Algoritması ile Eğitilmesi

Optimizasyon alanında ele alınan birçok zor problemin çözümünde bilim insanları çalışmalarında optimizasyon tekniklerini tek başlarına kullanmak yerine birkaç tekniği birarada kullanmayı denemişlerdir [189]. Yöntemlerin birleştirilmesiyle oluşan algoritmalar genelde hibrid (melez, karma) algoritmalar şeklinde tanımlanır ve aynı optimizasyon amacı için iki veya daha fazla algoritmanın birlikte kullanılmasını ifade ederler.

Algoritmaların birleştirilmesi; tekniklerin bazı mekanizmalarında başka tekniklerin

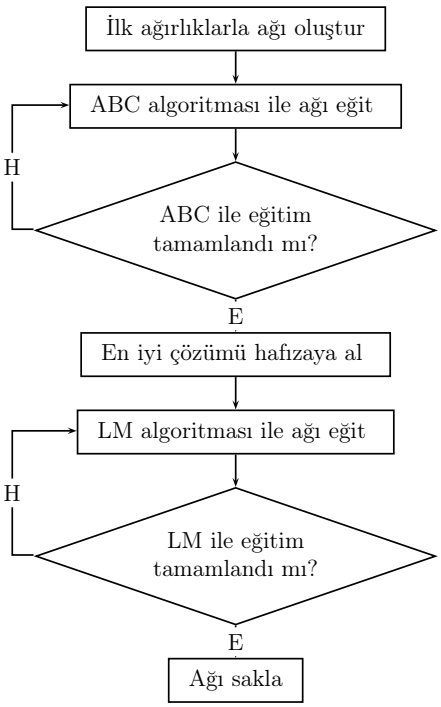
mekanizmalarından yararlanılması, kullanılan bir metodun çalışmasında bir bölümünün çıkartılarak o bölüme başka bir tekniğin bir kısımının monte edilmesi ve/veya algoritmaların güçlerini birleştirerek problemin çözümünün aranmasında algoritmalar tarafından iş bölümü yapılması şeklinde gerçekleştirilmektedir. Bu şekilde birleştirilen yöntemler genelde küresel araştırma yeteneği iyi olan algoritmalarla bölgesel araştırma yeteneği yüksek algoritmaların birlikte çalıştırılmasıyla ortaya çıkarılan hibrid tekniklerdir [190].

Yapay sinir ağlarının eğitimi konusunda da hibrid yapılar çeşitli şekillerde kullanılmıştır. Örneğin, Genetik algoritma ve Simplex bölgesel araştırma algoritmasını birleştirip XOR ve iris problemleri için yapay sinir ağlarının eğitiminde kullanılmıştır [191]. PSO ve BP algoritmalarının biraraya getirilmesi ile oluşturulan hibrid PSO-BP algoritması 3-bit parite problemi ve iris problemleri için [192]'de, deneysel üretilen verilerin sınıflandırılması için [193]'de ve kutup mod dağılımı problemi için [194]'de kullanılmıştır. Benzer şekilde PSO algoritmasının iki klasik teknikle melezleştirildiği model ile kanser, diyabet ve kalp hastalıkları sınıflandırılmaya çalışılmıştır [195]. Bu üç problemin çözümü için hibrid genetik algoritma [196]'de denenmiştir. Genetik algoritma ile BP algoritmasının karma yapısı ekonomik analiz ve modellemede [197], BP ve LM ile oluşturulan karma yapı kanser, diyabet ve kalp sınıflandırması problemlerinin çözümünde [181]'de araştırılmıştır. [198]'de ise RBF ağları için karma bir model oluşturulmuş ve bu modelin sınıflandırma performansı test edilmiştir.

5.3.1. ABC-LM Hibrid Algoritmasının MLP Ağlarının Eğitiminde Kullanılması

Karma yapılı algoritmaların yapay sinir ağlarının eğitilmesinde kullanımı bir önceki bölümde değinildiği üzere genellikle genetik algoritma ve parçacık sürü optimizasyon algoritması gibi küresel araştırma yetenekleri yüksek olan bir algoritma ile bölgesel araştırma yetenekleri kuvvetli olan BP, LM benzeri türev bilgisi kullanan algoritmalar veya matematiksel metodların arka arkaya çalıştırılmasıyla oluşturulan modeller ile araştırılmıştır. Tezin bu bölümünde, MLP ağlarının eğitilmesinde ABC algoritması ile elde edilen sonuçların daha da iyileştirilebilmesi amacıyla, ABC ile bölgesel arama başarısı yüksek ve hızlı olan LM algoritmasının melezleştirilmesi

sonucu oluşturulan model incelenmiştir. Bu çalışma da önerilen hibrid modelin blok diagramı Şekil 5.3'de verilmiştir. Bu ABC-LM modeli, XOR, 3-Bit parite ve Encoder-Decoder test problemlerinde ve daha önceki bölümlerde kullanılan 9 tane test sınıflandırma problemlerinde ABC algoritması ile eğitimin tamamlanmasının ardından elde edilen çözümün en iyi ağırlık setinin LM algoritmasına ön ağırlık seti olarak verilmesi ile test edilmiştir. ABC algoritmasının eğitimi tamamlamasından sonra LM algoritması 100 epok daha çalıştırılmış ve 30'ar koşma sonucunda elde edilen sonuçlar XOR, 3-Bit parite ve Encoder-Decoder problemleri için Tablo 5.12'de sunulmuştur. Sınıflandırma test problemlerinin ABC-LM yapısı ile eğitilmesi sonucu elde edilen ortalama eğitim hatası (SEP) ve sapma değerleri Tablo 5.13'de, test verisinin sınıflandırılması sonucu oluşan ortalama test sınıflandırma hata yüzde (CEP) ve standart sapma değerleri Tablo 5.14'de sunulmuştur. ABC-LM algoritmasının her problem için ortalama yakınsama grafikleri Şekil 5.4'de verilmiştir. ABC-LM modelinin performansının incelenmesi amacıyla yapılan bu çalışmalarda ayarlamalar, MLP ağlarının eğitilmesi bölümlerinde (Bölüm 4.2.4 ve Bölüm 5.1.2) ABC ve LM algoritmaları için yapılan ayarlamalar ile aynı olacak şekilde yapılmıştır.

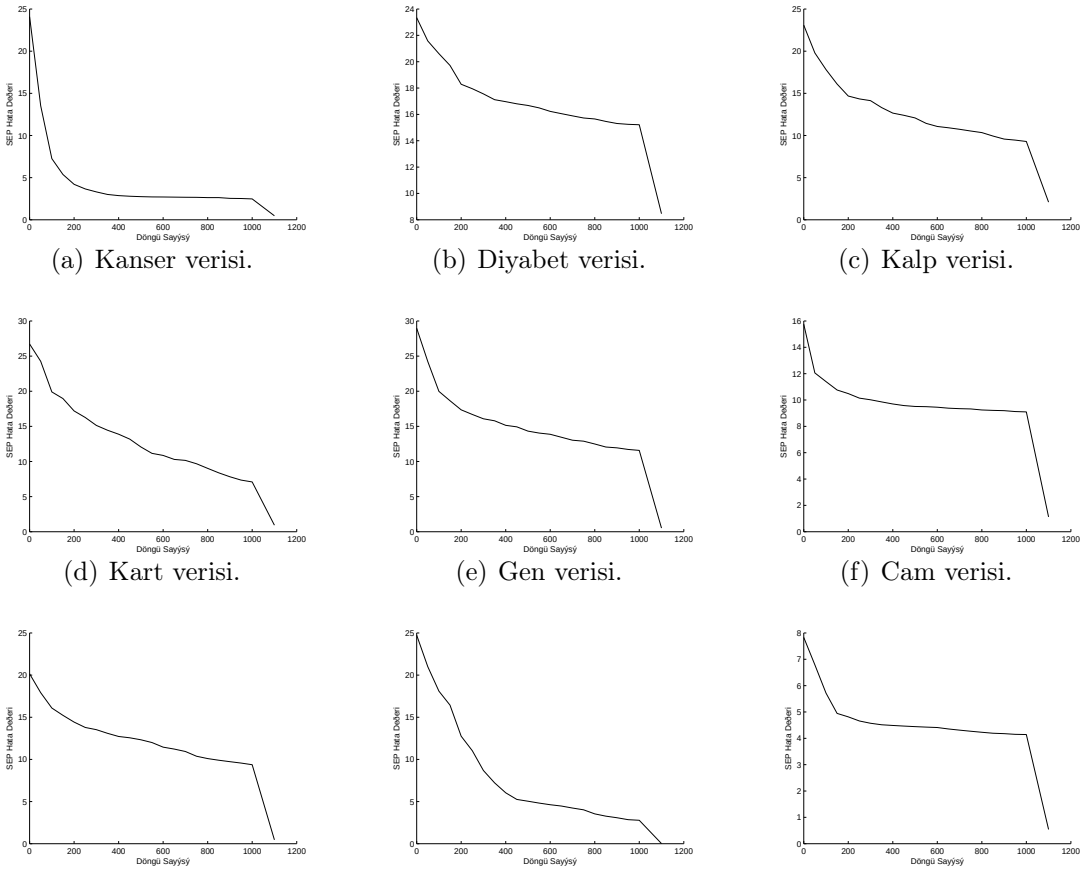


Şekil 5.3. ABC-LM Hibrid algoritmasının akış diyagramı.

Tablo 5.12. Hibrid ABC-LM algoritması ile MLP ağlarının eğitilmesi, ortalama MSE hata ve standart sapma değerleri.

Problem		**ABC**	**ABC-LM**	**LM**
XOR6	**MSE**	0.007051	**0.000752**	0.110700
	sd	0.00223	0.000980	0.063700
XOR9	**MSE**	0.006956	**2.1246E-09**	0.049100
	sd	0.002402	1.9579E-10	0.064600
XOR13	**MSE**	0.006079	**2.6111E-09**	0.007800
	sd	0.003182	1.2586E-09	0.022300
3-Bit Par.	**MSE**	0.006679	**6.3156E-07**	0.020900
	sd	0.002820	3.3189E-06	0.043000
Enc. Dec.	**MSE**	0.008191	**1.3007E-06**	0.024300
	sd	0.001864	8.8443E-07	0.042400

XOR, 3-Bit parite ve Encoder-Decoder test problemleri için yapılan eğitimlerin sonuçları incelendiğinde karma ABC-LM modeli büyük bir kazanç sağlamaktadır. Problemlerden en zoru olan 2-2-1 öndeğersiz (XOR6) probleminde LM'nin ortalama MSE değeri 0.11, ABC'nin ortalama MSE değeri 0.007 civarlarında iken ABC-LM ortalama MSE değerini 10 kat daha düşürerek 0.0007 değerlerine çekmiştir. Diğer XOR9, XOR13, 3-Bit parite ve Encoder-Decoder problemlerinde LM algoritmasının başarısı yüzde 2-6, ABC algoritmasının başarısı binde 6-8 aralığında iken ABC-LM hibrid algoritmasının ortalama MSE değerleri XOR9 ve XOR13 problemlerinde E-9'larda, 3-Bit parite probleminde E-7'lerde ve Encoder-Decoder probleminde E-6'larda kaydedilmiştir. Hata değerlerinde elde edilen kazanç benzeri hibrid yapının kararlığını gösteren standart sapma değerlerinde de kat ve kat düşmeler not edilmiştir. Sınıflandırma test problemlerinin çözümünde de ABC-LM hibrid modeli her iki algoritmanın tek başlarına kullanımlarından daha başarılı olmuştur. ABC algoritmasından daha başarılı sonuçlar vermesi LM algoritmasının 100 epokta olsa ABC algoritmasının bulduğu ağırlık setleri üzerinden eğitime devam ediyor olmasından, LM algoritmasının tek başına kullanılmasına göre daha iyi sonuçlar vermesi ise başlangıç değer bölgelerinin ABC algoritması ile ayarlanmış olmasından kaynaklanmaktadır. Tablo 5.13 ve Şekil 5.4 incelendiğinde eğitim standard hata değerinin her problem için hibrid yapı ile birlikte oldukça düştüğü



(a) Kanser verisi. (b) Diyabet verisi. (c) Kalp verisi.

(d) Kart verisi. (e) Gen verisi. (f) Cam verisi.

(g) At verisi. (h) Soya verisi. (i) Tiroit verisi.

Şekil 5.4. MLP ağlarının eğitilmesinde ABC-LM Hibrid algoritmasının ortalama eğitim hata yakınsama grafikleri.

ve eğitim aşamasında yakınsamanın arttığı rahatlıkla gözlemlenebilir. Tablo 5.14'den test verilerinin sınıflandırma hatası incelendiğinde eğitimde elde edilen başarının problemlerin çoğunda teste de yansıdığı açıkça görülmektedir. Kanser probleminde ABC-LM ile ABC'nin test CEP değerlerinin aynı olduğu, Gen, Cam, Soya ve Tiroit problemlerinin çözümünde ABC-LM modelinin test CEP değerlerinin ABC algoritmasının test CEP değerlerinden daha düşük olduğu kaydedilmiştir. Bunlarla birlikte, oluşturulan ABC-LM hibrid modeliyle elde edilen sonuçların LM algoritmasının kendi başına çalıştırılmasıyla elde edilen sonuçlarla kıyaslandığında hem eğitim hem de test hatasının bütün problemler için çok daha düştüğü gözlemlenmiştir. Ayrıca, ABC-LM modelinin sonuçları LM'nin sonuçlarına göre çok daha kararlı olup her problemde ABC-LM'nin standard sapma değerleri LM'nin standard sapma değerlerinden daha düşüktür. Yapılan çalışmalar sonucunda MLP

Tablo 5.13. Hibrid ABC-LM algoritması ile MLP ağlarının eğitilmesi, eğitim verisi ortalama SEP hata ve standart sapma değerleri.

No	Problem		ABC	ABC-LM	LM
1	**Kanser**	**SEP**	2.47	**0.47**	3.93
		sd	0.01	0.20	9.70
2	**Diyabet**	**SEP**	15.21	**8.46**	11.87
		sd	0.07	0.48	6.42
3	**Kalp**	**SEP**	9.28	**2.09**	6.18
		sd	0.17	0.54	10.55
4	**Kart**	**SEP**	7.08	**0.93**	5.24
		sd	1.07	0.32	10.02
5	**Gen**	**SEP**	11.57	**0.51**	0.73
		sd	0.52	0.21	0.34
6	**Cam**	**SEP**	9.09	**1.12**	4.04
		sd	0.07	0.26	3.69
7	**At**	**SEP**	9.36	**0.45**	2.48
		sd	0.32	0.18	5.22
8	**Soya**	**SEP**	2.77	**0.01**	0.73
		sd	0.13	0.01	0.64
9	**Tiroit**	**SEP**	4.14	**0.54**	4.02
		sd	0.02	0.39	1.48

ağlarının eğitiminde ağırlıkların belirlenmesinde hibrid ABC-LM algoritmasının kullanımının yakınsama hassasiyetini artırdığı ve test hata değerlerini düşürdüğü sonucuna varılmıştır.

Tablo 5.14. Hibrid ABC-LM algoritması ile MLP ağlarının eğitilmesi, test verisi ortalama sınıflandırma hata ve standart sapma değerleri.

No	**Problem**		**ABC**	**ABC-LM**	**LM**
1	**Kanser**	**CEP**	**1.14**	**1.14**	8.18
		sd	0.00	0.00	10.29
2	**Diyabet**	**CEP**	**25.22**	27.97	29.80
		sd	0.97	2.25	3.77
3	**Kalp**	**CEP**	**19.48**	22.91	24.96
		sd	1.41	2.31	7.40
4	**Kart**	**CEP**	**13.53**	20.29	21.59
		sd	1.17	3.24	7.98
5	**Gen**	**CEP**	29.50	**13.89**	14.35
		sd	1.88	1.07	2.48
6	**Cam**	**CEP**	45.62	**35.75**	42.99
		sd	3.11	5.54	11.55
7	**At**	**CEP**	**28.63**	33.68	33.79
		sd	2.61	3.21	4.50
8	**Soya**	**CEP**	38.63	**13.74**	25.51
		sd	3.18	2.19	9.89
9	**Tiroit**	**CEP**	6.95	**1.93**	7.22
		sd	0.01	0.23	0.07

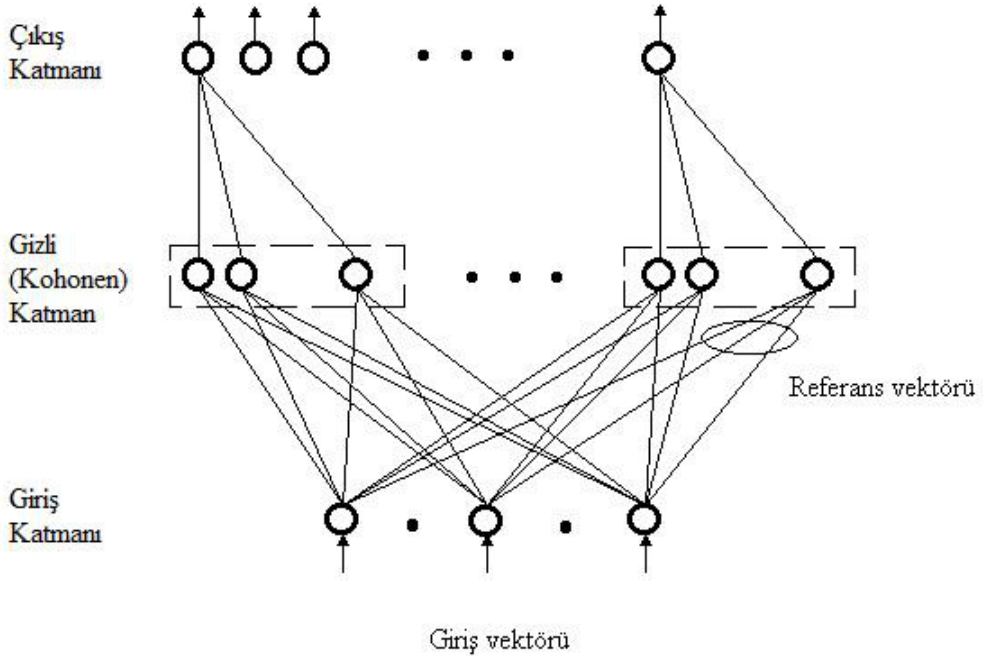
6. BÖLÜM

VEKTÖR KUANTALAMALI ÖĞRENME (LVQ) ve RADYAL TEMELLİ FONKSİYON (RBF) AĞLARININ YAPAY ARI KOLONİSİ ALGORİTMASI KULLANILARAK EĞİTİLMESİ

6.1. Vektör Kuantalamalı Öğrenme Ağları

Vektör Kuantalamalı Öğrenme (LVQ) ağları Kendiliğinden Organize olan Harita (SOM) ağlarının bir çeşidi olarak nitelendirilebilirler [199]. Sınıflandırma problemleri için geliştirilmiş olan LVQ modelinin temel felsefesi, n boyutlu bir vektörü bir vektörler setine karşı düşürmektedir. Bu model, "kazanan hepsini alır" (WTA) mantığına dayanılarak geliştirilmiştir. Kohonen tarafından geliştirilen LVQ [200], adından da anlaşılabileceği gibi vektörel olarak aldığı girişleri elde bulunan hazır vektör seti ile ifade etmeye çalışır. Bu hazır vektör setine "referans vektör" denilmektedir. Ağa gösterilen giriş setleri "en yakın komşu" kuralına tabi tutularak referans vektörlerinin gösterdiği sınıflardan en uygun olanına atanmaya çalışılır.

LVQ'nun yapısı genel hatlarıyla giriş, Kohonen ve çıkış katmanlarından oluşmaktadır, Şekil 6.1 örnek bir LVQ ağını göstermektedir [201]. LVQ ağı, giriş ile Kohonen olarak adlandırılan saklı katman arasında tamamen, gizli ve çıkış tabakası arasında da kısmen bağlıdır. Her çıkış işlemci elemanı farklı bir gizli işlemci eleman kümesine bağlıdır. Gizli ve çıkış işlemci elemanları arasındaki ağırlıklar 1'e sabitlenmiştir. Giriş-gizli işlemci eleman bağlantılarının ağırlıkları referans vektörlerinin elemanlarını oluşturur. Giriş katmanının görevi, giriş vektörünü ağa taşımaktır. Kohonen katmanının görevi ise gelen vektörü, işlemci elemanlarında, komşuluk durumlarına göre yarışa tabi tutmaktır. Her çıkış işlemci elemanı farklı bir gizli işlemci eleman kümesine bağlıdır ve belirtildiği gibi gizli ve çıkış işlemci elemanları arasındaki ağırlıklar 1'e eşittir. Giriş katmanı ile



Şekil 6.1. Vektör kuantalamalı öğrenme (LVQ) ağlarının yapısı.

Kohonen katmanı arasındaki bağlantıyı sağlayan ağırlıklandırılmış bağlantılar yani referans vektörleri, Kohonen katmanındaki proses elemanlarını girdi katmanındaki proses elemanlarına bağlayan ağırlık değerleridir [202]. Giriş-gizli işlemci eleman bağlantılarının ağırlıkları referans vektörlerinin elemanlarını oluşturur (her gizli işlemci elemana bir referans vektör atanmıştır). Ağın öğretilmesi esnasında bunlar yeniden değerler alırlar. Öğrenme esnasında sadece referans vektörlerin ağırlık değerleri değiştirilir ki LVQ ağının eğitilmesindeki amaç, her jenerasyonda girdi vektörüne en yakın referans vektörünü bulmaktır [203]. Bu sayede giriş vektörünün hangi alt sınıfa ait olduğu belirlenir. Bu katmanın, çıkış katmanına gönderdiği bilgi 1 veya 0'dır. Bu şekilde alınan bilgi ile giriş vektörünün sınıfı, çıkış katmanında belirlenmiş olur. Çıkış katmanında bulunan işlemci eleman sayısı en fazla Kohonen katmanındaki işlemci eleman sayısı kadardır. Bu katmandaki işlemci elemanlarından sadece biri 1 çıkışını verirken, diğerleri 0 çıkışı üretirler.

Kohonen katmanında bulunan işlemci elemanları bir kaç grup halinde bir araya gelerek çıkış katmanında bulunan işlemci elemanlarını işaret ederler. Bu işlemin amacı sınıfların doğrudan kazanan işlemci elemanı ile değil bir araya getirilmiş birbirine benzeyen vektörler tarafından belirlenmesidir. Bu nedenle Kohonen

katmanında belirlenen kazanan işlemci elemanı, giriş vektörünün bulunacağı sınıfı değil sadece bulunacağı alt sınıfı belirler. Alt sınıflar biraraya gelerek sınıfları oluştururlar ki bu alt sınıfları gösteren vektörün içerisinde sadece bir tane 1 bulunurken diğer elemanları sıfır olur. Hangi işlemci elemanın çıktısı 1 ise giriş vektörü o işlemci elemanın işaret ettiği sınıfa aittir şeklinde yorumlanır [204]. Özetle, Kohonen katmanında yarışı kazanan işlemci elemanı, çıkış katmanındaki hangi işlemci elemanını (sınıfı) işaret ederse giriş vektörü o sınıfa atanmış olur.

6.1.1. LVQ Öğrenme Algoritması

LVQ ağının eğitimi, kazanan işlemci elemanının referans vektör ağırlıklarında yapılan değişliklerle yapılır. Diğer referans vektör ağırlıkları değiştirilmezler. Kazanan işlemci elemanı için sınıflandırmanın başarılı olması veya başarılı olamaması şeklinde iki durum söz konusudur. Sınıflandırmanın başarılı olması halinde çıkışta gösterilen sınıf, giriş vektörünün olması gereken sınıftır ve kazanan işlemci elemanının referans vektörü, giriş vektörüne daha da yakınlaştırılmaya çalışılır. Bu sayede, aynı giriş vektörü ile yapılan bir sonraki örneklemede yarışmayı yine aynı işlemci elemanının kazanması garanti altına alınmış olmaktadır. Sınıflandırmanın başarısız olması durumunda ise çıkışta gösterilen sınıf yanlış sınıftır. Bu nedenle kazanan referans vektör, giriş vektöründen uzaklaştırılmaya çalışılır. [205]'de anlatılan bu yaklaşıma dayalı olarak yazılmış ve MATLAB'de hazır bulunan newlvq fonksiyonunun adımları aşağıdaki algoritmayla özetlenebilir:

1: Referans vektörlerin ağırlıklarına W_i başlangıç değerlerini ata
2: **repeat**
3: Bir giriş vektörü X'i ağa gir
4: X ile W_i arasındaki mesafeleri hesapla
5: En yakın referans vektörüne sahip işlemci elemanına yani kazanan işlemci elemana göre sınıflandırmayı yap
6: **If** Yapılan sınıflandırma doğru ise
then kazanan referans vektörünü düzenleyerek daha da yakınlaştır
else kazanan referans vektörünü düzenleyerek giriş vektöründen uzaklaştır
7: **until** Durma kriteri

Burada, kazanan referans vektörün giriş vektörüne yakınlaştırılması ya da giriş vektöründen uzaklaştırılması öğrenme katsayısı olan λ değerinin etkisi kadar eklenmesi veya çıkarılması ile gerçekleştirilir. Yakınlaştırırken Eşitlik 6.1'deki gibi λ değerinin etkisi kadar ekleme yapılır:

$$\vec{W'} = \vec{W} + \lambda(\vec{X} - \vec{W}) \tag{6.1}$$

Uzaklaştırma işleminde ise Eşitlik 6.2'de verildiği gibi λ katsayısının etkisi kadar çıkarılır:

$$\vec{W'} = \vec{W} - \lambda(\vec{X} - \vec{W}) \tag{6.2}$$

Öğrenme katsayısı olan λ'nın zaman içerisinde sıfıra doğru azalması istenir. Böylelikle, referans vektör çok yaklaştığı veya çok uzaklaştığı zamanlarda ağırlıkların değişme miktarı oldukça azalır ve sıfıra gider. Aksi durumda, büyük değişiklikler ters yönde uzaklaşmanın veya yakınlaşmanın tekrar başlayacak olmasına sebebiyet verir.

6.1.2. LVQ Ağlarının ABC ile Eğitilmesi ve Elde Edilen Sonuçlar

LVQ ağlarının eğitilmesi işlemi kısaca ağa verilen giriş deseni ile ağırlık vektörlerinin uyarlanması şeklinde tanımlanabilir. Her bir eğitim jenerasyonunda giriş desenleri ağa alınır ve her bir nöronun bu desen ile etkileşimi bulunur. Etkileşimi bulmak için genel YSA yapılarında yer alan aktivasyon fonksiyonunun benzeri olarak genellikle her bir ağırlık vektörünün giriş desenine olan Öklit uzaklığı hesaplanır. Bu hesaplamalar sonucunda en az etkileşime sahip olan birim ilgili eğitim jenerasyonunun kazanan birimi olarak kabul edilir. Bu çalışmada, yapay arı kolonisi algoritması giriş katmanı ile Kohonen katmanı arasındaki ağırlık vektörlerinin belirlenmesinde kullanılmıştır. LVQ ağlarının sınıflandırma doğasına uygunluğu gerekliliğinden eğitimlerde amaç fonksiyonu olarak sınıflandırma hata (CEP) değerleri alınmıştır. Eğitim aşamasının yanında test aşamasında da test sınıflandırma hatası (CEP) değerleri sınıflandırma başarısı

ölçütü olarak hesaplanmıştır. CEP değerinin eğitim verileri için hesaplanmasında hatalı sınıflandırılan eğitim veri sayısı, toplam eğitim veri sayısına bölünerek hesaplanmaktadır. Bu hesaplamayla ilgili formül Eşitlik 6.3 ile tanımlanmaktadır:

$$CEP = 100 \times \frac{\text{hatalı sınıflandırılan eğitim örnek sayısı}}{\text{toplam eğitim verisi sayısı}} \tag{6.3}$$

Bu bölümde yapılan çalışmalarda, Bölüm 5.1.1'de özetlenen ve MLP ağlarının eğitiminde kullanılan test veri kümelerinin sınıflandırılmasında yapay arı kolonisi algoritması LVQ ağlarının eğitimi için önerilmiştir. LVQ ağları tasarlanırken Kohonen katmanında her problem için sınıf sayısı kadar nöron konulmuş, yani her sınıf bir nöron ile temsil edilmiştir. Kanser, Diyabet, Kalp ve Kart verileri için kullanılan LVQ ağlarının Kohonen katmanlarında 2 nöron; Gen, At ve Tiriot verileri için tasarlanan ağlarda 3 nöron; Cam verisi için 6 nöron ve Soya verisi için yapılandırılan LVQ ağının Kohonen katmanında 19 nöron ile sınıflar ifade edilmiştir. Kısaca, Kohonen katmanlarında kullanılan nöron sayıları çıkış katmanlarında sınıf sayılarına eşit olan çıkış nöronlarının sayısıyla aynıdır ve her problem için optimize edilecek parametre sayısı giriş sayısı ile çıkış sayısının çarpımı kadardır. LVQ ağları için kullanılan ağ yapıları ve problemlerin özellikleri Tablo 6.1'de verilmiştir. Eğitim aşamalarında ABC ve newlvq öğrenme algoritmaları için döngü sayıları 1000, ABC algoritmasının koloni büyüklüğü 30, limit parametre değeri 1000 ve parametre aralığı [-2,2] olarak ayarlanmıştır. Bunlarla birlikte, veriler ağlara her problem için veri kümelerinin ilk %75'i eğitim verisi olarak ve kalan %25'lik kısım ise test verisi olarak yüklenmiştir.

LVQ ağlarının eğitiminde 50 döngüde bir kaydedilen ortalama sınıflandırma hataları ile ABC algoritmasının her problem için eğitim yakınsama hızı grafikleri Şekil 6.3'de verilmiştir. ABC ve newlvq algoritmaları için her problemde elde edilen ortalama eğitim ve test sınıflandırma hata değerleri ve bunların standart sapma değerleri Tablo 6.2'de ve bu değerlere ait grafikler eğitim verisi için Şekil 6.2(a)'de ve test verisi için Şekil 6.2(b)'de sunulmuştur.

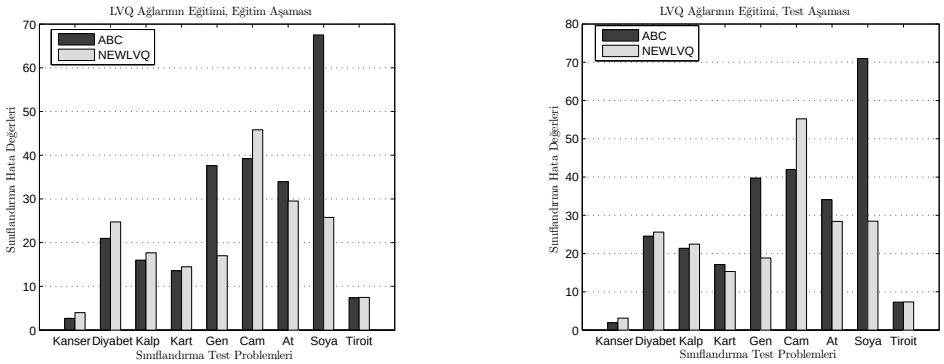
ABC algoritması LVQ ağlarının eğitilmesinde Kanser, Diyabet, Kalp, Cam ve Tiroit veri setleri üzerinde test verisi sınıflandırma başarılarına göre bu ağlar için

Tablo 6.1. Sınıflandırma problemlerinin ve kullanılan LVQ ağlarının parametreleri.

No	**Problem**	**YSA Yapısı**	**Büyüklük**	**Örnek Sayısı**
1	**Kanser**	9-2-2	18	699
2	**Diyabet**	8-2-2	16	768
3	**Kalp**	35-2-2	70	920
4	**Kart**	51-2-2	102	690
5	**Gen**	120-3-3	360	3175
6	**Cam**	9-6-6	54	214
7	**At**	58-3-3	168	364
8	**Soya**	82-19-19	1558	683
9	**Tiroit**	21-3-3	63	7200

Tablo 6.2. LVQ ağlarının eğitimi, ortalama sınıflandırma hata ve standart sapma değerleri

No	**Problem**		**Test Verisi**		**Eğitim Verisi**	
			ABC	**NEWLVQ**	**ABC**	**NEWLVQ**
1	**Kanser**	**CEP**	1.89	3.12	2.69	4.00
		sd	0.43	0.45	0.13	0.16
2	**Diyabet**	**CEP**	24.55	25.60	20.98	24.73
		sd	1.79	1.33	0.39	0.71
3	**Kalp**	**CEP**	21.38	22.46	16.01	17.68
		sd	1.27	0.81	0.78	0.60
4	**Kart**	**CEP**	17.15	15.31	13.55	14.48
		sd	1.50	1.26	1.35	1.07
5	**Gen**	**CEP**	39.75	18.83	37.61	17.01
		sd	1.89	0.98	1.40	0.83
6	**Cam**	**CEP**	41.98	55.19	39.21	45.81
		sd	7.53	2.20	3.84	2.07
7	**At**	**CEP**	34.07	28.41	33.91	29.51
		sd	4.16	1.17	0.66	0.89
8	**Soya**	**CEP**	70.96	28.46	67.51	25.78
		sd	6.73	0.34	2.60	0.34
9	**Tiroit**	**CEP**	7.31	7.35	7.37	7.46
		sd	0.06	0.06	0.06	0.01



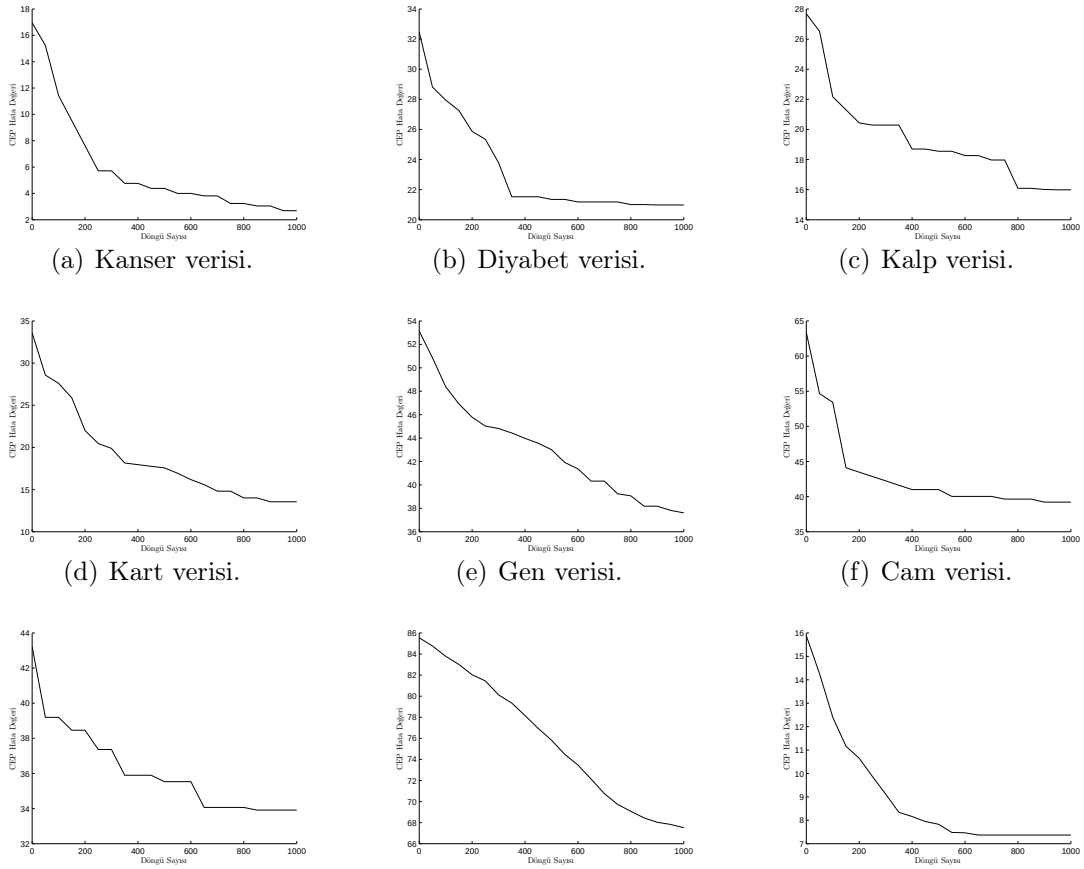
(a) Eğitim verisi. (b) Test verisi.

Şekil 6.2. LVQ ağlarının eğitimi, eğitim ve test verileri sınıflandırma hata değerleri grafikleri.

özel olarak geliştirilmiş olan newlvq eğitim fonksiyonundan daha başarılı olmuştur. Benzer şekilde eğitim verisi sınıflandırma hatası incelendiğinde bu problemlere ek olarak Kart verisiyle birlikte 6 problemde ABC algoritması daha düşük hata değerleri üretmiştir. Önceki bölümlerde yapılan çalışmalarda algoritmaların çalışma sürelerinin (hızlarının) birbirlerine oldukça yakın oldukları gözlemlendiğinden çalışma süreleri bakımından karşılaştırma yapılmamıştır. Ancak, LVQ öğrenme algoritmasının (newlvq) ABC algoritmasına oranla bütün problemlerde çok daha yavaş çalıştığı gözlemlenmiştir. Örnek verilecek olursa ABC algoritması kanser verisi üzerinde 1000 döngülük eğitim sürecini 5 dakika civarında tamamlarken newlvq fonksiyonunun aynı veri için 1000 jenerasyonlu eğitimi yaklaşık 15 dakika sürmektedir. Uygulamalar 6 GB Hafıza, 2.66 Intel Core 4 çekirdekli bilgisayarlarda MATLAB ortamlarında yapılan simülasyonlarda not edilmiştir. Sonuç olarak ABC algoritması LVQ ağlarının eğitiminde hızlı çalışan ve sınıflandırma yeteneği oldukça yüksek bir performans sergilemiştir.

6.2. Radyal Temelli Fonksiyon (RBF) Ağları

Radyal Temelli Fonksiyonlar (RBF) gizli katmanında yayılımcı bir aktivasyon fonksiyonuna sahip olduğundan yayılım temelli yapay sinir ağları olarakta adlandırılırlar. Radyal temelli fonksiyonlar, ilk olarak çok değişkenli gerçek bir interpolasyon probleminin çözümünde Powell tarafından 1985 yılında kullanılmıştır



(a) Kanser verisi. (b) Diyabet verisi. (c) Kalp verisi.

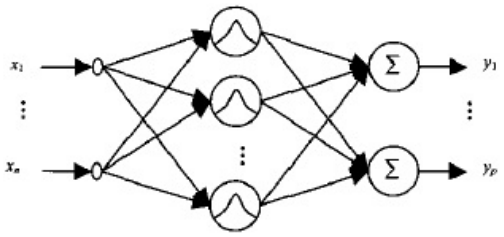
(d) Kart verisi. (e) Gen verisi. (f) Cam verisi.

(g) At verisi. (h) Soya verisi. (i) Tiroit verisi.

Şekil 6.3. LVQ ağlarının eğitilmesinde ABC algoritmasının ortalama eğitim hata yakınsama grafikleri.

[206]. 1988 yılında bu fonkfisyonu Broomhead ve Lowe sinir ağları dizaynı konusunda kullanmışlardır [207]. RBF ağları sınıflandırma amaçlı yaygın olarak kullanılmasının yanında karmaşık zaman serilerinin modellenmesi, sistem kimliklendirme, kontrol mühendisliği, elektronik aygıtların modellenmesi, ses tanıma, görüntü işleme, 3 boyutlu nesne modelleme, hareket algılama ve nesnelere hareket sağlama gibi uygulamalarda son yıllarda tercih edilmektedirler.

Şekil 6.4'de verildiği gibi RBF ağı bir giriş, bir gizli ve bir çıkış katmanından oluşur. Giriş katmanı, kaynak verilerinin ağa verildiği katmandır. İkinci katman olan gizli katman, ağın yapısına karar verecek lineer olmayan dönüşümün yani aktivasyon fonksiyonunun bulunduğu katmandır. Gizli katmanda belirlenen sonuç RBF ağı boyunca aktifleştirilmiş olan ve genellikle tercih edilen gauss fonksiyonudur, ancak bunun yerine başka bir fonksiyon da tercih edilebilir [208]. Çıkış katmanı ise ağın



Şekil 6.4. Radyal temelli fonksiyon (RBF) ağlarının yapısı.

giriş katmanının aktivasyon fonksiyonuna verdiği cevaba göre oluşan sonucu veren katmandır.

RBF ağ modelinde giriş katmanından gizli katmana kadar olan kısım denetlenemez ve gizli katmandan çıkış katmanına kadarki kısım ise denetlenebilir yapıdadır. Gizli katmandaki işlemci elemanlar girişlerin ağırlıklandırılmış şeklini kullanmamaktadır ve gizli katmandaki işlemci elemanların çıkışları YSA girişleri ile temel fonksiyonun merkezi arasındaki uzaklığa göre belirlenmektedirler. Radyal temelli fonksiyon ağı tasarımı çok boyutlu uzayda eğri uydurma yaklaşımıdır ve bu nedenle RBF'nin eğitimi, çok boyutlu uzayda eğitim verilerine en uygun bir yüzeyi bulma problemine dönüşebilir [209]. Radyal temelli fonksiyonlar, sayısal analizde çok değişkenli interpolasyon problemlerinin çözümünde kullanılmış ve YSA'nın gelişmesi ile birlikte bu fonksiyonlardan YSA tasarımında da yararlanılması fikriyle RBF ağları tasarlanmıştır. RBF ağları ileri beslemeli çok katmanlı YSA yapılarına benzerler ancak giriş katmanından ara katmana dönüşüm, radyal temelli aktivasyon fonksiyonları ile doğrusal olmayan sabit bir dönüşümdür. Ara katmandan çıkış katmanına ise uyarlamalı ve doğrusal bir dönüşüm gerçekleştirilir [210].

RBF'in genel matematiksel ifadesi Eşitlik 6.4'de verilen şekilde ifade edilir [211]:

$$s_j(x) = \sum_{i=1}^{N} w_{ij}\varphi_i(x) + b_j \tag{6.4}$$

Burada, w_{ij} gizli katman ile j. çıkış nöronu arası dönüşümdeki ağırlığı göstermekte, b_j muhtemel bir öndeğer, $\varphi(x)$ ise aktivasyon fonksiyonudur. Yukarıda belirtildiği

gibi RBF'de ara katman aktivasyon fonksiyonu genellikle standart oklit uzaklıklarını üstel fonksiyondan geçiren Gauss fonksiyonudur ve x giris vektörü, c_i merkez vektörü olduğu durumda Eşitlik 6.5'deki gibi tanımlanır:

$$\varphi_i(x) = \exp\left[-\frac{|x - c_i|^2}{2\sigma_i^2}\right] \tag{6.5}$$

6.2.1. RBF Öğrenme Algoritması

RBF yapay sinir ağılarını eğitmek amacıyla MATLAB'da yer alan newrb fonksiyonu RBF ağlarının tasarlanması amacıyla kullanılmaktadır. Newrb fonksiyonunun öğrenme yöntemi, RBF yapay sinir ağının gizli katmanında belirlenen en az hata hedefine ulaşıncaya kadar nöron eklemektir. Newrb fonksiyonunun kullanımında belirtilen parametreler: P, T, GOAL, SPREAD, MN ve DF değerleridir; bu parametrelerin açıklamaları ve ön değerleri aşağıda verilmiştir.

P: Q giriş vektörünün R x Q matrisi

T: Q hedef sınıf vektörünün SxQ matrisi

GOAL: hedeflenen minimum hata miktarı, temeli 0.0

SPREAD: RBF' nin yayılması, temeli 1.0

MN: Maksimum nöron sayısı, temeli Q

DF: Aralara eklenecek nöronların sayısı, temeli 25

Newrb fonksiyonu giriş katmanından sonra iki katmanlı bir ağ oluşturur: ilk katman (gizli katman), RADBAS transfer fonksiyonunun yüklü olduğu nöronlardan oluşur. Bu transfer fonksiyonu giriş katmanından gelen giriş değerleriyle gizli katmanın çıkışını Gauss fonksiyonuyla hesaplar [212]. İkinci katman purelin (doğrusal çıkış) nöronlarıdır ve ara katmandan gelen değerler ağırlıklandırılarak ağın çıkış değerleri hesaplanır. Bu öğrenme yönteminin en temel halinde radyal temelli fonksiyonlar için, M merkezlerin sayısına ve d seçilen merkezlerin aralarındaki uzaklığa bağlı olarak hesaplanan standart sapma (Eşitlik 6.6) değeri alınır yani standart sapması merkezlerin dağılımına göre sabit olan Gauss fonksiyonu uygulanır.

$$\sigma = \frac{d}{\sqrt{2M}} \tag{6.6}$$

Algoritmanın adımları şu şekilde özetlenebilir:

1: Ağı simule et
2: Gizli katmana nöron ekle
3: Gizli katmanın ağırlıklarını belirle
4: Çıkış katmanı için ağırlıkları ayarla
5: **If** Hedeflenen hata değerine ya da maksimum nöron sayısına ulaşıldı mı
then 2. adıma git
else Eğitimi sonlandır

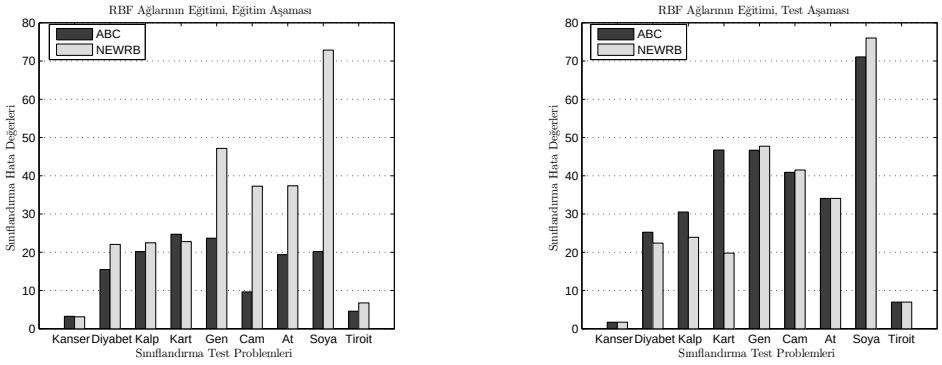
6.2.2. RBF Ağlarının ABC ile Eğitilmesi ve Elde Edilen Sonuçlar

Bu çalışmada Bölüm 5.1.1'de anlatılan test veri kümelerinin sınıflandırılması için yapay arı kolonisi kullanılarak RBF ağlarının optimizasyonu gerçekleştirilmiştir. RBF ağlarının eğitimi için önerilen newrb fonksiyonunda olduğu gibi standart sapma değeri sabit olan Gauss fonksiyonu tercih edilmiştir. ABC algoritması ağa uygulanan örnek örneklerin merkezlerini ve ağın ara katmanı ile çıkış katmanı arasındaki ağırlıkların belirlenmesinde kullanılmıştır. MLP ağlarında olduğu gibi RBF ağlarında da saklı katmanlara 6'şar nöron yerleştirilmiştir. Saklı katmanlarda yer alan nöronlara öndeğer uygulanmadığından dolayı kurulan yapılarda oluşan paremetre sayıları MLP ağlarında oluşan parametre sayılarından 6'şar adet eksiktir. RBF ağ yapıları her problem için Tablo 6.3'de verilmiştir. Bu aşamada da veri kümelerinden her problemin ilk %75'i veri eğitim seti, geriye kalan %25'lik kısmı ise test veri seti olarak kullanılmıştır. Eğitim aşamasında sınıflandırma hatası minimize edilmeye çalışılan amaç fonksiyonu olarak alınmış ve öğrenme algoritmaları ayrı ayrı 30'ar koşma sonucunda elde edilen sonuçlara göre değerlendirilmiştir. ABC algoritması için koloni büyüklüğü 30, limit değeri 1000 ve maksimum döngü sayısı 1000 olarak seçilmiştir.

Eğitim aşamasında her bir problem için elde edilen ortalama eğitim ve test sınıflandırma hataları ve bunların standart sapma değerleri Tablo 6.4'de ve bu

Tablo 6.3. Sınıflandırma problemlerinin ve kullanılan RBF ağlarının parametreleri.

No	Problem	YSA Yapısı	Büyüklük	Örnek Sayısı
1	**Kanser**	9-6-2	68	699
2	**Diyabet**	8-6-2	62	768
3	**Kalp**	35-6-2	224	920
4	**Kart**	51-6-2	320	690
5	**Gen**	120-6-3	741	3175
6	**Cam**	9-6-6	96	214
7	**At**	58-6-3	369	364
8	**Soya**	82-6-19	625	683
9	**Tiroit**	21-6-3	147	7200

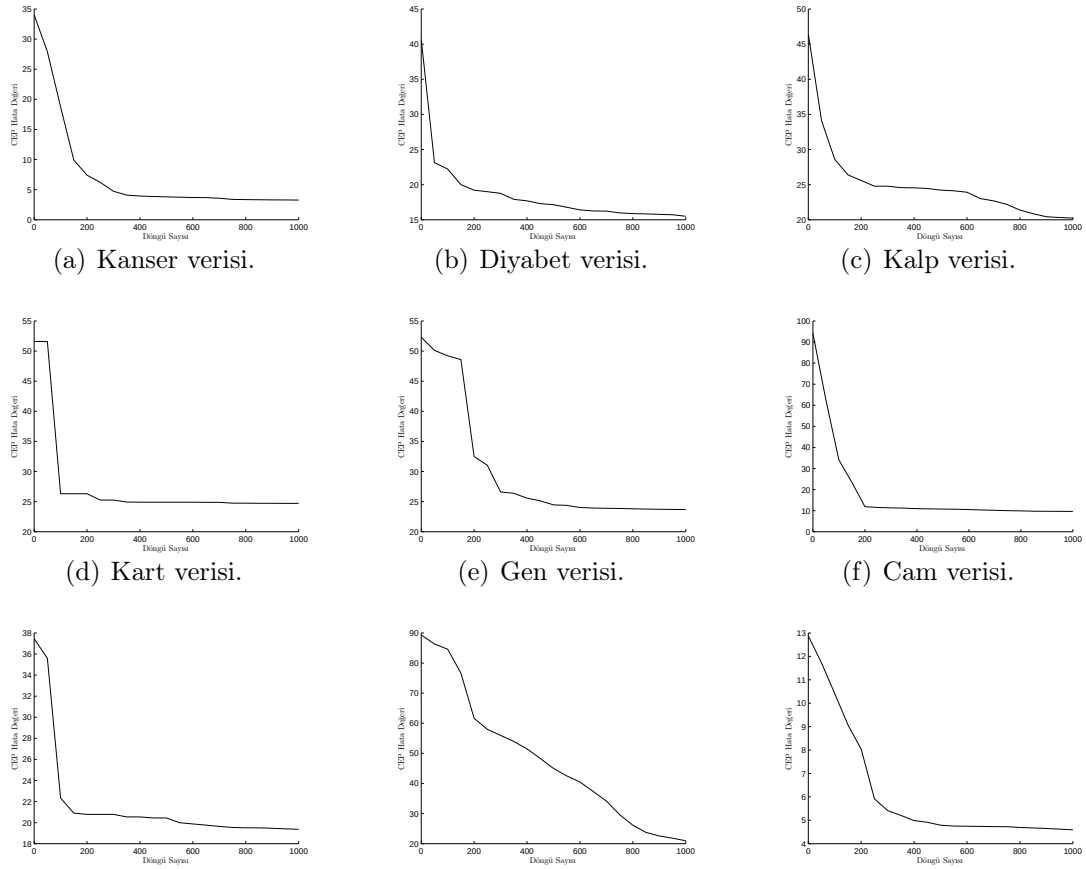


(a) Eğitim verisi. (b) Test verisi.

Şekil 6.5. RBF ağlarının eğitimi, eğitim ve test verileri sınıflandırma hata değerleri grafikleri.

değerlere ait grafikler eğitim verisi için Şekil 6.5(a)'de ve test verisi için Şekil 6.5(b)'de sunulmuştur. ABC algoritmasının her problem için eğitim aşamasındaki yakınsama grafiği her 50 döngüde bir kaydedilen ortalama sınıflandırma hatalarının çizimi ile Şekil 6.6'de verilmiştir.

Sonuçlar incelendiğinde ABC algoritmasının RBF ağları için özel olarak tasarlanmış newrb eğitim algoritması kadar başarılı olduğu söylenebilir. Ele alınan 9 problemin 3'ünde (Diyabet, Kalp ve Kart) newrb fonksiyonunun test verisi sınıflandırma başarısı ABC algoritmasının sınıflandırma başarısından daha iyidir. ABC algoritması Gen, Cam ve Soya veri setleri üzerinde newrb öğrenme stratejisine oranla daha düşük test sınıflandırma hata değerleriyle sınıflandırmada daha başarılı olmuştur. Kanser, At ve Trioit problemlerinde ise her iki öğrenme algoritması



(a) Kanser verisi. (b) Diyabet verisi. (c) Kalp verisi.

(d) Kart verisi. (e) Gen verisi. (f) Cam verisi.

(g) At verisi. (h) Soya verisi. (i) Tiroit verisi.

Şekil 6.6. RBF ağlarının eğitilmesinde ABC algoritmasının ortalama eğitim hata yakınsama grafikleri.

da aynı sınıflandırma başarısını göstermiştir. Bu problemler için sırasıyla %1.72, %34.07 ve %6.67 CEP hata değerleri elde edilmiştir. Vurgulanması gereken diğer bir nokta da ABC algoritmasının optimizasyon süresinin RBF ağlarının kendi öğrenme algoritmasına göre daha uzun olduğudur. Buna rağmen ABC algoritmasının RBF ağlarının eğitiminde sergilediği başarının yanında bu yavaşlık gözardı edilebilecek bir seviyededir.

6.3. ABC ile Eğitilen MLP, LVQ ve RBF Ağlarının Performanslarının Karşılaştırılması

Tezin bu bölümünde, ABC algoritması ile eğitilen MLP, LVQ ve RBF yapay sinir ağlarının sınıflandırma performansları birarada verilmiş ve karşılaştırılmıştır. Literatürde bu şekilde farklı YSA modellerinin benzer problemlerin sınıflandırılması

Tablo 6.4. RBF ağlarının eğitimi, ortalama sınıflandırma hata ve standart sapma değerleri.

No	**Problem**		**Test Verisi**		**Eğitim Verisi**	
			ABC	**NEWRB**	**ABC**	**NEWRB**
1	**Kanser**	**CEP**	1.72	1.72	3.26	3.13
		sd	0.00	0.00	0.06	0.00
2	**Diyabet**	**CEP**	25.24	22.40	15.49	22.04
		sd	1.80	0.00	0.28	0.00
3	**Kalp**	**CEP**	30.51	23.91	20.14	22.46
		sd	7.31	0.00	2.21	0.00
4	**Kart**	**CEP**	46.72	19.77	24.71	22.78
		sd	1.26	0.00	0.67	0.00
5	**Gen**	**CEP**	46.68	47.73	23.68	47.16
		sd	2.02	0.00	3.46	0.00
6	**Cam**	**CEP**	40.89	41.51	9.63	37.26
		sd	3.44	0.00	0.22	0.00
7	**At**	**CEP**	34.07	34.07	19.36	37.36
		sd	0.00	0.00	1.39	0.00
8	**Soya**	**CEP**	71.06	76.02	20.17	72.85
		sd	4.43	0.00	5.47	0.00
9	**Tiroit**	**CEP**	6.97	6.97	4.59	6.77
		sd	0.01	0.00	0.03	0.00

Tablo 6.5. MLP, LVQ, RBF ağlarının sınıflandırma test problemleri için ECEP ile eğitilmesi, ortalama eğitim hata (ECEP) ve standart sapma değerleri.

Problem		**MLP**	**LVQ**	**RBF**
Kanser	**ECEP**	2.84	**2.69**	3.26
	sd	0.19	0.13	0.06
Diyabet	**ECEP**	22.91	20.98	**15.49**
	sd	1.15	0.39	0.28
Kalp	**ECEP**	16.26	**16.01**	20.14
	sd	0.97	0.78	2.21

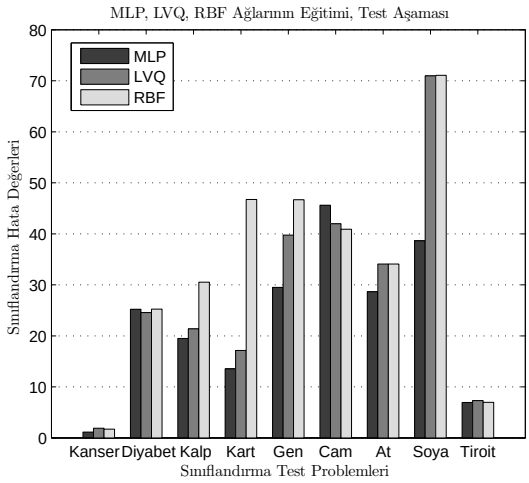
amacıyla kullanılmasına ve bunların birbirleriyle karşılaştırmasına rastlanılmamıştır. LVQ ağlarının "kazanan hepsini alır" mantığına dayalı yapılarından dolayı eğitim aşamasında ECEP hata değerinin kullanılmış olmasından ötürü bu üç model karşılaştırılırken MLP ve RBF ağlarının eğitiminde ECEP hata değerlerinin kullanıldığı kanser, diyabet ve kalp problemlerinin sınıflandırılması amaçlı yapılan eğitimlerin sonuçları eğitim verisi için Tablo 6.5'de, test verisi için Tablo 6.6'da verilmiştir. Bu sonuçlarla birlikte, tez çalışmasında ele alınan 9 sınıflandırma test problemlerinin çözülmesi amacıyla ABC algoritması ile eğitilen MLP ağlarının Bölüm 5.1.3'de, LVQ ağlarının Bölüm 6.1.2'de ve RBF ağlarının Bölüm 6.2.2'de verilen sınıflandırma başarıları (CEP değerleri) Tablo 6.7'de ve Şekil 6.7'de yeniden derlenmiştir.

Eğitim aşamasında sınıflandırma hatasının kullanılmasıyla yapılan eğitimlerin sonuçları incelendiğinde LVQ ağlarının hem eğitim hatası açısından hemde test hatası açısından diğer ağlara göre daha başarılı oldukları kaydedilmiştir. LVQ, kanser probleminde eğitimde %2.69, testte %1.89 sınıflandırma hatasıyla, diyabet probleminde eğitimde %20.98, testte %24.55 sınıflandırma hatasıyla, kalp probleminde eğitimde %16.01, testte %21.38 sınıflandırma hatasıyla genelde diğer iki modelden daha düşük hata değerleri üretmiştir. Ayrıca, diyabet ve kalp problemlerinde ABC ile eğitilen LVQ ağları ABC ile eğitilen MLP ve RBF ağlarına göre daha kararlı davranmıştır.

UCI veritabanından alınan 9 sınıflandırma test problemlerinin çözülmesi amacıyla ABC algoritması ile eğitilen MLP, LVQ ve RBF ağ modellerinin sınıflandırma

Tablo 6.6. MLP, LVQ, RBF ağlarının sınıflandırma test problemleri için ECEP ile eğitilmesi, ortalama test hata (CEP) ve standart sapma değerleri.

Problem		**MLP**	**LVQ**	**RBF**
Kanser	**CEP**	2.53	1.89	**1.72**
	sd	0.98	0.43	0.00
Diyabet	**CEP**	27.20	**24.55**	25.24
	sd	3.06	1.79	1.80
Kalp	**CEP**	23.91	**21.38**	30.51
	sd	2.42	1.27	7.31



Şekil 6.7. MLP, LVQ, RBF ağlarının eğitimi, test hata değerleri grafikleri.

başarıları Tablo 6.7 ve Şekil 6.7'den incelendiğinde MLP ağlarının LVQ ve RBF ağlarına göre çok daha başarılı olduğu; LVQ ağlarının da RBF ağlarına göre daha başarılı bir performans sergilediği gözükür. MLP ağlarının, özellikle diğer problemlere göre daha zor problemler olarak nitelendirebileceğimiz daha fazla örnek, giriş ve çıkışa sahip problemler olan kart, gen, at, soya ve tiroit problemlerinde LVQ ve RBF ağlarından daha başarılı sınıflandırma yaptığı sonucuna varılmıştır. Benzer şekilde, kanser, diyabet, kalp ve cam problemlerinde LVQ ağları ile yakın sonuçlar üretirken bu üç ağ modeli içerisinde en başarılı sınıflandırma performansını göstermiştir.

Tablo 6.7. MLP, LVQ, RBF ağlarının sınıflandırma test problemleri için eğitilmesi, ortalama test hata (CEP) ve standart sapma değerleri.

No	**Problem**		**MLP**	**LVQ**	**RBF**
1	**Kanser**	**CEP**	**1.14**	1.89	1.72
		sd	0.00	0.43	0.00
2	**Diyabet**	**CEP**	25.22	**24.55**	25.24
		sd	0.97	1.79	1.80
3	**Kalp**	**CEP**	**19.48**	21.38	30.51
		sd	1.41	1.27	7.31
4	**Kart**	**CEP**	**13.53**	17.15	46.72
		sd	1.17	1.50	1.26
5	**Gen**	**CEP**	**29.50**	39.75	46.68
		sd	1.88	1.89	2.02
6	**Cam**	**CEP**	45.62	41.98	**40.89**
		sd	3.11	7.53	3.44
7	**At**	**CEP**	**28.63**	34.07	34.07
		sd	2.61	4.16	0.00
8	**Soya**	**CEP**	**38.63**	70.96	71.06
		sd	3.18	6.73	4.43
9	**Tiroit**	**CEP**	**6.95**	7.31	6.97
		sd	0.01	0.06	0.01

7. BÖLÜM

SONUÇLAR VE ÖNERİLER

7.1. Çalışmanın Katkıları

Bu bölümde tez çalışmasının katkıları ve orjinalliği ortaya konulmuştur. Çalışmada her bölümünün sonunda, o bölümde yapılan çalışmaların ve uygulanan yöntemlerin sonuçlarıyla birlikte getirdiği yeniliklere dair değerlendirmelere yer verilmiştir. Bu bölümde bu değerlendirmeler bir arada ele alınmıştır.

Yapay sinir ağları fonksiyon tahmini, regresyon, örüntü tanıma, sınıflandırma, işlem kontrolu, robot teknolojisi vb. alanlarda başarıyla uygulanırlar. Birçok disiplinde sınıflandırma ve/veya tahmin amaçlı YSA uygulamalarına rastlanabilir. Bu tez çalışmasında sınıflandırma ve tahmin amaçlı yaygın kullanılan YSA modellerinin eğitilmesinde yapay arı kolonisi algoritmasının kullanılabilirliği araştırılmıştır.

Son zamanlarda zor problemlerin çözümünde çok yaygın olarak kullanılan yapay sinir ağlarının eğitimi doğrusal olmayan önemli bir optimizasyon problemidir. Optimizasyon, belli kısıtlar ya da sınırlar dahilinde olan parametre setlerinin değerlerinin belirlenmesi şeklinde tarif edilebilir. Öğrenme yeteneğine sahip olan yapay sinir ağlarının problem çözmedeki başarısı büyük ölçüde ağın doğru bir şekilde eğitimine ve eğitiminde kullanılan öğrenme algoritmasının performansına bağlıdır. Problemlerin boyutlarının çok büyük olması ve yapılarının lineer olmaması gibi özelliklerinden dolayı YSA'ların eğitiminde türeve dayalı klasik öğrenme algoritmaları yetersiz kalmış ve bu tür problemlerin çözümü için birçok sezgisel öğrenme algoritması önerilmiştir. Örneğin, Genetik Algoritma, Isıl İşlem, Tabu Araştırma, Evrimsel Stratejiler [53, 213–215] bunlardan en yaygın kullanılanlarındandır. Bu algoritmalarla birlikte doğada bir çok olay veya davranış

modellenmeye başlanmış ve bu modellere dayalı algoritmalar geliştirilmiştir. Son yıllarda karıncaların yiyecek kaynakları ile yuvaları arasındaki yol uzunluğunu en aza indirgeme davranışlarını modelleyen Karınca Koloni Optimizasyon [61] algoritmasında olduğu gibi doğadaki sürülerin davranışlarını temel alan sürü zekasına dayalı çeşitli optimizasyon yaklaşımları önerilmiştir. Sürü zekasına dayalı önerilen optimizasyon yaklaşımlardan birisi de bal arılarının yiyecek arama davranışlarını örnek alan yapay arı kolonisi algoritmasıdır [12].

Bu tez çalışmasında, ABC algoritması çok katmanlı ileri beslemeli, vektör kuantalamalı öğrenme ve radyal temelli fonksiyon yapay sinir ağlarının sınıflandırma ve tahmin test problemlerinin çözümüne uygulanması esnasında ağların eğitilmesi amacıyla kullanılmış ve sergilediği performans analiz edilmiştir.

Çok katmanlı ileri beslemeli algılayıcı ağları, XOR, 3-Bit Parite ve Encoder-Decoder test problemleri ve UCI makine öğrenme deposundan alınan 9 farklı sınıflandırma test probleminin çözülmesi amacıyla tasarlanmış ve eğitiminde de ABC algoritmasının performansı türeve dayalı algoritmalardan eğim düşme ve Levenberg-Marquardt geri yayılım algoritmalarının, evrimsel algoritmalardan Genetik Algoritma, Diferansiyel Gelişim ve Parçacık Sürü Optimizasyon algoritmalarının performansları ile karşılaştırılmıştır. Elde edilen sonuçlarda ABC algoritmasının MLP yapay sinir ağlarının sınıflandırma amaçlı eğitilmesinde üstün bir performans sergilediği sonucuna varılmıştır.

Zaman serilerinin tahmininde yaygınca kullanılan Box-Jenkins ve Mackey-Glass kaotik zaman serileri tahmin problemleri için toplama birimli ve çarpım birimli çok katmanlı ileri beselemeli algılayıcı yapay sinir ağ mimarileri ele alınmıştır. ABC, PSO ve DE algoritmaları kullanılarak saklı katmanlarında farklı sayılarda nöronlar bulunduran çeşitli ağ yapılarının eğitimi gerçekleştirilmiştir. Elde edilen sonuçlar incelendiğinde, ABC algoritmasının tahmin amaçlı yapay sinir ağlarının eğitiminde oldukça başarılı ve kararlı bir performans sergilediği sonucuna varılmıştır.

MLP ağlarının eğitilmesinde ABC algoritması ile LM algoritması birleştirilerek ABC algoritmasına dayalı ilk karma yapı olan ABC-LM hibrid modeli önerilmiştir. ABC algoritmasının küresel araştırma yeteneğinden ve LM algoritmasının bölgesel

araştırma yeteneğinden faydalanan ABC-LM yaklaşımının performansı test problemlerinin ve sınıflandırma test problemlerinin çözümlerinde ABC ve LM algoritmalarının performanslarıyla karşılaştırılmıştır. ABC-LM modelinin MLP ağlarının eğitiminde ele alınan problemlerin çoğunda ABC ve LM algoritmalarının kendi başlarına çalıştırılmalarına göre daha başarılı ve kararlı bir performans sergilediği ortaya konulmuştur.

Vektör Kuantalamalı Öğrenme ve Radyal Temelli Fonksiyon ağları özellikle sınıflandırma amaçlı son yıllarda yaygınca kullanılan ve üzerlerinde araştırmalar yapılan yapay sinir ağları modellerindendir. Bu tez çalışmasında kulllanılan sınıflandırma test problemlerinin sınıflandırılmasında ABC algoritmasının başarısı yapılarına özel olarak geliştirilen eğitim algoritmalarının başarılarıyla karşılaştırılmış ve elde edilen sonuçlar tartışılmıştır. ABC algoritmasının ele alınan çok sayıda farklı problemler için çeşitli yapay sinir ağları modellerinin eğitilmesinde literatürde iyi bilinen algoritmalar ve teknikler kadar başarılı olduğu ortaya konulmuştur.

7.2. Gelecekte Yapılacak Çalışmalar

Bu tez çalışmasında zor bir optimizasyon problemi olan yapay sinir ağlarının eğitimi ele alınmış ve yapay arı kolonisi algoritmasının bu amaçla kullanımı sunularak sergilediği performans araştırılmıştır. ABC algoritmasının genel yapısına sadık kalınarak standard ABC algoritmasının performansı ele alınan problemler için incelenmiştir. Algoritmanın mekanizmalarında problemlerin tiplerine göre değişiklikler veya yenilikler yapılarak performansındaki değişiklikler gözlemlenebilir. Özellikle algoritmanın adımlarına başka algoritmaların bazı mekanizmaları entegre edilerek tam bir karma yapı kurulabilir ve bu yapılar yapay sinir ağlarının eğitiminde kullanılabilir.

Yapay sinir ağlarının öğrenebilme yetenekleri temelde iki ana unsura bağlıdır; kullanılan YSA mimarisi (yapısal tasarım) ve kullanılan eğitim stratejisi (öğrenme algoritması). YSA mimarisinin, katman sayılarının, katmanlarda ki nöron sayılarının, nöronlar arası bağlantı şekillerinin ve aktivasyon fonksiyonlarının belirlenmesi yapısal tasarım aşamasında karar verilmesi gereken konulardır. ABC

algoritmasının yapısal tasarım optimizasyonunda performansı araştırılabilir.

Kurulan ağların eğitilmesinde aşırı öğrenmeyi (ezberleme) önleyici bir çalışma yapılmamıştır. Giles, [216]'de aşırı öğrenme davranışının en fazla eğitim algoritmasına bağımlı olduğunu ortaya koymuştur. Bu sebeple, ABC algoritmasının ezberleme eğiliminin diğer kullanılan eğitim algoritmalarının ezberleme eğilimlerine göre karşılaştırması yapılabilir.

Yapay sinir ağlarının eğitiminde probleme göre standart hata yüzdesi, ortalama karesel hata ve sınıflandırma hata yüzdesi amaç fonksiyonları olarak kullanılmıştır. Farklı amaç fonksiyonları kullanımının eğitimler üzerindeki etkileri incelenebilir.

Geri-beslemeli YSA modellerinde en az bir nöron bulunduğu katmandan sonraki katmanlardaki nöronlar tarafından beslenir. Bu ağların eğitimi de oldukça zor bir optimizasyon problemidir ve ABC algoritmasının bu ağların eğitilmesinde ki performansı araştırılabilir.

Bu çalışmayla birlikte veri madenciliği tekniklerinden kümeleme yöntemi dikkat çekmiş ve bu tezde ele alınan problemlerin sınıflandırmasında kümeleme yöntemi de kullanılmıştır. Çeşitli kümeleme yöntemlerinde yapay arı kolonisi algoritmasının kullanılması ve analizlerin yapılması planlanan çalışmalardandır.

KAYNAKLAR

1. Haykin, S., Neural Networks a Comprehensive Foundation, Prentice Hall, New Jersey, 1999.

2. Öztemel, E., Yapay Sinir Ağları, Papatya Yayıncılık, 2003.

3. Whitley, D., et al., Genetic algorithms and neural networks: Optimizing connections and connectivity, Technical Report 89-117, Computer Science, Colorado State University, 1989.

4. Verma, B., Ghosh, R., A novel evolutionary neural learning algorithm, evolutionary computation, In Proceedings of CEC'02, 1884–1889, May 12-17 2002.

5. Bäck, T., Schwefel, H.P., An overview of evolutionary algorithms for parameter optimization, Evolutionary Computation, 1(1), 1–23, 1993.

6. Corne, D., et al., New Ideas in Optimization, McGraw–Hill, 1999.

7. Bergh, F., An analysis of particle swarm optimizers, Ph.D. thesis, University of Pretoria, November 2001.

8. Michalewicz, Z., Schoenauer, M., Evolutionary algorithms for constrained parameter optimization problems, Evolutionary Computation, 4(1), 1– 32, 1995.

9. Dayhoff, J., Neural-Network Architectures: An Introduction, New York: Van Nostrand Reinhold, 1990.

10. Parekh, R., et al., Constructive neural-network learning algorithms for multicategory real-valued pattern classification, Technical Report ISU-CS-TR97-06, Dept. Computer Sci., Iowa State Univ., 1997.

11. Parekh, R., et al., Constructive neural-network learning algorithms for pattern classification, IEEE Transactions on Neural Networks, 11(2), 436–451, 2000.

12. Karaboga, D., An idea based on honey bee swarm for numerical optimization, Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

13. Basturk, B., Karaboga, D., An artificial bee colony (abc) algorithm for numeric function optimization, In Proceedings of IEEE International Conference on Neural Networks. IEEE Swarm Intelligence Symposium 2006, Indianapolis, Indiana, USA, May 2006.

14. Karaboga, D., Basturk, B., A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm, Journal of Global Optimization, 39(3), 459–471, 2007.

15. Karaboga, D., Akay, B., A survey: Algorithms simulating bee swarm intelligence, Artificial Intelligence Review, 31(1), 68–85, 2009.

16. Wasserman, P., Schwartz, T., Neural networks. what are they and why is everybody so interested in them now, IEEE Expert, 3(1), 10–15, 1988.

17. Zhang, C., et al., Convergence of bp algorithm for product unit neural networks with exponential weights, Neurocomputing, 72(1-3), 513–520, 2004.

18. Haykin, S., Neural Networks, Prentice Hall, New Jersey, 1994.

19. Grossberg, S., Neural Networks and Natural Intelligence, Cambridge, MA, MIT Press, 1988.

20. Elmas, C., Yapay Sinir Ağları, Seçkin Yayıncılık, Ankara, 2003.

21. Sağıroğlu, S., et al., Mühendislikte Yapay Zeka Uygulamaları-I: Yapay Sinir Ağları, Ufuk Yayıncılık, 2003.

22. Pala, M., Yapay sinir ağları ile dinamik zemin yapı etkileşim analizi, Ph.D. thesis, Sakarya Üniversitesi Fen Bilimleri Enstitüsü, 2003.

23. Matlab Neural Network Toolbox.

24. Pham, D., Karaboga, D., Intelligent Optimisation Techniques, Springer-Verlag, London, 2000.

25. Fausett, L., Fundamentals of Neural Networks, Prentice-Hall, 1994.

26. Lawrence, J., Introduction to Neural Networks, California Scientific, Nevada City, CA, 1993.

27. Yıldırım, M., Yapıştırıcı ile birleştirilmiş bindirme bağlantıların modal analizi ve titreşim kontrolü, Master's thesis, Fen Bilimleri Enstitüsü, Erciyes Üniversitesi, Türkiye, 2006.

28. Şen, Z., Yapay Sinir Ağları İlkeleri, Su Vakfı Yayınları, İstanbul, 2004.

29. Limin, F., Neural Networks in Computer Intelligence, McGraw-Hill, New York, 1994.

30. Guney, K., An Introduction to Neural Networks, UCL Press, 1997.

31. Gallant, S., Neural Network Learning and Expert systems, Massachusetts Inst. of Technology, 1993.

32. Wasserman, P., Neural computing: theory and practice, Van Nostrand Reinhold Co., New York, 1989.

33. Schalkhoff, R., Pattern Recognition, statistical, structural and neural approaches, John Wiley and Sons, New York, 1992.

34. Fukunaga, F., Introduction to Statistical Pattern Recognition, Academic Press, New York, 1990.

35. Ripley, B., Pattern Recognition and Neural Networks, Cambridge University Press, 1996.

36. Lawrence, S., et al., Lessons in neural network training: Training may be harder than expected, In Proceedings of National Conference on Artificial Intelligence, AAAI-97, 540–545, Menlo Park, California, 1997.

37. Prechelt, L., Early stopping-but when? neural networks: Tricks of the trade, 1998, URL `http://www.ipd.ira.uka.de/p̃rechelt/Biblio/`.

38. Kohonen, T., Self-Organizing Maps, Springer-Verlag, Berlin, 1995.

39. Singh, S., Kumar, D., A comparison of different neural network training algorithms for hydromechanical deep drawing, International Journal of Materials and Product Technology, 21, 186–199, 2004.

40. Abraham, A., Nath, B., Optimal design of neural nets using hybrid algorithms, In proceedings of 6 th Pacific Rim International Conference on Artificial Intelligence, PRICAI-00, LNAI, volume 1886, 510–520, Springer-Verlag, 2000.

41. Rumelhart, D., et al., Learning representations by backpropagation errors, Nature, 323, 533–536, 1986.

42. Lin, C., Lee, G., Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems, Prentice-Hall International, Englewood Cliffs, NJ, 1996.

43. Yao, X., Evolving artificial neural networks, In Proceeedings of the IEEE, volume 87(9), 1423–1447, 1999.

44. Hagan, M., Menhaj, M., Training feedforward networks with the marquardt algorithm, In Proceedings of IEEE Transactions on Neural Networks, volume 6, 989, 1994.

45. Rumelhart, D., McClelland, J., Parallel Distributed Processing, MIT Press, Cambridge, 1986.

46. Levenberg, K., A method for the solution of certain nonlinear problems in least squares, Quart. Appl. Math., 2, 164–168, 1944.

47. Marquardt, D., An algorithm for least-squares estimation of nonlinear parameters, Applied Mathematics, 11, 431–441, 1963.

48. Eiben, A., Smith, J., Introduction to Evolutionary Computing, Springer, 2003.

49. Lampinen, J., Zelinka, I., New Ideas in Optimization, chapter Mechanical Engineering Design Optimization by Differential Evolution, 127–146, Mc Graw-Hill, UK, 1999.

50. Corne, D., et al., eds., New ideas in optimization, McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999, ISBN 0-07-709506-5.

51. Belew, R., et al., Evolving networks: Using the genetic algorithm with connectionist learning, Technical Report CSE-CS-90-174, University of California, San Diego, 1990.

52. Bäck, T., Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms, Oxford University Press, 1996, ISBN 0-19-509971-0.

53. Holland, J., Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.

54. Tsukimoto, H., Hatano, H., The functional localization of neural networks using genetic algorithms, Neural Networks, 16, 55–67, 2003.

55. Storn, R., Price, K., Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces, Technical report, International Computer Science Institute, Berkley, 1995.

56. Storn, R., Price, K., Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces., Journal of Global Optimization, 11, 341–359, 1997.

57. Ronkkonen, J., et al., Real-parameter optimization with differential evolution, In Proceedings of Evolutionary Computation, the 2005 IEEE Congress, volume 1, 506–513, September 2005.

58. Price, K., et al., Differential Evolution A Practical Approach to Global Optimization, Springer Natural Computing Series, 2005.

59. Kennedy, J., Eberhart, R., Particle swarm optimization, In Proceedings of IEEE International Conference on Neural Networks, 1942–1948, Piscataway, NJ, 1995.

60. Bonabeau, E., et al., Swarm intelligence: from natural to artificial systems, Oxford University Press, Inc., New York, NY, USA, 1999.

61. Dorigo, M., et al., Positive feedback as a search strategy, Technical Report 91-016, Politecnico di Milano, Italy, 1991.

62. Farmer, J., et al., The immune system, adaptation, and machine learning, Physica D, 22, 187–204, 1986.

63. Passino, K.M., Biomimicry of bacterial foraging for distributed optimization and control, IEEE. Control System Magazine, 52–67, 2002.

64. Akay, B., Nümerik optimizasyon problemlerinde yapay arı kolonisi algoritmasının performans analizi, Ph.D. thesis, Erciyes University, Fen Bilimleri Enst., November 2009.

65. Lucic, P., Teodorovic, D., Bee system: Modeling combinatorial optimization transportation engineering problems by swarm intelligence, Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis, 441–445, Sao Miguel, Azores Islands, Portugal, June 2001.

66. Teodorovic, D., Dell'Orco, M., Bee colony optimization - a cooperative learning approach to complex transportation problems, Advanced OR and AI Methods in Transportation, 51– 60, 2005.

67. Yang, X.S., Engineering optimizations via nature-inspired virtual bee algorithms, Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach, LNCS, volume 3562/2005, 317– 323, June 2005.

68. Pham, D.T., et al., The bees algorithm, Technical report, Manufacturing Engineering Centre, Cardiff University, UK, 2005.

69. Waibel, M., et al., Division of labour and colony efficiency in social insects: effects of interactions between genetic architecture, colony kin structure and rate of perturbations, In Proceedings of the Royal Society B., 273, 1815–1837, 2006.

70. Bonabeau, E., et al., Adaptive task allocation inspired by a model of division of labor in social insects, In Proceedings of BCEC97, Biocomputing and emergent computation, 36–45, World Scientific Press, 1997.

71. Azeem, M., A novel parent selection operator in ga for tuning of scaling factors of fkbc, In Proceedings of IEEE International Conference on Fuzzy Systems, 1742–1747, 2006.

72. Karaboga, D., Artificial bee colony algorithm, Scholarpedia, 5(3)(6915), 2010, URL `http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm`.

73. Karaboga, D., Basturk, B., On the performance of artificial bee colony (abc) algorithm, Applied Soft Computing, 8(1), 687–697, 2008.

74. Karaboga, D., Akay, B., A comparative study of artificial bee colony algorithm, Applied Mathematics and Computation, 214, 108–132, 2009.

75. Karaboga, D., Basturk, B., Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing, LNCS, volume 4529/2007, chapter Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems, 789–798, Springer-Verlag, 2007.

76. Karaboga, D., Akay, B., Effect of region scaling on the initialization of particle swarm optimization differential evolution and artificial bee colony algorithms on multimodal high dimensional problems, International Conference on Multivariate Statistical Modelling and High Dimensional Data Mining, Kayseri, TURKEY, June 19-23 2008.

77. Karaboga, D., Akay, B., Solving large scale numerical problems using artificial bee colony algorithm, 6th International Symposium on Intelligent and Manufacturing Systems Features, Strategies and Innovation, Sakarya, Turkiye, October 14-17 2008.

78. Akay, B., Karaboga, D., Modified artificial bee colony algorithm for real-parameter optimization, Information Sciences, In Press, 2010.

79. Karaboga, D., Akay, B., Pid controller design by using artificial bee colony, harmony search and the bees algorithms, In Proceedings of the Institution of Mechanical Engineers, Part I, In Press, Journal of Systems and Control Engineering, 2010.

80. Karaboga, D., Ozturk, C., Neural networks training by artificial bee colony algorithm on pattern classification, Neural Nerwork World, 19(3), 279–292, 2009.

81. Karaboga, D., et al., Training neural networks with abc optimization algorithm on medical pattern classification, International Conference on Multivariate Statistical Modelling and High Dimensional Data Mining, Kayseri, TURKEY, June 19-23 2008.

82. Ozturk, C., Karaboga, D., Classification by neural networks and clustering with artificial bee colony (abc) algorithm, 6th International Symposium on Intelligent

and Manufacturin Systems Features, Strategies and Innovation, Sakarya, Turkiye, October 14-17 2008.

83. Ozturk, C., Karaboga, D., Sınıflandırma problemleri için yapay sinir ağlarının yapay arı koloni algoritması (abc) ile eğitilmesi, BUMAT 2008, Eskişehir Osmangazi Üniversitesi, Eskişehir, Turkiye, October 15-17 2008.

84. Ozturk, C., Karaboga, D., Artificial bee colony algorithm trained neural networks for time-series prediction problem, In Proceedings of International Symposium on INovations in Intelligent Systems and Applications, 286–291, Trabzon, Turkiye, 2009.

85. Karaboga, D., Ozturk, C., A novel clustering approach: Artificial bee colony (abc) algorithm, Applied Soft Computing, 11(1), 652–657, 2011.

86. Karaboga, D., Ozturk, C., Fuzzy clustering with artificial bee colony algorithm, Scientific Research and Essays, In Press, 2010.

87. Kurban, T., Besdok, E., A comparison of rbf neural network training algorithms for inertial sensor based terrain classification, Sensors, 9, 6312–6329, 2009.

88. Zhang, C., et al., An artificial bee colony approach for clustering, Expert Systems with Applications, 37(7), 4761–4767, 2010.

89. Marinakis, Y., et al., A hybrid discrete artificial bee colony - grasp algorithm for clustering, In Proceedings of International Conference on Computers and Industrial Engineering, 2009.

90. Singh, A., An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, Applied Soft Computing, 9 (2), 625–631, 2009.

91. Sundar, S., Singh, A., A swarm intelligence approach to the quadratic minimum spanning tree problem, Information Sciences, 180(17), 3182–3191, 2010.

92. Fenglei, L., et al., The parameter improvement of bee colony algorithm in tsp problem, Nov. 2007.

93. Karaboga, N., A new design method based on artificial bee colony algorithm for digital iir filters, Journal of The Franklin Institute, 346 (4), 328–348, 2009.

94. Bendes, E., Ozkan, C., Direk lineer trasformasyon yönteminde yapay zeka tekniklerinin uygulanması, UZALCBS08, Kayseri, Turkiye, September 13-15 2008.

95. Rao, R.S., et al., Optimization of distribution network configuration for loss reduction using artificial bee colony algorithm, International Journal of Electrical Power and Energy Systems Engineering, 1(2), 116–122, 2008.

96. Baykasoglu, A., et al., Swarm Intelligence Focus on Ant and Particle Swarm Optimization, chapter Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem, 113– 144, I-Tech Education and Publishing, Vienna, Austria, December 2007, ISBN 978-3-902613-09-7.

97. Akay, B., Karaboga, D., Artificial bee colony algorithm for large-scale problems and engineering design optimization, Journal of Intelligent Manufacturing, In Press, 2010.

98. Kang, F., et al., Structural inverse analysis by hybrid simplex artificial bee colony algorithms, Computers and Structures, In Press, 2010.

99. Pawar, P., et al., Multi-objective optimization of electro-chemical machining process parameters using artificial bee colony (abc) algorithm, Advances in Mechanical Engineering (AME-2008), Surat, India, December 15-17 2008.

100. Pawar, P., et al., Optimization of process parameters of abrasive flow machining process using artificial bee colony algorithm, Advances in Mechanical Engineering (AME-2008), Surat, India, December 15-17 2008.

101. Pawar, P., et al., Optimization of process parameters of milling process using particle swarm optimization and artificial bee colony algorithm, Advances in Mechanical Engineering (AME-2008), Surat, India, December 15-17 2008.

102. Hemamalini, S., Simon, S.P., Economic load dispatch with valve-point effect using artificial bee colony algorithm, XXXII National Systems Conference, India, Dec. 17-19 2008.

103. Omkar, S., et al., Artificial bee colony (abc) for multi-objective design optimization of composite structures, Applied Soft Computing, In Press, 2010.

104. Pan, Q., et al., A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem, Information Sciences, In Press, 2010.

105. Kang, F., et al., Structural inverse analysis by hybrid simplex artificial bee colony algorithms, Computers and Structures, 87(13-14), 861–870, 2009.

106. Xu, C., Duan, X., Artificial bee colony (abc) optimized edge potential function (epf) approach to target recognition for low-altitude aircraft, Pattern Recognition Letters, In Press, 2009.

107. Sabat, S., et al., Artificial bee colony algorithm for small signal model parameter extraction of mesfet, Engineering Applications of Artificial Intelligence, 23(5), 689–694, 2010.

108. Xu, C., et al., Chaotic artificial bee colony approach to uninhabited combat air vehicle (ucav) path planning, Aerospace Science and Technology, In Press, 2010.

109. Alatas, B., Chaotic bee colony algorithms for global numerical optimization, Computers and Structures, 37(8), 5682–5687, 2010.

110. Hetmaniok, E., et al., Solution of the inverse heat conduction problem by using the abc algorithm, Rough Sets and Current Trends in Computing, LNCS, Springer, 2010.

111. Zhao, H., et al., A hybrid swarm intelligent method based on genetic algorithm and artificial bee colony, Advances in Swarm Intelligence, LNCS, Springer, 2010.

112. Bernardino, A., et al., Efficient load balancing for a resilient packet ring using artificial bee colony, Applications of Evolutionary Computation, LNCS, Springer, 2010.

113. Udgata, S., et al., Sensor deployment in irregular terrain using artificial bee colony algorithm, In Proceedings of World Congress on Nature and Biologically Inspired Computing (NABIC09), 1309–1314, 2009.

114. Karaboga, D., et al., Cluster based wireless sensor network routings using artificial bee colony algorithm, International Conference on Autonomous and Intelligent Systems, Povoa de Varzim, Portugal, June 21-23 2010, ISBN 978-1-4244-7106-5.

115. Ozturk, C., et al., Artificial bee colony algorithm for dynamic deployment of wireless sensor networks, In Proceedings of International Symposium on INovations in Intelligent Systems and Applications, 564–569, Kayseri, Turkiye, 2010.

116. Pulikanti, S., Singh, A., An artificial bee colony algorithm for the quadratic knapsack problem, Neural Information Processing, LNCS, Springer, 2010.

117. Akay, B., Karaboga, D., Solving integer programming problems by using artificial bee colony algorithm, AI*IA 2009: Emergent Perspectives in Artificial Intelligence, LNCS, Springer, 2009.

118. Akay, B., Karaboga, D., Parameter tuning for the artificial bee colony algorithm, Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems, LNCS, Springer, 2009.

119. Duan, H., et al., A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems, Advances in Computational Intelligence, Advances in Intelligent and Soft Computing, Springer, 2009.

120. Narasimhan, H., Parallel artificial bee colony (pabc) algorithm, In Proceedings of World Congress on Nature and Biologically Inspired Computing (NaBIC09), 306–311, 2009.

121. Bahamish, H., et al., Protein tertiary structure prediction using artificial bee colony algorithm, In Proceedings of Third Asia International Conference on Modelling and Simulation, 258–263, 2009.

122. Chidambaram, C., Lopes, H., A new approach for template matching in digital images using an artificial bee colony algorithm, In Proceedings of World Congress on Nature and Biologically Inspired Computing (NaBIC09), 146–151, 2009.

123. Abu-Mouti, F., El-Hawary, M., Modified artificial bee colony algorithm for optimal distributed generation sizing and allocation in distribution systems, In Proceedings of Electrical Power and Energy Conference (EPEC), IEEE, 435–444, Montreal, CANADA, October 22-23 2009.

124. Liu, X., Cai, Z., Artificial bee colony programming made faster, In Proceedings of Fifth International Conference on Natural Computation (ICNC09), volume 4, 154–158, 2009.

125. Bao, L., Zeng, J., Comparison and analysis of the selection mechanism in the artificial bee colony algorithm, In Proceedings of Ninth International Conference on Hybrid Intelligent Systems (HIS09), volume 1, 411–416, 2009.

126. Benala, T., et al., A novel approach to image edge enhancement using artificial bee colony optimization algorithm for hybridized smoothening filters, In Proceedings of World Congress on Nature and Biologically Inspired Computing (NaBIC09), 1071–1076, 2009.

127. Nayak, S., et al., Application of artificial bee colony to economic load dispatch problem with ramp rate limits and prohibited operating zones, In Proceedings of World Congress on Nature and Biologically Inspired Computing (NaBIC09), 1237–1242, 2009.

128. Kang, F., et al., An improved artificial bee colony algorithm, In Proceedings of 2nd International Workshop on Intelligent Systems and Applications (ISA), 791–794, 2010.

129. Pansuwan, P., et al., Identifying optimum artificial bee colony (abc) algorithm's parameters for scheduling the manufacture and assembly of complex products, In Proceedings of 2nd International Conference on Intelligent Computer and Network Technology (ICCNT), 339–343, 2010.

130. Pacurib, J., et al., Solving sudoku puzzles using improved artificial bee colony algorithm, In Proceedings of Fourth International Conference on Innovative Computing, Information and Control (ICICIC), 885–888, 2010.

131. Dahiya, S., et al., Application of artificial bee colony algorithm to software testing, In Proceedings of 21st Australian Software Engineering Conference (ASWEC), 149–154, 2010.

132. Dos Santos Coelho, L., Alotto, P., Gaussian artificial bee colony algorithm approach applied to loney's solenoid benchmark problem, In Proceedings of 14th Biennial IEEE Conference on Electromagnetic Field Computation (CEFC), 1502–1505, 2010.

133. Krishnanand, K., et al., Comparative study of five bio-inspired evolutionary optimization techniques, In Proceedings of World Congress on Nature and Biologically Inspired Computing (NaBIC09), 1231–1236, 2009.

134. Mitchell, T., Machine Learning, McGraw-Hill, Portland, Oregon, 1997.

135. Bishop, C., Neural Networks for Pattern Recognition, Oxford University Press, 1995.

136. Weigend, A., On overfitting and the efective number of hidden units, In Proceedings of the Connectionist Models Summer School, 335–342, Hillsdale, NJ, 1994.

137. LeCun, Y., et al., Optimal brain damage, Advances in Neural Information Processing Systems, 2, 598–605, 1990.

138. Engelbrect, A., A new pruning based on variance analysis of sensitivity information, IEEE Transactions on Neural Networks, 12 (6), November 2001.

139. Fahlman, S., Lebiere, C., The cascade correlation learning architecture, Advances in Neural Information Processing Systems, 2, 524–532, 1990.

140. Lippmann, R., An introduction to computing with neural nets, IEEE, Acoustic, Speech and Signal Processing, 4–22, April 1987.

141. Minor, J., Parity with two layer feedforward nets, Neural Networks, 6, 705–707, 1993.

142. Brown, D., N-bit parity networks, Neural Networks, 6, 607–608, 1993.

143. Fahlman, S., An empirical study of learning speed in back-propagation networks, Technical Report CMU-CS-88-162, Carnegie Mellon University, Pittsburgh, 1988.

144. Stork, D., Allen, J., How to solve the n-bit encoder problem with just one hidden unit, Neurocomputing, 5, 1993.

145. Vesterstrom, J., Thomsen, R., A comparative study of differential evolution particle swarm optimization and evolutionary algorithms on numerical benchmark problems,

In Proceedings of the IEEE Congress on Evolutionary Computation (CEC'2004), volume 3, 1980–1987, Piscataway, New Jersey, June 2004.

146. Plummer, E., Time series forecasting with feed-forward neural networks: Guidelines and limitations, Master's thesis, Department of Computer Science, The Graduate School of The University of Wyoming, Jul 2000.

147. Bergh, F., Engelbrecht, A., Training product unit networks using cooperative particle swarm optimisers, In Proceedings of International Joint Conference on Neural Networks (IJCNN), volume 1, 126–131, 2001.

148. Durbin, R., Rumelhart, D., Product units: A computationally powerful and biologically plausible extension to backpropagation networks, Neural Computation, 1, 133–142, 1989.

149. Hertz, J., et al., Introduction to the Theory of Neural Computition, Redwwod City, CA: Addison Wesley, 1991.

150. Engelbrecht, A., Ismail, A., Training product unit neural networks, Stability and Control: Theory and Applications, 2, 59–74, 1999.

151. Ismail, A., Training and optimization of product unit networks, Master's thesis, Deptartment of Computer Science, University of Pretoria, South Africa, 2001.

152. Ismail, A., Global optimization algorithms for training product unit neural networks., In Proceedings of International joint conference on neural networks IJCNN, Italy, 2000.

153. D.J., J., Training product unit neural networks with genetic algorithms, 26–33, 1993.

154. Leerink, L., et al., Learning with product units., Advances in Neural Information Processing Systems, 7, 537–544, 1995.

155. Elliott, P., et al., Training reformulated product units in hybrid neural networks, In Proceedings of International joint conference on neural networks IJCNN, Canada, 5051–5058, Jul 16-21 2006.

156. Zhang, J., Working group on data modeling benchmarks.

157. Mackey, M., Glass, L., Oscillation and chaos in physiologist control systems, Science, 197, 287–299, 1977.

158. Box, G., Jenkins, G., Time series analysis: Forecasting and Control, Holden Day, San Francisco, 1970.

159. Hussain, A., Durbani, T., A new neural network for nonlinear time-series modelling, Neurovest Journal, 16–26, 1997.

160. Lang, K., et al., A time delay neural network architecture for isolated word recognition, Neural Networks, 3, 33–43, 1990.

161. LeCun, Y., et al., Backpropagation applied to handwritten zip code recognition, Neural Computation, 1(4), 1989.

162. Brameier, M., Banzhaf, W., A comparison of linear genetic programming and neural networks in medical data mining, IEEE Transactions on Evolutionary Computation, 5(1), 17–26, 2001.

163. Islam, M., et al., A constructive algorithm for training cooperative neural network ensembles, IEEE Transactions on Neural Networks, 14(3), 820–834, 2003.

164. Lehtokangas, M., Modeling with constructive backpropagation, Neural Networks, 12, 707–716, 1999.

165. Odri, S., et al., Evolutional development of a multilevel neural network, Neural Networks, 6, 583–595, 1993.

166. Petalas, Y., et al., Trajectory methods for neural network training, In Proceedings of IASTED, Artificial Intelligence and Applications, 400–404, Feb 16-18 2004.

167. Sexton, R., Dorsey, R., Reliable classification using neural networks: a genetic algorithm and backpropagation comparison, Decision Support Systems, 30, 11–22, 2000.

168. Socha, K., Blum, C., An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training, Neural Computation and Applications, 16, 235–247, 2007.

169. Blum, C., Socha, K., Training feed-forward neural networks with ant colony optimization: An application to pattern classification, 233–238, 2005.

170. Mendes, R., et al., Particle swarms for feedforward neural network training., In Proceedings of International joint conference on neural networks, 1895–1899, 2002.

171. Birattari, M., The problem of tuning metaheuristics as seen from a machine learning perspective, Ph.D. thesis, Dissertationen zur Kunstlichen Intelligenz. Akademische Verlagsgesellschaft Aka, Berlin, Germany, 2005.

172. Frank, A., Asuncion, A., Uci machine learning repository, Irvine, CA: University of California, School of Information and Computer Science, volume 21, 2010.

173. Prechelt, L., Proben1 a set of neural network benchmark problems and benchmarking rules, Technical report, 1994.

174. Duch, W., Korczak, J., Optimization and global minimization methods suitable for neural networks, Neural Computing Surveys, 2, 1–41, 1998.

175. Autio, L., et al., On the neural network classification of medical data and an endeavour to balance non-uniform data sets with artificial data extension, Computers in Biology and Medicine, 37, 388–397, 2007.

176. Ludermir, T., et al., An optimization methodology for neural network weights and architectures, IEEE Transactions on Neural Networks, 17(5), 1452–1460, 2006.

177. Ciocoiu, I., A modular feedforward network for solving classification problems, In Proceedings of NATO Advanced Research Workshop on: Limitations and Future Trends in Neural Computation, LFTNC01, 69–72, 2001.

178. Ciampi, A., Zhang, F., A new approach to training back-propagation artificial neural networks: empirical evaluation on ten data sets from clinical studies, Statistics in Medicine, 21, 1309–1330, 2002.

179. Liu, Z., et al., Evolving neural network using real coded genetic algorithm (ga) for multispectral image classification, Computer Systems, 20, 1119–1129, 2004.

180. Han, S., May, G., Optimization of neural network structure and learning parameters using genetic algorithms, IEEE, 8, 200–206, 1996.

181. Alba, E., Chicano, J., Training Neural Networks with GA Hybrid Algorithms, In Proc. of Gecco, LNCS, volume 3102, 852–863, Springer-Verlag, 2004.

182. Carvalho, M., Ludermir, T., Particle swarm optimization of feed-forward neural networks with weight decay, In Proceedings of Sixth International Conference on Hybrid Intelligent Systems (HIS06), 2662–2666, 2006.

183. Anastasiadis, A., Neural networks training and applications using biological data, Ph.D. thesis, School of Computer Science and Information Systems, Univertsity of London, December 2005.

184. Tallón-Ballesteros, A., et al., Distribution of the search of evolutionary product unit neural networks for classification, In Proceedings of the international conference on applied computing (IADIS), 266–273, Salamanca, Spain, 2007.

185. Wang, X., et al., Research on structure learning of product unit neural networks by particle swarm optimization, Inform. Technol. J., 7, 639–646, 2008.

186. Martínez-Estudillo, F., et al., Hybrid Evolutionary Algorithm with Product-Unit Neural Networks for Classification, volume 4507, 351–358, Springer-Verlag, 2007.

187. Hervás-Martínez, C., et al., Classification by means of evolutionary product-unit neural networks, In Proceedings of the International Joint Conference on Neural Networks (IJCNN), 1525–1532, Vancouver, Canada, July 16-21 2006.

188. Kohavi, R., A study of cross-validation and bootstrap for accuracy estimation and model selection, In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, volume 2(12), 1137–1143, 1995.

189. Quinlan, J., Combining instance-based and model-based learning, In Proceedings of ML93 (ed P.E. Utgoff), San Mateo: Morgan Kaufmann, 1993.

190. Weise, T., Global Optimization Algorithms – Theory and Application, Prentice Hall, New Jersey, 2nd edition, 2009, URL `http://www.it-weise.de/`.

191. Georgieva, A., Jordanov, I., Supervised neural network training with a hybrid global optimization technique, International Joint Conference on Neural Networks, Vancouver, BC, Canada, July 16-21 2006.

192. Zhang, J., et al., A hybrid particle swarm optimization backpropagation algorithm for feedforward neural network training, Applied Mathematics and Computation, 185, 1026–1037, 2007.

193. Wang, L., et al., Application of artificial neural network supported by bp and particle swarm optimization algorithm for evaluating the criticality class of spare parts, Third International Conference on Natural Computation (ICNC 2007), Haikou, China, August 24-27 2007.

194. Chen, Y., et al., A Novel Artificial Neural Network Based on Hybrid PSO-BP Algorithm in the Application of Adaptive PMD Compensation System, volume 4493, 311–319, Springer-Verlag, 2007.

195. Carvalho, M., Ludermir, T., Hybrid Training of Feed-Forward Neural Networks with Particle Swarm Optimization, LNCS, volume 4233, 1061–1070, Springer-Verlag, 2007.

196. Almeida, L., Ludermir, T., A hybrid method for searching near-optimal artificial neural networks, In Proceedings of the Sixth International Conference on Hybrid Intelligent Systems (HIS06), Auckland, New Zealand, December 13-15 2006.

197. Hasheminia, H., Niaki, S., A hybrid method of neural networks and genetic algorithmin economic modeling and analysis, Journal of Applied Science, 8(16), 2825–2833, 2008.

198. Ciocoiu, I., Hybrid feedforward neural networks for solving classification problems, Neural Processing Letters, 16, 81–91, 2002.

199. Schleif, F., Villmann, T., Neural maps and learning vector quantization theory and applications, In Proceedings of European Symposium on Artificial Neural Networks, ESANN, 509–516, Bruges, Belgium, 2009.

200. Kohonen, T., Self organizing and associative memory, Springer-Verlag, Berlin, 1989.

201. Kohonen, T., The self-organising map, In Proceedings of the IEEE, volume 78, 1464–1480, 1990.

202. Pham, D., Oztemel, E., Control chart pattern recognition using learning vector quantisation networks, International Journal of Production Research, 32(3), 721–729, 1994.

203. Kangas, J., On the analysis of pattern sequences by self-organizing maps, Ph.D. thesis, Helsinki University of Technology, Finland, 1994.

204. Merelo, J., Prieto, A., A combination of genetic algorithms and lvq, Artificial Neural Nets and Genetic Algorithms, 92–95, 1995.

205. Kohonen, T., et al., The learning vector quantization package, Technical Report A30, Helsinki University of Technology, 1996.

206. Powell, M., Radial basis functions for multivariable interpolation, Algorithms for Approximation, 143–167, 1987.

207. Broomhead, D., Lowe, D., Multivariable functional interpolation and adaptative networks, Complex Systems, 11, 321–355, 1988.

208. Harpham, C., et al., A review of genetic algorithms applied to training radial basis function networks, Neural Computing and Applications, 13(3), 193–201, 2004.

209. Poggio, T., Girosi, F., Networks for approximation and learning, In Proceedings of the IEEE, volume 78, 1481–1497, 1990.

210. Light, W., Some aspects of radial basis function approximation, Approximation Theory, Spline Functions and Applications, volume 356, 163–190, 1992.

211. Lacerda, E., et al., Radial Basis Function Networks 1, chapter Evolutionary Optimization of RBF Networks, 287–294, Physica Verlag, Heidelberg, New York, 2001.

212. Schwenker, F., et al., Three learning phases for radial-basis-function networks, Neural Networks, 14, 439–458, 2001.

213. Kirkpatrick, S., et al., Optimization by simulated annealing, Science, 220(4598), 671–680, 1983.

214. Glover, F., Tabu search - part i, ORSA Journal on Computing, 1(3), 190–206, 1989.

215. Schwefel, H.P., Kybernetische evolution als strategie der experimentellen forschung in der stromungstechnik, Master's thesis, Technical University of Berlin, Germany, 1965.

216. Giles, C., Lawrence, S., Overfitting and neural networks: Conjugate gradient and backpropagation, In Proceedings of IEEE International Joint Conference on Neural Networks, Como, Italy, July 2000.

EK -1

Tahmin Test Problemleri Verileri

1.1. Mackey-Glass Zaman Serisi Verileri

0 1.200000 1 1.085805 2 0.982477 3 0.888982 4 0.804384 5 0.727837 6 0.658574 7 0.595902 8 0.539195 9 0.487884 10 0.441455 11 0.399445 12 0.361433 13 0.327038 14 0.295916 15 0.267756 16 0.242276 17 0.219220 18 0.246463 19 0.307757 20 0.397178 21 0.494698 22 0.583265 23 0.655619 24 0.710732 25 0.750036 26 0.775630 27 0.789630 28 0.793964 29 0.790330 30 0.780201 31 0.764842 32 0.745342 33 0.722627 34 0.697486 35 0.675245 36 0.663863 37 0.668461 38 0.690802 39 0.728579 40 0.777080 41 0.830922 42 0.885404 43 0.937379 44 0.985353 45 1.028993 46 1.068476 47 1.104007 48 1.135587 49 1.162981 50 1.185804 51 1.203651 52 1.216485 53 1.225608 54 1.233433 55 1.242714 56 1.255372 57 1.271314 58 1.287582 59 1.298593 60 1.298236 61 1.282756 62 1.251977 63 1.208379 64 1.155518 65 1.096929 66 1.035636 67 0.974022 68 0.913880 69 0.856496 70 0.802634 71 0.752489 72 0.705736 73 0.661730 74 0.619840 75 0.579818 76 0.541986 77 0.507193 78 0.476675 79 0.452062 80 0.435599 81 0.430438 82 0.440628 83 0.470066 84 0.519976 85 0.586417 86 0.661005 87 0.734798 88 0.801487 89 0.857915 90 0.903110 91 0.937260 92 0.961090 93 0.975605 94 0.982012 95 0.981697 96 0.976237 97 0.967452 98 0.957519 99 0.949112 100 0.945426 101 0.949830 102 0.964947 103 0.991405 104 1.026797 105 1.065501 106 1.100331 107 1.125689 108 1.139706 109 1.143858 110 1.141285 111 1.135400 112 1.129145 113 1.124705 114 1.123387 115 1.125572 116 1.130703 117 1.137322 118 1.143128 119 1.144988 120 1.139072 121 1.121762 122 1.091658 123 1.050963 124 1.004313 125 0.956414 126 0.910743 127 0.869387 128 0.833268 129 0.802381 130 0.776007 131 0.752966 132 0.731924 133 0.711727 134 0.691683 135

0.671748 136 0.652640 137 0.635941 138 0.624249 139 0.621204 140 0.630916 141
0.656368 142 0.697366 143 0.749925 144 0.808058 145 0.866325 146 0.921143 147
0.970719 148 1.014423 149 1.052204 150 1.084218 151 1.110668 152 1.131784 153
1.147901 154 1.159582 155 1.167774 156 1.173954 157 1.180189 158 1.188916 159
1.202192 160 1.220445 161 1.241186 162 1.258763 163 1.266286 164 1.258944 165
1.235924 166 1.199665 167 1.153978 168 1.102639 169 1.048778 170 0.994755 171
0.942235 172 0.892280 173 0.845406 174 0.801591 175 0.760280 176 0.720531 177
0.681399 178 0.642465 179 0.604178 180 0.567762 181 0.534886 182 0.507477 183
0.487817 184 0.478731 185 0.483538 186 0.505308 187 0.545260 188 0.601152 189
0.667339 190 0.736852 191 0.803747 192 0.864109 193 0.915841 194 0.958028 195
0.990449 196 1.013370 197 1.027499 198 1.033953 199 1.034205 200 1.030058 201
1.023661 202 1.017569 203 1.014712 204 1.018122 205 1.030285 206 1.052185 207
1.082272 208 1.115781 209 1.145484 210 1.164577 211 1.169838 212 1.162239 213
1.145265 214 1.123018 215 1.099141 216 1.076439 217 1.056806 218 1.041191 219
1.029549 220 1.020763 221 1.012659 222 1.002189 223 0.985975 224 0.961373 225
0.927878 226 0.887701 227 0.844748 228 0.803028 229 0.765798 230 0.735436 231
0.713515 232 0.700766 233 0.696985 234 0.701059 235 0.711217 236 0.725485 237
0.742205 238 0.760487 239 0.780547 240 0.803796 241 0.832465 242 0.868391 243
0.911255 244 0.957877 245 1.003735 246 1.045223 247 1.080692 248 1.110207 249
1.134864 250 1.156158 251 1.175524 252 1.194054 253 1.212379 254 1.230695 255
1.248884 256 1.266640 257 1.283480 258 1.298500 259 1.309769 260 1.313696 261
1.305660 262 1.282615 263 1.245222 264 1.197020 265 1.142193 266 1.084148 267
1.025192 268 0.966703 269 0.909441 270 0.853817 271 0.800079 272 0.748416 273
0.698999 274 0.651991 275 0.607558 276 0.565929 277 0.527497 278 0.492962 279
0.463451 280 0.440647 281 0.426956 282 0.425606 283 0.440320 284 0.474057 285
0.526868 286 0.594386 287 0.668819 288 0.741983 289 0.807811 290 0.863005 291
0.906455 292 0.938388 293 0.959730 294 0.971732 295 0.975805 296 0.973487 297
0.966480 298 0.956735 299 0.946570 300 0.938761 301 0.936469 302 0.942809 303
0.959995 304 0.988265 305 1.024910 306 1.064101 307 1.098665 308 1.123405 309
1.137060 310 1.141552 311 1.140181 312 1.136291 313 1.132636 314 1.131148 315
1.132844 316 1.137804 317 1.145212 318 1.153459 319 1.160217 320 1.162401 321
1.156239 322 1.138207 323 1.107121 324 1.065472 325 1.018065 326 0.969583 327

0.923356 328 0.881269 329 0.844032 330 0.811458 331 0.782721 332 0.756648 333
0.732041 334 0.707977 335 0.684032 336 0.660379 337 0.637844 338 0.617994 339
0.603317 340 0.597306 341 0.604023 342 0.626663 343 0.665554 344 0.717249 345
0.775948 346 0.835981 347 0.893317 348 0.945695 349 0.992089 350 1.032149 351
1.065827 352 1.093216 353 1.114525 354 1.130140 355 1.140719 356 1.147307 357
1.151472 358 1.155376 359 1.161622 360 1.172652 361 1.189671 362 1.211384 363
1.233268 364 1.248462 365 1.250668 366 1.237057 367 1.208749 368 1.169213 369
1.122479 370 1.072115 371 1.020906 372 0.970856 373 0.923279 374 0.878864 375
0.837673 376 0.799122 377 0.762077 378 0.725223 379 0.687680 380 0.649539 381
0.611928 382 0.576625 383 0.545677 384 0.521332 385 0.506185 386 0.503249 387
0.515553 388 0.545016 389 0.590971 390 0.649578 391 0.714975 392 0.781321 393
0.844174 394 0.900712 395 0.949309 396 0.989066 397 1.019593 398 1.041016 399
1.054012 400 1.059763 401 1.059877 402 1.056343 403 1.051530 404 1.048181 405
1.049234 406 1.057364 407 1.074217 408 1.099463 409 1.129911 410 1.159358 411
1.180342 412 1.187519 413 1.179893 414 1.160138 415 1.132500 416 1.101156 417
1.069486 418 1.039921 419 1.013946 420 0.992104 421 0.973929 422 0.957909 423
0.941608 424 0.922086 425 0.896761 426 0.864540 427 0.826563 428 0.785791 429
0.745799 430 0.709874 431 0.680764 432 0.660696 433 0.651282 434 0.653232 435
0.666083 436 0.688225 437 0.717346 438 0.751092 439 0.787658 440 0.826121 441
0.866430 442 0.908937 443 0.953453 444 0.998343 445 1.040707 446 1.077710 447
1.107795 448 1.130955 449 1.148364 450 1.161881 451 1.173615 452 1.185533 453
1.199113 454 1.215074 455 1.233217 456 1.252395 457 1.270558 458 1.284811 459
1.291496 460 1.286632 461 1.267273 462 1.233267 463 1.187540 464 1.134458 465
1.078063 466 1.021283 467 0.965883 468 0.912684 469 0.861823 470 0.813029 471
0.765889 472 0.720090 473 0.675568 474 0.632548 475 0.591509 476 0.553138 477
0.518356 478 0.488427 479 0.465150 480 0.451069 481 0.449518 482 0.464138 483
0.497449 484 0.548889 485 0.613937 486 0.685516 487 0.756539 488 0.821649 489
0.877611 490 0.922941 491 0.957377 492 0.981433 493 0.996094 494 1.002626 495
1.002505 496 0.997416 497 0.989331 498 0.980616 499 0.974093 500 0.972911 501
0.980045 502 0.997456 503 1.025106 504 1.060061 505 1.096296 506 1.126435 507
1.145185 508 1.151448 509 1.147526 510 1.137132 511 1.123948 512 1.110994 513
1.100433 514 1.093503 515 1.090464 516 1.090554 517 1.092009 518 1.092154 519

1.087570 520 1.074588 521 1.050551 522 1.015530 523 0.972735 524 0.926852 525
0.882125 526 0.841574 527 0.807007 528 0.779199 529 0.758018 530 0.742566 531
0.731406 532 0.722900 533 0.715607 534 0.708656 535 0.702035 536 0.696789 537
0.695155 538 0.700526 539 0.716816 540 0.746707 541 0.789509 542 0.840822 543
0.894785 544 0.946675 545 0.993905 546 1.035658 547 1.072137 548 1.103946 549
1.131720 550 1.155965 551 1.177045 552 1.195250 553 1.210928 554 1.224643 555
1.237320 556 1.250254 557 1.264696 558 1.280754 559 1.295926 560 1.304798 561
1.301360 562 1.282431 563 1.248773 564 1.203585 565 1.150617 566 1.093152 567
1.033733 568 0.974216 569 0.915907 570 0.859682 571 0.806072 572 0.755304 573
0.707338 574 0.661923 575 0.618766 576 0.577790 577 0.539365 578 0.504340 579
0.473962 580 0.449885 581 0.434360 582 0.430460 583 0.441959 584 0.472278 585
0.522256 586 0.588109 587 0.662049 588 0.735564 589 0.802320 590 0.858901 591
0.904119 592 0.938081 593 0.961541 594 0.975573 595 0.981463 596 0.980670 597
0.974844 598 0.965872 599 0.955985 600 0.947869 601 0.944657 602 0.949571 603
0.965078 604 0.991748 605 1.027218 606 1.065921 607 1.100647 608 1.125785 609
1.139541 610 1.143502 611 1.140898 612 1.135167 613 1.129238 614 1.125241 615
1.124412 616 1.127041 617 1.132491 618 1.139256 619 1.145032 620 1.146719 621
1.140542 622 1.122941 623 1.092564 624 1.051655 625 1.004876 626 0.956938 627
0.911294 628 0.869992 629 0.833904 630 0.802975 631 0.776445 632 0.753120 633
0.731683 634 0.711024 635 0.690511 636 0.670155 637 0.650707 638 0.633757 639
0.621894 640 0.618745 641 0.628412 642 0.653867 643 0.694910 644 0.747548 645
0.805795 646 0.864214 647 0.919217 648 0.968989 649 1.012876 650 1.050798 651
1.082888 652 1.109334 653 1.130368 654 1.146335 655 1.157821 656 1.165790 657
1.171740 658 1.177759 659 1.186302 660 1.199457 661 1.217701 662 1.238616 663
1.256610 664 1.264766 665 1.258152 666 1.235818 667 1.200128 668 1.154887 669
1.103900 670 1.050333 671 0.996577 672 0.944317 673 0.894628 674 0.848034 675
0.804508 676 0.763481 677 0.723987 678 0.685046 679 0.646212 680 0.607917 681
0.571394 682 0.538323 683 0.510647 684 0.490642 685 0.481118 686 0.485367 687
0.506452 688 0.545643 689 0.600816 690 0.666462 691 0.735702 692 0.802596 693
0.863192 694 0.915333 695 0.958045 696 0.991048 697 1.014557 698 1.029238 699
1.036179 700 1.036836 701 1.033001 702 1.026813 703 1.020815 704 1.017924 705
1.021163 706 1.033033 707 1.054562 708 1.084289 709 1.117550 710 1.147159 711

1.166234 712 1.171403 713 1.163530 714 1.146078 715 1.123176 716 1.098510 717
1.074927 718 1.054358 719 1.037797 720 1.025239 721 1.015616 722 1.006796 723
0.995774 724 0.979204 725 0.954462 726 0.921015 727 0.881002 728 0.838263 729
0.796774 730 0.759792 731 0.729733 732 0.708225 733 0.696068 734 0.693122 735
0.698307 736 0.709853 737 0.725752 738 0.744290 739 0.764513 740 0.786554 741
0.811716 742 0.842063 743 0.879202 744 0.922590 745 0.968978 746 1.013982 747
1.054240 748 1.088328 749 1.116473 750 1.139879 751 1.160112 752 1.178656 753
1.196627 754 1.214649 755 1.232874 756 1.251102 757 1.268897 758 1.285593 759
1.300020 760 1.309900 761 1.311386 762 1.300050 763 1.273576 764 1.233379 765
1.183325 766 1.127546 767 1.069254 768 1.010541 769 0.952611 770 0.896083 771
0.841265 772 0.788335 773 0.737448 774 0.688763 775 0.642454 776 0.598723 777
0.557849 778 0.520299 779 0.486867 780 0.458804 781 0.437963 782 0.426964 783
0.429224 784 0.448421 785 0.486957 786 0.543819 787 0.613610 788 0.688229 789
0.759930 790 0.823357 791 0.875803 792 0.916518 793 0.945921 794 0.965020 795
0.975106 796 0.977616 797 0.974120 798 0.966370 799 0.956398 800 0.946634 801
0.939969 802 0.939606 803 0.948514 804 0.968502 805 0.999146 806 1.036925 807
1.075412 808 1.107591 809 1.129155 810 1.139786 811 1.141940 812 1.139048 813
1.134371 814 1.130494 815 1.129145 816 1.131129 817 1.136308 818 1.143660 819
1.151376 820 1.156916 821 1.156974 822 1.147708 823 1.126020 824 1.091693 825
1.048062 826 1.000093 827 0.952168 828 0.907222 829 0.866816 830 0.831413 831
0.800638 832 0.773536 833 0.748873 834 0.725456 835 0.702433 836 0.679494 837
0.656965 838 0.635864 839 0.618023 840 0.606259 841 0.604353 842 0.616305 843
0.644590 844 0.688261 845 0.742802 846 0.802186 847 0.861224 848 0.916571 849
0.966518 850 1.010389 851 1.048024 852 1.079467 853 1.104862 854 1.124459 855
1.138695 856 1.148304 857 1.154446 858 1.158842 859 1.163793 860 1.171892 861
1.185233 862 1.204187 863 1.226208 864 1.245613 865 1.255392 866 1.250477 867
1.229795 868 1.195680 869 1.151986 870 1.102598 871 1.050755 872 0.998896 873
0.948727 874 0.901310 875 0.857110 876 0.815980 877 0.777172 878 0.739509 879
0.701849 880 0.663688 881 0.625533 882 0.588764 883 0.555226 884 0.526980 885
0.506368 886 0.496179 887 0.499584 888 0.519434 889 0.556828 890 0.609727 891
0.673038 892 0.740417 893 0.806314 894 0.866906 895 0.919953 896 0.964268 897
0.999297 898 1.024988 899 1.041798 900 1.050690 901 1.053062 902 1.050689 903

1.045715 904 1.040676 905 1.038447 906 1.041976 907 1.053666 908 1.074494 909
1.103024 910 1.134696 911 1.162390 912 1.179128 913 1.181349 914 1.169841 915
1.148159 916 1.120591 917 1.090966 918 1.062233 919 1.036398 920 1.014532 921
0.996730 922 0.982037 923 0.968454 924 0.953157 925 0.933080 926 0.905956 927
0.871497 928 0.831772 929 0.790306 930 0.750769 931 0.716288 932 0.689355 933
0.671844 934 0.664875 935 0.668563 936 0.681910 937 0.703023 938 0.729645 939
0.759766 940 0.792114 941 0.826405 942 0.863243 943 0.903483 944 0.947046 945
0.991971 946 1.034902 947 1.072749 948 1.103931 949 1.128496 950 1.147655 951
1.163223 952 1.177157 953 1.191170 954 1.206433 955 1.223394 956 1.241735 957
1.260430 958 1.277822 959 1.291578 960 1.298522 961 1.294761 962 1.276946 963
1.244354 964 1.199550 965 1.146802 966 1.090183 967 1.032672 968 0.976085 969
0.921304 970 0.868572 971 0.817775 972 0.768693 973 0.721193 974 0.675320 975
0.631308 976 0.589552 977 0.550600 978 0.515212 979 0.484493 980 0.460074 981
0.444313 982 0.440392 983 0.451992 984 0.482084 985 0.530946 986 0.594807 987
0.666712 988 0.739091 989 0.805939 990 0.863560 991 0.910267 992 0.945774 993
0.970643 994 0.985912 995 0.992878 996 0.993014 997 0.987970 998 0.979654 999
0.970338 1000 0.962752 1001 0.960010 1002 0.965206 1003 0.980626 1004 1.006750
1005 1.041302 1006 1.078822 1007 1.112006 1008 1.134987 1009 1.145864 1010
1.146365 1011 1.139948 1012 1.130249 1013 1.120336 1014 1.112457 1015 1.107947
1016 1.107180 1017 1.109542 1018 1.113466 1019 1.116514 1020 1.115467 1021
1.106593 1022 1.086650 1023 1.054728 1024 1.013329 1025 0.967095 1026 0.920603
1027 0.877212 1028 0.838948 1029 0.806712 1030 0.780472 1031 0.759440 1032
0.742312 1033 0.727581 1034 0.713911 1035 0.700463 1036 0.687149 1037 0.674774
1038 0.665176 1039 0.661327 1040 0.667154 1041 0.686467 1042 0.720869 1043
0.768074 1044 0.822677 1045 0.878826 1046 0.932298 1047 0.980915 1048 1.023994
1049 1.061635 1050 1.094225 1051 1.122167 1052 1.145783 1053 1.165352 1054
1.181208 1055 1.193893 1056 1.204334 1057 1.213970 1058 1.224688 1059 1.238262
1060 1.255129 1061 1.272940 1062 1.286218 1063 1.288448 1064 1.275486 1065
1.247173 1066 1.206257 1067 1.156501 1068 1.101449 1069 1.043964 1070 0.986190
1071 0.929644 1072 0.875333 1073 0.823826 1074 0.775291 1075 0.729516 1076
0.685997 1077 0.644176 1078 0.603810 1079 0.565257 1080 0.529468 1081 0.497807
1082 0.471979 1083 0.454183 1084 0.447334 1085 0.455022 1086 0.480671 1087

0.525599 1088 0.586995 1089 0.658056 1090 0.730704 1091 0.798457 1092 0.857459 1093 0.906035 1094 0.943854 1095 0.971297 1096 0.989153 1097 0.998506 1098 1.000695 1099 0.997297 1100 0.990146 1101 0.981409 1102 0.973686 1103 0.970009 1104 0.973559 1105 0.986926 1106 1.011033 1107 1.044061 1108 1.080921 1109 1.114365 1110 1.138095 1111 1.149519 1112 1.149833 1113 1.142310 1114 1.130654 1115 1.118124 1116 1.107193 1117 1.099442 1118 1.095508 1119 1.095035 1120 1.096670 1121 1.098111 1122 1.096200 1123 1.087212 1124 1.067817 1125 1.036841 1126 0.996426 1127 0.950987 1128 0.905085 1129 0.862205 1130 0.824575 1131 0.793330 1132 0.768669 1133 0.749978 1134 0.736027 1135 0.725257 1136 0.716157 1137 0.707637 1138 0.699344 1139 0.691886 1140 0.687004 1141 0.687654 1142 0.697718 1143 0.720768 1144 0.757882 1145 0.806213 1146 0.860171 1147 0.914200 1148 0.964614 1149 1.009770 1150 1.049409 1151 1.083943 1152 1.113981 1153 1.140068 1154 1.162618 1155 1.181942 1156 1.198359 1157 1.212344 1158 1.224702 1159 1.236669 1160 1.249762 1161 1.265031 1162 1.281665 1163 1.295788 1164 1.301211 1165 1.292623 1166 1.268390 1167 1.230440 1168 1.182357 1169 1.127794 1170 1.069782 1171 1.010614 1172 0.951945 1173 0.894917 1174 0.840268 1175 0.788392 1176 0.739373 1177 0.693021 1178 0.648978 1179 0.606960 1180 0.567048 1181 0.529843 1182 0.496407 1183 0.468171 1184 0.447014 1185 0.435496 1186 0.436993 1187 0.455290 1188 0.493052 1189 0.549452 1190 0.618967 1191 0.693236 1192 0.764479 1193 0.827557 1194 0.880008 1195 0.921189 1196 0.951448 1197 0.971623 1198 0.982832 1199 0.986405 1200 0.983860

1.2. Box-Jenkins Zaman Serisi Verileri

1 5.31e+01 0.00e+00 5.27e+01 2 5.27e+01 1.78e-01 5.24e+01 3 5.24e+01 3.39e-01 5.22e+01 4 5.22e+01 3.73e-01 5.20e+01 5 5.20e+01 4.41e-01 5.20e+01 6 5.20e+01 4.61e-01 5.24e+01 7 5.24e+01 3.48e-01 5.30e+01 8 5.30e+01 1.27e-01 5.40e+01 9 5.40e+01 -1.80e-01 5.49e+01 10 5.49e+01 -5.88e-01 5.60e+01 11 5.60e+01 -1.055e+00 5.68e+01 12 5.68e+01 -1.421e+00 5.68e+01 13 5.68e+01 -1.52e+00 5.64e+01 14 5.64e+01 -1.302e+00 5.57e+01 15 5.57e+01 -8.14e-01 5.50e+01 16 5.50e+01 -4.75e-01 5.43e+01 17 5.43e+01 -1.93e-01 5.32e+01 18 5.32e+01 8.80e-02 5.23e+01 19 5.23e+01 4.35e-01 5.16e+01 20 5.16e+01 7.71e-01 5.12e+01 21

5.12e+01 8.66e-01 5.08e+01 22 5.08e+01 8.75e-01 5.05e+01 23 5.05e+01 8.91e-01 5.00e+01 24 5.00e+01 9.87e-01 4.92e+01 25 4.92e+01 1.263e+00 4.84e+01 26 4.84e+01 1.775e+00 4.79e+01 27 4.79e+01 1.976e+00 4.76e+01 28 4.76e+01 1.934e+00 4.75e+01 29 4.75e+01 1.866e+00 4.75e+01 30 4.75e+01 1.832e+00 4.76e+01 31 4.76e+01 1.767e+00 4.81e+01 32 4.81e+01 1.608e+00 4.90e+01 33 4.90e+01 1.265e+00 5.00e+01 34 5.00e+01 7.90e-01 5.11e+01 35 5.11e+01 3.60e-01 5.18e+01 36 5.18e+01 1.15e-01 5.19e+01 37 5.19e+01 8.80e-02 5.17e+01 38 5.17e+01 3.31e-01 5.12e+01 39 5.12e+01 6.45e-01 5.00e+01 40 5.00e+01 9.60e-01 4.83e+01 41 4.83e+01 1.409e+00 4.70e+01 42 4.70e+01 2.67e+00 4.58e+01 43 4.58e+01 2.834e+00 4.56e+01 44 4.56e+01 2.812e+00 4.60e+01 45 4.60e+01 2.483e+00 4.69e+01 46 4.69e+01 1.929e+00 4.78e+01 47 4.78e+01 1.485e+00 4.82e+01 48 4.82e+01 1.214e+00 4.83e+01 49 4.83e+01 1.239e+00 4.79e+01 50 4.79e+01 1.608e+00 4.72e+01 51 4.72e+01 1.905e+00 4.72e+01 52 4.72e+01 2.023e+00 4.81e+01 53 4.81e+01 1.815e+00 4.94e+01 54 4.94e+01 5.35e-01 5.06e+01 55 5.06e+01 1.22e-01 5.15e+01 56 5.15e+01 9.00e-03 5.16e+01 57 5.16e+01 1.64e-01 5.12e+01 58 5.12e+01 6.71e-01 5.05e+01 59 5.05e+01 1.019e+00 5.01e+01 60 5.01e+01 1.146e+00 4.98e+01 61 4.98e+01 1.155e+00 4.96e+01 62 4.96e+01 1.112e+00 4.94e+01 63 4.94e+01 1.121e+00 4.93e+01 64 4.93e+01 1.223e+00 4.92e+01 65 4.92e+01 1.257e+00 4.93e+01 66 4.93e+01 1.157e+00 4.97e+01 67 4.97e+01 9.13e-01 5.03e+01 68 5.03e+01 6.20e-01 5.13e+01 69 5.13e+01 2.55e-01 5.28e+01 70 5.28e+01 -2.80e-01 5.44e+01 71 5.44e+01 -1.08e+00 5.60e+01 72 5.60e+01 -1.551e+00 5.69e+01 73 5.69e+01 -1.799e+00 5.75e+01 74 5.75e+01 -1.825e+00 5.73e+01 75 5.73e+01 -1.456e+00 5.66e+01 76 5.66e+01 -9.44e-01 5.60e+01 77 5.60e+01 -5.70e-01 5.54e+01 78 5.54e+01 -4.31e-01 5.54e+01 79 5.54e+01 -5.77e-01 5.64e+01 80 5.64e+01 -9.60e-01 5.72e+01 81 5.72e+01 -1.616e+00 5.80e+01 82 5.80e+01 -1.875e+00 5.84e+01 83 5.84e+01 -1.891e+00 5.84e+01 84 5.84e+01 -1.746e+00 5.81e+01 85 5.81e+01 -1.474e+00 5.77e+01 86 5.77e+01 -1.201e+00 5.70e+01 87 5.70e+01 -9.27e-01 5.60e+01 88 5.60e+01 -5.24e-01 5.47e+01 89 5.47e+01 4.00e-02 5.32e+01 90 5.32e+01 7.88e-01 5.21e+01 91 5.21e+01 9.43e-01 5.16e+01 92 5.16e+01 9.30e-01 5.10e+01 93 5.10e+01 1.006e+00 5.05e+01 94 5.05e+01 1.137e+00 5.04e+01 95 5.04e+01 1.198e+00 5.10e+01 96 5.10e+01 1.054e+00 5.18e+01 97 5.18e+01 5.95e-01 5.24e+01 98 5.24e+01

-8.00e-02 5.30e+01 99 5.30e+01 -3.14e-01 5.34e+01 100 5.34e+01 -2.88e-01 5.36e+01 101 5.36e+01 -1.53e-01 5.37e+01 102 5.37e+01 -1.09e-01 5.38e+01 103 5.38e+01 -1.87e-01 5.38e+01 104 5.38e+01 -2.55e-01 5.38e+01 105 5.38e+01 -2.29e-01 5.33e+01 106 5.33e+01 -7.00e-03 5.30e+01 107 5.30e+01 2.54e-01 5.29e+01 108 5.29e+01 3.30e-01 5.34e+01 109 5.34e+01 1.02e-01 5.46e+01 110 5.46e+01 -4.23e-01 5.64e+01 111 5.64e+01 -1.139e+00 5.80e+01 112 5.80e+01 -2.275e+00 5.94e+01 113 5.94e+01 -2.594e+00 6.02e+01 114 6.02e+01 -2.716e+00 6.00e+01 115 6.00e+01 -2.51e+00 5.94e+01 116 5.94e+01 -1.79e+00 5.84e+01 117 5.84e+01 -1.346e+00 5.76e+01 118 5.76e+01 -1.081e+00 5.69e+01 119 5.69e+01 -9.10e-01 5.64e+01 120 5.64e+01 -8.76e-01 5.60e+01 121 5.60e+01 -8.85e-01 5.57e+01 122 5.57e+01 -8.00e-01 5.53e+01 123 5.53e+01 -5.44e-01 5.50e+01 124 5.50e+01 -4.16e-01 5.44e+01 125 5.44e+01 -2.71e-01 5.37e+01 126 5.37e+01 0.00e+00 5.28e+01 127 5.28e+01 4.03e-01 5.16e+01 128 5.16e+01 8.41e-01 5.06e+01 129 5.06e+01 1.285e+00 4.94e+01 130 4.94e+01 1.607e+00 4.88e+01 131 4.88e+01 1.746e+00 4.85e+01 132 4.85e+01 1.683e+00 4.87e+01 133 4.87e+01 1.485e+00 4.92e+01 134 4.92e+01 9.93e-01 4.98e+01 135 4.98e+01 6.48e-01 5.04e+01 136 5.04e+01 5.77e-01 5.07e+01 137 5.07e+01 5.77e-01 5.09e+01 138 5.09e+01 6.32e-01 5.07e+01 139 5.07e+01 7.47e-01 5.05e+01 140 5.05e+01 9.00e-01 5.04e+01 141 5.04e+01 9.93e-01 5.02e+01 142 5.02e+01 9.68e-01 5.04e+01 143 5.04e+01 7.90e-01 5.12e+01 144 5.12e+01 3.99e-01 5.23e+01 145 5.23e+01 -1.61e-01 5.32e+01 146 5.32e+01 -5.53e-01 5.39e+01 147 5.39e+01 -6.03e-01 5.41e+01 148 5.41e+01 -4.24e-01 5.40e+01 149 5.40e+01 -1.94e-01 5.36e+01 150 5.36e+01 -4.90e-02 5.32e+01 151 5.32e+01 6.00e-02 5.30e+01 152 5.30e+01 1.61e-01 5.28e+01 153 5.28e+01 3.01e-01 5.23e+01 154 5.23e+01 5.17e-01 5.19e+01 155 5.19e+01 5.66e-01 5.16e+01 156 5.16e+01 5.60e-01 5.16e+01 157 5.16e+01 5.73e-01 5.14e+01 158 5.14e+01 5.92e-01 5.12e+01 159 5.12e+01 6.71e-01 5.07e+01 160 5.07e+01 9.33e-01 5.00e+01 161 5.00e+01 1.337e+00 4.94e+01 162 4.94e+01 1.46e+00 4.93e+01 163 4.93e+01 1.353e+00 4.97e+01 164 4.97e+01 7.72e-01 5.06e+01 165 5.06e+01 2.18e-01 5.18e+01 166 5.18e+01 -2.37e-01 5.30e+01 167 5.30e+01 -7.14e-01 5.40e+01 168 5.40e+01 -1.099e+00 5.53e+01 169 5.53e+01 -1.269e+00 5.59e+01 170 5.59e+01 -1.175e+00 5.59e+01 171 5.59e+01 -6.76e-01 5.46e+01 172 5.46e+01 3.30e-02 5.35e+01 173 5.35e+01 5.56e-01 5.24e+01 174 5.24e+01 6.43e-01 5.21e+01 175

5.21e+01 4.84e-01 5.23e+01 176 5.23e+01 1.09e-01 5.30e+01 177 5.30e+01 -3.10e-01 5.38e+01 178 5.38e+01 -6.97e-01 5.46e+01 179 5.46e+01 -1.047e+00 5.54e+01 180 5.54e+01 -1.218e+00 5.59e+01 181 5.59e+01 -1.183e+00 5.59e+01 182 5.59e+01 -8.73e-01 5.52e+01 183 5.52e+01 -3.36e-01 5.44e+01 184 5.44e+01 6.30e-02 5.37e+01 185 5.37e+01 8.40e-02 5.36e+01 186 5.36e+01 0.00e+00 5.36e+01 187 5.36e+01 1.00e-03 5.32e+01 188 5.32e+01 2.09e-01 5.25e+01 189 5.25e+01 5.56e-01 5.20e+01 190 5.20e+01 7.82e-01 5.14e+01 191 5.14e+01 8.58e-01 5.10e+01 192 5.10e+01 9.18e-01 5.09e+01 193 5.09e+01 8.62e-01 5.24e+01 194 5.24e+01 4.16e-01 5.35e+01 195 5.35e+01 -3.36e-01 5.56e+01 196 5.56e+01 -9.59e-01 5.80e+01 197 5.80e+01 -1.813e+00 5.95e+01 198 5.95e+01 -2.378e+00 6.00e+01 199 6.00e+01 -2.499e+00 6.04e+01 200 6.04e+01 -2.473e+00 6.05e+01 201 6.05e+01 -2.33e+00 6.02e+01 202 6.02e+01 -2.053e+00 5.97e+01 203 5.97e+01 -1.739e+00 5.90e+01 204 5.90e+01 -1.261e+00 5.76e+01 205 5.76e+01 -5.69e-01 5.64e+01 206 5.64e+01 -1.37e-01 5.52e+01 207 5.52e+01 -2.40e-02 5.45e+01 208 5.45e+01 -5.00e-02 5.41e+01 209 5.41e+01 -1.35e-01 5.41e+01 210 5.41e+01 -2.76e-01 5.44e+01 211 5.44e+01 -5.34e-01 5.55e+01 212 5.55e+01 -8.71e-01 5.62e+01 213 5.62e+01 -1.243e+00 5.70e+01 214 5.70e+01 -1.439e+00 5.73e+01 215 5.73e+01 -1.422e+00 5.74e+01 216 5.74e+01 -1.175e+00 5.70e+01 217 5.70e+01 -8.13e-01 5.64e+01 218 5.64e+01 -6.34e-01 5.59e+01 219 5.59e+01 -5.82e-01 5.55e+01 220 5.55e+01 -6.25e-01 5.53e+01 221 5.53e+01 -7.13e-01 5.52e+01 222 5.52e+01 -8.48e-01 5.54e+01 223 5.54e+01 -1.039e+00 5.60e+01 224 5.60e+01 -1.346e+00 5.65e+01 225 5.65e+01 -1.628e+00 5.71e+01 226 5.71e+01 -1.619e+00 5.73e+01 227 5.73e+01 -1.149e+00 5.68e+01 228 5.68e+01 -4.88e-01 5.56e+01 229 5.56e+01 -1.60e-01 5.50e+01 230 5.50e+01 -7.00e-03 5.41e+01 231 5.41e+01 -9.20e-02 5.43e+01 232 5.43e+01 -6.20e-01 5.53e+01 233 5.53e+01 -1.086e+00 5.64e+01 234 5.64e+01 -1.525e+00 5.72e+01 235 5.72e+01 -1.858e+00 5.78e+01 236 5.78e+01 -2.029e+00 5.83e+01 237 5.83e+01 -2.024e+00 5.86e+01 238 5.86e+01 -1.961e+00 5.88e+01 239 5.88e+01 -1.952e+00 5.88e+01 240 5.88e+01 -1.794e+00 5.86e+01 241 5.86e+01 -1.302e+00 5.80e+01 242 5.80e+01 -1.03e+00 5.74e+01 243 5.74e+01 -9.18e-01 5.70e+01 244 5.70e+01 -7.98e-01 5.64e+01 245 5.64e+01 -8.67e-01 5.63e+01 246 5.63e+01 -1.047e+00 5.64e+01 247 5.64e+01 -1.123e+00 5.64e+01 248 5.64e+01 -8.76e-01 5.60e+01 249 5.60e+01 -3.95e-01 5.52e+01 250 5.52e+01 1.85e-01

5.40e+01 251 5.40e+01 6.62e-01 5.30e+01 252 5.30e+01 7.09e-01 5.20e+01 253 5.20e+01 6.05e-01 5.16e+01 254 5.16e+01 5.01e-01 5.16e+01 255 5.16e+01 6.03e-01 5.11e+01 256 5.11e+01 9.43e-01 5.04e+01 257 5.04e+01 1.223e+00 5.00e+01 258 5.00e+01 1.249e+00 5.00e+01 259 5.00e+01 8.24e-01 5.20e+01 260 5.20e+01 1.02e-01 5.40e+01 261 5.40e+01 2.50e-02 5.51e+01 262 5.51e+01 3.82e-01 5.45e+01 263 5.45e+01 9.22e-01 5.28e+01 264 5.28e+01 1.032e+00 5.14e+01 265 5.14e+01 8.66e-01 5.08e+01 266 5.08e+01 5.27e-01 5.12e+01 267 5.12e+01 9.30e-02 5.20e+01 268 5.20e+01 -4.58e-01 5.28e+01 269 5.28e+01 -7.48e-01 5.38e+01 270 5.38e+01 -9.47e-01 5.45e+01 271 5.45e+01 -1.029e+00 5.49e+01 272 5.49e+01 -9.28e-01 5.49e+01 273 5.49e+01 -6.45e-01 5.48e+01 274 5.48e+01 -4.24e-01 5.44e+01 275 5.44e+01 -2.76e-01 5.37e+01 276 5.37e+01 -1.58e-01 5.33e+01 277 5.33e+01 -3.30e-02 5.28e+01 278 5.28e+01 1.02e-01 5.26e+01 279 5.26e+01 2.51e-01 5.26e+01 280 5.26e+01 2.80e-01 5.30e+01 281 5.30e+01 0.00e+00 5.43e+01 282 5.43e+01 -4.93e-01 5.60e+01 283 5.60e+01 -7.59e-01 5.70e+01 284 5.70e+01 -8.24e-01 5.80e+01 285 5.80e+01 -7.40e-01 5.86e+01 286 5.86e+01 -5.28e-01 5.85e+01 287 5.85e+01 -2.04e-01 5.83e+01 288 5.83e+01 3.40e-02 5.78e+01 289 5.78e+01 2.04e-01 5.73e+01 290 5.73e+01 2.53e-01 5.70e+01

EK -2

Sınıflandırma Test Problemleri Örnek Verileri

2.1. Kanser Hastalığı Verileri

0.2 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.2 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.1 0.1
0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.5 0.4 0.6 0.8 0.4 0.1 0.8 1 0.1 0 1 0.5 0.3 0.3 0.1 0.2 0.1
0.2 0.1 0.1 1 0 0.2 0.3 0.1 0.1 0.3 0.1 0.1 0.1 0.1 1 0 0.3 0.5 0.7 0.8 0.8 0.9 0.7 1 0.7 0
1 1 0.5 0.6 1 0.6 1 0.7 0.7 1 0 1 1 0.9 0.8 0.7 0.6 0.4 0.7 1 0.3 0 1 0.4 0.1 0.1 0.1 0.2
0.1 0.3 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.8 1 1 0.1 0.3 0.6 0.3 0.9 0.1
0 1 0.1 0.1 0.3 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 0.1 1 0 0.3 0.4
0.5 0.2 0.6 0.8 0.4 0.1 0.1 0 1 0.4 0.3 0.3 0.1 0.2 0.1 0.3 0.3 0.1 1 0 0.3 0.3 0.2 0.1 0.3
0.1 0.3 0.6 0.1 1 0 0.2 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1
0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.8 1 1 1 0.5 1 0.8 1 0.6 0 1 0.8 0.7 0.4
0.4 0.5 0.3 0.5 1 0.1 0 1 0.1 0.1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 1 0 0.2 0.1 0.1 0.1 0.2 0.1
0.1 0.1 0.1 1 0 1 0.8 0.8 0.4 1 1 0.8 0.1 0.1 0 1 0.5 0.1 0.1 0.2 0.2 0.1 0.2 0.1 0.1 1 0
0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.2 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.3 0.3
0.3 0.6 1 0.3 0.1 0.1 0 1 0.4 0.8 0.6 0.3 0.4 1 0.7 0.1 0.1 0 1 0.4 0.3 0.2 0.1 0.3 0.1 0.2
0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.8 0.5 0.5 0.5 0.2 1 0.4 0.3 0.1 0 1
0.3 0.3 0.5 0.2 0.3 1 0.7 0.1 0.1 0 1 0.2 0.3 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.3 0.1 0.1
0.1 0.2 0.5 0.1 0.1 0.1 1 0 0.3 0.2 0.1 0.1 0.2 0.2 0.3 0.1 0.1 1 0 0.2 0.7 1 1 0.7 1 0.4
0.9 0.4 0 1 0.1 0.1 0.2 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 1
0 0.5 1 1 1 1 0.2 1 1 1 0 1 0.4 0.1 0.2 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2
0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.3 0.7 0.7 0.4 0.4 0.9 0.4
0.8 0.1 0 1 0.1 0.1 0.1 0.1 0.1 0.1 0.3 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1
0 0.6 0.8 0.7 0.5 0.6 0.8 0.8 0.9 0.2 0 1 1 0.7 0.7 0.4 0.5 1 0.5 0.7 0.2 0 1 0.1 0.1 0.1

0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.8 1 0.4 0.4 0.8 1 0.8 0.2 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.2
0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.8 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1
0 0.1 0.1 0.4 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.3 0.1
0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.1
0.35 0.1 0.1 0.1 1 0 0.8 1 1 0.8 0.7 1 0.9 0.7 0.1 0 1 1 0.5 0.8 1 0.3 1 0.5 0.1 0.3 0 1
1 1 1 0.2 1 1 0.5 0.3 0.3 0 1 1 0.8 0.8 0.2 0.8 1 0.4 0.8 1 0 1 1 0.5 0.5 0.3 0.6 0.7 0.7
1 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.2 0.1 0.1 0.2 0.2 0.1 0.1 0.1 0.1 1 0
0.1 0.1 0.2 0.2 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.1 0.1
0.1 0.2 0.35 0.3 0.1 0.1 1 0 0.4 0.2 0.3 0.5 0.3 0.8 0.7 0.6 0.1 0 1 0.2 0.1 0.1 0.1 0.1
0.1 0.1 0.1 0.1 1 0 1 1 1 1 0.6 1 0.8 0.1 0.5 0 1 0.8 0.6 0.5 0.4 0.3 1 0.6 0.1 0.1 0 1 0.4
0.1 0.1 0.1 0.2 0.2 0.3 0.2 0.1 1 0 0.5 0.1 0.1 0.6 0.3 0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1
0.2 0.1 0.2 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2
0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.3 0.1 0.2 0.1 0.2 0.1 0.1 1 0
0.3 0.4 0.4 1 0.5 0.1 0.3 0.3 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.7 0.8 0.7
0.2 0.4 0.8 0.3 0.8 0.2 0 1 1 0.5 0.7 0.3 0.3 0.7 0.3 0.3 0.8 0 1 0.1 0.1 0.1 0.1 0.2 0.1
0.2 0.1 0.1 1 0 0.2 0.5 0.7 0.6 0.4 1 0.7 0.6 0.1 0 1 0.9 0.8 0.8 0.5 0.6 0.2 0.4 1 0.4 0
1 0.5 1 1 1 0.4 1 0.5 0.6 0.3 0 1 0.3 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.1 0.1 0.1 0.1
0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.5 0.5 0.8 1 0.8 0.7 0.3 0.7 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.1
0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.2 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1
0 0.1 0.2 0.2 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.4 0.3 0.1 0.1 0.2 0.1 0.4 0.8 0.1 1 0 0.3 0.3
0.2 0.6 0.3 0.3 0.3 0.5 0.1 1 0 0.3 1 0.8 0.7 0.6 0.9 0.9 0.3 0.8 0 1 0.8 0.3 0.4 0.9 0.3
1 0.3 0.3 0.1 0 1 1 0.8 0.4 0.4 0.4 1 0.3 1 0.4 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1
0 0.8 0.4 0.6 0.3 0.3 0.1 0.4 0.3 0.1 1 0 0.5 0.6 0.6 0.8 0.6 1 0.4 1 0.4 0 1 0.1 0.1 0.1
0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.1 0.1 0.1 0.2 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.2
0.2 0.1 0.1 1 0 0.5 0.1 0.3 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.8 0.3 0.5 0.4 0.5 1 0.1 0.6 0.2 0
1 0.5 1 1 0.3 0.8 0.1 0.5 1 0.3 0 1 1 1 1 0.7 1 1 0.8 0.2 0.1 0 1 0.4 0.2 0.2 0.1 0.2 0.1
0.2 0.1 0.1 1 0 0.2 0.3 0.1 0.1 0.5 0.1 0.1 0.1 0.1 1 0 0.5 0.2 0.2 0.2 0.2 0.1 0.1 0.1 0.2
1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.2 0.2 0.1 0.1 1 0 0.3 1
0.7 0.8 0.5 0.8 0.7 0.4 0.1 0 1 0.3 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.4 0.5 0.1 0.8
0.1 0.3 0.6 0.1 1 0 0.5 0.1 0.1 0.1 0.1 0.1 0.3 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.2 0.1
0.1 1 0 0.5 1 0.6 0.1 1 0.4 0.4 1 1 0 1 0.5 0.1 0.1 0.1 0.3 0.2 0.2 0.2 0.1 1 0 0.8 0.4 0.4
0.1 0.2 0.9 0.3 0.3 0.1 0 1 0.5 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.2 1 0 0.4 0.8 0.8 0.5 0.4 0.5

1 0.4 0.1 0 1 0.9 0.1 0.2 0.6 0.4 1 0.7 0.7 0.2 0 1 0.8 0.5 0.6 0.2 0.3 1 0.6 0.6 0.1 0 1
0.1 0.1 0.1 0.1 0.1 0.1 0.3 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.3 0.3 0.1 0.1 1 0 0.4 1 0.8
0.5 0.4 0.1 1 0.1 0.1 0 1 0.2 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.1 0.3 0.1 0.2 0.1
0.3 0.1 0.1 1 0 1 0.3 0.5 0.1 1 0.5 0.3 1 0.2 0 1 0.4 0.6 0.5 0.6 0.7 0.35 0.4 0.9 0.1 1
0 0.3 0.1 0.1 0.2 0.3 0.4 0.1 0.1 0.1 1 0 0.5 0.2 0.2 0.2 0.2 0.1 0.2 0.2 0.1 1 0 0.5 0.5
0.5 0.6 0.3 1 0.3 0.1 0.1 0 1 0.5 0.6 0.7 0.8 0.8 1 0.3 1 0.3 0 1 0.6 1 1 1 0.8 1 1 1 0.7 0
1 0.4 0.2 0.2 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.3 0.2 0.2 0.2 0.2 0.1 0.4 0.2 0.1 1 0 0.5 0.1
0.1 0.6 0.3 0.1 0.2 0.1 0.1 1 0 0.7 0.8 0.3 0.7 0.4 0.5 0.7 0.8 0.2 0 1 0.1 0.1 0.1 0.1 0.2
0.1 0.2 0.1 0.2 1 0 0.8 0.4 0.4 0.5 0.4 0.7 0.7 0.8 0.2 1 0 0.5 0.6 0.5 0.6 1 0.1 0.3 0.1
0.1 0 1 0.7 0.9 0.4 1 1 0.3 0.5 0.3 0.3 0 1 0.3 0.3 0.2 0.2 0.3 0.1 0.1 0.2 0.3 1 0 0.2
0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.2 0.1 0.1 0.1
0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.3
0.2 0.1 1 0 0.5 1 0.8 1 0.8 1 0.3 0.6 0.3 0 1 0.5 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.4
0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.8 0.7 0.8 0.5
1 1 0.7 0.2 0.1 0 1 0.1 0.1 0.1 0.1 0.1 0.35 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.2
0.3 0.1 1 0 0.2 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.1 0.2 0.2 0.2 0.3 0.1 0.1 1
0 0.3 0.1 0.1 0.3 0.8 0.1 0.5 0.8 0.1 1 0 0.2 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.7 0.6
0.3 0.2 0.5 1 0.7 0.4 0.6 0 1 0.4 1 0.4 0.7 0.3 1 0.9 1 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1
0.3 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1 1 0.1 0.1 0.1 0.1
1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.3 0.4 0.1 0.4 0.1 0.3 0.1 0.1 1 0 1 0.5
0.5 0.6 0.8 0.8 0.7 0.1 0.1 0 1 1 1 1 0.4 0.8 0.1 0.8 1 0.1 0 1 0.8 1 1 1 0.6 1 1 1 1 0 1
0.3 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.6 1 0.5 0.5 0.4 1 0.6 1 0.1 0 1 0.8 0.3 0.8 0.3
0.4 0.9 0.8 0.9 0.8 0 1 0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 1 0.4 0.3 0.1 0.3 0.3 0.6
0.5 0.2 0 1 0.3 0.1 0.1 0.1 0.2 0.2 0.3 0.1 0.1 1 0 0.3 0.1 0.3 0.1 0.3 0.4 0.1 0.1 0.1 1
0 0.2 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.5 1 0 0.8 0.2 0.1 0.1 0.5 0.1 0.1 0.1 0.1 1 0 0.4 0.1
0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 1 0.3 0.5 0.4 0.3 0.7 0.3 0.5 0.3 0 1 0.1 0.1 0.1 0.1 0.2
0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 1 0.3 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1
0.1 1 0 1 0.4 0.3 0.2 0.3 1 0.5 0.3 0.2 0 1 0.2 0.5 0.3 0.3 0.6 0.7 0.7 0.5 0.1 0 1 0.3
0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.6 0.6 0.7 1
0.3 1 0.8 1 0.2 0 1 0.2 0.1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 1 0 0.3 0.3 0.2 0.1 0.2 0.3 0.3
0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.4 0.4 0.4 0.4 0.6 0.5 0.7 0.3 0.1 1
0 0.2 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.3 0.2 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.1

0.2 1 0.4 0.5 0.2 0.1 0.1 1 0 0.2 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2
0.1 0.1 0.1 0.1 1 0 0.3 0.1 0.4 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2
0.1 0.1 1 0 0.1 0.1 0.3 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1
0 0.3 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.3 0.1 0.2 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1
0.1 0.3 0.2 0.1 0.1 0.1 0.1 1 0 0.3 0.2 0.2 0.3 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.3 0.3 0.1 0.3
0.3 0.3 0.3 0.3 0 1 0.6 0.6 0.6 0.5 0.4 1 0.7 0.6 0.2 0 1 0.1 0.1 0.1 0.1 0.5 0.1 0.3 0.1
0.1 1 0 0.5 0.8 0.8 0.8 0.5 1 0.7 0.8 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.5
0.1 0.1 0.1 0.2 0.1 0.2 0.2 0.1 1 0 0.4 0.5 0.5 1 0.4 1 0.7 0.5 0.8 0 1 0.3 0.1 0.1 0.4 0.3
0.1 0.2 0.2 0.1 1 0 1 0.8 1 1 0.6 0.1 0.3 0.1 1 0 1 0.6 0.2 0.1 0.1 0.1 0.1 0.7 0.1 0.1 1 0
0.1 0.6 0.8 1 0.8 1 0.5 0.7 0.1 0 1 0.4 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.9 0.5 0.5 0.2
0.2 0.2 0.5 0.1 0.1 0 1 0.3 0.1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 1 0 0.3 0.6 0.6 0.6 0.5 1 0.6
0.8 0.3 0 1 0.2 0.1 0.1 0.1 0.1 0.1 0.3 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1
0 1 1 0.9 0.3 0.7 0.5 0.3 0.5 0.1 0 1 0.4 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.3 0.1 0.1
0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.3 0.1 0.1 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1
0.2 0.3 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 1 0.3 0.4 0.5 0.3 1 0.4 0.1 0.1 0 1
0.3 0.1 0.1 0.3 0.2 0.1 0.1 0.1 0.1 1 0 0.7 0.4 0.4 0.3 0.4 1 0.6 0.9 0.1 0 1 0.8 0.8 0.8
0.1 0.2 0.35 0.6 1 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.3 0.1 0.2 0.1 0.2 0.1
0.2 0.1 0.1 1 0 0.5 1 1 1 0.6 1 0.6 0.5 0.2 0 1 0.6 0.8 0.7 0.8 0.6 0.8 0.8 0.9 0.1 0 1 0.3
0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.6 0.3 0.3 0.5 0.3 1 0.3 0.5 0.3 1 0 0.5 0.1 0.1 0.1
0.2 0.2 0.3 0.3 0.1 1 0 0.8 0.6 0.7 0.3 0.3 1 0.3 0.4 0.2 0 1 0.5 0.3 0.3 0.2 0.3 0.1 0.3
0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1
0 0.4 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.2 0.1 0.1 0.1 0.3 0.1 0.2 0.1 0.1 1 0 0.1 0.1
0.3 0.1 0.1 0.1 0.2 0.1 0.1 1 0 0.8 1 0.8 0.8 0.4 0.8 0.7 0.7 0.1 0 1 0.7 0.1 0.2 0.3 0.2
0.1 0.2 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.3
0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.3 0.1 1
0 0.6 1 1 0.2 0.8 1 0.7 0.3 0.3 0 1 0.5 0.1 0.2 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.7 0.5 0.6 1
0.4 1 0.5 0.3 0.1 0 1 0.3 0.1 0.1 0.1 0.2 0.35 0.3 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2
0.1 0.1 1 0 0.5 0.3 0.5 0.5 0.3 0.3 0.4 1 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0
0.3 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.7 0.9 0.8 0.6 1 0.8 1 0.1 0 1 1 0.4 0.2 0.1
0.3 0.2 0.4 0.3 1 0 1 0.8 1 1 0.8 0.5 1 0.7 0.8 0.1 0 1 0.8 1 0.3 0.2 0.6 0.4 0.3 1 0.1 0
1 1 0.1 0.1 0.1 0.2 1 0.5 0.4 0.1 0 1 0.2 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.3 0.6 0.4
1 0.3 0.3 0.3 0.4 0.1 0 1 0.3 0.1 0.1 0.1 0.2 0.2 0.7 0.1 0.1 1 0 0.3 0.3 0.1 0.1 0.2 0.1

0.1 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.7 0.4 0.7 0.4 0.3 0.7 0.7 0.6 0.1
0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.9 1 1 0.1 1 0.8 0.3 0.3 0.1 0 1 0.8 0.7 0.8
0.7 0.5 0.5 0.5 1 0.2 0 1 0.4 0.2 0.4 0.3 0.2 0.2 0.2 0.1 0.1 1 0 1 0.6 0.4 0.1 0.3 0.4 0.3
0.2 0.3 0 1 0.7 0.4 0.6 0.4 0.6 0.1 0.4 0.3 0.1 0 1 0.1 0.1 0.1 0.1 0.1 0.1 0.3 0.1 0.1 1
0 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.5 0.1
0.3 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 1 0.4 0.7 0.2 0.2
0.8 0.6 0.1 0.1 0 1 0.5 0.7 0.4 0.1 0.6 0.1 0.7 1 0.3 0 1 0.3 0.1 0.1 0.1 0.2 0.1 0.1 0.1
0.1 1 0 0.5 0.3 0.4 0.1 0.8 1 0.4 0.9 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.3
0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.6 0.9 0.7 0.5 0.5 0.8 0.4 0.2 0.1 1 0 0.2 0.1 0.1 0.1
0.2 0.1 0.2 0.2 0.1 1 0 1 1 1 1 1 0.1 0.8 0.8 0.8 0 1 0.3 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1
0 0.4 0.1 0.2 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.6 1 1
1 1 1 0.8 1 1 0 1 1 1 0.8 0.6 0.4 0.5 0.8 1 0.1 0 1 0.5 0.3 0.3 0.3 0.2 0.3 0.4 0.4 0.1 0
1 0.1 0.1 0.3 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.3 0.1 0.2 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1
0.3 0.1 0.2 0.35 0.2 0.1 0.1 1 0 1 0.3 0.6 0.2 0.3 0.5 0.4 1 0.2 0 1 0.5 0.1 0.1 0.3 0.2
0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.5 0.5 0.1 0.1 1 0 0.1 0.3 0.3 0.2 0.2 0.1 0.7
0.2 0.1 1 0 0.1 0.2 0.2 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.2 0.1 0.2 0.1 0.1 0.1 0.1 1
0 0.1 0.1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.7 0.5
0.6 0.3 0.3 0.8 0.7 0.4 0.1 0 1 0.1 0.3 0.1 0.2 0.2 0.2 0.5 0.3 0.2 1 0 0.4 0.1 0.1 0.1 0.3
0.1 0.1 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.7 0.2 0.4 0.1 0.6 1 0.5 0.4
0.3 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 1 1 1 1 0.3 1 1 0.6 0.1 0 1 0.5 0.4 0.4
0.5 0.7 1 0.3 0.2 0.1 1 0 0.4 0.1 0.1 0.3 0.1 0.5 0.2 0.1 0.1 0 1 1 1 1 0.6 0.8 0.4 0.8
0.5 0.1 0 1 1 1 0.8 1 0.6 0.5 1 0.3 0.1 0 1 0.3 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.5
0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.3 0.1 0.2 0.2 0.1 1 0 0.2 0.1 0.1 0.2
0.2 0.1 0.3 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.1 0.1 0.2 0.1 0.1 0.2
0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.3 0.3 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1
0 1 0.4 0.6 0.4 0.5 1 0.7 0.1 0.1 0 1 0.8 0.8 0.9 0.4 0.5 1 0.7 0.8 0.1 0 1 0.8 1 1 0.7 1
1 0.7 0.3 0.8 0 1 0.5 0.7 0.7 0.1 0.5 0.8 0.3 0.4 0.1 1 0 1 1 1 0.1 0.6 0.1 0.2 0.8 0.1 0
1 0.2 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 1 0.3 0.3 1 0.2 1 0.7 0.3 0.3 0 1 0.6 0.8 0.8
0.1 0.3 0.4 0.3 0.7 0.1 1 0 0.8 1 1 1 0.8 1 1 0.7 0.3 0 1 0.3 0.1 0.1 0.1 0.2 0.35 0.3 0.1
0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.2 0.2 0.1 1 0 0.1
0.1 0.1 0.1 0.2 0.4 0.1 0.1 0.1 1 0 0.5 0.4 0.6 0.7 0.9 0.7 0.8 1 0.1 0 1 0.1 0.1 0.1 0.2
0.2 0.1 0.3 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.1

0.1 0.1 1 0 0.8 0.2 0.4 0.1 0.5 0.1 0.5 0.4 0.4 0 1 0.1 0.2 0.3 0.1 0.2 0.1 0.1 0.1 0.1 1
0 1 0.7 0.7 0.6 0.4 1 0.4 0.1 0.2 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.3 0.2
0.1 0.3 0.1 0.1 0.1 0.1 1 0 1 1 1 1 1 1 0.4 1 1 0 1 0.1 0.1 0.1 0.1 0.2 0.2 0.1 0.1 0.1 1
0 0.5 0.1 0.2 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.4 0.4 0.4 0.2 0.2 0.3 0.2 0.1 0.1 1 0 0.6 0.2
0.3 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.8 0.4 1 0.5 0.4 0.4 0.7 1 0.1 0 1 0.1 0.2 0.3 0.1 0.2 0.1
0.3 0.1 0.1 1 0 0.3 0.2 0.2 0.1 0.2 0.1 0.2 0.3 0.1 1 0 1 0.5 0.7 0.4 0.4 1 0.8 0.9 0.1 0
1 0.6 0.3 0.3 0.3 0.3 0.2 0.6 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 1 1 1
0.8 0.6 0.8 0.7 1 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.5 0.1 0.1 0.1 1 0 1 0.5 1 0.3 0.5 0.8 0.7
0.8 0.3 0 1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 1 0 0.8 0.9 0.9 0.5 0.3 0.5 0.7 0.7 0.1 0
1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.3 0.3
0.5 0.2 0.3 1 0.7 0.1 0.1 0 1 0.1 0.2 0.1 0.3 0.2 0.1 0.2 0.1 0.1 1 0 0.4 0.1 0.3 0.3 0.2
0.1 0.1 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.3 0.1 0.1 0.1 1 0 0.7 0.5 1 1 1 1 0.4 1 0.3 0 1
0.2 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.3 0.2 0.2 0.2 0.2 0.1 0.3 0.2 0.1 1 0 0.3 0.1 0.1
0.1 0.1 0.1 0.2 0.1 0.1 1 0 0.5 0.3 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.8 0.8 0.9 0.6 0.6 0.3
1 1 0.1 0 1 0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.2 0.1 1 0 1 0.5 0.5 0.6 0.3 1 0.7 0.9 0.2 0 1
0.4 0.1 0.1 0.1 0.2 0.1 0.3 0.6 0.1 1 0 1 0.4 0.5 0.4 0.3 0.5 0.7 0.3 0.1 0 1 0.5 0.3 0.5
0.1 0.8 1 0.5 0.3 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.7 0.8 0.8 0.7 0.3 1 0.7
0.2 0.3 0 1 0.2 0.1 0.1 0.1 0.2 0.5 0.1 0.1 0.1 1 0 0.6 0.6 0.6 0.9 0.6 0.35 0.7 0.8 0.1 1
0 0.4 0.5 0.5 0.8 0.6 1 1 0.7 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.3 0.1 0.1
0.1 0.2 0.1 0.2 0.1 0.1 1 0 1 1 1 0.3 1 1 0.9 1 0.1 0 1 0.8 1 0.5 0.3 0.8 0.4 0.4 1 0.3 0 1
0.5 0.1 0.3 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.5 0.4 0.4 0.9 0.2 1 0.5 0.6 0.1 0 1 0.5 0.5 0.5
0.2 0.5 1 0.4 0.3 0.1 0 1 0.3 0.1 0.1 0.1 0.3 0.2 0.1 0.1 0.1 1 0 1 0.7 0.7 0.3 0.8 0.5 0.7
0.4 0.3 0 1 0.4 0.8 0.7 1 0.4 1 0.7 0.5 0.1 0 1 0.5 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.3
0.4 0.5 0.3 0.7 0.3 0.4 0.6 0.1 1 0 0.6 0.1 0.1 0.3 0.2 0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1
0.2 0.1 0.3 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.4 0.1
0.1 0.1 1 0 0.5 0.4 0.6 0.6 0.4 1 0.4 0.3 0.1 0 1 0.4 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0
0.5 0.1 0.2 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.3 0.2 0.1 0.1 0.2 0.1 0.2 0.2 0.1 1 0 0.1 0.1 0.1
0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.4 0.8 0.6 0.4 0.3 0.4 1 0.6 0.1 0 1 0.5 0.2 0.2 0.2 0.3 0.1
0.1 0.3 0.1 1 0 1 0.6 0.6 0.2 0.4 1 0.9 0.7 0.1 0 1 0.3 0.2 0.2 0.1 0.4 0.3 0.2 0.1 0.1 1
0 0.6 1 0.2 0.8 1 0.2 0.7 0.8 1 0 1 1 0.4 0.4 1 0.2 1 0.5 0.3 0.3 0 1 0.1 0.1 0.1 0.3 0.2
0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.1
0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 1

0.4 0.3 1 0.3 1 0.7 0.1 0.2 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.3 0.1 0.1 0.3 0.2
0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.1 0.1
0.1 1 0 0.4 0.1 0.4 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.8 0.8 0.7 0.4 1 1 0.7 0.8 0.7 0 1 0.1 0.1
0.2 0.1 0.3 0.35 0.1 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.6 0.1 0.1 0.1
0.2 0.1 0.3 0.1 0.1 1 0 0.6 0.3 0.4 0.1 0.5 0.2 0.3 0.9 0.1 0 1 0.4 0.1 0.1 0.3 0.2 0.1 0.3
0.1 0.1 1 0 0.2 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1
0 0.5 0.8 0.7 0.7 1 1 0.5 0.7 0.1 0 1 0.6 0.1 0.3 0.2 0.2 0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1
0.1 0.4 0.3 0.1 0.1 0.1 1 0 0.2 0.1 0.3 0.2 0.2 0.1 0.2 0.1 0.1 1 0 0.5 0.3 0.2 0.4 0.2 0.1
0.1 0.1 0.1 1 0 0.6 0.1 0.3 0.1 0.4 0.5 0.5 1 0.1 0 1 0.4 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1
0 0.2 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.2 0.1
0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.7 0.6 1 0.5 0.3 1 0.9 1 0.2 0 1 0.2 0.1 0.1 0.1 0.2 0.1
0.1 0.1 0.1 1 0 0.8 1 1 0.8 0.6 0.9 0.3 1 1 0 1 0.1 0.1 0.1 0.3 0.1 0.3 0.1 0.1 0.1 1 0 0.3
0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 1 0.4 0.6 0.1
0.2 1 0.5 0.3 0.1 0 1 0.1 0.1 0.1 0.1 0.1 0.1 0.3 0.1 0.1 1 0 0.6 0.1 0.1 0.1 0.1 0.1 0.1
0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.3 0.2 0.2 0.1 0.1 1 0 0.4 0.1 0.1 0.3 0.2 0.1 0.3 0.1 0.1 1
0 0.3 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 1 0 0.5 0.1
0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.2 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 1 0.3 0.3 0.1 0.2 1
0.7 0.6 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.5 0.2 0.3 0.1 0.6 1 0.5 0.1 0.1
0 1 0.4 0.7 0.8 0.3 0.4 1 0.9 0.1 0.1 0 1 0.3 0.2 0.1 0.2 0.2 0.1 0.3 0.1 0.1 1 0 0.1 0.1
0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2
0.1 0.3 0.2 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.2 0.1 0.1 0.1 0.2 0.1 0.2
0.1 0.1 1 0 0.4 0.2 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.5 0.2 0.2 0.2 0.2 0.2 0.3 0.2 0.2 1
0 0.4 0.1 0.1 0.2 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.3 0.4 0.3 0.4 0.5 0.4 0.7 0.1 1 0 0.4 0.1
0.1 0.2 0.2 0.1 0.2 0.1 0.1 1 0 0.3 0.1 0.2 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.7 0.3 0.2 1 0.5
1 0.5 0.4 0.4 0 1 0.4 0.1 0.1 0.1 0.2 0.1 0.3 0.2 0.1 1 0 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
0.1 1 0 0.7 0.5 0.3 0.7 0.4 1 0.7 0.5 0.5 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.8
0.7 0.5 1 0.7 0.9 0.5 0.5 0.4 0 1 0.5 0.1 0.2 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.3 0.1 0.1 0.1
0.1 0.1 0.2 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.5 0.3 0.3 0.4 0.2 0.4 0.3
0.4 0.1 0 1 1 0.6 0.6 0.3 0.4 0.5 0.3 0.6 0.1 0 1 0.2 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0
0.1 0.1 0.1 0.1 0.1 0.1 0.3 0.1 0.1 1 0 0.5 1 1 0.5 0.4 0.5 0.4 0.4 0.1 0 1 0.4 0.1 0.1 0.1
0.2 0.1 0.3 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.1 0.2 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.1
0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.6 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1

0 0.8 0.4 0.7 0.1 0.3 1 0.3 0.9 0.2 0 1 0.8 1 1 1 0.7 0.5 0.4 0.8 0.7 0 1 1 0.6 0.4 0.3 1
1 0.9 1 0.1 0 1 1 0.8 0.7 0.4 0.3 1 0.7 0.9 0.1 0 1 1 0.8 1 0.1 0.3 1 0.5 0.1 0.1 0 1 0.8
0.7 0.6 0.4 0.4 1 0.5 0.1 0.1 0 1 0.8 0.2 0.3 0.1 0.6 0.3 0.7 0.1 0.1 0 1 0.1 0.1 0.1 0.1
0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.1 0.1 0.2
0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.3 0.3 0.2 0.2 0.2 0.3 0.1 1
0 1 1 1 1 0.7 1 0.7 1 0.4 0 1 0.4 0.1 0.2 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.5 0.8 0.6 0.5
0.8 0.7 1 0.1 0 1 0.4 0.1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1
0.1 1 0 0.5 0.5 0.7 0.8 0.6 1 0.7 0.4 0.1 0 1 0.1 0.1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 1 0 0.4
0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.1 0.1 0.1 0.1
0.2 0.5 0.1 0.1 0.1 1 0 1 0.9 0.7 0.3 0.4 0.2 0.7 0.7 0.1 0 1 0.2 0.1 0.1 0.1 0.2 0.1 0.3
0.1 0.1 1 0 0.5 0.8 0.9 0.4 0.3 1 0.7 0.1 0.1 0 1 0.1 0.1 0.1 0.3 0.2 0.3 0.1 0.1 0.1 1 0
0.4 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.4 0.2 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.4 0.3
0.1 0.2 0.35 0.2 0.3 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.3 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2
0.1 0.3 0.1 0.1 1 0 1 0.6 0.3 0.6 0.4 1 0.7 0.8 0.4 0 1 0.5 0.2 0.2 0.2 0.1 0.1 0.2 0.1 0.1
1 0 0.3 0.1 0.4 0.1 0.2 0.35 0.3 0.1 0.1 1 0 0.5 1 1 0.6 1 1 1 0.6 0.5 0 1 0.4 0.1 0.1 0.1
0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.3 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.2 0.1 0.3 0.1
0.1 0.7 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.2 0.2 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1
0 0.3 0.1 0.1 0.1 0.3 0.1 0.2 0.1 0.1 1 0 1 0.2 0.2 0.1 0.2 0.6 0.1 0.1 0.2 0 1 0.5 1 1 1
1 1 1 0.1 0.1 0 1 0.5 0.1 0.1 0.3 0.2 0.1 0.1 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.2 0.1
0.1 1 0 0.5 0.1 0.1 0.3 0.4 0.1 0.3 0.2 0.1 1 0 0.7 0.6 0.4 0.8 1 1 0.9 0.5 0.3 0 1 0.2 0.1
0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.5 0.2 0.3 0.4 0.2 0.7 0.3 0.6 0.1 0 1 0.9 1 1 0.1 1 0.8
0.3 0.3 0.1 0 1 0.4 0.1 0.2 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.6 0.5 0.5 0.8 0.4 1 0.3 0.4 0.1
0 1 1 0.4 0.4 0.6 0.2 1 0.2 0.3 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 1 0.8 0.8
0.2 0.3 0.4 0.8 0.7 0.8 0 1 1 0.4 0.4 1 0.6 1 0.5 0.5 0.1 0 1 0.8 0.6 0.4 0.3 0.5 0.9 0.3
0.1 0.1 0 1 0.7 0.6 0.6 0.3 0.2 1 0.7 0.1 0.1 0 1 0.2 0.1 0.2 0.1 0.2 0.1 0.3 0.1 0.1 1 0
0.3 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.2 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.9 0.9 1
0.3 0.6 1 0.7 1 0.6 0 1 0.3 0.1 0.1 0.1 0.2 0.1 0.3 0.2 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.3
0.1 0.1 1 0 0.5 1 1 1 0.5 0.2 0.8 0.5 0.1 0 1 1 1 1 0.8 0.6 0.1 0.8 0.9 0.1 0 1 0.3 0.1 0.1
0.1 0.3 0.1 0.2 0.1 0.1 1 0 0.5 0.8 0.8 1 0.5 1 0.8 1 0.3 0 1 0.1 0.1 0.3 0.1 0.2 0.1 0.1
0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.3 0.2 0.1 1 0 0.6 1 0.7 0.7 0.6 0.4 0.8 1 0.2 0 1 0.3
0.1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 1 0 1 1 1 0.7 0.9 1 0.7 1 1 0 1 0.5 0.3 0.2 0.8 0.5 1 0.8
0.1 0.2 0 1 1 1 0.6 0.3 0.3 1 0.4 0.3 0.2 0 1 0.3 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.9

0.4 0.5 1 0.6 1 0.4 0.8 0.1 0 1 0.1 0.1 0.1 0.1 0.1 0.35 0.2 0.1 0.1 1 0 0.4 0.1 0.1 0.1
0.2 0.1 0.1 0.1 0.1 1 0 0.8 0.3 0.3 0.1 0.2 0.2 0.3 0.2 0.1 1 0 0.8 0.4 0.4 0.1 0.6 1 0.2
0.5 0.2 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 1 0
0.2 0.3 0.4 0.4 0.2 0.5 0.2 0.5 0.1 0 1 0.3 0.2 0.1 0.1 0.1 0.1 0.2 0.1 0.1 1 0 0.3 0.3 0.6
0.4 0.5 0.8 0.4 0.4 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.35 0.2 0.1 0.1 1 0 0.5 0.2 0.1 0.1 0.2
0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1
0.1 1 0 0.3 0.1 0.3 0.1 0.2 0.35 0.2 0.1 0.1 1 0 1 1 1 1 0.5 1 1 1 0.7 0 1 0.5 0.1 0.2 0.1
0.2 0.1 0.3 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.1 0.1 0.2
0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.4 0.1 0.3 0.1 0.2 0.1 0.2 0.1 0.1 1
0 1 1 1 0.3 1 0.8 0.8 0.1 0.1 0 1 0.4 0.1 0.1 0.3 0.1 0.1 0.2 0.1 0.1 1 0 0.3 0.1 0.1 0.1
0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.3
0.1 0.1 1 0 0.5 0.2 0.4 0.1 0.1 0.1 0.1 0.1 0.1 1 0 0.2 0.3 0.2 0.2 0.2 0.2 0.3 0.1 0.1 1
0 0.5 1 1 0.3 0.7 0.3 0.8 1 0.2 0 1 0.8 0.4 0.5 0.1 0.2 0.35 0.7 0.3 0.1 0 1 0.9 1 1 1 1 1
1 1 0.1 0 1 0.4 0.4 0.2 0.1 0.2 0.5 0.2 0.1 0.2 1 0 0.7 0.2 0.4 0.1 0.3 0.4 0.3 0.3 0.1 0
1 0.4 0.1 0.1 0.2 0.2 0.1 0.2 0.1 0.1 1 0 0.8 0.6 0.4 1 1 0.1 0.3 0.5 0.1 0 1 0.5 0.1 0.1
0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.7 0.4 0.5 1 0.2 1 0.3 0.8 0.2 0 1 0.5 0.2 0.2 0.4 0.2 0.4 0.1
0.1 0.1 1 0 0.3 0.2 0.2 0.3 0.2 0.3 0.3 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0
0.6 0.1 0.3 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 1 0.4 0.5
0.5 0.5 1 0.4 0.1 0.1 0 1 0.5 0.1 0.1 0.3 0.2 0.1 0.1 0.1 0.1 1 0 0.6 0.1 0.1 0.1 0.2 0.1
0.3 0.1 0.1 1 0 0.4 0.1 0.1 0.3 0.2 0.1 0.1 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1
1 0 0.5 1 1 0.8 0.5 0.5 0.7 1 0.1 0 1 1 0.6 0.5 0.8 0.5 1 0.8 0.6 0.1 0 1 0.2 0.1 0.1 0.1
0.3 0.1 0.2 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.1 0.1 0.2 0.1 0.2 0.1 0.2
0.1 0.1 1 0 0.1 0.3 0.1 0.1 0.2 0.1 0.2 0.2 0.1 1 0 0.2 0.2 0.2 0.1 0.1 0.1 0.7 0.1 0.1 1
0 0.9 0.8 0.8 0.9 0.6 0.3 0.4 0.1 0.1 0 1 0.5 0.1 0.4 0.1 0.2 0.1 0.3 0.2 0.1 1 0 0.1 0.1
0.1 0.1 0.1 0.1 0.3 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 1 1 0.7 0.8 0.7 0.1
1 1 0.3 0 1 0.2 0.1 0.1 0.2 0.2 0.1 0.3 0.1 0.1 1 0 0.1 0.2 0.1 0.3 0.2 0.1 0.1 0.2 0.1 1
0 0.5 0.3 0.6 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.1 0.1 0.3 0.1 0.1 1 0 0.2 0.1
0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.3 0.1 0.1 0.3 0.1 0.1 0.3 0.1 0.1 1 0 0.5 1 1 0.9 0.6 1
0.7 1 0.5 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1
1 0 0.5 0.6 0.6 0.2 0.4 1 0.3 0.6 0.1 0 1 0.3 0.1 0.2 0.2 0.2 0.1 0.1 0.1 0.1 1 0 0.9 0.6
0.9 0.2 1 0.6 0.2 0.9 1 0 1 0.5 0.7 1 0.6 0.5 1 0.7 0.5 0.1 0 1 0.6 1 1 0.2 0.8 1 0.7 0.3
0.3 0 1 0.9 1 1 1 1 0.5 1 1 1 0 1 0.6 0.5 0.4 0.4 0.3 0.9 0.7 0.8 0.3 0 1 1 1 1 0.8 0.2

1 0.4 0.1 0.1 0 1 0.1 0.2 0.3 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.9 0.5 0.5 0.4 0.4 0.5 0.4 0.3 0.3 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.3 0.1 0.1 0.2 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.1 0.1 0.3 0.2 0.1 0.1 0.1 0.1 1 0 0.4 0.1 0.2 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.8 1 1 1 0.6 1 1 1 0.1 0 1 0.4 0.2 0.1 0.1 0.2 0.2 0.3 0.1 0.1 1 0 0.2 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.7 0.3 0.4 0.4 0.3 0.3 0.3 0.2 0.7 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.6 0.1 0.1 0.3 0.2 0.1 0.1 0.1 0.1 1 0 0.3 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.7 0.8 0.7 0.6 0.4 0.3 0.8 0.8 0.4 0 1 0.8 0.7 0.8 0.2 0.4 0.2 0.5 1 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.6 1 1 1 0.4 1 0.7 1 0.1 0 1 1 0.4 0.3 1 0.4 1 1 0.1 0.1 0 1 0.4 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.4 0.6 0.6 0.5 0.7 0.6 0.7 0.7 0.3 0 1 0.3 0.1 0.1 0.2 0.2 0.1 0.1 0.1 0.1 1 0 0.1 0.2 0.2 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.5 0.8 0.4 1 0.5 0.8 0.9 1 0.1 0 1 0.5 0.4 0.6 1 0.2 1 0.4 0.1 0.1 0 1 0.8 0.7 0.8 0.5 0.5 1 0.9 1 0.1 0 1 0.3 0.1 0.1 0.1 0.2 0.5 0.5 0.1 0.1 1 0 0.3 0.1 0.2 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.9 0.5 0.8 0.1 0.2 0.3 0.2 0.1 0.5 0 1 0.5 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.5 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.5 0.2 0.1 0.1 0.2 0.1 0.3 0.1 0.1 1 0 0.5 0.1 0.1 0.4 0.2 0.1 0.3 0.1 0.1 1 0 0.1 0.1 0.2 0.1 0.2 0.2 0.4 0.2 0.1 1 0 0.9 0.7 0.7 0.5 0.5 1 0.7 0.8 0.3 0 1 0.6 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.3 1 0.3 1 0.6 1 0.5 0.1 0.4 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.1 0.4 0.3 1 0.4 1 0.5 0.6 0.1 0 1 0.2 0.1 0.1 0.2 0.3 0.1 0.2 0.1 0.1 1 0 0.4 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 0.6 0.3 0.2 0.1 0.3 0.4 0.4 0.1 0.1 0 1 0.5 0.1 0.2 0.1 0.2 0.1 0.1 0.1 0.1 1 0 0.7 0.5 0.6 1 0.5 1 0.7 0.9 0.4 0 1 0.6 1 1 1 0.8 1 0.7 1 0.7 0 1 0.5 0.7 1 1 0.5 1 1 1 0.1 0 1 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0

2.2. Diyabet Hastalığı Verileri

0.176471 0.605 0.42623 0 0 0.536513 0.0209223 0.0666667 1 0 0.352941 0.72 0.590164 0.27 0.269504 0.505216 0.0755764 0.316667 0 1 0.117647 0.875 0.721311 0 0 0.341282 0.105892 0.0166667 0 1 0.705882 0.605 0.639344 0.17 0 0.394933 0.0772844 0.683333 0 1 0.117647 0.535 0.606557 0.3 0.118203 0.500745 0.139197 0.0333333 0 1 0.882353 0.68 0.57377 0.32 0.130024 0.552906 0.0320239 0.366667 1 0 0.470588 0.545 0.622951 0.39 0.134752 0.415797 0.239966 0.166667 1 0 0.117647 0.405 0.491803 0.22 0 0.412817 0.0905209 0.0666667 0 1 0.235294 0.985 0.57377 0.39 0.879433 0.546945 0.961144 0.166667 0 1 0 0.525 0.737705 0 0 0.441133 0.0508113 0.416667 0 1

0.411765 0.545 0.655738 0.31 0 0.535022 0.447908 0.366667 1 0 0.235294 0.415 0.704918 0.19 0 0.436662 0.10205 0.216667 0 1 0.352941 0.77 0.606557 0.32 0.228132 0.436662 0.324936 0.3 0 1 0.176471 0.79 0.622951 0.36 0.289598 0.470939 0.33006 0.116667 1 0 0.647059 0.69 0.622951 0 0 0.494784 0.146029 0.233333 0 1 0.176471 0.5 0.557377 0.23 0.0957447 0.470939 0.371904 0.116667 0 1 0.176471 0.855 0.590164 0.33 0.159574 0.496274 0.0516652 0.05 1 0 0.235294 0.77 0.508197 0.31 0.335697 0.488823 0.0678907 0.0333333 0 1 0.0588235 0.435 0.639344 0.27 0.0378251 0.515648 0.00982067 0.0166667 0 1 0.0588235 0.485 0.57377 0.15 0 0.271237 0.029462 0 0 1 0.411765 0.51 0.606557 0.4 0.124113 0.554396 0.0538002 0.4 0 1 0.176471 0.845 0.606557 0.19 0.147754 0.445604 0.0811272 0.166667 1 0 0.470588 0.63 0.606557 0.38 0.0886525 0.385991 0.0358668 0.3 0 1 0.470588 0.325 0.590164 0.23 0 0.4769 0.222886 0.35 0 1 0.588235 0.575 0 0 0 0 0.0781383 0.15 1 0 0.117647 0.42 0 0 0 0 0.0964987 0 0 1 0.0588235 0.555 0.770492 0 0 0.488823 0.0798463 0.4 0 1 0 0.545 0.721311 0.3 0 0.484352 0.331768 0.283333 1 0 0.0588235 0.465 0.57377 0.31 0 0.453055 0.101196 0.0333333 0 1 0 0.605 0.540984 0.3 0.195035 0.511177 0.0533732 0.2 1 0 0.117647 0.42 0.409836 0.23 0.0898345 0.453055 0.380017 0 0 1 0.352941 0.74 0.590164 0.35 0 0.500745 0.234415 0.483333 1 0 0 0.69 0.491803 0.35 0.1974 0.515648 0.194705 0 1 0 0.0588235 0.53 0.57377 0.28 0.159574 0.509687 0.0273271 0.0166667 0 1 0 0.505 0.508197 0 0 0.326379 0.110162 0.0666667 0 1 0.117647 0.64 0.52459 0.42 0 0.596125 0.436806 0.05 0 1 0.411765 0.57 0.52459 0 0 0.408346 0.279249 0.216667 1 0 0.176471 0.6 0.57377 0.3 0.159574 0.639344 0.159693 0.15 0 1 0.235294 0.71 0.704918 0 0 0.655738 0.242101 0.0166667 1 0 0.176471 0.64 0.590164 0.25 0.224586 0.482861 0.20111 0.1 1 0 0.235294 0.915 0 0 0 0.423249 0.0572161 0.25 1 0 0.0588235 0.435 0.491803 0.37 0.0886525 0.554396 0.184031 0.0166667 0 1 0.352941 0.49 0.47541 0.33 0.224586 0.506706 0.150299 0.366667 0 1 0.0588235 0.735 0.770492 0.41 0 0.734724 0.119556 0.1 1 0 0.0588235 0.675 0.442623 0 0 0.397914 0.260034 0.683333 0 1 0.0588235 0.655 0.52459 0.14 0.490544 0.353204 0.132792 0 0 1 0.588235 0.645 0.622951 0.28 0.144208 0.535022 0.0862511 0.3 0 1 0 0.81 0.622951 0.56 0.118203 0.792846 0.290777 0.0666667 1 0 0.176471 0.44 0.47541 0.11 0.0638298 0.369598 0.0807003 0.0166667 0 1 0.0588235 0.405 0.606557 0.41 0.0673759 0.690015 0.434671 0.183333 0 1 0 0.475 0.655738 0.45 0.108747 0.543964 0.1076 0.0833333 0 1 0.235294 0.615 0.655738 0.15 0.208038 0.4769 0.15585 0.216667 0 1 0 0.825 0.737705

0.33 0.803783 0.779434 0.149018 0.0333333 0 1 0 0.895 0.737705 0.27 0 0.657228 0.259607 0.0333333 1 0 0.176471 0.41 0.57377 0 0 0.314456 0.132792 0.0666667 0 1 0 0.9 0.639344 0.63 0.0165485 0.885246 1 0.0666667 1 0 0 0.335 0.622951 0 0 0.675112 0.0495303 0.416667 0 1 0 0.625 0.786885 0 0 0.33532 0.0785653 0 0 1 0.117647 0.46 0.508197 0.28 0 0.470939 0.0222032 0.05 0 1 0.529412 0.445 0.508197 0 0 0.33532 0.0273271 0.2 0 1 0.529412 0.85 0.606557 0.31 0 0.655738 0.13877 0.366667 1 0 0.235294 0.515 0.491803 0.33 0.22695 0.357675 0.379163 0.2 0 1 0.235294 0.645 0.491803 0.12 0.27305 0.409836 0.191716 0.166667 0 1 0 0.525 0.688525 0 0 0.415797 0.283091 0.683333 1 0 0.176471 0.865 0.688525 0.33 0.560284 0.532042 0.0768574 0.0166667 1 0 0.117647 0.645 0 0 0 0.57377 0.0964987 0.333333 0 1 0.117647 0.495 0.57377 0.16 0.0520095 0.304024 0.0670367 0.1 0 1 0.0588235 0.45 0.508197 0.12 0.0508274 0.405365 0.214347 0.05 0 1 0.411765 0.795 0.540984 0 0 0.453055 0.130231 0.25 1 0 0.529412 0.78 0.704918 0 0 0.369598 0.0649018 0.533333 1 0 0.0588235 0.365 0.409836 0.1 0 0.342772 0.0725875 0 0 1 0 0.39 0.721311 0.29 0.0472813 0.549925 0.152007 0 0 1 0.411765 0.53 0.754098 0.18 0 0.338301 0.0670367 0.45 0 1 0 0.505 0.52459 0.17 0 0.312966 0.0742955 0 0 1 0.0588235 0.4 0.45082 0 0 0.28465 0.0768574 0 0 1 0 0.825 0.622951 0.43 0.301418 0.71386 0.0772844 0.0833333 0 1 0.176471 0.865 0.672131 0.48 0.549645 0.57228 0.879163 0.0666667 1 0 0.117647 0.45 0.57377 0.17 0 0.406855 0.0029889 0.0166667 0 1 0.235294 0.47 0.532787 0.22 0 0.368107 0.029889 0 0 1 0.176471 0.79 0.52459 0.13 0.457447 0.464978 0.0926558 0.05 0 1 0.294118 0.83 0.622951 0 0 0.681073 0.11187 0.1 1 0 0.235294 0.55 0.622951 0.2 0.118203 0.423249 0.0170794 0.1 0 1 0.0588235 0.84 0.721311 0.29 0 0.52161 0.353117 0.516667 1 0 0.294118 0.58 0.606557 0 0 0.38152 0.0525192 0.15 0 1 0.0588235 0.445 0.196721 0.19 0.0295508 0.414307 0.20538 0 0 1 0.411765 0.685 0.737705 0.41 0 0.4769 0.133646 0.3 0 1 0.176471 0.65 0.52459 0 0 0.344262 0.100769 0.0166667 0 1 0 0.465 0.491803 0 0 0.52608 0.0789923 0.0666667 0 1 0 0.51 0.639344 0.4 0.106383 0.514158 0.0683177 0.05 0 1 0.294118 0.44 0.540984 0.21 0.0271868 0.363636 0.112724 0.15 0 1 0.0588235 0.46 0.508197 0.25 0.0484634 0.290611 0.172502 0.0666667 0 1 0.0588235 0.465 0.459016 0.11 0 0.33532 0.144748 0.0166667 0 1 0.0588235 0.72 0.672131 0.46 0.212766 0.687034 0.109735 0.416667 1 0 0.470588 0.985 0.606557 0 0 0.385991 0.475235 0.3 1 0 0.294118 0.575 0.622951 0 0 0.464978 0.113151 0.383333 1 0 0.352941 0.495 0.491803 0.19 0.0638298 0.400894 0.178907 0.183333 0 1 0.235294 0.38

0.508197 0 0 0.506706 0.133646 0.0666667 0 1 0.176471 0.935 0.57377 0.22 0.236407 0.542474 0.140905 0.25 1 0 0.0588235 0.45 0.557377 0.08 0 0.365127 0.452605 0.25 0 1 0.176471 0.56 0.606557 0.3 0 0.470939 0.0508113 0.0666667 1 0 0 0.63 0.688525 0.29 0.254137 0.457526 0.188728 0.05 0 1 0.235294 0.735 0.606557 0.25 0.346336 0.520119 0.131085 0.15 0 1 0.117647 0.55 0.606557 0.29 0.147754 0.482861 0.264731 0.1 0 1 0 0.47 0.57377 0.27 0.135934 0.648286 0.114859 0 0 1 0 0.595 0 0 0 0.482861 0.0269001 0.05 1 0 0 0.67 0.47541 0.2 0.343972 0.393443 0.116994 0 0 1 0.235294 0.485 0.491803 0.23 0 0.420268 0.15585 0.0166667 0 1 0.411765 0.75 0.639344 0.29 0.148936 0.52459 0.262169 0.55 1 0 0.0588235 0.485 0.540984 0.15 0.165485 0.345753 0.174637 0.0166667 0 1 0.294118 0.545 0.508197 0.41 0.152482 0.533532 0.186166 0.0666667 1 0 0.529412 0.36 0.639344 0.25 0 0.470939 0.0862511 0.283333 0 1 0.235294 0.67 0.590164 0 0 0.354694 0.0849701 0.65 1 0 0.352941 0.595 0.409836 0.22 0.208038 0.403875 0.529462 0.2 1 0 0.764706 0.38 0.491803 0 0 0.488823 0.0435525 0.333333 0 1 0.588235 0.665 0.557377 0 0 0.402385 0.0713066 0.25 0 1 0 0.465 0.491803 0.25 0.108747 0.42772 0.193851 0.0166667 0 1 0.764706 0.725 0.672131 0.19 0.130024 0.330849 0.0713066 0.6 0 1 0.235294 0.61 0.557377 0 0 0.52161 0.134927 0.133333 0 1 0.0588235 0.545 0.47541 0.18 0.137116 0.424739 0.060205 0.0166667 0 1 0.588235 0.575 0 0 0 0.52608 0.0239112 0.133333 0 1 0.529412 0.51 0.622951 0.37 0 0.490313 0.25064 0.416667 1 0 0.352941 0.48 0 0 0 0.353204 0.0478224 0.116667 0 1 0.176471 0.53 0.442623 0.21 0.186761 0.460507 0.0913749 0.05 0 1 0.117647 0.435 0 0.23 0 0.4307 0.296755 0.0666667 0 1 0.470588 0.425 0.45082 0.2 0 0.363636 0.0247652 0.35 0 1 0.352941 0.975 0.57377 0 0 0.460507 0.106746 0.166667 1 0 0.0588235 0.905 0.639344 0.42 0.346336 0.596125 0.503843 0.0166667 1 0 0.294118 0.365 0.491803 0 0 0.399404 0.0811272 0.1 0 1 0.705882 0.53 0.655738 0 0 0.351714 0.0251921 0.383333 0 1 0.117647 0.455 0.508197 0 0 0.406855 0.190863 0.0166667 0 1 0.0588235 0.535 0.557377 0.19 0 0.394933 0.0371477 0.05 0 1 0.470588 0.5 0.622951 0 0 0.576751 0.0478224 0.35 0 1 0.0588235 0.62 0.491803 0.32 0 0.533532 0.186166 0 0 1 0.117647 0.495 0.491803 0.17 0.189125 0.545455 0.16012 0 0 1 0.294118 0.39 0.393443 0 0 0.502235 0.245944 0.0666667 0 1 0.470588 0.475 0.590164 0 0 0.548435 0.173783 0.6 0 1 0.0588235 0.995 0.622951 0.43 0 0.639344 0.561913 0.0166667 1 0 0.0588235 0.355 0.393443 0.18 0.0898345 0.304024 0.104611 0.0166667 0 1 0.0588235 0.65 0.491803 0.23 0.200946 0.42623 0.262169 0 0 1 0.0588235 0.48 1 0 0 0.33383 0.0550811 0.1

0 1 0.411765 0.905 0.688525 0.21 0.22695 0.535022 0.216909 0.5 1 0 0.117647 0.34 0.508197 0.13 0.0177305 0.299553 0.0764304 0.0333333 0 1 0.470588 0.525 0.819672 0.36 0 0.645306 0.0687447 0.4 1 0 0.294118 0.545 0.614754 0.26 0 0.536513 0.199829 0.65 0 1 0.294118 0.475 0.590164 0.33 0 0.561848 0.12468 0.1 0 1 0.352941 0.46 0.754098 0 0 0.296572 0.0469684 0.116667 0 1 0.0588235 0.68 0.606557 0.5 0.241135 0.557377 0.137062 0.05 0 1 0.294118 0.44 0.639344 0.3 0 0.411326 0.0768574 0.266667 0 1 0.352941 0.4 0.540984 0.3 0 0.390462 0.100342 0.333333 0 1 0.176471 0.71 0.655738 0.15 0 0.482861 0.0520922 0.7 0 1 0 0.5 0.721311 0.6 0.130024 0.697466 0.377455 0.166667 0 1 0.352941 0.77 0.639344 0.41 0.165485 0.687034 0.210504 0.1 0 1 0.0588235 0.545 0.311475 0.18 0.141844 0.344262 0.140478 0.0833333 0 1 0.352941 0.555 0.52459 0.39 0 0.509687 0.0777114 0.05 0 1 0.352941 0.915 0.770492 0 0 0.608048 0.590521 0.4 0 1 0 0.52 0.52459 0.23 0.137116 0.414307 0.160547 0.0333333 0 1 0.117647 0.415 0.540984 0.23 0.0591017 0.479881 0.178907 0.0166667 0 1 0 0.485 0.52459 0.36 0.118203 0.548435 0.222886 0.0666667 0 1 0 0.595 0.52459 0.18 0.108747 0.520119 0.27626 0.0333333 0 1 0 0.53 0.57377 0.37 0.174941 0.587183 0.225021 0.0166667 0 1 0 0.99 0.540984 0.32 0.323877 0.615499 0.181042 0.116667 1 0 0.647059 0.675 0 0 0 0.779434 0.213493 0.316667 1 0 0.294118 0.525 0.590164 0.29 0.384161 0.549925 0.0345858 0.116667 0 1 0 0.615 0.590164 0 0 0.540984 0.0768574 0.516667 1 0 0.235294 0.475 0.52459 0 0 0.4769 0.0354398 0.166667 1 0 0.235294 0.615 0.508197 0 0 0.4769 0.0631939 0.233333 1 0 0.0588235 0.475 0.491803 0.18 0.0685579 0.356185 0.0777114 0.0166667 0 1 0.470588 0.54 0.57377 0 0 0.454545 0.374466 0.2 1 0 0.0588235 0.64 0.672131 0.17 0.216312 0.409836 0.0157985 0.0166667 0 1 0.411765 0.68 0.737705 0 0 0.445604 0.0563621 0.483333 0 1 0.764706 0.76 0.737705 0.33 0.034279 0.399404 0.278822 0.366667 1 0 0.235294 0.585 0.52459 0.27 0.141844 0.494784 0.0649018 0.05 0 1 0.117647 0.445 0.737705 0.3 0 0.499255 0.0913749 0.35 0 1 0.0588235 0 0.606557 0.2 0.0271868 0.412817 0.0943638 0 0 1 0.0588235 0.755 0.491803 0 0 0.388972 0.0431255 0.0166667 0 1 0.235294 0.555 0.590164 0.47 0.244681 0.552906 0.560205 0.583333 1 0 0.176471 0.87 0.47541 0.22 0.229314 0.490313 0.219898 0.25 1 0 0.470588 0.895 0.590164 0.42 0.153664 0.487332 0.273698 0.25 1 0 0.235294 0.705 0.606557 0 0 0.411326 0.0708796 0.316667 0 1 0.705882 0.46 0.508197 0.07 0.304965 0.411326 0.362084 0.383333 1 0 0 0.365 0 0 0 0.314456 0.112724 0.0666667 0 1 0.588235 0.645 0.508197 0.36 0 0.614009 0.154996 0.283333 1

0 0.352941 0.81 0.508197 0 0 0.362146 0.0426985 0.483333 1 0 0.117647 0.53 0.52459 0.35 0.140662 0.454545 0.564475 0.216667 0 1 0.117647 0.645 0.688525 0 0 0.417288 0.087959 0.1 0 1 0 0.705 0.688525 0.26 0 0.482861 0.15158 0.0166667 0 1 0.117647 0.49 0.491803 0.17 0.141844 0.517139 0.0512383 0.0166667 0 1 0 0.54 0.557377 0.2 0 0.406855 0.302733 0.183333 0 1 0.117647 0.56 0.557377 0.22 0.111111 0.508197 0.101196 0.0833333 0 1 0.411765 0.485 0.622951 0.32 0.107565 0.609538 0.338599 0.183333 1 0 0 0.76 0.672131 0.39 0.321513 0.61848 0.0819812 0.1 0 1 0.117647 0.73 0 0 0 0.409836 0.0691716 0.116667 1 0 0.0588235 0.57 0.540984 0.36 0.236407 0.567809 0.0900939 0 0 1 0.470588 0.5 0.606557 0.4 0.254137 0.587183 0.248933 0.366667 1 0 0 0.725 0 0 0 0.658718 0.235696 0.166667 1 0 0.647059 0.6 0.655738 0.37 0.177305 0.630402 0.301879 0.45 1 0 0.235294 0.42 0.737705 0.23 0.0661939 0.588674 0.0345858 0.0666667 0 1 0.470588 0.97 0.655738 0 0 0.388972 0.201964 0.766667 0 1 0.235294 0.685 0.688525 0 0 0.464978 0.0742955 0.15 0 1 0.470588 0.55 0.622951 0 0 0.414307 0.0678907 0.616667 0 1 0.470588 0.755 0.639344 0.32 0.248227 0.639344 0.18702 0.25 1 0 0.647059 0.775 0.622951 0.28 0.177305 0.496274 0.544406 0.5 1 0 0.0588235 0.7 0.606557 0.26 0.212766 0.359165 0.320239 0.0333333 0 1 0.294118 0.515 0.885246 0.37 0 0.584203 0.0969257 0.733333 0 1 0.411765 0.525 0 0 0 0 0.0969257 0.05 0 1 0.117647 0.61 0.42623 0.43 0.186761 0.539493 0.315115 0.116667 0 1 0.588235 0.895 0.57377 0 0 0.5231 0.0520922 0.266667 0 1 0.411765 0.76 0.721311 0.44 0 0.745156 0.110589 0.25 1 0 0.235294 0.72 0.47541 0.28 0.165485 0.439642 0.08924 0.266667 0 1 0.352941 0.625 0.557377 0.3 0.141844 0.447094 0.164816 0.183333 0 1 0.0588235 0.9 0 0 0 0.645306 0.087105 0.333333 1 0 0.235294 0.425 0.47541 0.22 0.0579196 0.414307 0.0973527 0.116667 0 1 0 0.62 0.459016 0.13 0.124113 0.324888 0.159693 0 0 1 0.235294 0.78 0.614754 0 0 0.719821 0.0683177 0.183333 1 0 0.294118 0.65 0.672131 0 0 0.582712 0.374893 0.266667 1 0 0.0588235 0.445 0.622951 0.34 0.0437352 0.464978 0.0486763 0.0333333 0 1 0.0588235 0.61 0.737705 0.51 0.260047 0.740686 0.105465 0.166667 1 0 0.176471 0.955 0.557377 0.15 0.153664 0.460507 0.0943638 0.216667 0 1 0.117647 0.73 0.57377 0.38 0.425532 0.417288 0.110589 0.133333 1 0 0.823529 0.5 0.639344 0.25 0.217494 0.545455 0.142613 0.416667 1 0 0.117647 0.5 0.540984 0.2 0.106383 0.490313 0.336892 0.116667 1 0 0.117647 0.6 0.442623 0 0 0.399404 0.160974 0.1 0 1 0.0588235 0.64 0.803279 0.41 0.0685579 0.4769 0.530743 0.2 1 0 0.0588235 0.395 0.655738 0.25 0.0437352 0.378539 0.215628 0.0166667 0 1 0.0588235

0.495 0.47541 0.1 0 0.378539 0.201964 0 0 1 0.294118 0.945 0.52459 0.33 0.384161 0.464978 0.215628 0.133333 1 0 0.588235 0.34 0.868852 0.23 0.0579196 0.529061 0.088386 0.433333 0 1 0.235294 0.68 0.57377 0 0 0.464978 0.471392 0.0166667 1 0 0.235294 0.495 0.622951 0.15 0.0602837 0.345753 0.0619129 0 0 1 0.588235 0.555 0.57377 0.27 0 0.409836 0.0269001 0.316667 1 0 0 0.94 0.672131 0.14 0.218676 0.4769 0.257899 0.0166667 1 0 0.117647 0.71 0.672131 0.18 0.0756501 0.368107 0.291631 0 0 1 0.529412 0.56 0.672131 0.32 0.206856 0.509687 0.0777114 0.25 1 0 0.470588 0.775 0.508197 0.26 0.585106 0.506706 0.198548 0.416667 1 0 0.0588235 0.715 0.688525 0.23 0.36643 0.631893 0.426132 0.0166667 0 1 0.588235 0.81 0.688525 0 0 0.412817 0.0444065 0.55 0 1 0.470588 0.98 0.622951 0.29 0.330969 0.558867 0.225021 0.6 1 0 0.176471 0.54 0.508197 0.24 0 0.387481 0.0619129 0.0666667 0 1 0.235294 0.855 0.590164 0 0 0.649776 0.171221 0.0833333 1 0 0.294118 0.715 0.639344 0 0 0.670641 0.0478224 0.433333 0 1 0.352941 0.535 0.721311 0 0 0.548435 0.277114 0.166667 0 1 0.470588 0.495 0.688525 0 0 0.527571 0.132365 0.483333 0 1 0.294118 0.68 0.688525 0.41 0.104019 0.52161 0.088813 0.233333 1 0 0.0588235 0.54 0.721311 0.19 0 0.403875 0.137489 0.05 0 1 0.176471 0.42 0.590164 0.32 0 0.554396 0.0807003 0.116667 0 1 0.235294 0.635 0.721311 0.11 0.183215 0.514158 0.222032 0.116667 0 1 0 0.585 0.655738 0.31 0.0626478 0.673621 0.00469684 0.05 0 1 0 0.945 0.852459 0.25 0 0.511177 0.152434 0.333333 1 0 0.176471 0.415 0.47541 0.31 0.0212766 0.511177 0.110162 0.0666667 0 1 0.647059 0.69 0.606557 0.26 0.170213 0.538003 0.204526 0.483333 1 0 0.470588 0.42 0.606557 0.31 0 0.57079 0.161827 0.3 0 1 0.411765 0.665 0.688525 0 0 0.599106 0.263877 0.266667 0 1 0 0.51 0.704918 0.17 0.124113 0.436662 0.26345 0.1 0 1 0.411765 0.57 0.622951 0.17 0.130024 0.354694 0.16567 0.166667 0 1 0.0588235 0.65 0.57377 0.13 0.124113 0.385991 0.168232 0.0166667 0 1 0.235294 0.59 0.57377 0 0 0.663189 0.35269 0.0833333 0 1 0.823529 0.875 0.508197 0.3 0 0.500745 0.0572161 0.283333 1 0 0.117647 0.475 0.442623 0.14 0.104019 0.388972 0.28608 0.0166667 0 1 0.0588235 0.44 0.508197 0.24 0.0520095 0.445604 0.146883 0.0333333 0 1 0.470588 0.835 0.868852 0.46 0.27305 0.560358 0.0371477 0.366667 1 0 0.176471 0.39 0.57377 0 0 0.484352 0.0819812 0.3 0 1 0 0.585 0.540984 0.31 0.222222 0.459016 0.177199 0.0166667 0 1 0.411765 0.535 0.606557 0 0 0.441133 0.0751494 0.166667 1 0 0.294118 0.695 0.52459 0.35 0.165485 0.42623 0.142186 0.0833333 0 1 0.117647 0.45 0.655738 0.14 0.0650118 0.363636 0.0730145 0.05 0 1 0.117647 0.72 0.47541

0.33 0.159574 0.470939 0.146883 0.0666667 1 0 0.529412 0.65 0.57377 0 0 0.509687 0.24509 0.4 1 0 0.0588235 0.98 0.622951 0.36 0.294326 0.543964 0.340307 0.133333 1 0 0.411765 0.92 0.688525 0.33 0 0.529061 0.118275 0.333333 1 0 0.294118 0.615 0.606557 0.4 0.0910165 0.508197 0.0815542 0.116667 0 1 0.352941 0.465 0.409836 0.3 0.0756501 0.42772 0.118702 0.0333333 0 1 0 0.735 0.696721 0.54 0 0.637854 0.126815 0.05 0 1 0.352941 0.67 0.655738 0.37 0.437352 0.688525 0.0683177 0.416667 1 0 0.705882 0.5 0.688525 0.33 0.124113 0.447094 0.175064 0.416667 0 1 0.411765 0.62 0.57377 0.33 0.254137 0.38003 0.0354398 0.266667 0 1 0.705882 0.44 0.606557 0.4 0.0638298 0.52608 0.128096 0.45 0 1 0.647059 0.425 0.606557 0 0 0.448584 0.0947908 0.233333 0 1 0.588235 0.45 0.696721 0.32 0 0.520119 0.318958 0.583333 1 0 0 0.685 0.557377 0.14 0.174941 0.369598 0.0277541 0 0 1 0.117647 0.785 0.606557 0.35 0.520095 0.587183 0.0239112 0.15 0 1 0.117647 0.635 0.47541 0.24 0.325059 0.412817 0.649872 0.0666667 0 1 0 0.43 0.557377 0.32 0 0.533532 0.0683177 0.0666667 0 1 0.0588235 0.785 0.590164 0.21 0.198582 0.38152 0.0192143 0.05 0 1 0.117647 0.585 0.737705 0.19 0.0839243 0.375559 0.100342 0 0 1 0.529412 0.56 0.672131 0.24 0 0.420268 0.514091 0.483333 1 0 0.0588235 0.475 0.606557 0.21 0.0862884 0.385991 0.254056 0.25 0 1 0.470588 0.77 0.639344 0.32 0 0.482861 0.15585 0.4 1 0 0.0588235 0.56 0.590164 0.3 0.208038 0.512668 0.192143 0.0666667 0 1 0.411765 0.735 0.622951 0 0 0.587183 0.0764304 0.366667 1 0 0.470588 0.665 0.590164 0 0 0.490313 0.0819812 0.3 1 0 0.470588 0.535 0.655738 0 0 0.366617 0.332195 0.216667 0 1 0.294118 0.52 0.606557 0 0 0.42921 0.0320239 0.45 0 1 0.0588235 0.415 0.557377 0 0 0.271237 0.233134 0.1 0 1 0.411765 0.895 0.778689 0.31 0 0.509687 0.0367208 0.65 0 1 0.0588235 0.5 0.540984 0.15 0.0661939 0.351714 0.251067 0.0833333 0 1 0.529412 0.615 0.57377 0.44 0.111111 0.493294 0.126388 0.316667 0 1 0.352941 0.685 0.5 0 0 0.360656 0.0311699 0.566667 0 1 0.0588235 0.515 0.245902 0.38 0.0981087 0.645306 0.0448335 0.2 0 1 0.764706 0.53 0.57377 0 0 0.509687 0.0738685 0.516667 0 1 0.294118 0.735 0.614754 0 0 0.445604 0.152007 0.116667 0 1 0.235294 0.495 0.590164 0.17 0 0.38152 0.0922289 0.116667 0 1 0 0.57 0.655738 0.34 0.336879 0.658718 0.0380017 0.1 0 1 0 0.81 0.622951 0.36 0 0.739195 0.122118 0.0833333 1 0 0.176471 0.555 0.508197 0 0 0.336811 0.0273271 0 0 1 0 0.5 0.57377 0.26 0.0591017 0.459016 0.221605 0 0 1 0.294118 0.79 0.57377 0 0 0.444113 0.0550811 0.7 0 1 0 0.505 0.622951 0 0 0.532042 0.0512383 0.0833333 0 1 0.235294 0.66 0.704918 0.31 0 0.417288 0.145602 0.7 0 1

0.0588235 0.515 0.655738 0.11 0.0969267 0.289121 0.176345 0.0166667 0 1 0.0588235 0.44 0.245902 0.42 0.117021 0.819672 0.17848 0.0833333 1 0 0 0.645 0.901639 0.46 0.153664 1 0.102904 0.0833333 1 0 0.235294 0.945 0.901639 0.31 0 0.424739 0.257045 0.266667 0 1 0.411765 0.515 0.540984 0.32 0 0.582712 0.113578 0.166667 1 0 0.117647 0.73 0.622951 0.35 0.229314 0.5693 0.107173 0.133333 0 1 0 0.625 0.557377 0 0 0.368107 0.0546541 0 0 1 0.176471 0.4 0.672131 0.31 0.0827423 0.509687 0.51836 0.1 1 0 0.0588235 0.765 0.672131 0.42 0.573286 0.605067 0.260034 0.0333333 0 1 0.0588235 0.64 0.393443 0.45 0.229314 0.603577 0.228437 0.05 1 0 0.0588235 0.475 0.540984 0.13 0.0449173 0.292101 0.109308 0.0666667 0 1 0.647059 0.715 0.770492 0.33 0.172577 0.545455 0.0751494 0.5 1 0 0.588235 0.74 0.688525 0.48 0.280142 0.560358 0.394108 0.5 1 0 0.411765 0.89 0.688525 0 0 0.594635 0.108027 0.333333 1 0 0 0.585 0 0 0 0.503726 0.364646 0.383333 0 1 0.176471 0.615 0.819672 0.35 0.283688 0.853949 0.342442 0.0166667 0 1 0.352941 0.455 0 0 0 0.444113 0.180615 0.166667 0 1 0.352941 0.735 0.655738 0 0 0.439642 0.0426985 0.483333 1 0 0.117647 0.415 0.532787 0.28 0.0780142 0.548435 0.235269 0.05 0 1 0.588235 0.695 0.655738 0 0 0.403875 0.581981 0.6 0 1 0.117647 0.45 0.557377 0.42 0 0.5693 0.181469 0.1 1 0 0.176471 0.75 0.622951 0 0 0.312966 0.0550811 0.266667 0 1 0 0.885 0.491803 0.29 0.565012 0.515648 0.424424 0 1 0 0.470588 0.63 0.721311 0.36 0.12766 0.57377 0.115713 0.466667 0 1 0.235294 0.92 0.639344 0.39 0.327423 0.551416 0.0794193 0.166667 1 0 0.0588235 0.61 0.52459 0.32 0.184397 0.5231 0.262169 0.15 1 0 0.411765 0.795 0.52459 0 0 0.408346 0.0922289 0.316667 0 1 0.0588235 0.5 0.590164 0.12 0.0827423 0.377049 0.247652 0.116667 0 1 0.176471 0.74 0.540984 0.25 0 0.484352 0.0760034 0.0166667 0 1 0.294118 0.575 0.803279 0 0 0.788376 0.0559351 0.116667 1 0 0.117647 0.51 0.704918 0.36 0.141844 0.678092 0.0209223 0.0333333 1 0 0.176471 0.61 0.639344 0 0 0.342772 0.0751494 0.316667 0 1 0 0.51 0.52459 0.46 0.0921986 0.605067 0.17848 0 0 1 0.117647 0.62 0.557377 0.28 0.242317 0.490313 0.340307 0.15 1 0 0.529412 0.7 0.770492 0 0 0.487332 0.280102 0.4 1 0 0.0588235 0.665 0.836066 0.28 0.165485 0.488823 0.0666097 0.4 1 0 0.0588235 0.45 0.508197 0.18 0.06974 0.374069 0.508113 0.0666667 0 1 0.294118 0.72 0.672131 0.26 0.336879 0.4769 0.159693 0.616667 1 0 0.588235 0.575 0.803279 0 0 0.357675 0.403074 0.216667 0 1 0.117647 0.64 0.639344 0.37 0.21513 0.645306 0.489325 0.166667 1 0 0.117647 0.56 0.704918 0.42 0.189125 0.57228 0.0717336 0.116667 0 1 0 0.64 0.557377 0.19 0.212766 0.454545 0.560632 0.0666667 1 0 0.529412

0.53 0.42623 0 0 0.464978 0.12895 0.35 0 1 0.117647 0.605 0.57377 0.32 0.112293 0.582712 0.345004 0.0333333 0 1 0.529412 0.82 0.688525 0.21 0 0.459016 0.32152 0.183333 1 0 0.117647 0.575 0.52459 0.22 0 0.459016 0.146456 0 0 1 0.0588235 0.695 0.508197 0.41 0.567376 0.606557 0.195559 0 0 1 0.117647 0.775 0.42623 0.27 0.638298 0.576751 0.0691716 0.0666667 1 0 0.176471 0.555 0.737705 0.12 0.0921986 0.423249 0.178053 0.133333 0 1 0.235294 0.755 0.737705 0.38 0 0.442623 0.0922289 0.25 0 1 0.0588235 0.595 0.721311 0.41 0.200946 0.675112 0.183177 0.0833333 0 1 0.0588235 0.63 0.459016 0.29 0.179669 0.42772 0.308711 0 0 1 0.117647 0.44 0.47541 0.26 0.0189125 0.423249 0.293766 0.0166667 0 1 0 0.695 0.508197 0.17 0.248227 0.329359 0.0550811 0 0 1 0.176471 0.53 0.590164 0 0 0.384501 0.0550811 0.1 0 1 0.117647 0.495 0 0 0 0.330849 0.0128096 0.0333333 0 1 0.294118 0.605 0.590164 0.23 0.132388 0.390462 0.0713066 0.15 0 1 0.0588235 0.59 0.47541 0.36 0.111111 0.496274 0.0781383 0.0333333 0 1 0.235294 0.45 0.721311 0.47 0.0638298 0.561848 0.121264 0.133333 0 1 0 0.895 0.409836 0.36 0.187943 0.563338 0.160974 0.0166667 1 0 0.0588235 0.595 0.704918 0.39 0.260047 0.679583 0.311699 0.133333 1 0 0.764706 0.765 0.721311 0.37 0.165485 0.605067 0.467976 0.3 0 1 0.176471 0.9 0.52459 0.25 0.0827423 0.506706 0.0824082 0.0833333 0 1 0.470588 0.59 0.590164 0.19 0 0.344262 0.596926 0.416667 0 1 0.0588235 0.62 0.606557 0.36 0 0.414307 0.00939368 0.15 0 1 0.0588235 0.905 0.52459 0.3 0.212766 0.508197 0.106746 0.283333 1 0 0.294118 0.585 0.704918 0.3 0.124113 0.582712 0.0738685 0.35 0 1 0.117647 0.775 0.606557 0.17 0.113475 0.396423 0.15158 0.1 1 0 0.117647 0.54 0.52459 0 0 0.459016 0.0341588 0 0 1 0.117647 0.54 0.42623 0.26 0.0744681 0.484352 0.102477 0.0166667 0 1 0.0588235 0.585 0.721311 0.24 0.171395 0.514158 0.13877 0.316667 1 0 0.470588 0.94 0.639344 0 0 0.71386 0.0251921 0.366667 1 0 0.235294 0.495 0.557377 0.38 0 0.488823 0.028608 0.2 0 1 0.0588235 0.535 0.409836 0.19 0 0.421759 0.0439795 0.133333 0 1 0 0.505 0.532787 0.28 0 0.366617 0.0678907 0.0166667 0 1 0.176471 0.51 0.360656 0.2 0.111111 0.459016 0.137489 0.0833333 0 1 0.352941 0.575 0.491803 0.39 0 0.502235 0.0713066 0.316667 1 0 0.0588235 0.6 0.655738 0.48 0.236407 0.579732 0.462852 0.333333 0 1 0.294118 0.585 0.754098 0 0 0.508197 0.110589 0.283333 0 1 0.647059 0.68 0.688525 0.35 0.153664 0.421759 0.0777114 0.35 1 0 0.0588235 0.58 0.57377 0.28 0 0.408346 0.0538002 0 0 1 0.588235 0.47 0.590164 0.18 0 0.344262 0.220751 0.583333 0 1 0.176471 0.79 0.57377 0.3 0.387707 0.529061 0.113578 0.233333 1 0 0.529412

0.825 0.721311 0 0 0.453055 0.0956447 0.466667 1 0 0.117647 0.405 0.590164 0.15 0.0898345 0.448584 0.200256 0.0666667 0 1 0.0588235 0.355 0.508197 0 0 0.324888 0.144321 0.0833333 0 1 0.117647 0.46 0.42623 0 0 0.448584 0.0269001 0.0166667 0 1 0 0.73 0.672131 0 0 0.603577 0.727156 0.383333 0 1 0.0588235 0.405 0.590164 0.18 0.0472813 0.396423 0.087532 0.05 0 1 0.176471 0.575 0.540984 0.39 0.165485 0.567809 0.030743 0.116667 0 1 0.117647 0.625 0.491803 0.2 0.165485 0.503726 0.00426985 0.166667 0 1 0.176471 0.555 0.459016 0.39 0 0.448584 0.204526 0.15 0 1 0.176471 0.65 0.639344 0.23 0.0933806 0.423249 0.104611 0.216667 1 0 0.294118 0.425 0.606557 0.22 0 0.432191 0.489325 0.183333 1 0 0.117647 0.5 0.442623 0.28 0.124113 0.563338 0.179334 0.05 0 1 0.235294 0.6 0.557377 0 0 0.441133 0.269428 0.216667 0 1 0.0588235 0.53 0.622951 0 0 0.558867 0.0508113 0.0833333 0 1 0.117647 0.5 0.52459 0.23 0 0.442623 0.123826 0 0 1 0.352941 0.585 0.786885 0 0 0.42772 0.0337319 0.15 0 1 0.176471 0.66 0.655738 0 0 0.512668 0.138343 0.383333 1 0 0.529412 0.455 0.557377 0 0 0.360656 0.0520922 0.616667 0 1 0.235294 0.72 0.672131 0.32 0 0.57377 0.203245 0.266667 1 0 0 0.51 0.614754 0.23 0 0 0.210931 0 0 1 0.235294 0.79 0.639344 0 0 0.490313 0.309564 0.166667 1 0 0.352941 0.83 0.606557 0 0 0.396423 0.0964987 0.75 0 1 0.0588235 0.385 0.459016 0.3 0.0661939 0.496274 0.500854 0.05 0 1 0.0588235 0.5 0.540984 0.29 0.231678 0.4769 0.156277 0.35 0 1 0.176471 0.515 0.590164 0.3 0.179669 0.411326 0.278395 0.1 0 1 0.176471 0.405 0.704918 0.16 0.0780142 0.409836 0.0973527 0.0166667 0 1 0.470588 0.37 0.57377 0.4 0.0579196 0.52608 0.26772 0.3 0 1 0.176471 0.445 0.606557 0.16 0.100473 0.453055 0.201964 0.283333 0 1 0.352941 0.515 0.540984 0 0 0.362146 0.0730145 0.133333 0 1 0.0588235 0.625 0.409836 0.4 0.1974 0.496274 0.377455 0.116667 1 0 0.470588 0.455 0.672131 0 0 0.530551 0.217336 0.783333 0 1 1 0.815 0.590164 0.41 0.134752 0.609538 0.315542 0.433333 1 0 0.294118 0.53 0.672131 0.3 0 0.588674 0.088813 0.283333 0 1 0.352941 0.425 0.639344 0 0 0.464978 0.129804 0.35 0 1 0.235294 0.625 0.57377 0.18 0.144208 0.4307 0.455167 0.4 1 0 0 0.52 0.622951 0 0 0.274218 0.215201 0.1 0 1 0.117647 0.47 0.622951 0.18 0.0780142 0.470939 0.243809 0.0333333 0 1 0.235294 0.56 0.639344 0.4 0 0.587183 0.0674637 0.283333 0 1 0.529412 0.725 0.721311 0.34 0.195035 0.451565 0.295901 0.533333 1 0 0.176471 0.645 0.52459 0.29 0.135934 0.393443 0.060205 0.116667 1 0 0.117647 0.375 0.52459 0.24 0.0650118 0.442623 0.12468 0.2 0 1 0.705882 0.42 0.590164 0.31 0 0.442623 0.0935098 0.416667 1 0 0.176471 0.305 0.672131 0.28 0 0.512668 0.0704526

0.416667 0 1 0.0588235 0.455 0.442623 0.25 0.118203 0.375559 0.0666097 0.0333333 0 1 0.0588235 0.82 0.672131 0.43 0.0791962 0.488823 0.112297 0.483333 0 1 0 0.705 0 0 0 0.631893 0.0542272 0.133333 1 0 0.0588235 0.595 0.360656 0.47 0.0744681 0.529061 0.0862511 0.0666667 0 1 0.0588235 0.48 0.52459 0.27 0.102837 0.494784 0.0900939 0 0 1 0.294118 0.695 0.655738 0.35 0.189125 0.470939 0.120837 0.0666667 1 0 0.470588 0.905 0.557377 0.36 0.585106 0.448584 0.229291 0.65 1 0 0.411765 0.405 0.639344 0.4 0.0567376 0.695976 0.0781383 0.35 0 1 0.529412 0.725 0.655738 0.46 0.153664 0.564829 0.238685 0.316667 1 0 0 0.635 0.655738 0.37 0.248227 0.540984 0.309991 0.0333333 0 1 0.647059 0.555 0.688525 0.4 0 0.697466 0.361657 0.4 1 0 0.411765 0.71 0.491803 0.33 0.224586 0.42921 0.260034 0.666667 0 1 0.411765 0.595 0 0 0 0.375559 0.0559351 0.266667 0 1 0.0588235 0.715 0.606557 0.22 0.072104 0.390462 0.0760034 0 0 1 0.0588235 0.435 0.557377 0.34 0.0910165 0.560358 0.137916 0.05 0 1 0.411765 0.415 0.639344 0.26 0.0839243 0.436662 0.294193 0.25 0 1 0.470588 0.6 0.639344 0 0 0.372578 0.141332 0.716667 0 1 0.0588235 0.445 0.540984 0.23 0.111111 0.418778 0.0380017 0 0 1 0.470588 0.915 0.52459 0 0 0.347243 0.253629 0.183333 1 0 0.352941 0.625 0.639344 0.31 0 0.411326 0.207942 0.466667 1 0 0.0588235 0.355 0.639344 0.5 0.0531915 0.494784 0.146883 0 0 1 0.176471 0.63 0.721311 0.41 0.277778 0.585693 0.267293 0.1 0 1 0.588235 0.625 0.57377 0.26 0.135934 0.463487 0.0542272 0.333333 1 0 0 0.535 0.491803 0.25 0 0.393443 0.0234842 0.0333333 0 1 0.529412 0.285 0.655738 0.37 0 0.488823 0.00768574 0.333333 0 1 0.294118 0.43 0.557377 0.28 0.0839243 0.450075 0.122118 0.05 0 1 0.117647 0.44 0.606557 0.19 0.0626478 0.432191 0.0644748 0.0166667 0 1 0.176471 0.565 0.360656 0.13 0 0.33383 0.0264731 0.0166667 0 1 0.0588235 0.41 0.52459 0.13 0.112293 0.315946 0.143894 0.0333333 0 1 0.235294 0.45 0 0 0 0.417288 0.227156 0.166667 0 1 0.588235 0.46 0.508197 0 0 0.385991 0.0380017 0.166667 0 1 0.235294 0.77 0.590164 0.29 0.148936 0.466468 0.111016 0.266667 0 1 0.176471 0.85 0.52459 0.37 0.265957 0.514158 0.118702 0.15 1 0 0.764706 0.79 0.934426 0 0 0.630402 0.0764304 0.383333 1 0 0.352941 0.97 0.639344 0 0 0.350224 0.0217763 0.633333 1 0 0.0588235 0.575 0.57377 0.3 0.113475 0.515648 0.19257 0.183333 1 0 0.294118 0 0.655738 0.32 0 0.611028 0.114432 0.266667 1 0 0.235294 0.46 0.655738 0 0 0.628912 0.0678907 0.133333 0 1 0.411765 0.805 0.704918 0 0 0.453055 0.0371477 0.433333 1 0 0.764706 0.63 0.737705 0 0 0.646796 0.215628 0.35 1 0 0.176471 0.695 0.442623 0 0 0.38152 0.138343 0.0166667 1 0 0.0588235 0.815 0.590164

0 0 0.581222 0.488471 0.2 1 0 0.411765 0.625 0.704918 0 0 0.560358 0.0964987 0.5 0 1 0 0.655 0 0 0 0.643815 0.0819812 0.0833333 1 0 0.470588 0.93 0.737705 0.35 0.265957 0.514158 0.14731 0.266667 1 0 0.294118 0.61 0.704918 0 0 0.517139 0.0905209 0.2 0 1 0 0.645 0.655738 0 0 0.464978 0.266866 0.133333 0 1 0.352941 0.525 0.57377 0.32 0.0803783 0.459016 0.0187874 0.266667 0 1 0.235294 0.64 0.57377 0 0 0.511177 0.0960717 0.05 0 1 0.529412 0.76 0.639344 0.34 0.202128 0.509687 0.347993 0.2 1 0 0.588235 0.375 0.672131 0 0 0.496274 0.0789923 0.283333 0 1 0.117647 0.61 0.57377 0.27 0 0.548435 0.11187 0.1 0 1 0.352941 0.51 0.672131 0 0 0.459016 0.0435525 0.25 1 0 0.117647 0.615 0.393443 0.32 0.195035 0.627422 0.188728 0.0833333 0 1 0.0588235 0.64 0.721311 0.39 0.130024 0.543964 0.418019 0.266667 1 0 0.176471 0.81 0.42623 0.38 0 0.554396 0.24509 0.05 1 0 0.176471 0.705 0 0 0 0.447094 0.291631 0.1 1 0 0.117647 0.705 0.47541 0.34 0.1513 0.378539 0.265158 0.05 0 1 0 0.835 0 0 0 0.481371 0.324936 0.15 1 0 0.176471 0.865 0.639344 0.39 0.218676 0.503726 0.380871 0.166667 1 0 0.0588235 0.545 0.491803 0.08 0.21513 0.378539 0.37105 0 0 1 0.352941 0.57 0.721311 0 0 0.414307 0.0721605 0.75 0 1 0 0.655 0.540984 0.4 0 0.511177 0.0503843 0.0166667 1 0 0.0588235 0.715 0.704918 0.3 0.390071 0.448584 0.347566 0.0333333 0 1 0.117647 0.45 0.491803 0 0 0.350224 0.0482494 0.0666667 0 1 0 0.535 0.622951 0 0 0.675112 0.259607 0.05 0 1 0 0.685 0.688525 0.27 0 0.406855 0.0653288 0.633333 0 1 0.411765 0.8 0.442623 0.32 0.206856 0.454545 0.217763 0.3 1 0 0.117647 0.505 0.47541 0.17 0.313239 0.360656 0.228864 0.0333333 0 1 0.235294 0.625 0.655738 0 0 0.481371 0.195559 0.1 1 0 0.294118 0.775 0.688525 0.44 0.644208 0.576751 0.230999 0.216667 0 1 0 0.69 0 0 0 0.540984 0.365073 0.0666667 1 0 0.411765 0.97 0.557377 0.28 0 0.535022 0.284799 0.333333 1 0 0.588235 0.505 0.622951 0.48 0.212766 0.490313 0.0397096 0.7 0 1 0.176471 0.48 0.459016 0.34 0.135934 0.368107 0.369769 0.3 0 1 0 0.495 0 0 0 0.372578 0.0747225 0.0166667 0 1 0 0.675 0.770492 0.46 0.171395 0.605067 0.087959 0.0833333 0 1 0.0588235 0.44 0.639344 0.29 0.0898345 0.4769 0.122545 0.133333 0 1 0.294118 0.22 0.508197 0 0 0.372578 0.217336 0.25 0 1 0.117647 0.435 0.47541 0.16 0.0614657 0.487332 0.0375747 0.0666667 0 1 0.0588235 0.945 0.491803 0.23 1 0.448584 0.136635 0.633333 1 0 0.411765 0.5 0 0 0 0.447094 0.173356 0.183333 1 0 0.294118 0.57 0.606557 0 0 0.371088 0.284372 0.6 0 1 0.176471 0.62 0.655738 0.33 0.153664 0.494784 0.0969257 0.0833333 0 1 0.235294 0.455 0.57377 0.32 0.104019 0.493294 0.157131 0.0166667 0 1 0.117647 0.54 0.508197 0.32 0.0661939

0.375559 0.0213493 0 0 1 0.0588235 0.625 0.57377 0.24 0.130024 0.362146 0.0610589 0.0666667 0 1 0.352941 0.51 0.737705 0.39 0 0.532042 0.254483 0.116667 0 1 0.294118 0.935 0.622951 0.27 0.244681 0.649776 0.408198 0.533333 1 0 0.117647 0.495 0.42623 0.15 0.111111 0.366617 0.238685 0 0 1 0.411765 0.68 0.606557 0.26 0.159574 0.387481 0.242955 0.5 0 1 0 0.51 0.42623 0 0 0.374069 0 0 0 1 0 0.7 0.532787 0.26 0.153664 0.634873 0.150726 0.05 1 0 0 0.675 0.557377 0.42 0.295508 0.630402 0.122545 0.05 1 0 0.117647 0.525 0.614754 0 0 0.347243 0.205807 0.533333 0 1 0.117647 0.695 0.614754 0 0 0.38152 0.0380017 0.133333 0 1 0 0.475 0.52459 0.39 0.124113 0.66468 0.122972 0.0166667 0 1 0.352941 0.525 0.655738 0.28 0 0.484352 0.341588 0.0833333 0 1 0.117647 0.53 0.459016 0.27 0.195035 0.432191 0.148591 0.0166667 0 1 0.117647 0.985 0.57377 0.45 0.641844 0.454545 0.0341588 0.533333 1 0 0.588235 0.84 0.606557 0 0 0.566319 0.195986 0.216667 1 0 0.0588235 0.565 0.52459 0.35 0 0.500745 0.198548 0 1 0 0.411765 0.935 0.557377 0.39 0.359338 0.561848 0.0751494 0.333333 1 0 0.764706 0.645 0 0.3 0 0.594635 0.20965 0.383333 1 0 0.235294 0.645 0.704918 0.2 0.319149 0.5231 0.0653288 0.0333333 0 1 0.117647 0.47 0.557377 0.18 0.0898345 0.387481 0.206234 0 0 1 0.235294 0.55 0.540984 0 0 0.47541 0.167805 0.133333 0 1 0.117647 0.48 0.557377 0.13 0.0579196 0.314456 0.242955 0.0833333 0 1 0 0.535 0.508197 0.3 0.0874704 0.545455 0.289923 0.0666667 1 0 0.176471 0.42 0.557377 0.3 0.125296 0.47541 0.219044 0.0666667 0 1 0 0.66 0.639344 0 0 0.482861 0.1345 0 0 1 0.117647 0.465 0.52459 0.32 0.189125 0.566319 0.254483 0.0333333 1 0 0.411765 0.665 0.721311 0.15 0.183215 0.482861 0.0785653 0.266667 0 1 0.529412 0.92 0.696721 0.15 0 0.447094 0.484629 0.466667 1 0 0.0588235 0.555 0.704918 0.19 0 0.448584 0.0277541 0.0333333 0 1 0.470588 0.625 0.786885 0 0 0 0.0657558 0.55 1 0 0.0588235 0.5 0.606557 0.12 0.0543735 0.290611 0.030316 0.116667 0 1 0 0.62 0.57377 0.2 0 0.408346 0.0751494 0.25 1 0 0.235294 0.475 0.57377 0.32 0 0.47839 0.22801 0.05 0 1 0.411765 0.84 0.721311 0.42 0.379433 0.5693 0.302733 0.316667 1 0 0.352941 0.545 0.491803 0.27 0 0.372578 0.0546541 0.1 0 1 0.235294 0.475 0.491803 0.32 0 0.527571 0.087959 0.116667 0 1 0.0588235 0.86 0.557377 0.49 0.684397 0.631893 0.266439 0.116667 1 0 0 0.865 0.639344 0.32 0.313239 0.692996 0.461571 0.616667 0 1 0.294118 0.485 0.622951 0.27 0 0.530551 0.128096 0.516667 1 0 0.117647 0.555 0.491803 0 0 0.390462 0.113151 0.0333333 0 1 0.0588235 0.475 0.672131 0.25 0.212766 0.52161 0.0661827 0.366667 1 0 0.294118 0.48 0.606557 0.18 0.0791962 0.500745 0.3924 0.366667 0 1

0.117647 0.34 0.57377 0.32 0.0780142 0.372578 0.0465414 0.0666667 0 1 0.0588235 0.965 0.409836 0.16 0.443262 0.385991 0.246371 0.05 0 1 0.294118 0.84 0.52459 0 0 0.490313 0.0243382 0.333333 1 0 0.352941 0.625 0.622951 0 0 0.503726 0.0183604 0.55 1 0 0.117647 0.525 0.655738 0.45 0.225768 0.502235 0.270282 0.133333 1 0 0.235294 0.655 0.557377 0.21 0.196217 0.493294 0.0350128 0.116667 0 1 0.176471 0.495 0.442623 0.19 0.101655 0.38152 0.0324509 0.05 0 1 0.235294 0.575 0.590164 0 0 0.4307 0.127242 0.416667 1 0 0.529412 0.6 0.590164 0.22 0.0661939 0.309985 0.279675 0.45 0 1 0.411765 0.53 0.491803 0.24 0 0.394933 0.0930828 0.133333 1 0 0.117647 0.54 0.508197 0.1 0.328605 0.377049 0.342869 0.0166667 0 1 0.176471 0.565 0.409836 0.1 0.100473 0.439642 0.233988 0.0666667 0 1 0.117647 0.87 0.721311 0.37 0.141844 0.663189 0.242528 0.05 1 0 0.0588235 0.485 0.57377 0.4 0 0.567809 0.059778 0.15 0 1 0.235294 0.66 0 0 0 0.490313 0.0956447 0.0333333 1 0 0.294118 0.495 0.606557 0.27 0 0.432191 0.0533732 0.183333 0 1 0.0588235 0.73 0.459016 0 0 0.442623 0.207515 0.133333 0 1 0 0.73 0.57377 0 0 0.564829 0.109308 0.116667 1 0 0.117647 0.525 0.47541 0.4 0.111111 0.520119 0.0627669 0.0666667 0 1 0.117647 0.46 0.622951 0.2 0 0.360656 0.691716 0.116667 0 1 0.470588 0.62 0.622951 0.24 0.70922 0.42772 0.260034 0.516667 1 0 0 0.455 0.655738 0 0 0.482861 0.223313 0.1 0 1 0 0.755 0.737705 0.46 0 0.627422 0.125107 0 1 0 0.411765 0.935 0.409836 0.33 0.463357 0.505216 0.319385 0.216667 1 0 0.235294 0.55 0.754098 0 0 0.560358 0.0482494 0.15 0 1 0.411765 0.75 0.540984 0.42 0.404255 0.517139 0.273271 0.35 0 1 0.352941 0.645 0.737705 0.07 0.385343 0.292101 0.215201 0.65 0 1 0.176471 0.435 0.491803 0.18 0 0.324888 0.156277 0 0 1 0.0588235 0.595 0.442623 0.13 0.0591017 0.33234 0.0542272 0.05 0 1 0.588235 0.505 0.704918 0.37 0 0.679583 0.451751 0.283333 1 0 0.352941 0.62 0.590164 0 0 0.411326 0.123826 0.133333 1 0 0.529412 0.61 0.459016 0 0 0.496274 0.442357 0.2 1 0 0.176471 0.91 0.606557 0 0 0.454545 0.114005 0.133333 1 0 0.176471 0.4 0 0 0 0 0.0409906 0.0166667 0 1 0.117647 0.6 0.622951 0.37 0.124113 0.591654 0.058497 0.133333 0 1 0.176471 0.625 0.47541 0 0 0.470939 0.0311699 0.05 0 1 0.352941 0.52 0.606557 0.18 0.184397 0.445604 0.274979 0.333333 1 0 0.470588 0.6 0 0 0 0.447094 0.0448335 0.283333 1 0 0.117647 0.355 0.57377 0.27 0 0.417288 0.216909 0.0166667 0 1 0.0588235 0.42 0.52459 0.23 0.135934 0.549925 0.167805 0.116667 0 1 0.705882 0.7 0.672131 0.43 0.384161 0.584203 0.192143 0.616667 1 0 0.235294 0.48 0.459016 0.17 0.0579196 0.309985 0.11187 0.0833333 0 1 0.176471 0.495 0.508197 0.19

0.0874704 0.324888 0.0858241 0.0833333 0 1 0.117647 0.985 0.57377 0.99 0 0.517139 0.212212 0.683333 1 0 0.0588235 0.525 0.47541 0 0 0.362146 0.0465414 0 0 1 0 0.615 0.721311 0.37 0 0.52459 0.0508113 0.133333 0 1 0.117647 0.595 0 0 0 0.292101 0.321947 0.85 0 1 0.117647 0.65 0.786885 0 0 0.336811 0.0811272 0 0 1 0 0.525 0.557377 0.22 0 0.298063 0.0674637 0.0166667 0 1 0.352941 0.57 0 0 0 0 0.0473954 0.0833333 0 1 0.352941 0.46 0.508197 0.32 0.148936 0.4769 0.0029889 0.416667 0 1 0.0588235 0.485 0.557377 0.21 0 0.405365 0.434244 0.0166667 0 1 0.647059 0.635 0.868852 0 0 0.581222 0.0478224 0.5 0 1 0.0588235 0.535 0.590164 0.3 0.0969267 0.459016 0.31725 0.05 0 1 0.588235 0.805 0.557377 0.23 0.156028 0.38003 0.105892 0.433333 1 0 0.529412 0.82 0.639344 0 0 0.488823 0.029889 0.4 1 0 0.470588 0.56 0.590164 0 0 0.351714 0.325363 0.616667 0 1 0.294118 0.68 0.672131 0 0 0 0.239966 0.8 0 1 0.176471 0.965 0.57377 0.31 0 0.520119 0.0695986 0.0666667 1 0 0.235294 0.58 0.590164 0.12 0.102837 0.329359 0.164389 0.266667 0 1 0.0588235 0.63 0.491803 0 0 0.448584 0.115713 0.433333 1 0 0.235294 0.73 0.754098 0 0 0.464978 0.19684 0.666667 1 0 0.0588235 0.555 0.508197 0.13 0.21513 0.357675 0.0256191 0.0333333 0 1 0.352941 0.755 0.508197 0.31 0.141844 0.529061 0.262169 0.116667 0 1 0.176471 0.37 0.557377 0.28 0.0531915 0.442623 0.0918019 0.0333333 0 1 0.0588235 0.505 0.409836 0.15 0.0425532 0.360656 0.191289 0.0833333 0 1 0.0588235 0.485 0.52459 0.19 0.0969267 0.271237 0.0943638 0 0 1 0.352941 0.615 0.590164 0.45 0.271868 0.500745 0.279675 0.216667 0 1 0.647059 0.515 0.557377 0.4 0 0.688525 0.0204953 0.35 0 1 0.0588235 0.4 0.606557 0.11 0.070922 0.447094 0.191716 0.0166667 0 1 0.529412 0.77 0.639344 0.3 0.118203 0.460507 0.0367208 0.4 0 1 0.705882 0.7 0.696721 0.33 0 0.557377 0.0708796 0.333333 0 1 0 0.565 0.622951 0 0 0.496274 0.0853971 0.0333333 1 0 0 0.475 0.696721 0.25 0.0425532 0.557377 0.0721605 0.05 1 0 0.294118 0.55 0.557377 0 0 0.387481 0.0913749 0.15 0 1 0.529412 0.78 0.704918 0.28 0.183215 0.511177 0.474381 0.35 1 0 0.588235 0.61 0.639344 0.31 0 0.411326 0.185312 0.4 0 1 0.294118 0.83 0.590164 0.19 0.206856 0.384501 0.217336 0.5 1 0 0.235294 0.73 0.639344 0 0 0.57377 0.188728 0.766667 1 0 0 0.595 0.540984 0.27 0 0.578241 0.0772844 0.0166667 0 1 0 0.525 0.52459 0.41 0.167849 0.61848 0.0405636 0.0166667 0 1 0.117647 0.59 0.655738 0 0 0.639344 0.262596 0 1 0 0.294118 0.62 0.606557 0 0 0.506706 0.0606319 0.283333 1 0 0.176471 0.815 0.57377 0.18 0.124113 0.470939 0.0811272 0.116667 1 0 0.352941 0.95 0.754098 0 0 0.529061 0.0853971 0.75 1 0 0.176471 0.495 0.655738 0.11 0.0756501

0.28763 0.087959 0.15 0 1 0.352941 0.4 0.655738 0.36 0 0.593145 0.0422716 0.116667 0 1 0.117647 0.645 0.606557 0.26 0.242317 0.494784 0.219044 0.0666667 0 1 0.235294 0.73 0.696721 0.27 0.118203 0.4307 0.0473954 0.1 0 1 0.411765 0.47 0.52459 0.25 0.0933806 0.496274 0.28181 0.333333 0 1 0.0588235 0.54 0.491803 0.46 0.210402 0.529061 0.143894 0.05 0 1 0.235294 0.585 0.508197 0.12 0 0.442623 0.12895 0.15 1 0 0.0588235 0.605 0.639344 0.39 0.0874704 0.581222 0.0781383 0.116667 0 1 0 0.9 0.540984 0.39 0 0.625931 0.774979 0.0666667 1 0 0 0.905 0.721311 0.44 0.602837 0.645306 0.0614859 0.0833333 1 0 0.0588235 0.395 0.614754 0.3 0 0.4769 0.135781 0.0166667 0 1 0.176471 0.88 0.704918 0.27 0.184397 0.496274 0.459436 0.516667 1 0 0.352941 0.435 0.655738 0 0 0.345753 0.00256191 0.183333 0 1 0.117647 0.61 0.491803 0.18 0.125296 0.444113 0.272844 0.0166667 0 1 0 0.37 0.42623 0.1 0.0425532 0.414307 0.0815542 0.0166667 0 1 0.352941 0 0.557377 0.41 0 0.581222 0.277114 0.333333 1 0 0.176471 0.39 0.409836 0.32 0.104019 0.461997 0.0725875 0.0833333 1 0 0.235294 0.74 0.491803 0.27 0.375887 0.460507 0.030743 0.133333 1 0 0.0588235 0 0.393443 0.2 0 0.368107 0.0264731 0.0166667 0 1 0.0588235 0.545 0.459016 0.21 0.159574 0.375559 0.322374 0.0333333 0 1 0.117647 0.635 0.377049 0.21 0.395981 0.512668 0.0418446 0.0166667 0 1 0.470588 0.6 0.704918 0 0 0.423249 0.0772844 0.0166667 1 0 0.176471 0.58 0.606557 0.15 0.124113 0.391952 0.0123826 0.05 0 1 0.117647 0.79 0.737705 0 0 0.470939 0.310418 0.75 1 0 0.411765 0.71 0.737705 0.24 0.567376 0.453055 0.0213493 0.366667 1 0 0.235294 0.865 0.57377 0.14 0.198582 0.442623 0.120837 0.2 1 0 0.0588235 0.455 0.52459 0.24 0 0.435171 0.0486763 0 0 1 0.764706 0.53 0.590164 0.54 0 0.545455 0.0426985 0.4 0 1 0 0.63 0.704918 0.27 0.141844 0.408346 0.186593 0 0 1 0.0588235 0.395 0.491803 0.42 0.0567376 0.648286 0.256191 0.0333333 0 1 0 0.565 0.655738 0.16 0 0.461997 0.33988 0 0 1 0.176471 0.535 0.508197 0.13 0.0567376 0.341282 0.256191 0.0333333 1 0 0 0.59 0.52459 0.23 0.105201 0 0.705807 0 0 1 0.235294 0.57 0.52459 0 0 0.4307 0.0204953 0.05 0 1 0.0588235 0.72 0.672131 0.4 0 0.615499 0.225875 0.116667 0 1 0.176471 0.64 0.639344 0 0 0.314456 0.0811272 0.566667 0 1 0.294118 0.54 0.590164 0.43 0.0886525 0.538003 0.0789923 0.2 0 1 0.117647 0.67 0.57377 0 0 0.4307 0.198121 0.0333333 1 0 0.470588 0.715 0.540984 0 0 0.520119 0.0217763 0.333333 1 0 0.411765 0.98 0.737705 0 0 0.593145 0.159266 0.333333 1 0 0.117647 0.56 0.614754 0.32 0 0.532042 0.029889 0 0 1 0.117647 0.56 0.540984 0.22 0 0.372578 0.0977797 0.05 0 1 0.117647 0.425 0.532787 0

0 0.590164 0.363792 0.1 0 1 0.0588235 0.585 0.491803 0.23 0.125296 0.503726 0.16567 0.1 0 1 0.0588235 0.695 0.377049 0.19 0.0981087 0.42772 0.245944 0.0166667 0 1 0 0.6 0.606557 0.18 0.0744681 0.454545 0.088386 0.0833333 0 1 0 0.805 0.409836 0 0 0.326379 0.0751494 0.733333 0 1 0.176471 0.645 0.754098 0.49 0.183215 0.542474 0.380017 0.183333 1 0 0.0588235 0.425 0.540984 0.29 0 0.396423 0.116567 0.166667 0 1 0 0.655 0.721311 0 0 0.470939 0.283945 0.183333 1 0 0.0588235 0.69 0.672131 0 0 0.597615 0.0674637 0.116667 0 1 0.352941 0.825 0.557377 0.26 0.198582 0.500745 0.236123 0.466667 0 1 0.588235 0.61 0.557377 0 0 0.464978 0.0768574 0.333333 0 1 0 0.285 0.491803 0 0 0.323398 0.280529 0.766667 0 1 0.0588235 0.56 0.655738 0.45 0.156028 0.518629 0.059351 0.05 0 1 0.0588235 0.58 0.639344 0.29 0.212766 0.538003 0.17848 0.0666667 0 1 0 0.555 0.532787 0 0 0.366617 0.248506 0.166667 0 1 0.294118 0.495 0.442623 0.28 0.0981087 0.506706 0.179761 0.15 0 1 0 0.47 0 0 0 0 0.0760034 0.0666667 0 1 0.294118 0.58 0.606557 0.29 0 0.481371 0.248506 0.233333 1 0 0.235294 0.57 0.532787 0 0 0.326379 0.151153 0.266667 0 1 0.588235 0.54 0.540984 0 0 0.482861 0.0828352 0.35 1 0 0.294118 0.555 0.590164 0.28 0 0.356185 0.140478 0.1 0 1 0 0.59 0.688525 0.47 0.271868 0.682563 0.201964 0.166667 1 0 0.117647 0.41 0.42623 0.22 0.135934 0.424739 0.692143 0.0666667 0 1 0.176471 0.555 0.47541 0.31 0.0520095 0.439642 0.150299 0.0166667 0 1 0.294118 0.66 0.655738 0 0 0.399404 0.0461144 0.8 0 1 0.529412 0.67 0.606557 0.33 0.070922 0.385991 0.163108 1 0 1 0.117647 0.5 0.557377 0.25 0.0839243 0.57377 0.105038 0.0833333 0 1 0.764706 0.52 0.590164 0 0 0.464978 0.165243 0.283333 1 0 0.117647 0.5 0.57377 0.52 0.0673759 0.603577 0.255764 0.0666667 0 1 0 0.42 0.672131 0.31 0.147754 0.5693 0.0661827 0.0333333 0 1 0.117647 0.505 0.47541 0.35 0.106383 0.324888 0.0328779 0.0166667 0 1 0.294118 0.735 0.639344 0 0 0.502235 0.059778 0.733333 0 1 0.529412 0.855 0.901639 0.24 0.283688 0.676602 0.274552 0.55 1 0 0.705882 0.755 0.57377 0.4 0.320331 0.622951 0.283518 0.283333 1 0 0.294118 0.81 0.852459 0 0 0.561848 0.0311699 0.516667 1 0 0.529412 0.62 0.57377 0.33 0.475177 0.527571 0.087105 0.216667 0 1 0.0588235 0.835 0.606557 0.17 0.170213 0.348733 0.157558 0.2 1 0 0.294118 0.64 0.655738 0 0 0.515648 0.028181 0.4 0 1 0.411765 0.31 0.639344 0 0 0.485842 0.133646 0.333333 0 1 0.0588235 0.495 0.590164 0.3 0.0212766 0.575261 0.142613 0 0 1 0.470588 0.88 0.737705 0.34 0.35461 0.502235 0.166097 0.616667 1 0 0 0.49 0.672131 0.15 0.0992908 0.375559 0.0943638 0.0166667 0 1 0.0588235 0.745 0.557377 0.29 0.150118 0.436662

0.115713 0.35 1 0 0.294118 0.56 0.540984 0 0 0.563338 0.0781383 0.333333 1 0 0 0.9 0.737705 0.26 0.106383 0.543964 0.100769 0.233333 1 0 0.0588235 0 0.557377 0.35 0 0.4769 0.132792 0.0166667 0 1 0.294118 0.63 0.639344 0.27 0.0260047 0.441133 0.154142 0.316667 0 1 0.235294 0.725 0.672131 0.18 0 0.484352 0.0670367 0.816667 1 0 0.411765 0.57 0.540984 0 0 0.488823 0.0768574 0.35 1 0 0.0588235 0.43 0.540984 0.52 0.0768322 0.615499 0.358241 0.133333 0 1 0.0588235 0.865 0.606557 0 0 0.548435 0.00426985 0.283333 1 0 0.352941 0.54 0.360656 0.2 0.153664 0.357675 0.313834 0.233333 0 1 0.294118 0.79 0.688525 0.41 0.248227 0.587183 0.135354 0.133333 1 0 0.117647 0.61 0.622951 0.27 0.236407 0.535022 0.172929 0.0833333 0 1 0 0.685 0.327869 0.35 0.198582 0.642325 0.943638 0.2 1 0 0.411765 0.645 0.557377 0.49 0.147754 0.57377 0.154142 0.366667 1 0 0.352941 0.67 0.57377 0.23 0.153664 0.527571 0.198121 0.133333 1 0 0.117647 0.57 0.557377 0.22 0 0.42772 0.0059778 0.0666667 0 1 0 0.52 0.52459 0.37 0.0756501 0.500745 0.184458 0.0166667 1 0 0.176471 0.51 0.606557 0 0 0.439642 0.0183604 0.183333 0 1 0.294118 0.385 0.672131 0.41 0.0496454 0.533532 0.0333049 0.233333 0 1 0 0.455 0.557377 0.32 0.248227 0.594635 0.129377 0.0666667 0 1 0.235294 0.545 0.52459 0.44 0.117021 0.518629 0.353117 0.0833333 1 0 0.117647 0.37 0 0 0 0 0.0102477 0.0166667 0 1 0.117647 0.56 0.639344 0.5 0.165485 0.587183 0.0414176 0.05 0 1 0.117647 0.54 0.655738 0 0 0.402385 0.0772844 0.516667 1 0 0.411765 0.975 0.57377 0.33 0.171395 0.374069 0.0362938 0.566667 1 0 0.176471 0.58 0 0 0 0.350224 0.0465414 0.0333333 0 1 0 0.42 0.52459 0.22 0.0780142 0.533532 0.199402 0 0 1 0 0.685 0.57377 0.38 0 0.494784 0.0392827 0.0166667 0 1 0 0.465 0.819672 0.39 0.0851064 0.646796 0.402647 0.233333 0 1 0.294118 0.685 0.885246 0 0 0.727273 0.0636208 0.266667 1 0 0.529412 0.595 0.655738 0.35 0 0.432191 0.0789923 0.133333 1 0 0.176471 0.48 0.639344 0.39 0 0.555887 0.0683177 0.316667 0 1 0.117647 0.28 0.459016 0.28 0.0531915 0.360656 0.108454 0.0166667 0 1 0.117647 0.545 0.754098 0 0 0.636364 0.327498 0.55 0 1 0.176471 0.45 0.639344 0 0 0.636364 0.20538 0 0 1 0.0588235 0.51 0.606557 0 0 0.588674 0.0918019 0.35 1 0 0.352941 0.515 0.590164 0.32 0.224586 0.561848 0.105038 0.566667 0 1

EK -3

Yapay Sinir Ağları Eğitiminde Kullanılan ABC Algoritmasının Kodları

3.1. Yapay Sinir Ağları Eğitiminde Kullanılan ABC Algoritmasının Kodları

```
begin
for ks:=1 to kosmasayisi do
begin
gen:=0;
randomize;
initial;
REPEAT
gen:=gen+1;
employed;
seleksiyon;
normalize;
onlooker;
scout;
min[gen]:=min[gen]+minf;
bestfit[gen]:=bestfit[gen]+enfit;
UNTIL ((gen=maxgen)or (minf<=tol));
if (minf<=tol) then success:=success+1;
testofruns[ks]:=CalculateTestSetError();
minort:=minort+(minf);
minsofruns[ks]:=minf;
gensofruns[ks]:=gen;
genort:=genort+gen;
```

```
testerr:=testerr+testofruns[ks];
end;
minort:=minort/kosmasayisi;
genort:=genort/kosmasayisi;
testerr:=testerr/kosmasayisi;
for ks:=1 to kosmasayisi do
begin
stdsapma:=stdsapma+sqr(minsofruns[ks]-minort);
stdsapmagens:=stdsapma+sqr(gensofruns[ks]-genort);
teststd:=teststd+sqr(testofruns[ks]-testerr);
end;
stdsapma:=sqrt(stdsapma/(kosmasayisi));
stdsapmagens:=sqrt(stdsapmagens/(kosmasayisi));
teststd:=sqrt(teststd/kosmasayisi);
end;
procedure initial;
var
i,j:integer ;
begin
randomize;
enfit:=0;
minf:=0;
for i:=1 to d do
begin
xhi[i]:=xh;
xlo[i]:=xl;
xt[i]:=0;
end;
for i:=1 to np do
begin
pfit[i]:=0;
pf[i]:=0;
```

```
bas[i]:=0;
for j:=1 to d do
begin
x[i,j]:=xlo[j]+random*(xhi[j]-xlo[j]);
p[i,j]:=x[i,j];
end;
end;
end;
procedure employed;
var
i,j:integer;
begin
for j:=1 to d do
xt[j]:=0;
for j:=1 to d do
begin
for i:=1 to np do
xt[j]:=xt[j]+p[i,j]/np;
end;
for i:=1 to np do
begin
par1:=trunc(random*d+1);
par2:=trunc(random*d+1);
for j:=1 to d do
begin
if (j=par1) then
begin
repeat
aday:=trunc(random*np+1);
until(aday<>i);
x[i,j]:=p[i,j]+(p[i,j]-p[aday,j])*(0.5-random)*2;
end
```

```
else
x[i,j]:=p[i,j];
if x[i,j]<xl then x[i,j]:=xl;
if x[i,j]>xh then x[i,j]:=xh;
end;
CalculateFunction(i);
avgfit[gen]:=avgfit[gen]+(pfit[i]/(2*np));
end;
end;
procedure seleksiyon;
var
i,j:integer;
begin
for i:=1 to np do
begin
if fit[i]>pfit[i] then
begin
bas[i]:=0;
pfit[i]:=fit[i];
pf[i]:=f[i];
for j:=1 to d do
p[i,j]:=x[i,j];
end
else
bas[i]:=bas[i]+1;
end;
end;
procedure normalize;
var
i,j:integer;
begin
maxim:=pfit[1];
```

```
index:=1;
for i:=1 to np do
begin
if maxim<pfit[i] then
begin
maxim:=pfit[i];
index:=i;
end;
end;
for j:=1 to d do
pg[j]:=p[index,j];
if enfit<maxim then
begin
enfit := maxim;
minf:=pf[index];
for j:=1 to d do
en[j]:=pg[j];
end;
toplama:=0;
for i:=1 to np do
begin
toplama:=toplama+pfit[i];
end;
for i:=1 to np do
begin
nfit[i]:=0.9*pfit[i]/maxim+0.1;
end;
end;
procedure onlooker;
var
i,j,k:integer;
begin
```

```
for j:=1 to d do
xt[j]:=0;
for j:=1 to d do
begin
for i:=1 to np do
xt[j]:=xt[j]+p[i,j]/np;
end;
t:=0;
i:=1;
for k:=1 to np do
mem[k]:=1;
repeat
if random<nfit[i] then
begin
mem[i]:=mem[i]+1;
t:=t+1;
par1:=trunc(random*d+1);
par2:=trunc(random*d+1);
for j:=1 to d do
begin
if (j=par1) then
begin
repeat
aday:=trunc(random*np+1);
until(aday<>i);
x[i,j]:=p[i,j]+(p[i,j]-p[aday,j])*(random-0.5)*2;
end
else
x[i,j]:=p[i,j];
if x[i,j]<xl then x[i,j]:=xl;
if x[i,j]>xh then x[i,j]:=xh;
end;
```

```
CalculateFunction(i);
if fit[i]>pfit[i] then
begin
bas[i]:=0;
pfit[i]:=fit[i];
pf[i]:=f[i];
for j:=1 to d do
p[i,j]:=x[i,j];
end
else
bas[i]:=bas[i]+1;
avgfit[gen]:=avgfit[gen]+(pfit[i]/(2*np));
end;
i:=i+1;
if i=np+1 then i:=1;
until(t=np);
end;
procedure scout;
var
i,j:integer;
begin
maxbas:=bas[1];
basindex:=1;
for i:=1 to np do
begin
if maxbas<bas[i] then
begin
maxbas:=bas[i];
basindex:=i;
end;
end;
for i:=1 to np do
```

```
begin
if ((bas[i]=maxbas)and (bas[i]>lim)) then
begin
bas[basindex]:=0;
scoutoccured[gen]:=true;
scoutcikti[ks]:=scoutcikti[ks]+1;
for j:=1 to d do
begin
x[basindex,j]:=(xh-xl)* 2 * (0.5-random);
end;
calculateFunction(i);
pfit[i]:=fit[i];
pf[i]:=f[i];
for j:=1 to d do
p[i,j]:=x[i,j];
end;
end;
end;
function calculateSetError(i:integer):real;
var
epoch,m,n,k:integer;
err:real;
begin
err:=0;
for epoch:=1 to network.traindatasayisi do
begin
SetWeigthMatrix(1,i);
for m:=1 to network.katmanlar[1].noronsayisi do
begin
network.katmanlar[1].cikis[m]:=0;
for n:=1 to network.girissayisi do
begin
```

```
network.katmanlar[1].cikis[m]:=network.katmanlar[1].cikis[m]+katmanlar[1].W[n,m]*InputData
end;
network.katmanlar[1].cikis[m]:=network.katmanlar[1].cikis[m]+network.katmanlar[1].W[n,m];
network.katmanlar[1].cikis[m]:=TransferFcn(katmanlar[1].cikis[m],katmanlar[1].trnfcn);
end;
for k:=2 to network.katmansayisi do
begin
SetWeigthMatrix(k,i);
for m:=1 to network.katmanlar[k].noronsayisi do
begin
network.katmanlar[k].cikis[m]:=0;
for n:=1 to network.katmanlar[k-1].noronsayisi do
begin
network.katmanlar[k].cikis[m]:=network.katmanlar[k].cikis[m]+katmanlar[k].W[n,m]*katmanla
end;
network.katmanlar[k].cikis[m]:=network.katmanlar[k].cikis[m]+network.katmanlar[k].W[n,m];
network.katmanlar[k].cikis[m]:=TransferFcn(katmanlar[k].cikis[m],katmanlar[k].trnfcn);
end;
end;
case network.pf of
0: err:=err+CalculateMSEError(epoch);
1: err:=err+CalculateSSEError(epoch);
2: err:=err+CalculateSEPError(epoch);
3: err:=err+CalculateCEPError(epoch);
end;
end;
case network.pf of
2: err:=err * (omax-omin);
3: err:=100*(err/network.traindatasayisi);
end;
result:=err;
end;
```

```
function calculateTestSetError():real;
var
epoch,m,n,k:integer;
err:real;
begin
err:=0;
for epoch:=network.traindatasayisi+1 to network.traindatasayisi+network.testdatasayisi
do
begin
SetBestWeigthMatrix(1);
for m:=1 to network.katmanlar[1].noronsayisi do
begin
network.katmanlar[1].cikis[m]:=0;
for n:=1 to network.girissayisi do
begin
network.katmanlar[1].cikis[m]:=network.katmanlar[1].cikis[m]+katmanlar[1].W[n,m]*network.I
end;
network.katmanlar[1].cikis[m]:=network.katmanlar[1].cikis[m]+network.katmanlar[1].W[n,m];
network.katmanlar[1].cikis[m]:=TransferFcn(network.katmanlar[1].cikis[m],network.katmanlar[
end;
for k:=2 to network.katmansayisi do
begin
SetBestWeigthMatrix(k);
for m:=1 to network.katmanlar[k].noronsayisi do
begin
network.katmanlar[k].cikis[m]:=0;
for n:=1 to network.katmanlar[k-1].noronsayisi do
begin
network.katmanlar[k].cikis[m]:=network.katmanlar[k].cikis[m]+katmanlar[k].W[n,m]*katmanla
end;
network.katmanlar[k].cikis[m]:=network.katmanlar[k].cikis[m]+katmanlar[k].W[n,m];
network.katmanlar[k].cikis[m]:=TransferFcn(network.katmanlar[k].cikis[m],katmanlar[k].trnfcn
```

```
end;
end;
err:=err+CalculateCEPError(epoch);
end;
result:=100*(err/network.testdatasayisi);
end;
procedure CalculateFunction(i:integer);
begin
f[i]:=calculateSetError(i);
if f[i]<=0 then
fit[i]:=1+abs(f[i])
else
fit[i]:=1/(1+f[i]);
end;
function TransferFcn(x:real;trnfcnno:integer):real;
begin
case trnfcnno of
0: result:=Logsig(x);
end;
end;
function Logsig(x:real):real;
begin
result:=1/(1+exp(-x))
end;
end.
function CalculateMSEError(e:integer):real;
var
cs:integer;
r:real;
begin
r:=0;
for cs:=1 to network.cikissayisi do
```

```
begin
r:=r+sqr(network.OutputData[e,cs]-network.katmanlar[network.katmansayisi].cikis[cs]);
end;
r:=r/(network.traindatasayisi);
result:=r;
end;
function CalculateSSEError(e:integer):real;
var
cs:integer;
r:real;
begin
r:=0;
for cs:=1 to network.cikissayisi do
begin
r:=r+sqr(network.OutputData[e,cs]-network.katmanlar[network.katmansayisi].cikis[cs]);
end;
result:=r;
end;
function CalculateSEPError(e:integer):real;
var
cs:integer;
r:real;
begin
r:=0;
for cs:=1 to network.cikissayisi do
begin
r:=r+sqr(network.OutputData[e,cs]-network.katmanlar[network.katmansayisi].cikis[cs]);
end;
r:=(100*r)/(network.traindatasayisi*network.cikissayisi);
result:=r;
end;
function CalculateCEPError(e:integer):real;
```

```
var
cs:integer;
r:real;
maxind:integer;
f:real;
begin
result:=0;
maxind:=1;
for cs:=2 to network.cikissayisi do
begin
if (katmanlar[katmansayisi].cikis[cs]>katmanlar[katmansayisi].cikis[maxind]) then
maxind:=cs;
end;
r:=0;
f:=0;
for cs:=1 to network.cikissayisi do
begin
if (cs=maxind) then
network.katmanlar[network.katmansayisi].cikis[cs]:=1
else
network.katmanlar[network.katmansayisi].cikis[cs]:=0;
f:=f+abs(network.OutputData[e,cs]-network.katmanlar[network.katmansayisi].cikis[cs]);
end;
if (f<>0) then result:=result+1;
end;
procedure SetWeigthMatrix(katmanno:integer;cozumno:integer);
var
ps,dd,m,n:integer;
WMT:Tmatris;
begin
if (katmanno=1) then
begin
```

```
ps:=(network.girissayisi+1)*(network.katmanlar[1].noronsayisi);
for dd:=0 to ps-1 do
begin
n:=dd mod (network.katmanlar[1].noronsayisi);
m:=trunc(((dd+1)-n)div(network.katmanlar[1].noronsayisi))+1;
network.katmanlar[katmanno].W[m,n+1]:=x[cozumno,dd+1];
end;
end
else
begin
ps:=(network.girissayisi+1)*(network.katmanlar[1].noronsayisi);
for dd:=2 to katmanno-1 do
ps:=ps+(network.katmanlar[dd-1].noronsayisi+1)*(network.katmanlar[dd].noronsayisi);
for dd:=ps to ps+((katmanlar[katmanno-1].noronsayisi+1)*(katmanlar[katmanno].noronsayisi))
do
begin
n:=(dd-ps) mod (network.katmanlar[katmanno].noronsayisi);
if (network.katmanlar[katmanno].noronsayisi=1) then
m:=trunc((((dd-ps)+1)-n)div(network.katmanlar[katmanno].noronsayisi))
else
m:=trunc((((dd-ps)+1)-n)div(network.katmanlar[katmanno].noronsayisi))+1;
network.katmanlar[katmanno].W[m,n+1]:=x[cozumno,dd+1];
end;
end;
end;
procedure SetBestWeigthMatrix(katmanno:integer);
var
ps,dd,m,n:integer;
WMT:Tmatris;
begin
if (katmanno=1) then
begin
```

```
ps:=(network.girissayisi+1)*(network.katmanlar[1].noronsayisi);
for dd:=0 to ps-1 do
begin
n:=dd mod (network.katmanlar[1].noronsayisi);
m:=trunc(((dd+1)-n)div(network.katmanlar[1].noronsayisi))+1;
network.katmanlar[katmanno].W[m,n+1]:=en[dd+1];
end;
end
else
begin
ps:=(network.girissayisi+1)*(network.katmanlar[1].noronsayisi);
for dd:=2 to katmanno-1 do
ps:=ps+(network.katmanlar[dd-1].noronsayisi+1)*(network.katmanlar[dd].noronsayisi);
for dd:=ps to ps+((katmanlar[katmanno-1].noronsayisi+1)*(katmanlar[katmanno].noronsayisi))
do
begin
n:=(dd-ps) mod (network.katmanlar[katmanno].noronsayisi);
if (network.katmanlar[katmanno].noronsayisi=1) then
m:=trunc((((dd-ps)+1)-n)div(network.katmanlar[katmanno].noronsayisi))
else
m:=trunc((((dd-ps)+1)-n)div(network.katmanlar[katmanno].noronsayisi))+1;
network.katmanlar[katmanno].W[m,n+1]:=en[dd+1];
end;
end;
end;
GetInputsOutputsFromData();
end;
close(F);
end;
procedure GetInputsOutputsFromData();
var i,j:integer;
begin
```

```
for i:=1 to network.traindatasayisi+network.testdatasayisi do
begin
for j:=1 to network.girissayisi do
network.InputData[i,j]:=network.Data[i,j];
for j:=1 to network.cikissayisi do
begin
network.OutputData[i,j]:=network.Data[i,j+network.girissayisi];
end;
end;
end;
function CalculateNumberofParams(network:Tnet):integer;
var
i,ps:integer;
begin
ps:=(network.girissayisi+1)*(network.katmanlar[1].noronsayisi);
for i:=2 to network.katmansayisi do
ps:=ps+(network.katmanlar[i-1].noronsayisi+1)*(network.katmanlar[i].noronsayisi);
result:=ps;
end;
```

ÖZGEÇMİŞ

1977 yılında Kayseri'de doğdu. İlköğrenimini 60. Yıl Cumhuriyet İlköğretim Okulu'nda, orta ve lise öğrenimini Sami Yangın Anadolu Ticaret Lisesinde okudu ve dönem ikincisi olarak 1995 yılında mezun oldu. Aynı yıl Erciyes Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nü kazandı. 3. sınıf öğrencisi iken Şahin Yazılım A.Ş.'de analist programcı olarak çalışmaya başladı. 2000 yılında Erciyes Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümününden Fakülte Birincisi derecesiyle mezun oldu. Lisansüstü eğitim sınavında (LES) Türkiye'de ilk 500'e girdi ve bu başarısı ile Milli Eğitim Bakanlığı yurt dışı yüksek lisans bursunu almaya hak kazandı. Eylül 2002'de Rutgers University, New Jersey'de Elektrik ve Bilgisayar Mühendisliği bölümünde Yazılım Mühendisliği alanında mastır öğrenimine başladı ve Temmuz 2004'te Kablosuz Sensor Ağları Dolaşımında Veri Gizliliği konusunda hazırladığı tezini savunarak mastır derecesini aldı. Yüksek lisans eğitimi sırasında üniversite araştırma-geliştirme labaratuarlarında proje asistanı olarak çalıştı ve bölümden öğretici asistanlık teklifi aldı. Eylül 2004'te Türkiye'ye döndü ve mecburi hizmetini yapmak üzere Dumlupınar Üniversitesi, Mühendislik Fakültesi Bilgisayar Mühendisliği Bilgisayar Yazılımı Anabilim Dalında araştırma görevlisi olarak görevlendirildi. 2005 Güz döneminde Erciyes Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda doktora öğrenimine başlamasının ardından Kasım 2005'te Dumlupınar Üniversitesinde ki görevinden ayrıldı ve Şahin Yazılım A.Ş.'de projelerin yönetiminden ve koordinasyonundan sorumlu genel müdür yardımcısı olarak yeniden çalışmaya başladı. 2007 yılı mart ayında Erciyes Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü Yazılım Anabilim Dalına öğretim görevlisi olarak atandı. Halen bu görevde çalışmakta olup Yazılım Mühendisliği, Sistem Analizi, Dosya Organizasyon Teknikleri, Veritabanı Yönetim Sistemleri ve Programlama ile ilgili dersler vermekte ve Yapay Arı Kolonisi algoritması ile veri madenciliği ve kablosuz sensör ağları konularında optimizasyon ile ilgili araştırmalar yapmaktadır. Evli ve bir çocuk babasıdır.

İletişim Bilgileri

Adres	:	Erciyes Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 38039 Melikgazi/KAYSERİ
Telefon	:	0 352 437 49 01 - 32581
Belgegeçer	:	0 352 437 57 84
E-posta	:	celal@erciyes.edu.tr