T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRONİK BURUN VERİLERİNİN YAPAY ZEKA TABANLI ALGORİTMALARLA SINIFLANDIRILMASI

DOKTORA TEZİ

Muhammed Fatih ADAK

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Prof. Dr. Nejat YUMUŞAK**

Haziran 2016

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRONİK BURUN VERİLERİNİN YAPAY ZEKA TABANLI ALGORİTMALARLA SINIFLANDIRILMASI

DOKTORA TEZİ

Muhammed Fatih ADAK

Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ

Bu tez 22/06/2016 tarihinde aşağıdaki jüri tarafından oybirliği ile kabul edilmiştir.

Doç. Dr. **Pakize ERDOĞMUŞ** **Jüri Başkanı**	**Prof. Dr.** **Nejat YUMUŞAK** **Üye**	**Prof. Dr.** **Celal ÇEKEN** **Üye**
Doç. Dr. **Resul KARA** **Üye**	**Yrd. Doç. Dr.** **Ahmet Yahya TEŞNELİ** **Üye**	

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Muhammed Fatih ADAK

07.06.2016

TEŞEKKÜR

Gaz sensörleri ve elektronik burun yardımıyla elde edilen kokunun, optimize edilmiş yapay zekâ yöntemleri ile sınıflandırılması konusunda bana çalışma fırsatı veren, bilgi ve deneyimleri ile yol gösteren değerli danışmanım Prof. Dr. Nejat YUMUŞAK hocama aynı zamanda destekleri için tez izleme komitesi üyeleri, Prof. Dr. Celal ÇEKEN ve Prof. Dr. İbrahim ÇİL hocalarıma teşekkür ederim.

Elektronik burun verilerinin elde edilmesinde ve düzenlenmesinde destek veren TÜBİTAK Gaz Sensörleri Laboratuvarından Dr. Cihat TAŞALTIN'a ve Viyana Üniversitesi Gaz Sensörleri Laboratuvarı sorumlusu Prof. Dr. Peter Lieberzeit'e teşekkür ederim.

Tez süresince ve hayatım boyunca benden maddi ve manevi desteklerini esirgemeyen sevgili eşim H. Büşra ADAK'a, annem ve babama sonsuz minnet duygularımı sunarım.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

ABC	: Yapay arı koloni algoritması
AC	: Aseton, kloroform ikili gaz karışımı, kloroform sabit
ANN	: Yapay sinir ağı
BMO	: Kuş eşleşme algoritması
BP	: Geri yayılım
CA	: Kloroform ve aseton ikili gaz karışımı, aseton sabit
CM	: Kloroform ve methanol ikili gaz karışımı, metanol sabit
CO	: Karbonmonoksit gaz sensörü
feedforward	: İleri besleme
GA	: Genetik algoritma
GC-MS	: Gas choromatography – mass spectrometry
H2S	: Hidrojen sülfit gaz sensörü
HCA	: Hiyerarşil küme analizi
LDA	: Lineer ayrışım analizi
MA	: Metanol ve aseton ikili gaz karışımı, aseton sabit
MAE	: Ortalama mutlak hata
max	: Maksimum
maks	: Maksimum
MC	: Metanol ve kloroform ikili gaz karışımı, kloroform sabit
MIP	: Moleküler baskılanmış nano parçacık
min	: Minimum
ml	: Mili litre
MOS	: Metal oksit yarı iletken gaz sensörü
MSE	: Karesel ortalama hata
NH3	: Amonyak gaz sensörü
NO	: Nitrik oksit gaz sensörü

NP	: Nano parçacık
PCA	: Temel bileşen analizi
PNN	: İstatistiksel yapay sinir ağı
PSO	: Parçacık sürü optimizasyonu
QCM	: Quartz crystal microbalance gaz sensörü
rand	: Rastgele sayı üreten fonksiyon
SAW	: Yüzey akustik dalga gaz sensörü
SO2	: Sülfür dioksit gaz sensörü
SVM	: Destek vektör makinesi
TH	: Toplam hata
YSA	: Yapay sinir ağı

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Yapay sinir ağı, yapay arı koloni algoritması, genetik algoritma, elektronik burun, koku sınıflandırması, makine öğrenmesi

Günümüzde elektronik burun teknolojisi, yiyecek kalitesinin belirlenmesi, sağlık, savunma sanayi ve çevre gibi birçok farklı alanda başarıyla kullanılmaktadır. Adı geçen bu alanlarda genellikle koku verisinin sınıflandırıldığı görülmektedir. Gaz Sensörleri yardımıyla elde edilen koku verisinin sınıflandırılmasında birçok yöntem kullanılmakla birlikte literatürde özellikle yapay sinir ağlarına (YSA) sıklıkla rastlanmaktadır. YSA'da geleneksel olarak kullanılan geri yayılım algoritmasının (YSA-BP) bilindiği üzere, lokal minimuma takılma ve eğitim verisini ezberleme gibi zayıf yönleri bulunmaktadır. Bu tez kapsamında bu zayıf yönleri aşmak için, gaz sensörlerinden elde edilen koku verisinin sınıflandırılmasında YSA'nın eğitim kısmı yapay arı koloni (YSA-ABC) ve genetik algoritma (YSA-GA) ile optimize edilmiştir. Geliştirilen yazılım sayesinde, veri seti okutulup, YSA tasarlanıp ve eğitim modeli seçilerek algoritma çalıştırılabilmektedir.

YSA'nin eğitilmesindeki geleneksel yöntemde (BP) karesel ortalama hata (MSE) baz alınırken, geliştirilen yazılım sayesinde, YSA-ABC ve YSA-GA eğitiminde, istenirse MSE, ortalama mutlak hata (MAE) veya R^2 kullanılabilir. YSA-ABC'nin eğitilmesinde MAE'nin kullanılması MSE'ye göre daha başarılı sonuçlar verdiği görülmüştür. Bu yöntemler kullanılarak 4 farklı çalışma yapılmıştır. Bu 4 çalışmada eğitim hata değerleri olarak YSA-BP E-06, YSA-GA E-03 ile E-18 ve YSA-ABC ise E-16 düzeylerinde başarım göstermişlerdir. Test verisindeki başarımları ise YSA-BP E-06, YSA-GA E-03 ile E-09 ve YSA-ABC E-08 ile E-16 düzeylerinde başarım göstermişlerdir. Bu sonuçlar YSA-ABC'nin 4 çalışmanın 3'ünde diğer iki eğitim modeline göre daha başarılı eğitim ve test sonucu ürettiğini YSA-GA'nın ise sadece 1 çalışmada başarılı olduğunu göstermiştir. Eğitim süreleri karşılaştırıldığında, bütün çalışmalarda en hızlı eğitim modelinin saniyeler içerisinde tamamlanan YSA-BP olduğu daha sonra dakikalar düzeyinde süren YSA-ABC geldiği ve en yavaş modelin ise saatler süren YSA-GA olduğu görülmüştür. Eğitim modellerinin, eğitim süresi ve test verilerinde gösterdikleri başarı bir arada düşünüldüğünde, YSA-ABC'nin koku verisinin sınıflandırılmasında kullanımının daha uygun olacağı sonucuna varılmıştır.

CLASSIFICATION OF E-NOSE DATA BY USING ARTIFICIAL INTELLIGENCE-BASED ALGORITHMS

SUMMARY

Keywords: Artificial neural network, artificial bee colony algorithm, genetic algorithm, electronic nose, odour classification, machine learning

Today, electronic nose technology is successfully used in a wide range of areas such as food quality, health system, defense industry, and in environment. Usually, it is required to classify odour data during the use of e-nose technology in these fields. There are many methods used in classification of gas sensor data. Artificial neural networks (ANN) are especially come across in numerous studies as a classification method. Back propagation algorithm (ANN-BP) which is traditionally used in ANN, is known to get stuck in local minima and overfit the training data. In this thesis, during the classification of gas sensor data, artificial bee colony (ANN-ABC) and genetic algorithm (ANN-GA) are used to optimize ANN training in order to overcome these weaknesses of ANN-BP. A software is developed to run the algorithm after dataset is given to the network, ANN is designed and training method is determined.

Mean squared error (MSE) is traditionally used as the only performance measure in ANN training (with BP). However, in the software developed here, mean absolute error (MAE) and R^2 are also measured during ANN-ABC and ANN-GA training. It is observed that using MAE in ANN-ABC training as a performance measure gives more successful results compared to MSE use. Four different studies are conducted using this method. In these studies, ANN-BP had training error values in the level of E-06, while ANN-GA had E-03 and E-18, and ANN-ABC achieved E-16 level. The success levels of the networks in the test data were E-06 for ANN-BP, E-03 and E-09 for ANN-GA, and E-08 and E-16 for ANN-ABC. These results showed that, ANN-ABC produced more satisfactory training and test results in three of the four studies, compared to other two training methods. ANN-GA is found to be the most successful in only one of the studies. In terms of training time, ANN-BP is seen to be the fastest in all of the studies with a completion time in seconds; it is followed by ANN-ABC with a minutes-level completion time, while ANN-GA is observed to be the slowest by lasting for hours. When training time and performance in test data measures are both considered simultaneously, it can be concluded that ANN-ABC is more suitable to be used in classification of odour data.

BÖLÜM 1. GİRİŞ

Günümüz bilişim çağında, bilgisayarın her alana girmesi, her işlemin çevrimiçi yapılabilmesi iş süreçlerini hızlandırmış, zaman kaynaklı maliyetleri düşürmüştür. Gelişen teknoloji her alanda getirdiği yeniliklerle özellikle hesaplama ve sonuca varma işlemlerini hızlandırmıştır. Buna paralel olarak robot endüstrisinde de çok önemli gelişmeler yaşanmakta olup robotlar üzerinde sıra dışı çalışmalara imza atılmaktadır. Bu kapsamda değerlendirilebilecek yapay zekâ ve makine öğrenmesi, araştırmacıların ve bilim adamlarının birçok yeni teknik geliştirip var olan tekniklerde iyileştirmelere gittiği, önemini koruyan ve hatta her geçen gün daha da arttıran bir alan olmuştur. Makina öğrenmesi en genel tanımıyla, bilgisayar bilimi ile istatistiğin kesiştiği, verinin analiz edilip buradan bir sonuca varıldığı, sınıflama veya kümelemenin yapılabildiği bir çalışma alanıdır [1]. Makina öğrenmesi, geliştirilen araçlar ve yöntemler sayesinde, sağlık alanından, fabrikalara, savunma sanayinden, yiyecek endüstrisine kadar çok geniş bir yelpazede kullanılmaktadır. Makine öğrenmesi tekniklerinin bu alanlarda kullanılmasındaki en büyük amaç, insan gücüne olan ihtiyacın en aza indirilmesi ve hesaplamaların hızlı yapılması ile sınıflandırma ve kümeleme işlemlerinin normal insanın hesaplama süresinden çok daha hızlı bir şekilde yapılmasına olanak sağlamaktır.

Makine öğrenmesi teknikleri belirtilen birçok alanda kullanıldığı gibi sensörlerde ve özellikle gaz sensörleri alanında da sıklıkla kullanılmaktadır. Gaz sensörleri, koku verisinin belli koşul ve ortamda yapılan deneylerle, ilgili sensör üzerinde bırakacağı etkinin sayısal veriye dönüştürülmesini sağlayan sensörlerdir. Piyasada birçok sensör bulunmakla birlikte, birbirine benzer yapıdaki sensörlerin bir araya gelip oluşturdukları sensör dizisi ile koku verisinin bu sensörlerden geçip hızlı bir şekilde sayısal veriye dönüştürüldüğü cihazlara elektronik burun adı verilir [2]. Gaz sensörlerinden elde edilen koku verisi üzerinde uygulanan makine öğrenme

teknikleri ile koku verisinin sınıflandırılması, ayrıştırılması ve analiz edilmesi mümkündür. Literatürde bununla ilgili sayısız çalışma bulunmaktadır [3]–[5]. Literatürdeki çalışmalardan, bu tezin çalışma alanında olan veya yakınlık gösteren çalışmalara bakıldığında, yiyecek kalite kontrolünün elektronik burunlar yardımıyla yapıldığı çalışmaların [6], tarım ve ormancılık alanında yapılan gaz sensörleri çalışmalarının [7], meyve sınıflandırılmasının elektronik burun ile yapıldığı çalışmaların [8] ve süt ürünlerinin elektronik burun yardımıyla analiz edildiği çalışmaların [9] listelendiği birçok çalışma alanı görülebilecektir.

Elektronik burunun içerdiği gaz sensörlerinin kalitesi, elektronik burunun da kalitesini belirleyecektir. Satın alınacak bir elektronik burunda bu mutlaka düşünülmeli ve hangi alanda kullanılacaksa, o alanda daha başarılı olduğu bilinen elektronik burun tercih edilmelidir. Buna ek olarak Padilla ve arkadaşları sinyal işleme kısmının da çok büyük önem arz ettiğini belirtmişlerdir [10]. Bunun yanında kişi gaz sensörlerini kendi tasarlayıp veya temin edip, kendi elektronik burun cihazını tasarlayabilir. Bu yolu tercih edip başarılı sonuçlar alındığı görülmüştür [11]. Başka elektronik burun tasarımları da, aroma sınıflandırmasında yapılmış ve başarılı sonuçlar elde edilmiştir [12], [13].

Elektronik burundan elde edilen veri, ham veri formunda olup, bir ön işlem yapılmadan, veri seti oluşturulamayacak ve istenen sınıflandırma yöntemi uygulanamayacaktır. Elektronik burundan elde edilen veri parametre bazında çok boyutlu olabilmektedir. Bu boyutu indirgemek için genelde, Temel Bileşen Analizi (PCA), Hiyerarşik Küme Analizi (HCA) veya Lineer Ayrışım Analizi (LDA) kullanılmaktadır [14]. Nadir de olsa karar ağacı yaklaşımı gibi farklı yöntemleri, veri boyutu indirgemede kullanan çalışmalar da bulunmaktadır [15]. PCA, HCA veya LDA yöntemlerinin kullanıldığı çalışmalar incelendiğinde birçok farklı elektronik burun çalışmasında kullanıldığı görülecektir. Örneğin, Çin'in ünlü likör aroma bileşiklerinin sınıflandırıldığı çalışmada, 86 adet aroma bileşiği kullanılmış, PCA ve HCA ile sınıflandırma yapılmıştır [16]. Gupta ve arkadaşlarınn yapmış oldukları çalışmada, üzüm ve elma meyvelerinden elde edilen veri üzerine, PCA ve ardından LDA işlemi uygulanıp kemometrik yaklaşım ile sınıflandırma yapılmış ve %100'lük

bir başarı elde edilmiştir [17]. Yine kemometrik bir yaklaşım ile Versari ve arkadaşları, PCA ve LDA'yı kullanıp şarabın kalitesini analiz etmişlerdir [18]. PCA ve LDA'nın kullanıldığı bir diğer çalışmada "Orthosiphon stamineus" isimli bitkiyi, veri füzyonu ile daha iyi sınıflandırmışlardır [19]. Elektronik burundan elde edilen veriler ile mango meyvesinin olgunluk analizi, gaz kromatografisi ile yapılmıştır [20].

Elektronik burun çok farklı alanlarda kullanıldığında da yine boyut azaltmak için PCA kullanılmıştır. Örneğin bal türlerinin sınıflandırılmasında [21], içkilerin kalite kontrolünde [22], bakteri sınıflandırmasında [23], kanser teşhis ve analizinde [24], hasar görmüş bitkilerin ayrıştırılmasında [25], [26], aroma ve yiyeceklerin kalite kontrolünde [27]–[33], kâğıt endüstrisinde [34], kullanımı görülmüştür. Etin tazeliğinin değerlendirildiği bir diğer çalışmada PCA yerine LDA kullanılmıştır [35]. Bu kadar fazla çalışmada bu ayrıştırma yöntemlerinin kullanılıp başarılı sonuçlar elde edilmesinin yanında, Liu ve arkadaşları, özellikle aroma sınıflandırılmasında sadece analiz yöntemlerinin yüksek performans elde etmede yeterli olamayacağını belirtip, Çin içeceklerinin kalite kontrol ve lezzet değerlendirmesinde, 8 farklı lezzetli Çin içeceklerinin sınıflandırmasında PCA ve HCA'nın yanıda, geri yayılım algoritması kullanan yapay sinir ağı (YSA-BP) ve destek vektör makinesi (SVM) algoritmalarını kullanmışlardır [36]. Aynı düşüncede olan diğer araştırmacılar da ikili gaz karışımlarının tanımlanmasında PCA yanında YSA veya sadece YSA kullanmışlardır [37]–[39].

Makine öğrenmenin temel araçlarından biri olan Yapay sinir ağları (YSA), beynin çalışmasını örnek alır ve birçok sınıflandırma ve kümeleme yöntemlerinde kullanılırlar. YSA, katmanlar, katmanları oluşturan nöronlar ve nöronların birbirine ağırlıklar ile ifade edilen bağlarla bağlanması ile oluşur. Öğrenme aşamasında geleneksel yöntem olarak geri yayılım (BP) algoritması kullanılır [40]. Bu algoritma ağda ileri besleme ile oluşan hata değerini, ağda geri hareket ettirerek bütün nöronlara iletir ve hata oranına göre ağırlıklar güncellenir.

Yine birçok çalışmada boyut indirgeme ve analiz tekniklerinin yanında yapay sinir ağları kullanılmıştır. Yapay sinir ağının (YSA) elektronik burun çalışmalarında kullanıldığında yüksek performans alındığı görülmektedir. Örneğin parfüm kokusunun sınıflandırılmasında elektronik burundan elde edilen veri ve YSA kullanılmış, geliştirilen sistem ile 20 farklı koku algılanabilmiştir. Bu çalışmada YSA eğitiminde çevrimiçi eğitim kullanılmıştır [41]. Bir başka çevrimiçi eğitime benzer bir eğitimin kullanıldığı çalışmada siyah çayın optimum fermantasyon süresinin belirlenmesi yapılmıştır [42]. Peynir sınıflandırılmasında da sensör verisinin YSA ile eğitimi başarılı sonuçlar vermiştir [43]–[45]. Sağlık alanında yine YSA'nın koku sensörü verisinde kullanıldığını görmek mümkündür. Örneğin Tüberküloz tespitinde 14 adet sensör içeren bir elektronik burun kullanılmış, tasarlanan 3 katmanlı YSA, 25 kompleks örneği öğrenmesinin yanında 12 bilinmeyen örneği tanımlayabilmiştir [46]. Sağlık alanında yapılan diğer bir çalışmada anestezi dozu seviyesinin tespiti için elektronik burun kullanılmış, tek ara katmanlı YSA'nın başarı oranı en iyi durumda %95 olmuştur [47].

Elektronik burun verisinin sınıflandırılmasında her ne kadar YSA yaygın olarak kullanılsa da, YSA'nın lokal minimuma takılma veya eğitim verisini ezberleme durumu söz konusudur [1], [40]. Bu durum, araştırmacıların hybrid algoritmalar ya da YSA'nın eğitimini en iyileme yoluna yönelmesine neden olmuştur. Optimizasyon, bir problemin çözümü karşısında metodu iyileştirerek daha iyi bir çözüm ortaya koymak ve hata oranını düşürmektir [48]. YSA'nın eğitiminin optimize edilmesinde, araştırmacıların asıl amacı, YSA eğitimindeki hata oranını aşağı çekmenin yanında, algoritmanın asıl performansını belirleyen test verisinde hata oranını aşağı çekmektir [40]. Örneğin YSA'nın parçacık sürü optimizasyonu (PSO) ile hybrid kullanıldığı çalışmada klasik YSA'ya göre daha başarılı sonuçlar alındığı görülmüştür [49]–[51]. Yan ve arkadaşları ise yaradaki iltihap tespitinde elektronik burun kullanmış ve YSA'nın bu alanda performans veremediği bundan dolayı PSO'nun destek vektör makineleri (SVM) ile hybrid çalışmasını kullanmış ve sınıflandırma başarısı %97,5 olarak elde etmiştir [52]. Destek vektör makineleri yeni sayılabilecek bir makine öğrenme metodu olarak istatistiksel öğrenme teorisine dayanır ve Vapnik tarafından geliştirilmiştir [53]. Destek vektör makineleri, elektronik burun kullanılarak farklı

alanlarda da sınıflandırma için kullanılmıştır. Örneğin tıp alanında [54], aroma, yiyecek ve tütün alanında [55]–[58], hava kalitesinin belirlenmesi alanında [59] başarıyla kullanılmıştır. Destek vektör makineleri, elektronik burun alanında veri madenciliği ile birlikte de [60] kullanıldığı gibi tek başına da kullanımında başarı elde edilmiştir [61]. Destek vektör makineleri, tezin ileriki çalışması olarak modele dahil edilmesi planlanmakla birlikte literatürdeki bazı çalışmalarda alınmış sonuçlar SVM'nin bu teze dahil edilmemesine neden olmuştur. Örneğin bu tezin kapsamına giren benzer bir çalışma olan, Qiu ve arkadaşlarının yaptığı meyve suyu sınıflandırmasında SVM'nin düşük doğruluk değerleri ürettiğini belirtmişlerdir [55]. Yine YSA'nın ağırlıklarının optimal düzeyde belirlenme olayının bir optimal özellik seçme olduğu ve bu seçme işlemlerinin NP-hard problem olarak tanımlandığı [62] bu tarz problemlerde sezgisel algoritmaların kullanılmasının daha başarılı sonuçlar ürettiği belirtilmiştir [63].

Sezgisel algoritmalar kullanılarak YSA eğitiminin optimize edildiği çalışmalara bakıldığında, Örneğin Nasimi ve arkadaşları, YSA'nın eğitimini Karınca koloni algoritması ile gerçekleştirmiş ve sondajın tabanında oluşan basıncın tespitinde %99'luk bir başarı elde etmişlerdir [64]. Fakat karınca kolonisi, özellikle elektronik burun verisini sınıflandırılırken, YSA eğitimini optimize etmede nadiren kullanılmıştır. Farklı veri setleri üzerinde yapılan testlerde, YSA, Kuş eşleşme algoritması (BMO) ile eğitilmiş ve bazı veri setlerinde GA ve SVM'den daha iyi bazılarında ise daha kötü sonuç vermiştir [65]. Gıda işlemede termal basıncın incelendiği bir çalışmada YSA, Levenberg–Marquardt algoritması kullanılarak eğitilmiş ve başarılı sonuçlar elde edilmiştir [66]. YSA'nın eğitiminde kullanılan birçok algoritmanın karşılaştırıldığı bir çalışmada eğitim-öğretim tabanlı optimizasyon algoritmasının başarılı sonuçlar verdiğine vurgu yapılmıştır [67].

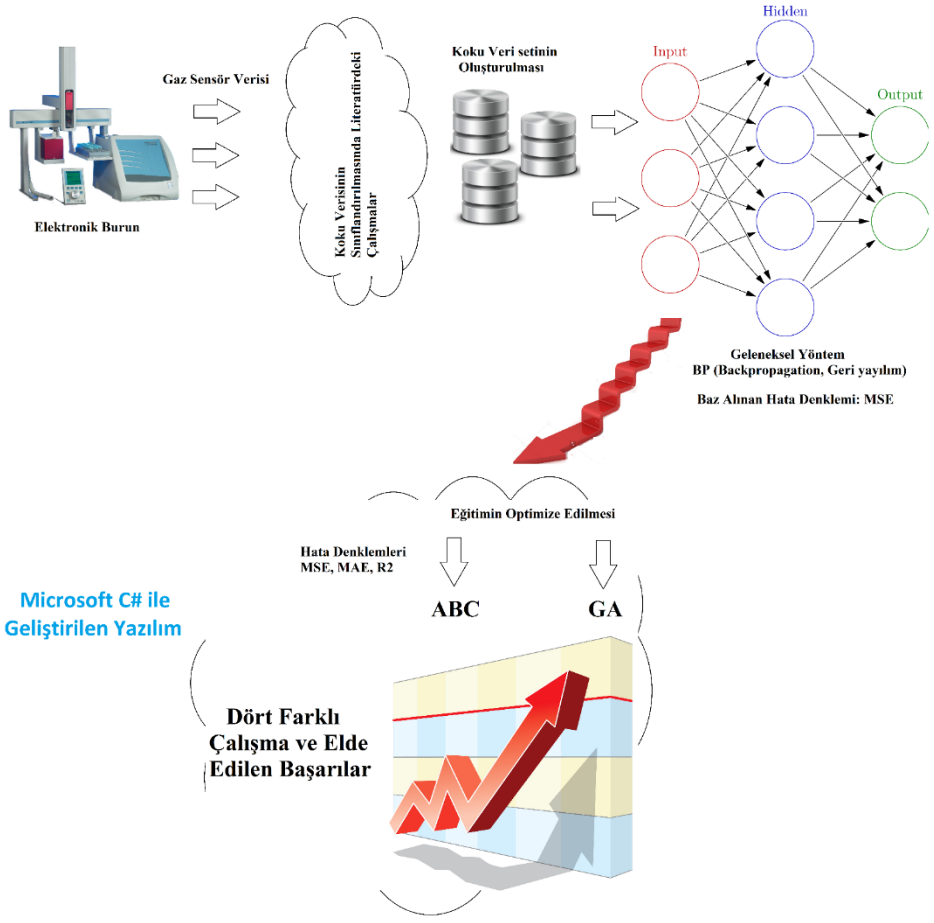
Literatürde, Genetik algoritma (GA), YSA ile birlikte sıklıkla kullanıldığı görülmektedir. GA, gerçek hayattaki doğal seçim mekanizması ve gen yapılarını örnek alan çaprazlama ve mutasyon içeren sezgisel arama algoritmasıdır [68]. GA kullanılarak Yapay sinir ağlarının (YSA) eğitildiği çalışmalarda başarılı sonuçlar elde edilmektedir. Örneğin ağın ağırlıklarının, kromozomları oluşturduğu bir

çalışmada en iyi ağırlık değerleri çaprazlama ve mutasyon işlemleri sonucunda belirlenmektedir [69]. Gizli katmanda kaç adet nöron bulunması gerektiğini belirlemek için GA kullanılmış, eğitim ve test işleminin süresi geleneksel yönteme göre iyileştirilmiştir [70]. Yine buna benzer bir çalışmada Zaji ve arkadaşları, çalışmalarında iki gizli katman kullanmışlardır. Gizli katmandaki nöron sayısı ve aktivasyon fonksiyonunu GA'yı temel alan Genetik programlama ile bulmuşlardır [71]. Genetik algoritmanın YSA'nın tasarımını belirlemede kullanıldığı bir diğer çalışmada klasik yönteme göre %1 ve 2'lik bir iyileşme sağlamışlardır [72]. Kromozomların sadece girdi ve çıktılardan oluşan YSA-GA hiybrid çalışmasında başarılı sonuçlar alınmıştır [73]. Kim ve arkadaşları, buzdolabındaki yiyeceklerin çürüme derecesinin tespiti için sekiz gaz sensörü kullanarak elde ettikleri veri üzerinde Genetik yapay sinir ağını (GANN) çalıştırarak başarılı sonuçlar elde etmişlerdir [74]. Gaz sensörleri üzerinde yapılan çalışmada yapay sinir ağının ağırlıkları GA tarafından optimize edilerek belirlenmiş ve BP'ye göre çok düşük hata oranı elde edilmiştir [75]–[77]. Bu çalışmalara yakın fakat sonlanma kriterinde farklılık içeren Baklacioglu'nun çalışmasında YSA, BP ile eğitimi sağlanmış, eğitim sonucunda sonlanma kriteri sağlanmadıysa GA çalıştırılıp yeni ağırlık değerleri belirlenmiştir [78]. GA'nın, elektronik burun yardımıyla çaydan elde edilen sensor verisi üzerinde de başarılı olduğu görülmüştür [79]. GA'nın YSA ile değil de farklı optimizasyon algoritmaları ve özellikle ABC ile hybrid bir şekilde kullanılıp başarılı sonuçlar aldığı da görülmüştür [63], [80], [81].

Yapay Arı Koloni (ABC) algoritması, arıların yiyecek bulma davranışlarını örnek alır. Bir sürü optimizasyonu olan ABC ilk olarak Karaboga tarafından geliştirilmiştir [82]. ABC kullanılarak yapay sinir ağının eğitildiği çalışmalar incelendiğinde başarılı sonuçlar alındığı görülecektir [83]. Özellikle Karaboga'nın yapmış olduğu bir çalışmada %100'lük bir başarı elde edilmiştir fakat bu başarı sadece eğitim veri seti üzerindeki başarıdır, test veri seti çalışmada bulunmamaktadur [84]. Buna benzer diğer çalışmalarda da iyi sonuçlar elde edilmiştir. Örneğin Uzlu ve arkadaşları hidroelektrik üretim tahmininde Yapay Sinir ağlarını ABC ile birlikte kullanmış ve daha iyi sonuç elde etmişlerdir [85]. Yine Yapay sinir ağının ABC ile eğitildiği bir çalışmada Ozkan ve arkadaşları petrol sızıntı tespiti için optimize edilmiş yapay sinir

ağına resim verisini vermişlerdir. Yapılan testlerde BP yerine ABC ile yapılan eğitimin daha başarılı olduğu görülmüştür [86]. Kayaların patlatılarak çıkarılması sırasında parçalanma oranını azaltmak için, parçalanma ve arka aşınma değerleri YSA tarafından tahmin edilmiştir. Patlama deseninin optimize edilmesinde ise ABC kullanılmıştır [87]. DNA mikro dizilerinin sınıflandırılması için veri boyutu indirgeme yöntemi olarak PCA yerine ABC kullanılmış sonra elde edilen veri YSA ile eğitilmiştir [88]. Bunun yanında YSA'nın ABC ile eğitildiği ve suç sınıflandırmasında bile başarılı sonuç verdiği görülmektedir [89].

Bu tez çalışması kapsamında, elektronik burun verisi sınıflandırılmasında kullanılan YSA'nın eğitimi, daha başarılı sınıflandırma yapabilmek için, sezgisel algoritmalardan, Yapay arı koloni (ABC) ve Genetik algoritma (GA) kullanılarak ayrı ayrı yapılmıştır. YSA'nın ABC ile eğitilip 4 farklı aroma türünün sınıflandırılmasında başarılı sonuçlar elde edilmiştir [90].



Şekil 1.1. Tezde yapılan çalışmaları özetleyen genel kapsam

Bu tez çalışmasında, önce elektronik burun ve gaz sensörleri hakkında genel bilgi verilmiş, çalışma şekilleri anlatılmış ve koku verisinin nasıl elde edildiği açıklanmıştır. Yapay sinir ağları genel çalışma prensibinden bahsedilmiş, hatayı minimize etmek için kullanılan ve geleneksel yöntem olan geri yayılım (BP) algoritması açıklanmıştır. Ardından, tezin asıl amacı, kullanılan yöntemler açıklanmış, ABC ve GA'nın çalışma prensipleri incelenmiş, yapay sinir ağlarının bu iki algoritma ile ayrı ayrı nasıl eğitildikleri gösterilmiştir. Eğitim ve test aşamalarının yapıldığı ve bu tez kapsamında, Microsoft Visual C# ortamında geliştirilen program detaylıca anlatılmıştır (Şekil 1.1.). Bu tez kapsamında 3 adet pilot çalışma yapılmıştır. Birinci pilot çalışmada, 4 farklı meyve kokusunun elektronik burundan elde edilen veriler kullanılarak YSA-ABC ile sınıflandırılması yapılmıştır. İkinci çalışmada, solvent yapımında sıklıkla kullanılan kloroform, aseton ve metanol'un ikili gaz karışımlarından elde edilen gaz sensör verisinin YSA-ABC ve YSA-GA kullanılarak sınıflandırılması ve karşılaştırılması yapılmıştır. Son pilot çalışma olarak 5 farklı gazdan elde edilen ve gaz sensörü olan QCM (Quartz Crystal Microbalance) kullanılarak veriler toplanmış, YSA-ABC ve YSA-GA ayrı ayrı uygulanarak sınıflandırma yapılmıştır. Elde edilen sonuçlar detaylı bir şekilde verilmiştir. Eğitim modellerinde her ne kadar eğitim bir kez yapılsa da çevrimiçi eğitim söz konusu olduğunda eğitim süreleri önem arz edebilmektedir. Bundan dolayı yapılan çalışmalardaki eğitim modellerinin eğitim zaman analizi yapılmıştır.

BÖLÜM 2. GAZ SENSÖRLERİ VE ELEKTRONİK BURUN

Günümüzde, savunma sanayinden, sağlık alanına, kozmetikten, yiyecek kalitesine kadar birçok alanda kullanılan elektronik burun, canlıların koku alma duyusunu taklit eden sensörlere sahiptir. Bu sensörler tasarlanırken kokuların içerdiği moleküller düşünülmüştür. Çünkü moleküller uçucudur dolaysıyla koku algılayıcılara (sensörlere) iletilmesi için taşıyıcılara ihtiyaç vardır [91]. Bu taşıyıcılık görevi sensörlere giden borular ve hava yardımıyla yapılmaktadır. Sensörler koku verisini dijital sinyallere çevirir. Elektronik burunun tarihi 1970'lere kadar dayanmaktadır. Sensörlerin verdiği tepkileri elektronik burun paralel olarak ve birleştirip kodlayarak iletir [4], [14].

2.1. Elektronik Burun Çeşitleri

Koku verisinin elde edilip üzerinde başarılı sınıflandırma işlemlerinin yapılabilmesi için, çalışılan alana özgü sensör teknolojisi kullanılmalıdır. Bir sensör tasarlanıp bütün çalışma alanlarında kullanılabilmesi söz konusu değildir. Bundan dolayı piyasada çok farklı çeşit ve markada elektronik burun ve gaz sensörü bulunmaktadır. Bunların birçoğu akademik çalışmalarda sıklıkla kullanılmakla birlikte çalışmalarda çok sık rastlananlar Tablo 2.1.'de listelenmiştir [2]. Tablo 2.1.'de listelenen elektronik burunların bazıları geniş boyutlu ve hantal bazıları ise taşınabilir avuç içi cihazlardır. Bazılarında birkaç sensör bulunurken bazılarında 20, 30'luk sensör dizileri bulunmaktadır. Kullanmış oldukları sensör teknolojilerinin ayrıntılı bilgilerine ilgili sensör geliştirici firmaların web sitelerinden erişilebilir. Tablo 2.1.'de listelenen 10 adet elektronik burun incelenmiş ve yapılan çalışmalar listelenmiştir [92].

Tablo 2.1. Akademik çalışmalarda sıklıkla kullanılan elektronik burunlar

Firma	Sensör Türü	Sistem
Agilent Technologies	MS	4440
Alpha M.O.S.	MOS, CP, SAW MS and MS-EN electronic tongue	Fox, Centauri Kronos & Prometheus Astree
Applied Sensor	MOSFET, MOS, QCM 4 x MOS, 8 x QCM QCM	3320, 3310 VOCseries VOCcheck
Bloodhound Sensors	CP	BH114
Cyrano Sciences Inc.	CP (composite)	Cyranose 320
Daimler Chrysler Aerospace	QCM, SAW, MOS	SAM system
Electronic Sensor Technology	SAW	zNose
Element	MOS	FreshSense
Environics Industry	IMCELL	MGD-1
Forschungszentrum Karlsruhe	MOS, SAW	Sagas
HKR Sensorsysteme	QCM, MS	QMB6
Lennartz Electronic	QCM, MOS, electrochemical	MosesII
Marconi Applied Technologies	CP, MOS, QCM	e-Nose 5000
Microsensor Systems	SAW	ProSat
Osmetech	CP	OMA and core sensor modüle
Quartz Technology	QCM	QTS-1
SMart Nose	MS	Smartnose-300
WMA Airsense Analysentechnik	MOS	PEN

2.1.1. Alpha MOS (FOX-4000)

Genellikle 12 adet sensörün kullanıldığı bu elektronik burunda sıvı bir kısım bulunmaktadır. Genellikle yiyecek kalitesi ve kozmetik sınıflandırılmasında kullanılmıştır. Örneğin parfüm temizleyicilerinin, nitel ve nicel analizinde bu elektronik burun kullanılmıştır [93]. Bir diğer çalışmada kimyasal parametrelerin arasındaki ilişkinin tahmin edilmesinde kullanılmış ve başarılı sonuçlar elde edilmiştir [94]. YSA ile PCA'nın birlikte kullanıldığı ve zeytinyağının sınıflandırıldığı [95] ve SVM' kullanılıp domuz etindeki bakteri miktarının tahmin edildiği [58] çalışmalarda da bu elektronik burun kullanılmıştır. GC-MS'nin kullanıldığı çalışmada yine başarılı sonuçlar elde edilmiştir [16].

2.1.2. Bloodhound sensors

Bu elektronik burun ilk olarak bir üniversitede laboratuvar ortamında geliştirilmiştir. İçerisinde 14 adet sensör bulunmaktadır ve sensör teknolojisi sıvı kristallere dayanmaktadır. Bu elektronik burun sıklıkla tıp alanında kullanılır, nadiren de olsa yiyecek kalitesinde kullanıldığı da görülmüştür [96]. Kullanıldığı bazı çalışmalara bakıldığında, YSA'nın yardımıyla mikroorganizma tespiti [97], idrardaki bakteri sınıflandırılması [98], balgam örnekleri kullanılarak tüberküloz teşhisi [46], gibi çalışmalar örnek verilebilir.

2.1.3. Cyranose 320

Bu elektronik burun taşınabilir bir elektronik burun olmakla birlikte birçok farklı alanda kullanımı görülmüştür. Taşınabilir küçük boyutu olmasına rağmen içerisinde 32 adet sensör barındırmaktadır. Tıp alanındaki çalışmalarda, örneğin Sarkoidoz hastalığının teşhisinde [99], akciğer ve solunum yolu hastalıklarının tespitinde [100]–[102] kullanılmıştır. Yiyecek kalitesinin tespitinde de birçok çalışmada kullanılmıştır [6].

2.1.4. zNose

Yiyecek kalitesinin değerlendirilmesinde sıklıkla kullanılan bu elektronik burun SAW (Surface Acoustic Wave Sensor) teknolojisini temel alır. Armudun toplanma zamanına göre kalitesinin değerlendirildiği bir çalışmada bu elektronik burun kullanılmıştır [103]. Yine yiyecek kalitesinin analiz edildiği bir çalışmada Zhang, zNose kullanmıştır [104].

2.1.5. FreshSense

Bu elektronik burun yiyecek kalitesinin değerlendirilmesinde sıklıkla kullanılmaktadır ve 5 adet sensör (CO, H2S, NO, SO2 ve NH3) içermektedir. Bu

elektronik burunun kullanıldığı çalışmalar incelendiğinde özellikle balık ürününün kalite değerlendirilmesinde kullanıldığı görülecektir [5], [105], [106].

2.1.6. MGD-1

Bu elektronik burunun büyük boyutlu olduğu gibi taşınabilir çeşitleri de bulunmaktadır. 6 adet sensör içerir ve çeşitli peynirlerin bulunduğu bir çalışmada sınıflandırma yapılmıştır [45].

2.1.7. Sagas

MOS veya SAW sensör çeşitleri bu elektronik burunda bulunabilir. Bu cihaz çeşitli koku analizlerinde kullanılmıştır. Örneğin farklı çeşitlerdeki domuz etinin kalitesinin değerlendirildiği ve PCA ile PNN (Probabilistic Neuronal Network) kullanıldığı çalışmada başarılı sonuçlar alınmıştır [107]. Medikal alanında yapılan diğer bir çalışmada ise solunum yolu ve sistematik hastalıkların teşhisi ve izlenmesi için Sagas elektronik burunu kullanılmıştır [108].

2.1.8. QMB6

QCM sensörlerin kullanıldığı bu elektronik burunu gıda ve çevre çalışmalarında görmek mümkündür. Günlük süt ürünlerinin analizi için Ampuero ve Bosset bu elektronik burunu kullanmışlardır [9]. Gıda ambalaj kısmında, mürekkep analizinde [109] ve benzer çalışma fakat çevre alanındaki, plastik materyallerin kalite değerlendirmesinde [110] kullanılmıştır.

2.1.9. E-nose 5000

Sıklıkla çevre analizlerinde kullanılan bu elektronik burun başarılı sonuçlar verdiği görülmüştür. Örneğin içme ve atık suyunun çevrimiçi değerlendirilmesinde kullanılmıştır [111].

2.1.10. ProSat

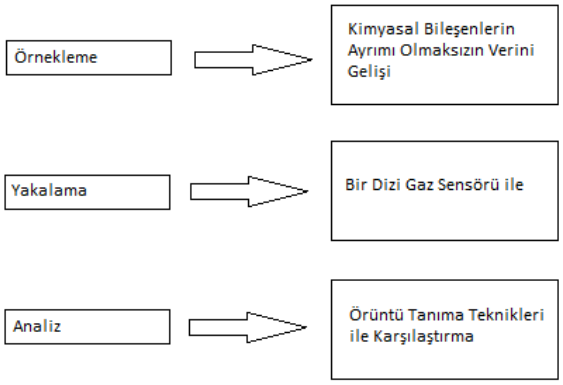
Çevresel kokuların analizinde sıklıkla kullanılan bu elektronik burun özellikle endüstriyel atıkların, atık suların analizinde başarılı sonuçlar vermiştir [112]. Yine atık su analizinin yapıldığı bir diğer çalışmada 12 aylık periyod süresince atık su bu elektronik burun kullanılarak izlenmiştir [113].

2.1.11. Smartnose-300

Bu elektronik burun daha çok tıp alanında kullanılmıştır. MS (mass spectrometry-based) sensörleri kullanılır. Örneğin hidrojen gazına maruz bırakılan fareler üzerinde inceleme yapılırken bu elektronik burun kullanılmıştır [114].

2.2. Elektronik Burundan Verinin Elde Edilmesi

Elektronik burunlar deneylerde kullanılmaya başlanmadan önce, elektronik burunun içerdiği sensörler belli bir süre hava vererek temizlenmelidir. Elektronik burun içerisinde bulunan sensörler, iyonları, molekülleri, atomaları veya sıvıları kullanarak, gazı dijital sinyallere çeviriler [2]. Bir çalışmada elektronik burun kullanılarak veri elde edilmek isteniyorsa Şekil 2.1.'deki adımlar kullanılmalıdır [115]. Koku verisi alınacak örnek bir cam tüpe veya kaba benzer bir yere konarak örnekleme safhası gerçekleştirilir.



Şekil 2.1. Elektronik burunda işlem adımları

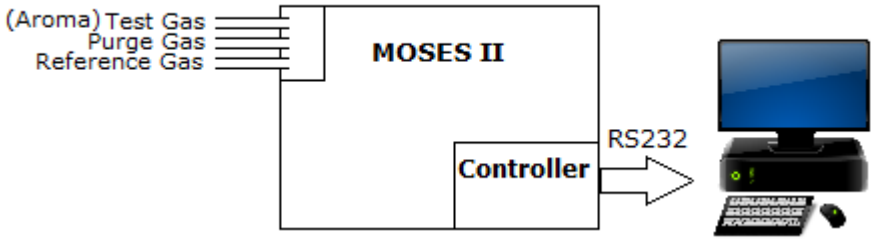
Elektronik burunun sahip olduğu sensörler yardımıyla örnek maddenin kokusu yakalanır. Son aşama olarak veriler kullanılabilecek formata sokulup üzerinde analiz yapılır.

2.3. Tez Kapsamında Kullanılan Gaz Sensörleri

Tez kapsamında, 3 farklı ölçüm yapılmıştır. Bu 3 ölçümde de farklı gaz sensör teknolojisi kullanılmıştır. Bölüm 2.1.'de detayları verilen elektronik burunlarının kullanılmamasının nedeni yüksek satın alma maliyetleri ve çalışılan laboratuvarlarda bulunmamalarıdır. Fakat aynı ölçümler, ilgili elektronik burunlar ile de yapıldığında başarılı sonuçlar alınacaktır. Nitekim incelenen literatürde ilgili elektronik burunlarda benzer çalışmalara rastlanmaktadır.

2.3.1. MOSES II

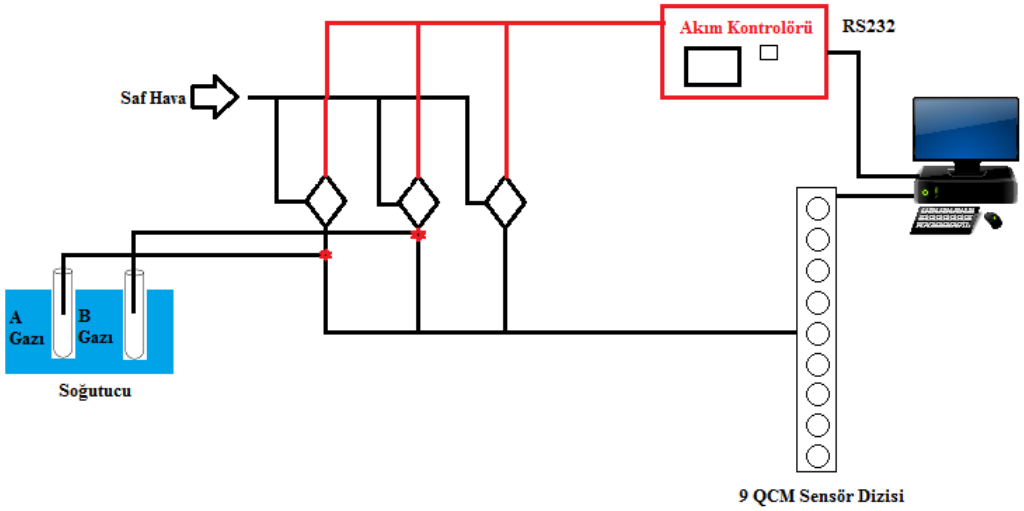
Tez kapsamında yapılan ölçüm çalışmalarından ilkinde MOSES II elektronik burunu kullanılmıştır. Bu elektronik burunun yapısı Şekil 2.2.'de gösterilmektedir. Şekil 2.2.'de elektronik buruna giren 3 adet boru görünmektedir. Bu borulardan iki tanesi koku verisi almak için kullanılırken bir tanesi sensörleri temizlemek için verilen hava için kullanılmaktadır. RS232 arabirimi ile bilgisayara bağlanıp dijital veriler elde edilebilmektedir. MOSES II elektronik burunu, Dr. Cihat TASALTIN'ın sorumlusu olduğu TÜBİTAK, Malzeme Araştırma Merkezi kimyasal sensör laboratuvarında bulunmaktadır. Ölçümler bu laboratuvarda yapılmıştır. MOSES II'deki 8 adet metal oksit sensör bulunmaktadır. Bu elektronik burunda ayrıca sıcaklık ve nem sensörleri de bulunmaktadır. MOSES II elektronik burunu literatürde birçok çalışmada kullanılmış ve başarılı sonuçlar alınmıştır [116]–[118].



Şekil 2.2. MOSES II elektronik burunu

2.3.2. Dokuz QCM sensöre sahip elektronik burun

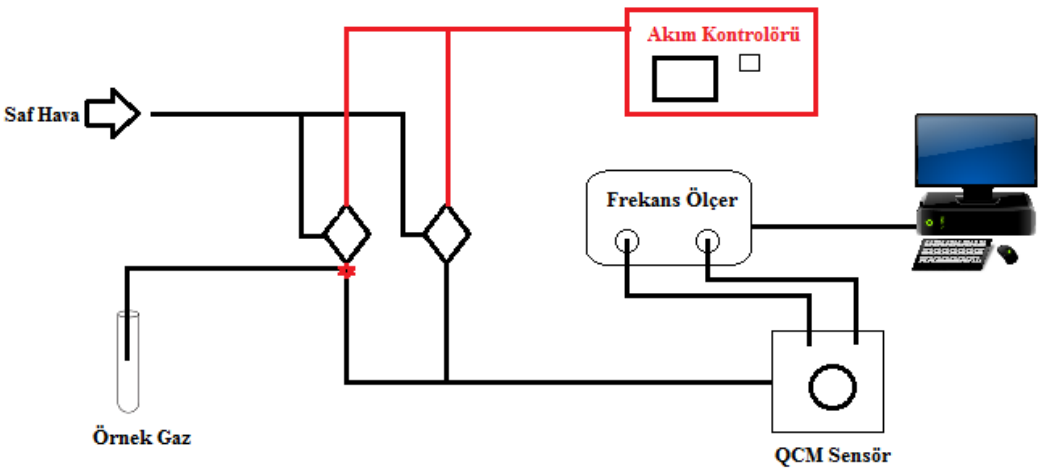
Tez kapsamında yapılan ölçüm çalışmalarının diğerinde ikili gaz karışımları ölçülmüştür. Şekil 2.3.'te görülecek olan bu ölçüm sisteminde ikili gaz karışımını sağlayacak akım kontrolörü bulunmaktadır. Bu cihaz A gazından ve B gazından belli oranlar alarak karışımı sağladıktan sonra gaz taşınmasını sağlayacak borulardan saf hava verilir. Bu hava belli oranlarda karışıma uğramış ikili gaz karışımını alarak 9 adet QCM sensörden oluşan sensör dizisine getirir. Sensörlerin bu gaz karışımına vermiş oldukları tepki bilgisayar yardımıyla kayıt altına alınır. A ve B gazları soğutucu yardımıyla istenilen soğuklukta sensörlere iletilebilmektedir. Şekil 2.3.'te gösterilen bu düzenek yine Dr. Cihat TASALTIN'ın sorumlusu olduğu TÜBİTAK, Malzeme Araştırma Merkezi kimyasal sensör laboratuvarında bulunmaktadır. Ölçümler bu laboratuvarda yapılmıştır.



Şekil 2.3. İkili gaz karışımlarının ölçüldüğü sensör düzeneği

2.3.3. Tek QCM sensöre sahip elektronik burun

Tez kapsamında yapılan ölçüm çalışmalarının sonuncusu, Avusturya'nın Viyana kentinde bulunan Viyana Üniversitesi, gaz sensörleri laboratuvarında, Prof. Dr. Peter Lieberzeit öncülüğünde yapılmıştır. Burada farklı oranlarda molekül içeren QCM sensörler kullanılmıştır. Ölçümlerin yapıldığı düzenek bir adet QCM sensör içermekle birlikte bir dizi ölçüm yapıldıktan sonra farklı yapıdaki QCM sensörler takılmıştır. Bu çalışmada 5 farklı QCM yapısına sahip sensör kullanılmıştır. İkili gaz karışımında yapılan çalışmaya benzer fakat ölçüm anında tek gaz kullanılmıştır. Bu çalışmada amaçlanan bireysel olarak koklatılan 5 farklı gazın sınıflandırılmasıdır. Kullanılan QCM sensörler üzerinde 2 farklı kanal bulunmaktadır. Birinci kanalda moleküler baskılanmış polimer (MIP), ikinci kanalda ise nanoparçacık (NP) bulunur (Şekil 2.4.).



Şekil 2.4. Tek QCM sensöre sahip düzenek

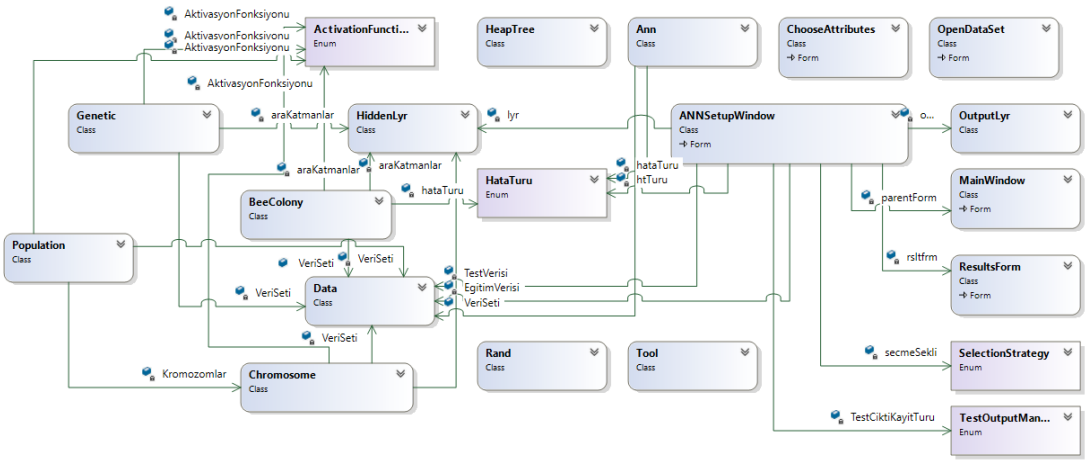
MIP ve NP'nin oranları değiştirilerek 5 farklı QCM sensör elde edilmiştir. Bu tarz QCM sensörlerin kullanılıp başarılı sonuçların elde edildiği birçok çalışma bulunmaktadır [119]–[121].

BÖLÜM 3. YAPAY SİNİR AĞININ ABC VE GA İLE EĞİTİMİNİN GERÇEKLEŞTİRİLDİĞİ YAZILIMIN GELİŞTİRİLMESİ

Tez kapsamında YSA'nın, geri yayılım algoritması (BP), ABC ve GA ile eğitilmesini sağlayan ve sonuçları dosyaya veya grafik ekranına veren bir yazılım geliştirilmiştir. Bu yazılım Microsoft Visual C# kullanılarak Windows uygulaması olarak tasarlanmıştır [122]. Bu yazılım sayesinde veri seti, excel ya da yazı dosyasından okutularak programa dâhil edilir. Yapay sinir ağı istenildiği gibi tasarlandıktan sonra eğitim modeli ve parametreler ayarlanır. Algoritma çalıştırılarak sonuçlar elde edilir. Geliştirilen yazılımın ana ekranı Şekil 3.1.'de gösterilmiştir. Şekil 3.1.'den görüldüğü üzere yukarı tarafta araçlar ve işlemlere ulaşmayı sağlayan menü ve kısa yollar bulunmaktadır. Açılan bütün pencereler bu ana ekranın içerisinde açılmaktadır. Bu işlemlerin daha kolay yapılmasını sağlamaktadır.

Şekil 3.1. Geliştirilen yazılımın ana ekranı

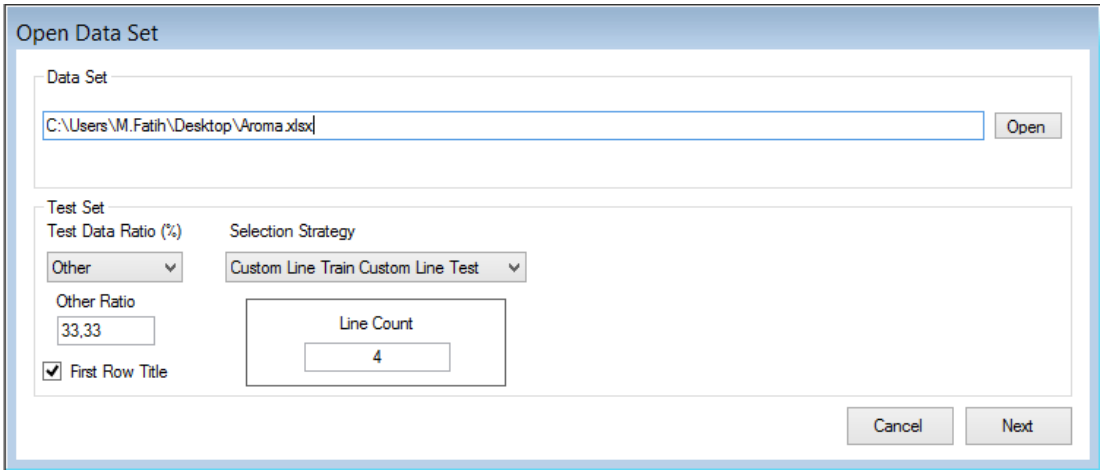
Geliştirilen yazılımın sınıf diyagramı incelendiğinde Şekil 3.2.'deki gibi bir yapı ortaya çıkmaktadır. MainWindow sınıfı OpenDataSet sınıfını çağırarak veri setinin seçilmesini sağlamakta, ChooseAttributes sınıfı çağrılarak nitelikler ayarlanmakta daha sonra ANNSetupWindow sınıfı çağrılarak yapay sinir ağının yapısı oluşturulup eğitim modeli ve parametreler ayarlanmaktadır. Algoritma çalıştığında Ann sınıfı çağrılacak ve seçilmiş olan eğitim türüne göre BeeColony ya da Genetic sınıfı çağrılacaktır.



Şekil 3.2. Geliştirilen programın sınıf diyagramı

3.1. Veri Setinin Okutulması

Geliştirilen uygulama YSA'yı eğitebilmesi için veri setine ihtiyacı vardır. Bu veri setini sağlama işi Şekil 3.3.'teki pencereden yapılır. Bu pencere açılabilmesi için ana penceredeki File > New menüsü takip edilmelidir. Şekil 3.3.'ten görüleceği üzere veri seti dosyasının seçildiği bir bölüm bulunmaktadır. Dosya türü olarak xls, xlsx ve txt kabul edilmektedir. Her bir nitelik bir sütuna gelecek şekilde yerleştirilen veri seti eğer nitelik başlıklarını içeriyorsa pencereden ilgili alan seçilmelidir.



Şekil 3.3. Uygulama yardımıyla veri setinin okutulması

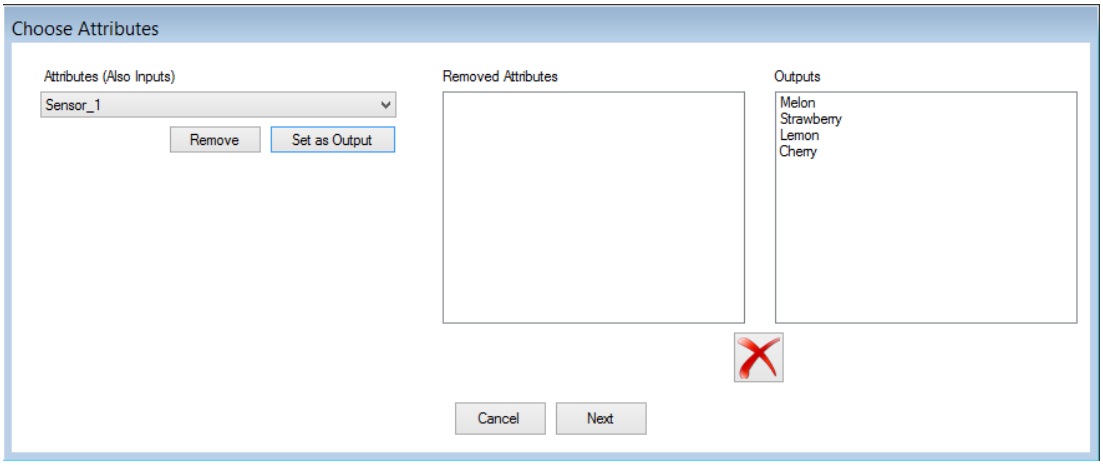
Verilen veri setinin yüzde kaçı eğitim, yüzde kaçı test olduğu pencerenin test bölümünden ayarlanmalıdır. Veri setinden test kısmının nasıl seçileceği de bu pencereden ayarlanmaktadır. Geliştirilen yazılımın sunduğu 3 seçenek bulunmaktadır.

- Rastgele
- Bir satır eğitim, bir satır test
- Belirli satır eğitim, belirli satır test

İlki yani Rastgele'nin seçilmesi halinde uygulama veri setinden belirlenen yüzdelik kadar test verisini rastgele seçecektir. İkinci seçenek veri setinden bir satır eğitim bir satır test şeklinde seçmeye başlayacak ve test yüzdeliğine erişildiğinde seçme işlemi bitecektir. Üçüncü seçenekte ise kullanıcı belirtmiş olduğu satır kadar eğitim ve yine o satır kadar test verisi seçilecektir.

Yazılıma okutulan veri seti Excel formatında olmayıp yazı dosyası (txt) formatında ise bu durumda nitelikler, aralarındaki boşluklar sayesinde ayırt edilebilecek ve okutulacaktır. Bu tez kapsamında yapılan uygulamaların tümünde veri seti excel formatında okutulmuştur. Şekil 3.3.'teki bütün ayarlamalar yapıldıktan sonra ileri butonuna basılır. Bundan sonra niteliklerin ayarlandığı pencere gelecektir. Şekil 3.4.'te görülen bu pencerede algoritmaya dâhil edilmeyecek nitelikler çıkartılabilir.

Hangi nitelikler çıktı niteliği ise bunun yine bu pencereden ayarlanması gerekir. Nitelikler seçildikten sonra ileri butonuna basılır.



Şekil 3.4. Niteliklerin ayarlandığı uygulama penceresi

3.2. Ağ Yapısının Tasarlanması

Ağ yapısının tasarlandığı pencerede yapay sinir ağına ait bütün detaylı ayarlamalar yapılabilir. Şekil 3.5.'te bir bölümü görülen bu pencerenin baş kısmında YSA'nın kaç adet gizli katmandan oluştuğu belirtilir. Bu ayarlandığı anda Şekil 3.6.'da diğer bölümü görülen pencerede YSA'nın ön izlemesi oluşacaktır. Şekil 3.5.'teki hidden layer bölümünden oluşturulan gizli katmanlara nöronlar eklenmelidir. Bunun için ilgili ara katman seçilip Add Node butonuna basılmalı ve Hidden Layer Details bölümünden nöronlar eklenmelidir. Her nöron eklendiğinde, ilgili nörona ait eşik değerine başlangıç değeri atanabilir ya da rastgele atanması sağlanabilir. Şekil 3.5.'teki Range of Values bölümünden ağırlıklar ve eşik değerlerinin alabileceği maksimum ve minimum değerler ayarlanır. Aktivasyon fonksiyonu olarak 3 farklı fonksiyon seçilebilir.

- Sigmoid
- Hiperbolik Tanjant
- Step

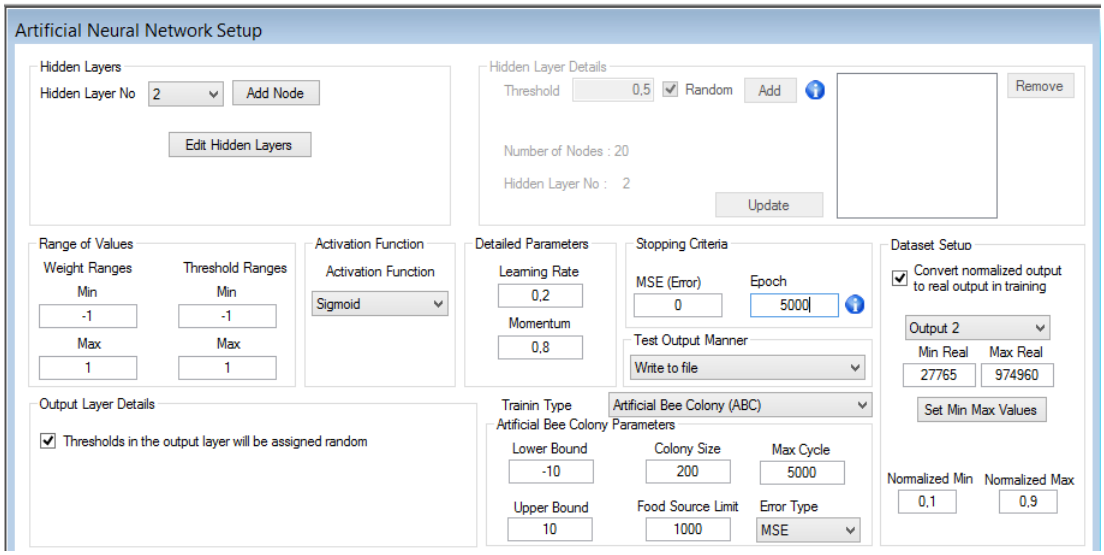
Bu üç farklı aktivasyon fonksiyonunun hesaplanma şekilleri sırasıyla (Denklem 3.1) (Denklem 3.2) ve (Denklem 3.3)'te görülebilir.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.2}$$

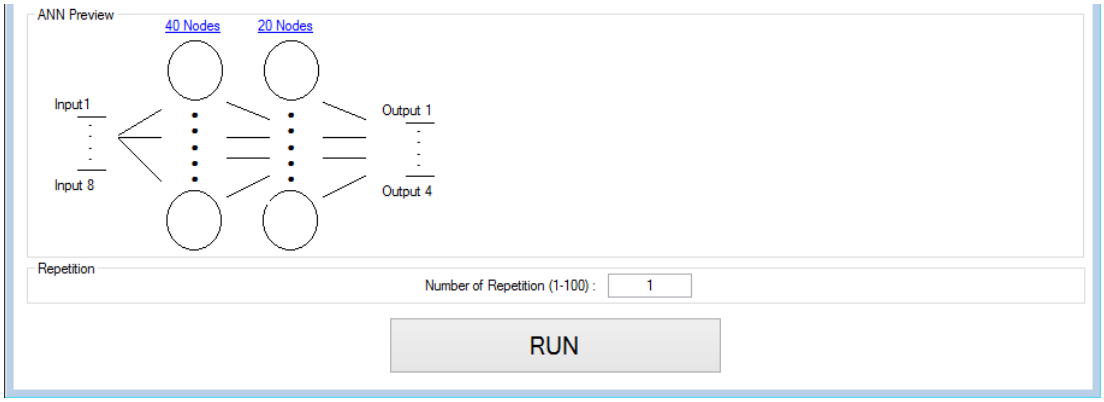
$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.3}$$

Geri yayılım algoritmasının parametreleri Şekil 3.5.'teki Detailed Parameters ve Stopping Criteria'dan ayarlanmalıdır. Bu bölümden öğrenme oranı, momentum değeri, durma kuralı olarak, MSE hata değeri veya kaç epoch çalışacağı belirlenebilir.



Şekil 3.5. Yapay sinir ağının tasarlandığı pencere

Eğer veri seti normalize edilmiş ise ve eğitimde gerçek değerler ile hata hesaplanması isteniyorsa, Dataset Setup bölümünden bu ayarlanabilir.



Şekil 3.6. Yapay sinir ağının ön izleme yapıldığı pencere

Tabi bunun yapılabilmesi için gerçek min. ve maks. değerleri programa verilmeli ve normalize edilmiş min. ve maks. değeri ayrıca belirtilmelidir.

3.3. Eğitim Yönteminin Seçilmesi

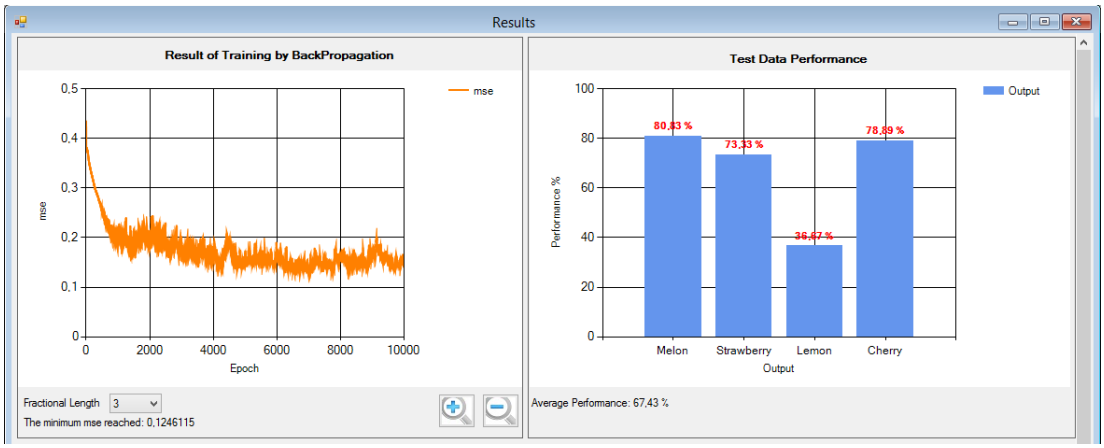
Program, eğitilmiş ağlara vermiş olduğu test verisinden elde ettiği sonuçları istenirse dosyaya yazabilir veya kendisi yorumlayarak performans grafiklerini oluşturabilir. YSA'nın eğitiminin optimize edilmesi ABC veya GA ile yapılabilir. Bunun Şekil 3.5.'teki pencereden seçilip belirtilmesi gerekmektedir. Hangi optimizasyon yöntemi seçilirse ilgili parametreler girilmesi için ekrana gelecektir. Yine bu bölümden hangi hata türüne göre eğitimin gerçekleşmesi ve global minimuma erişilmesi amaçlandığı gösterilebilir. Bu hata türleri üç adettir.

- MSE
- MAE
- R^2

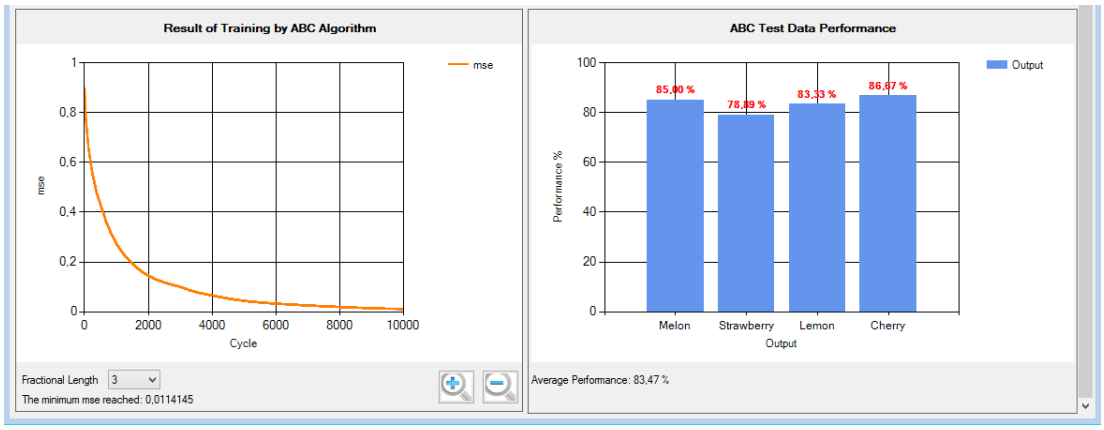
Algoritmanın kaç kez tekrar edeceği Şekil 3.6.'daki pencerenin alt kısmından yapılabilir. Bu penceredeki bütün ayarlamalar yapıldıktan sonra RUN butonuna basılıp algoritma çalıştırılır.

3.4. Çıktıların Alınması ve Sonuç Ekranlarının Gösterilmesi

Algoritmanın çalışması süresince anlık olarak algoritmanın bitimine ne kadar süre kaldığı programın bilgi bölümünde gösterilmektedir. Algoritma çalışmasını bitirdikten sonra eğer sonuçlandırma olarak dosya seçeneği seçilmiş ise test sonuçları dosyaya yazılacaktır. Bu durumda performans grafiklerini oluşturmak kullanıcıya kalmaktadır. Eğer performansın program tarafından hesaplanması seçilmiş ise program performansı hesaplayıp grafiksel olarak gösterecektir. Örnek sonuç grafikleri Şekil 3.7. ve Şekil 3.8.'de verilmiştir. Şekil 3.7.'deki YSA'nın BP ile eğitiminde MSE'nin grafiği ve test verisinin performansı, Şekil 3.8.'de ise YSA'nın ABC ile eğitiminde MSE'nin grafiği ve test verisinin performansı gösterilmiştir.



Şekil 3.7. Uygulamanın BP performans grafikleri



Şekil 3.8. Uygulamanın ABC performans grafikleri

Şekil 3.7. ve Şekil 3.8.'den de görüleceği üzere performansların ve sonuç grafiklerinin kolay bir şekilde okunabilirliği sağlanmıştır. Büyütme ve küçültme butonları ile ve ondalık hassasiyeti combobox'ı ile grafik daha anlaşılır bir şekilde tekrar oluşturulabilir. Yine arzu edilirse grafikler tek bir grafik performans göstergesi tek bir göstergede birleştirilip karşılaştırma olanağı oluşturulabilir.

BÖLÜM 4. MATERYAL VE METOTLAR

Bu bölümde tez kapsamında kullanılan yöntemler ve çalışma şekilleri anlatılmıştır. Yapay sinir ağının çalışma şekli ve nasıl sonuç üretildiği, geleneksel olarak kullanılan geri yayılım algoritması (BP) anlatılmıştır. Bu bölümde yeni sayılabilecek Yapay arı koloni algoritması (ABC) anlatılmış ve nasıl problemlere uygulandığı gösterilmiştir. Çok farklı çalışmalarda kullanılan ve başarılı sonuçlar elde edilen Genetik algoritma (GA) detaylı bir şekilde anlatılmıştır. Daha sonra tez çalışmasının asıl amacı olan yapay sinir ağının eğitiminin optimize edilme kısmı anlatılmıştır. İki farklı yöntem kullanılmış, yapay sinir ağı ABC ve GA ile ayrı ayrı eğitilmiştir. Eğitilme modelleri detaylı bir şekilde anlatılmıştır. Bölümün sonunda yapılan 4 çalışma ve elde edilen bulgular verilmiştir. Dört farklı aroma türünün sınıflandırıldığı çalışmada diğer eğitim modellerine göre daha başarı sağlayan YSA-ABC kullanılmış, ikili gaz ölçüm çalışmasında bazı gaz karışımlarında YSA-GA bazılarında ise YSA-ABC başarılı olduğu için her ikisi de kullanılmıştır. Son çalışmada farklı QCM sensör yapılarında 5 farklı gazın sınıflandırılması yapılmış ve diğerlerine göre başarılı performans gösteren YSA-ABC kullanılmıştır.

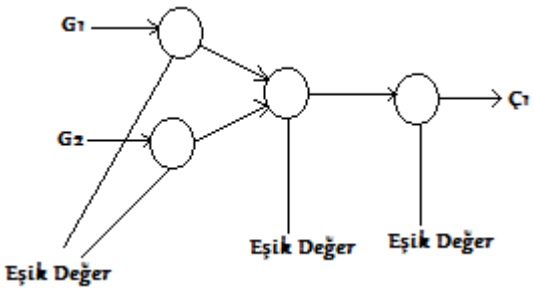
4.1. Yapay Sinir Ağları (YSA)

Yapay sinir ağları beynin çalışma prensibini uygulama amacıyla ortaya çıkmıştır. Basit işlem yapan elemanların (düğümlerin) birbirleriyle bağlı bir şekilde veri işleme ve hesaplama yapabilme kabiliyetine sahiptirler [123]. YSA, katmanlar, katmanları oluşturan nöronlar ve nöronların birbirine ağırlıklar ile ifade edilen bağlarla bağlanması ile oluşur. Öğrenme aşamasında geleneksel yöntem olarak geri yayılım (BP) algoritması kullanılır. YSA 3 temel katmandan oluşur. Bunlar; girdi, ara ve çıktı katmanlarıdır. Ara katman bir veya birden fazla katmandan oluşuyor olabilir. Yapay

sinir ağı ağırlıkların güncellenmesi için BP algoritmasını kullanır. Bu algoritmanın çalışması için önce ağda ileri besleme yapılmalıdır.

4.1.1. İleri doğru hesaplama (feedforward)

BP'nin uygulanıp ağırlıkların güncellenebilmesi için önce ağda ileri doğru hesaplama ya da besleme (feedforward) denilen işlem yapılmalıdır. Bu işlem için Eğitim setindeki bir örnek ya rastgele ya da sıradan seçilir. Bu seçilen veri örneğin Şekil 4.1.'deki gibi tasarlanmış olan YSA'ya verilir. G1 ve G2'deki girdiler girdi katmanındaki ağırlıklar ile çarpılıp ara net değerleri hesaplanır.



Şekil 4.1. Örnek bir yapay sinir ağı

Net değerlerinin hesaplanması için (Denklem 4.1)'deki formül kullanılmaktadır. Bu formülde ilgili nörona bağlı ağırlıklar A_{kj} ile çıktı değerleri $Ç_k^i$ çarpılıp toplanmaktadır.

$$NET_j^a = \sum_{k=1}^{n} A_{kj} Ç_k^i \tag{4.1}$$

Hesaplanan Net değerler bir aktivasyon fonksiyonundan geçirilerek ara çıkış değerleri hesaplanır. Ara çıkış değerleri, ara katmandan çıktı katmanına bağlanan ağırlıklar ile çarpılıp, çıkış net değerleri hesaplanır. Aynı şekilde bu Net değerleri de aktivasyon fonksiyonundan geçirilip ağın çıktısı hesaplanır (Denklem 4.2). Buradaki β_j^a değeri ilgili nörondaki eşik değeri olmaktadır.

$$Ç_j^a = \frac{1}{1 + e^{-(NET_j^a + \beta_j^a)}} \tag{4.2}$$

4.1.2. Geri yayılım algoritması (BP)

BP ile eğitimin kullanıldığı YSA'da ağırlıklar ve eşik değerleri ilk başta rastgele belirlenir. Rastgele belirlenme aralığı kullanıcı tarafından sınırlandırılabilir. İleri besleme sonucu ağda oluşan çıktı $Ç_m$ ile beklenen çıktı B_m arasındaki fark hata E_m şeklinde tanımlanır. Bu hata değeri her iterasyonda hesaplanır (Denklem 4.3).

$$E_m = B_m - Ç_m \tag{4.3}$$

Epoch ise bütün eğitim veri setinin ağa verilmesiyle oluşan durumdur. Her bir epoch'taki hata değerini bulmak için her iterasyonda elde edilen hata değerlerinin karelerin toplamını bulmak gerekir (Denklem 4.4).

$$TH = \frac{1}{2}\sum_m E_m^2 \tag{4.4}$$

Epoch Hata = TH / Veri sayısı

En son epoch'ta bulunan hata değeri (MSE) eğitimin sonunda elde edilen MSE değeri olup bu değerin sıfıra yakın olması beklenmektedir.

Her iterasyonda bulunan hata kabul edilebilir hata oranından daha büyük ise ağırlıklar ve eşik değerleri güncellenip tekrar ileri besleme yapılarak hata ölçümüne devam edilir. İlk önce çıktı katman ile ara katman arasındaki ağırlıklar güncellenir. Bu güncelleme yapılabilmesi için önce, (Denklem 4.5) gösterildiği gibi ağırlıkların değişim miktarları bulunmalıdır. Bu hesaplama yapılırken momentum katsayısı α ve öğrenme katsayısı λ hesaba dâhil edilir. Öğrenme katsayısı her iterasyondaki öğrenme oranını belirleyecektir. Bu oran ne kadar küçük verilirse, öğrenme o kadar yavaş gerçekleşir. Momentum katsayısının kullanılması ise ağı lokal minimuma düşmekten kurtarmaya çalışır. Çalışmalarda genellikle öğrenme katsayısı küçük verilirken momentum buna karşılık büyük bir değer verilmektedir. Böylelikle

öğrenme yavaş ve lokal minimuma takılma riski düşürülmüş olacaktır. δ_m ifadesi lokal gradyanı gösterir ve (Denklem 4.6)'daki gibi hesaplanır.

$$\Delta A_{jm}^{a}(t) = \lambda \delta_m Ç_j^a + \alpha \Delta A_{jm}^{a}(t-1) \tag{4.5}$$

$$\delta_m = f'(NET) E_m \tag{4.6}$$

(Denklem 4.6)'daki $f'(NET)$ ifadesi, aktivasyon fonksiyonunun türevidir ve eğer aktivasyon fonksiyonu Sigmoid olarak seçilirse, denklemin açılımı (Denklem 4.7)'deki gibi olacaktır.

$$\delta_m = Ç_m(1 - Ç_m).E_m \tag{4.7}$$

Ağırlık değişim miktarları hesaplandıktan sonra, ağırlıklar (Denklem 4.8) yardımıyla güncellenir.

$$A_{jm}^{a}(t) = A_{jm}^{a}(t-1) + \Delta A_{jm}^{a}(t) \tag{4.8}$$

Aynı şekilde eşik değerleri de ağırlıklar gibi güncellenecektir (Denklem 4.9).

$$\begin{aligned} \Delta \beta_m^{ç}(t) &= \lambda \delta_m + \alpha \Delta \beta_m^{ç}(t-1) \\ \beta_m^{ç}(t) &= \beta_m^{ç}(t-1) + \Delta \beta_m^{ç}(t) \end{aligned} \tag{4.9}$$

Ara katman ile girdi katmanlar arasındaki ağırlıkların güncelleştirilmesi işlemi de (Denklem 4.8) ve (Denklem 4.9) ile aynı mantıkta yapılıp bu sefer çıktı ifadeleri ara katmanın çıktıları olmaktadır. Ağırlıkların değişim miktarları için (Denklem 4.10) kullanılmaktadır.

$$\Delta A_{kj}^{i}(t) = \lambda \delta_j^a Ç_k^i + \alpha \Delta A_{kj}^{i}(t-1)$$
$$\delta_m = f'(NET) \sum_m \delta_m A_{jm}^a \qquad (4.10)$$
$$\delta_j^a = Ç_j^a \left(1 - Ç_j^a\right) \sum_m \delta_m A_{jm}^a$$

4.2. Yapay Arı Koloni Algoritması (ABC)

Yapay arı koloni algoritması bir sürü optimizasyon algoritması olarak arıların nektar arama davranışlarını örnek alır. İlk olarak Karaboğa tarafından geliştirilen ABC algoritması 3 çeşit arı türünü içerir [82]. Bunlar; işçi, gözcü ve kaşif arılardır. Bu algoritmanın başlangıçta bazı varsayımları bulunmaktadır. Bu varsayımlar, her bir kaynak sadece bir görevli arı tarafından kontrol edilir. İşçi arıların sayısı görevi olmayan arıların sayısına eşittir. Kaynağı tükenen işçi arı kaşif arıya dönüşmektedir. ABC'nin çalışmasını anlatan algoritma Şekil 4.2.'de verilmiştir.

```
Yiyecek kaynaklarını üret
do
   İşçi arıları gönder
   Olasılıkları hesapla
   Gözcü arıları, gelen olasılık bilgilerine göre gönder
   En iyi sonucu sakla
   Kaşif arıları  gönder
while(cycle <= max_cycle)
```

Şekil 4.2. ABC'nin çalışmasını özetleyen algoritmanın sözde kodu

Üretilen yiyecek kaynaklarındaki nektarlar, işçi arılar tarafından toplanır. Gözcü arılar, işçi arılardan gelen bilgiler doğrultusunda yeni nektar arayışına çıkarlar. Kaşif arılar ise rastgele yeni nektar bölgeleri aralar ve nektar bölgesi bulduklarında işçi arıya dönüşürler. İşçi arı nektar kaynağını bitirdiğinde kaşif arıya dönüşür. Yiyecek kaynakları, ABC algoritmasının ilk adımında, verilen alt ve üst sınır içerisinde rastgele üretilir. (Denklem 4.11)'den de görülebilen bu üretim işlemi rastgele sayı üreten *rand* fonksiyonundan yararlanır. (Denklem 4.11)'de *i* yiyecek kaynağını, *j* optimize edilecek parametre sayısını ifade eder.

$$x_{ij} = x_{\min j} + rand(0,1) * (x_{\max j} - x_{\min j}) \qquad (4.11)$$

İşçi arılar nektar kaynağı tükendiğinde, kâşif arılara dönüşürler ve yeni nektar kaynağını, gözcü arılardan gelen mevcut nektar kaynak bilgisini referans alarak belirlerler. Bu işlemi yaparken (Denklem 4.12)'den faydalanırlar. (Denklem 4.12)'de verilen v_{ij} bulunacak olan yeni nektar yerini ifade eder. Rastgele seçilen indeks olan i yiyecek kaynağını, j optimize edilecek parametre sayısını ifade eder. Rastgele üretilen bir sayı olan ϕ_{ij} komşu yiyecek kaynaklarının üretimini kontrol eder.

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{4.12}$$

Yeni bulunan yiyecek kaynağının uygunluk değeri (Denklem 4.13)'te verilen uygunluk fonksiyonu ile hesaplanır. Eğer yeni bulunan yiyecek kaynağındaki nektar miktarı mevcut yiyecek kaynağındaki nektar miktarından düşük ise yiyecek kaynağı değişmez ve yeni yiyecek kaynağı arayışına devam edilir.

$$fitness_i = \begin{cases} \frac{1}{1+f_i} & f_i \geq 0 \\ \frac{1}{abs(f_i)} & f_i < 0 \end{cases} \tag{4.13}$$

(Denklem 4.13)'te verilen f_i, i çözümünün uygunluk değeridir ve i. konumdaki yiyecek kaynağındaki nektar miktarı ile ilişkilidir. ABC algoritması belirlenen iterasyon sayısı kadar çalışır ve global minimumu bulmayı hedefler.

4.3. Genetik Algoritma (GA)

Genetik algoritma, gerçek hayattaki doğal seçim mekanizması ve gen yapılarını örnek alan çaprazlama ve mutasyon içeren sezgisel arama algoritmasıdır [68]. Her bir iterasyonda çaprazlama ve mutasyon sonucu oluşan yeni güçlü bireyler bir sonraki nesillere aktarılır. Kromozomların çaprazlama ve mutasyona uğrayabilme olasılıklarını ayarlayan, çaprazlama ve mutasyon oranları bulunmaktadır.

4.3.1. Çaprazlama

Genetik algoritmada bilgi taşıyan genler kromozomları oluşturur. Güçlü nesiller elde edebilmek için kromozomlardaki bilgilerin birbirlerine aktarılması gerekir. Bundan dolayı çaprazlama yapılır. Literatürde birçok farklı çaprazlama tekniği kullanılmıştır. Fakat 3 tanesine sıklıkla rastlanılır.

- Tek noktalı çaprazlama
- İki noktalı çaprazlama
- Kes ve ekle çaprazlama

Bu tez kapsamında tek noktalı çaprazlama tekniği kullanıldığı için sadece bu teknikten bahsedilecektir.

4.3.1.1. Tek noktalı çaprazlama

Bu çaprazlama tekniğinde seçilen kromozomlar, rastgele belirlenmiş olan nokta baz alınarak çaprazlama uygulanır. Örneğin Şekil 4.3.'te verilen kromozomlar, kırmızı ile rastgele belirlenen nokta etrafında çaprazlanıyorlar. Oluşan yeni bireyler sağ tarafta görülmektedir.

Şekil 4.3. Tek noktalı çaprazlama örneği

4.3.2. Mutasyon

Belli bir iterasyondan sonra, yapılan çaprazlamalar sonucu kromozomlardaki genlerin birbirini tekrar edebilme olasılığı bulunmaktadır. Bu durumu ortadan kaldırmak ve kromozom çeşitliliğini sağlamak için, bazı kromozomlar mutasyona

tabi tutulur. Örneğin Şekil 4.4.'te kromozomun sadece iki geni mutasyona uğruyor (değer sıfır ise bir, bir ise sıfır oluyor).

Şekil 4.4. Mutasyona uğrayan kromozom örneği

4.3.3. Seçim yöntemi

Yeni nesle aktarılacak, çaprazlamaya girmeyecek veya girecek bireyleri seçmek için çeşitli yöntemler kullanılmaktadır. Sıklıkla kullanılanlar şu şekilde listelenebilir.

- Turnuva seçimi
- Sabit durum seçimi
- Elitist seçim

Bu tez kapsamında elitist seçim kullanıldığı için bu seçim yönteminden bahsedilecektir.

4.3.3.1. Elitist seçim

Çaprazlama ve mutasyon sonucunda oluşan ve daha önce mevcut olan bireyler arasından, popülasyon sayısı kadar en iyi uygunluk değerine sahip bireyler bir sonraki nesle aktarılmak üzere seçilir.

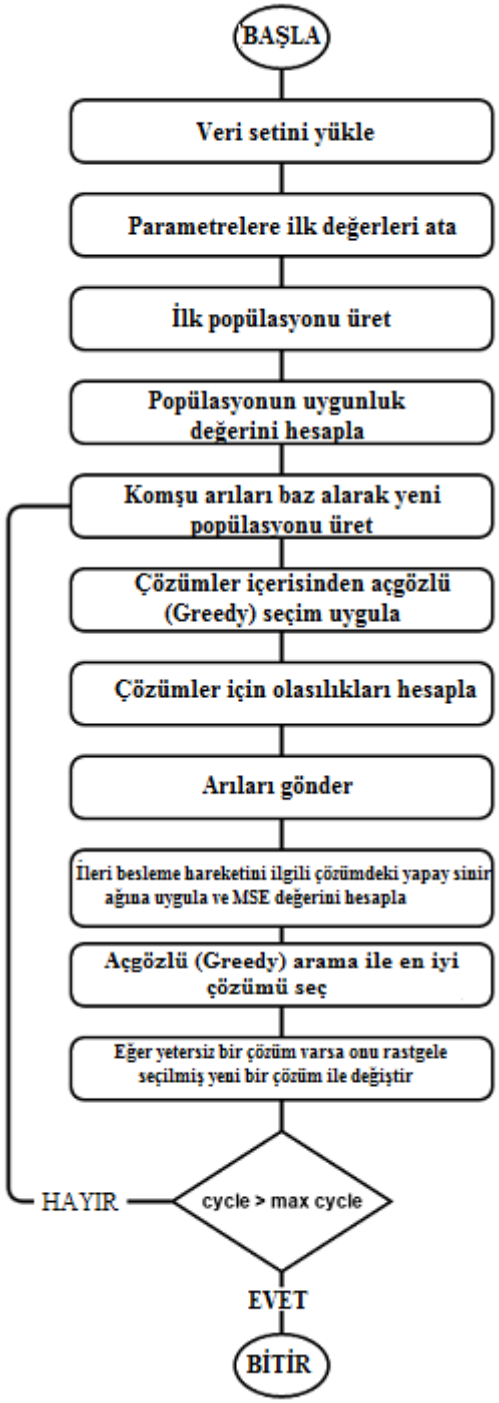
Bu bölümde anlatılan Genetik algoritmanın çalışma prensibi Şekil 4.5.'te verilmiştir. Çalışma prensibinden görüleceği üzere, çaprazlamaya girmeyecek ve yeni nesle aktarılacak kromozomlar elitist yöntemi ile seçilmektedir.

```
BASLA
    Kromozomları rastgele değerler ile oluştur.
   for iterasyon=1 to max_iterasyon do
      mevcut popülasyondan elitist seçim yap
     geri kalan kromozomlardan çaprazlama oranına girenleri çaprazla
     mutasyon oranına girenleri mutasyona uğrat
    yeni oluşan ve mevcut popülasyondan, popülasyon sayısı kadar en iyi uygunluk değerine sahip
    kromozomları bir sonraki nesle aktar
  end
BİTİR
```

Şekil 4.5. Genetik algoritmanın çalışma prensibi

4.4. Yapay Sinir Ağlarının ABC ile Eğitilmesi (YSA-ABC)

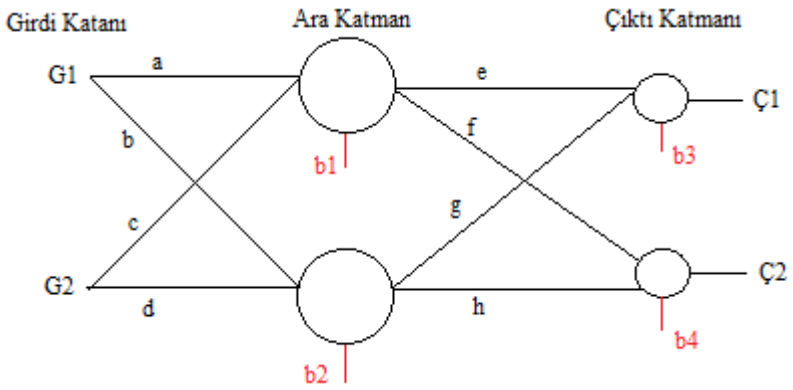
Yapay sinir ağları her ne kadar çok geniş kullanım alanına sahip olsa da eğitim setini ezberleme veya lokal minimuma takılma gibi problemleri bulunmaktadır [1]. Bu problemi gidermek adına literatürde yapılan çok fazla çalışma bulunmaktadır. Literatür incelendiğinde bunu daha çok melez algoritmalar kullanarak yapıldığı görülecektir. Bu tez kapsamında yapay sinir ağının eğitim kısmı geri yayılım (BP) algoritması yerine ABC algoritması ile eğitilmiştir. ABC'nin seçilmesinin nedeni YSA'ya uyarlanması kolay ve gerekli parametre sayısının az olmasından kaynaklanmaktadır. ABC algoritmasının keşif (exploration) yönünden güçlü olduğundan lokal minimuma takılmadan global minimuma erişmeyi sağlayacaktır. YSA'nın ABC ile nasıl eğitildiğini gösteren akış diyagramı Şekil 4.6.'da verilmiştir.



Şekil 4.6. YSA-ABC'nin çalışmasını gösteren akış diyagramı

ABC, ilk etapta rastgele üretilmiş ağırlıklar ve eşik değerlerini bir dizi içerisinde alıp bu değerleri optimize edip en uygun ağırlık ve eşik değerlerini bulmaktadır. Burada uygunluğu, her iterasyonda bulmuş olduğu ağırlık ve eşik değerlerini ağa yerleştirerek ileri besleme (feedforward) hareketi ile hesaplamaktadır.

Örneğin Şekil 4.7.'deki yapay sinir ağı ele alınırsa ABC bu ağın ağırlıklarını ve eşik değerlerini optimize etmeye çalışacaktır. Algoritmaya verilecek dizi, Şekil 4.7. için [a, b, c, d, b1, b2, e, f, g, h, b3, b4] olacaktır.



Şekil 4.7. YSA-ABC için örnek bir yapay sinir ağı

Her iterasyonda ağ üzerinde yapılan ileri beslemeden oluşan hata, kullanıcının belirlemiş olduğu hata türüne göre hesaplanmakta ve uygunluk fonksiyonunun döndürdüğü değer olmaktadır. ABC ile yapılan eğitimde 3 farklı hata türü kullanılabilmektedir. Bunlardan MSE, hata karelerin toplamının eğitim veri seti sayısına bölümü ile elde edilir (Denklem 4.14). *T* gerçek çıktıyı, *O* ağda hesaplanan çıktıyı, *n* ise veri seti sayısını ifade eder.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(T - O)^2 \tag{4.14}$$

MAE ise, hatanın hedef çıktıya bölümünden elde edilen değerlerin toplamının veri seti sayısına bölünmesi ile elde edilir (Denklem 4.15). *T* gerçek çıktıyı, *O* ağda hesaplanan çıktıyı, *n* ise veri seti sayısını ifade eder.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\frac{|T - O|}{T} \tag{4.15}$$

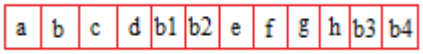
R^2'de ise bağımsız değişken ve gerçek çıktıların ortalaması kullanılır. Hesaplama formülü (Denklem 4.16)'da verilmiştir. *T* gerçek çıktıyı, T_{ort} gerçek çıktıların

ortalamasını, O ağda hesaplanan çıktıyı, n veri seti sayısını, k ise bağımsız değişken sayısını ifade eder.

$$R^2 = 1 - \left[1 - \frac{\sum(T-O)^2}{\sum(T-T_{ort})^2}\right]\frac{n-1}{n-k-1} \tag{4.16}$$

4.5. Yapay Sinir Ağlarının GA ile Eğitilmesi

Bu tez çalışması kapsamında YSA'nın eğitimi Genetik algoritma ile de yapılmış ve başarılı sonuçlar alınmıştır. ABC'de diziyi oluşturan ağırlık ve eşik değerleri GA'da kromozomları oluşturacaktır. Şekil 4.7.'deki yapay sinir ağı ele alınırsa, kromozomu oluşturan genler Şekil 4.8.'deki gibi olacaktır.



Şekil 4.8. YSA-GA için örnek bir kromozom

Çaprazlama ve mutasyon sonucu oluşan kromozomlardaki genler teker teker ağın üzerine yerleştirilir. İleri besleme hareketi yapılarak hata değeri ölçülür. Hata değeri en düşük çıkan popülasyon sayısı kadar kromozom seçilerek bir sonraki nesle aktarılır. Bu şekilde iterasyon sayısı kadar nesil üretimi olacaktır. İterasyon bittiğinde en iyi genlere sahip kromozom optimal çözüm olmaktadır. YSA-GA'nın çalışma prensibini ifade eden algoritma Şekil 4.9.'da verilmiştir.

```
BASLA
    Kromozomları rastgele değerler ile oluştur.
   for iterasyon=1 to max_iterasyon do
     mevcut popülasyondan elitist seçim yap
     geri kalan kromozomlardan çaprazlama oranına girenleri çaprazla
     mutasyon oranına girenleri mutasyona uğrat
    her kromozomu tasarlanan ağa yerleştir, ağda ileri besleme yap
    ileri beslemelerden aktivasyon fonksiyonlarını hesapla
    popülasyon sayısı kadar en iyi aktivasyon değerine sahip
    kromozomları bir sonraki nesle aktar
  end
BİTİR
```

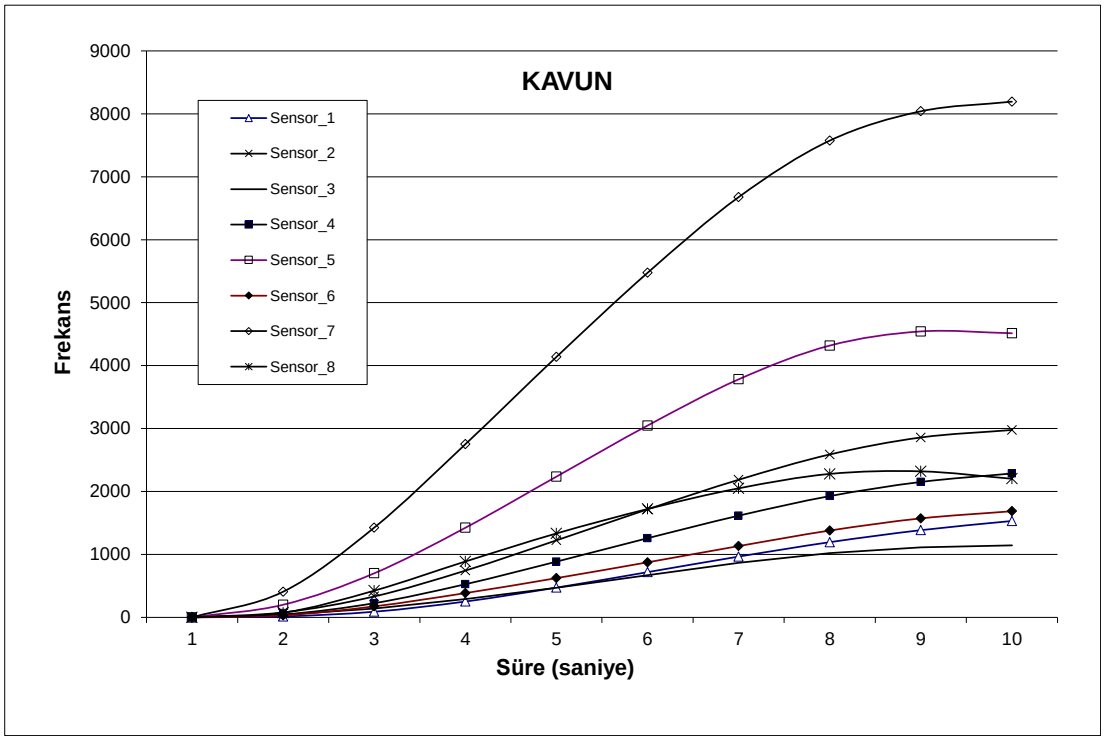
Şekil 4.9. YSA-GA algoritmasının çalışma prensibi

4.6. Dört Farklı Aroma Verisinin ABC Tabanlı YSA ile Sınıflandırılması

Meyve aromalarının sınıflandırılabilmesi, sanayi alanında meyve tanımlama ve kalite belirleme çalışmalarında büyük kolaylıklar sağlayacaktır. Bu kapsamda, kokuları birbirine yakın dört farklı meyve aromasından (kavun, kiraz, limon, çilek) elektronik burun yardımıyla veri elde edilmiş ve sınıflandırma çalışması yapılmıştır. Kokuları birbirine yakın olan bu meyvelerin başarılı sınıflandırılabilmesi, kullanılan yöntemin birbirine yakın farklı kokularda da başarı göstermesi mümkündür.

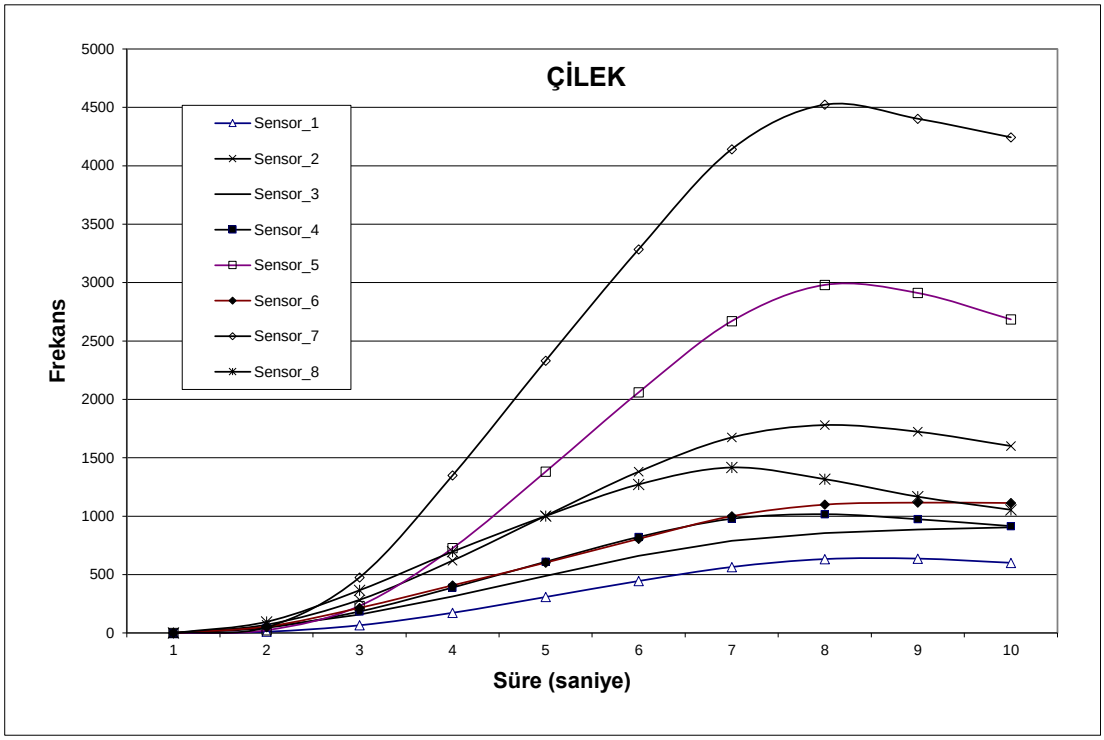
4.6.1. Dört farklı aroma verisinin elde edilmesi

Dört farklı meyve aromasından (kavun, kiraz, limon, çilek) gaz verisi, MOSES II elektronik burunu kullanılarak elde edilmiştir. Meyve örnekleri 25 ml cam şişelerde, 153◦ C sıcaklıkta, elektronik buruna koklatılmıştır. Deney süresince sıcaklık ve nem sensörleride aktif olarak çalışmıştır. Her ölçüm aynı oranlarla dört kez tekrarlanmıştır. Her meyve örneği için, örnek sensöre 10 saniye süresince koklatılmış ve ardından 10 dakika saf hava ile sensörler temizlenmiştir. Elektronik burunun içerdiği sekiz farklı sensörün, kavun meyvesine verdiği tepki Şekil 4.10.'da verilmiştir.



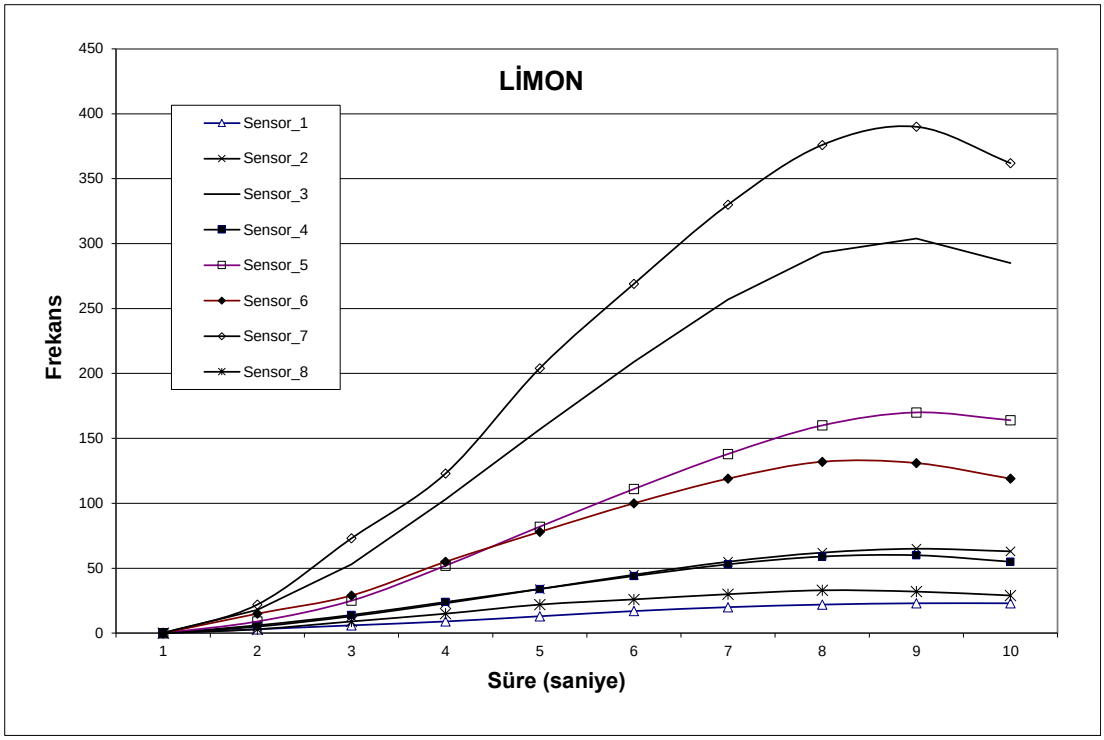
Şekil 4.10. Sekiz farklı gaz sensörünün kavun meyvesine verdiği tepki

Sensörlerin çilek meyvesine verdiği tepkiye bakıldığında frekans değerlerinin kavun meyvesine göre daha aşağıda olduğu görülecektir (Şekil 4.11.).

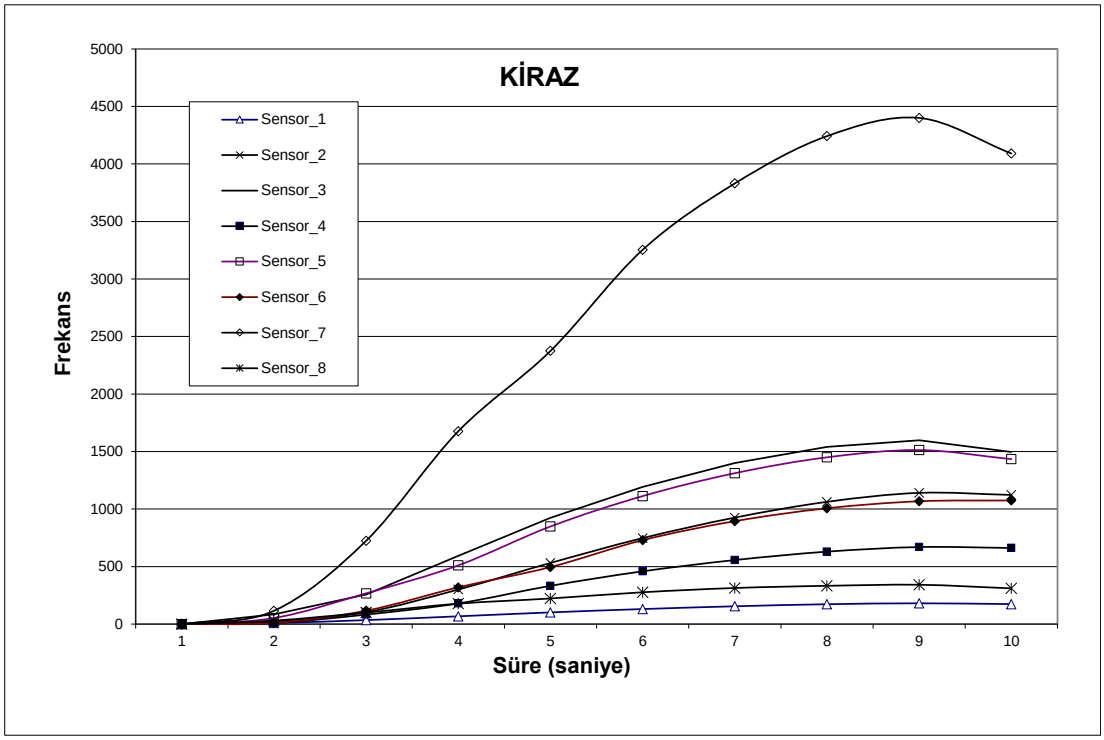


Şekil 4.11. Sekiz farklı gaz sensörünün çilek meyvesine verdiği tepki

Sensörlerin limon meyvesine verdiği tepki diğer üç meyveye göre en düşük oranda olduğu Şekil 4.12.'de görülmektedir.



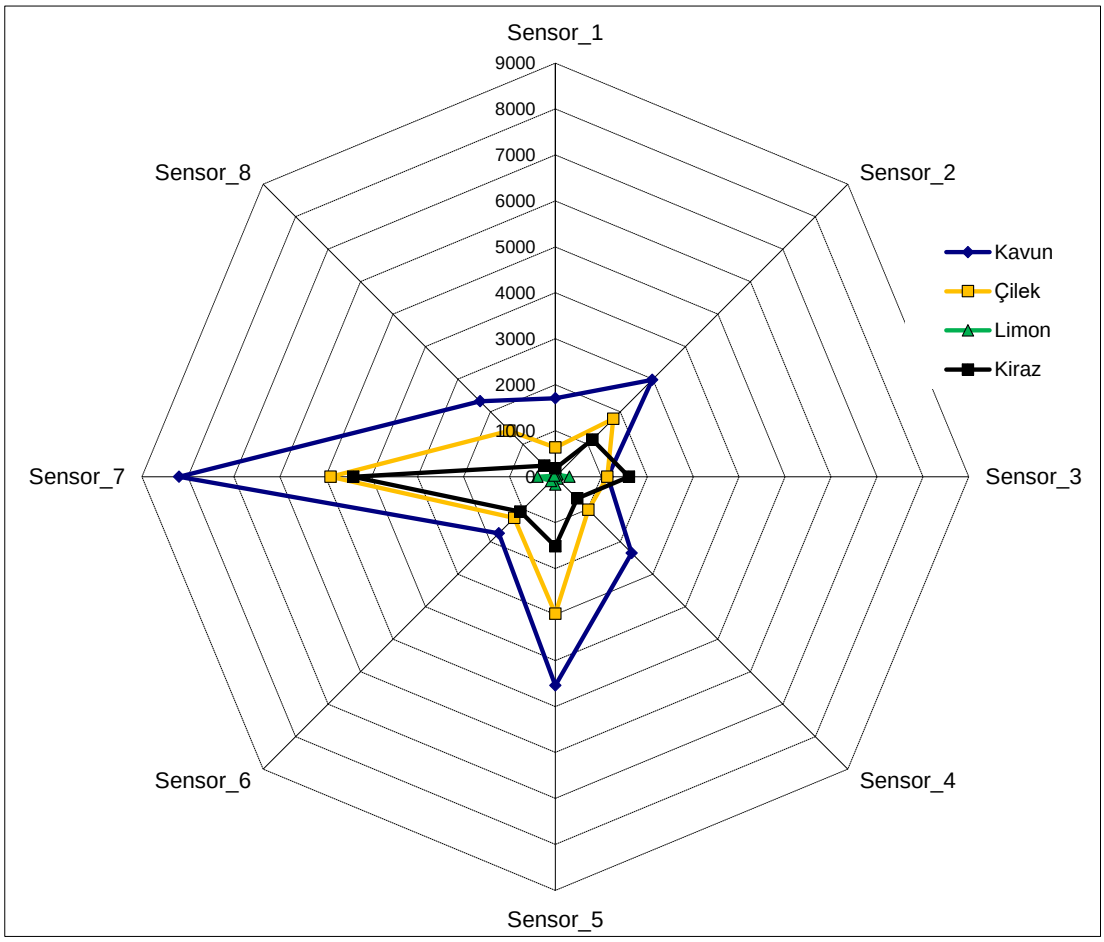
Şekil 4.12. Sekiz farklı gaz sensörünün limon meyvesine verdiği tepki



Şekil 4.13. Sekiz farklı gaz sensörünün kiraz meyvesine verdiği tepki

Şekil 4.13.'e bakıldığında kiraz meyvesine sekiz sensörün verdiği tepkiden, ilk sensörün tepkisi, çilek meyvesine verdiği tepkiye benzemektedir.

Sensörlerin meyve örneklerine verdikleri 10 saniyelik tepkiden sadece tepe nokta değeri alınıp sekiz sensörün 4 meyve ile ilgili radar grafiği oluşturulmuş ve Şekil 4.14.'teki grafik elde edilmiştir. Bu oluşan görüntü meyve aromalarının parmak izleri olarak tabir edilir. Şekil 4.14.'te görülen meyve parmak izlerinin birbirinden kolaylıkla ayırt edilebilecek yapıda olduğu görülmektedir.



Şekil 4.14. Sensör tepkilerinin tepe nokta değerlerinin oluşturduğu radar grafiği

Şekil 4.14.'ten de görüleceği üzere sensör değerleri büyük sayılardan oluşmaktadır. Fakat bu tez çalışması kapsamında yapay sinir ağı eğitiminde aktivasyon fonksiyonu olarak sigmoid fonksiyonu kullanılmıştır. Sigmoid fonksiyonu girdi olarak sadece [0,1] aralığında değer kabul eder. Değerlerin [0,1] aralığına indirgenmesi için sütun temelli min-max normalizasyonu uygulanmıştır (Denklem 4.17). Burada x_i

normalize edilecek sayıyı, x_{min} sayıların en küçüğünü, x_{max} sayıların en büyüğünü ifade etmektedir..

$$x_{i,0-1} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{4.17}$$

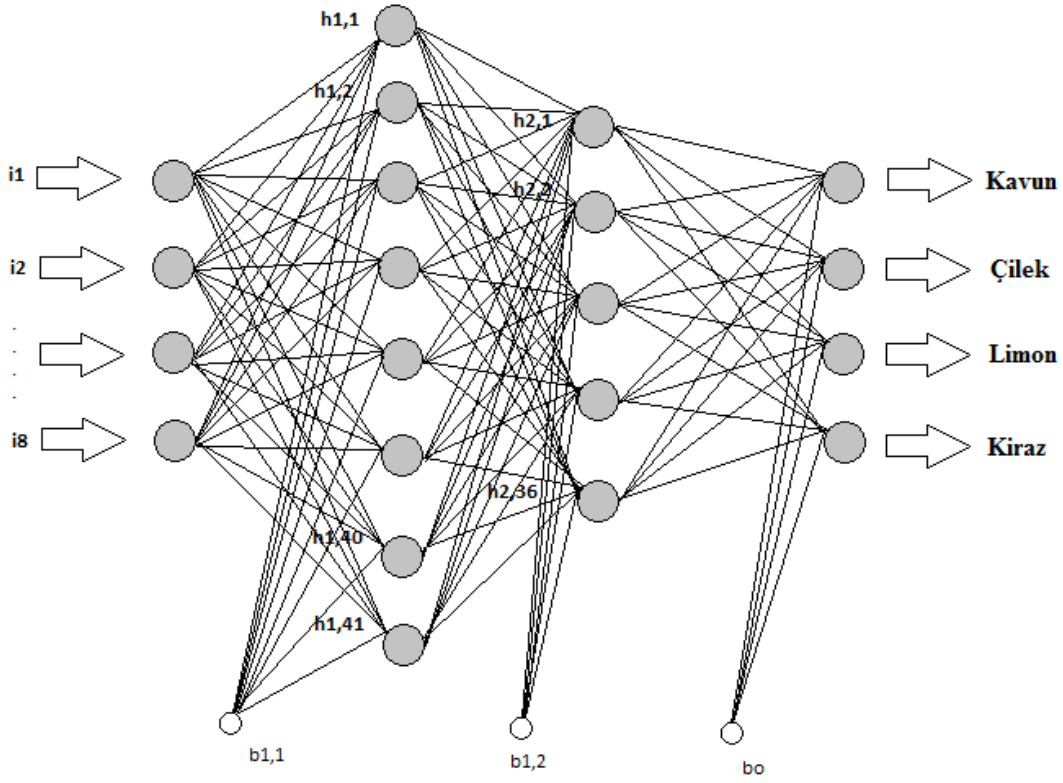
Veri setinin oluşturulması tepe noktalarının seçilmesi ve min-max normalizasyonunun yapılması, Tablo 4.1.'deki kavun meyvesinin sensör 1'e verdiği tepki üzerinden örnek olarak gösterilmiştir. Tablo 4.1.'deki ölçümde frekans tepe noktasına 10 saniyede erişmiştir. Normalizasyon sütunu, genel veri setinde kavun meyvesinin 1 değerini aldığı değerleri gösterir.

Tablo 4.1. Sensör 1'in kavun meyvesine vermiş olduğu tepki

Saniye	Frekans								**Norm.**
1	0	0	0	0	0	0	0	0	**0,0000**
2	0	0	0	0	0	0	0	14	**0,0091**
3	0	0	0	0	0	0	14	89	**0,0581**
4	0	0	0	0	0	14	89	251	**0,1639**
5	0	0	0	0	14	89	251	472	**0,3083**
6	0	0	0	14	89	251	472	718	**0,4690**
7	0	0	14	89	251	472	718	965	**0,6303**
8	0	14	89	251	472	718	965	1194	**0,7799**
9	14	89	251	472	718	965	1194	1385	**0,9046**
10	89	251	472	718	965	1194	1385	1531	**1,0000**

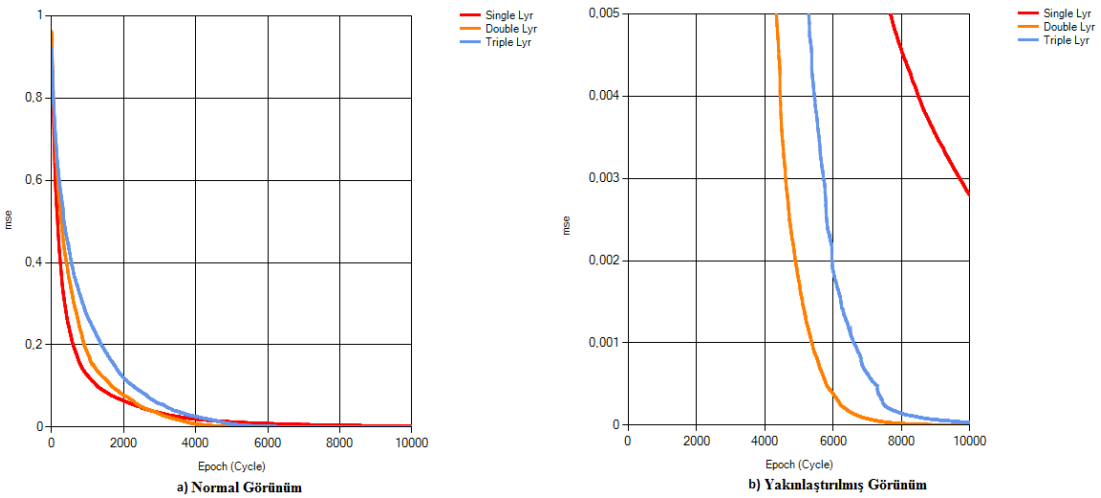
4.6.2. Tasarlanan yapay sinir ağı

Dört farklı aroma tipinin sınıflandırılması için 4 katmanlı yapay sinir ağı tasarlanmıştır. Şekil 4.15.'te verilen bu YSA, bir giriş, iki gizli bir de çıkış katmanı içermektedir. Giriş katmanında sekiz adet sensörün vermiş olduğu sekiz adet giriş bulunmaktadır. İlk gizli katmanda 41 ikinci gizli katmanda 36 adet nöron bulunmaktadır. Çıkış katmanında dört aromanın ifade edildiği dört adet çıkış bulunmaktadır. YSA'ya gelen veri hangi sınıfı temsil ediyorsa ilgili aroma çıkış değeri 1 diğerleri 0 olmaktadır.



Şekil 4.15. Dört farklı aroma türünün sınıflandırılması için tasarlanan YSA

YSA yapısında iki gizli katman seçilmesinin sebebi yapılan testlerde en başarılı sonuçları vermiş olmasıdır. Katman sayısını ikiden fazla arttırmak hata oranın artmasına neden oluştur. Tek, iki ve üç gizli katmanın test edildiği ve MSE grafiklerinin oluşturdukları yapı Şekil 4.16.'da verilmiştir.



Şekil 4.16. Gizli katman sayısının MSE üzerindeki etkisi

Şekil 4.15.'teki YSA hem BP hem de ABC ile ayrı ayrı eğitilmiştir. BP ile eğitilirken Tablo 4.2.'deki parametreler kullanılmıştır.

Tablo 4.2. YSA'nın BP ile eğitilirken kullanılan parametreler

Parametre	Değer
Ağırlık aralıkları	[-1,1]
Eşik değer aralıkları	[-1,1]
Aktivasyon fonksiyonu	Sigmoid
Öğrenme katsayısı	0,2
Momentum değeri	0,8
Durma kuralı (epoch)	10000

YSA'nın ABC ile eğitilirken kullanılan parametreler Tablo 4.3.'te verilmiştir. ABC'nin üreteceği parametre değerleri [-10,10] arasında olacaktır. Üretilecek maksimum arı sayısı 200'dür.

Tablo 4.3. YSA'nın ABC ile eğitilirken kullanılan parametreler

Parametre	Değer
Alt sınır değeri	-10
Üst sınır değeri	10
Koloni boyutu	200
Nektar sayısı	1.000
İterasyon sayısı	10.000

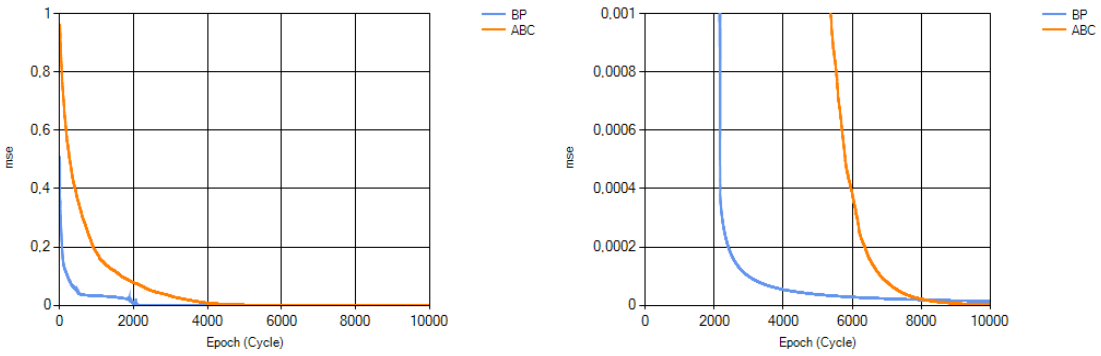
Şekil 4.15.'teki YSA yapısı kullanıldığı için ABC ile eğitildiğinde, ABC 2029 adet parametreyi optimize etmeye ve global minimumu bulmaya çalışacaktır. Bu 2029 adet parametrenin nereden geldiği Tablo 4.4.'te verilmiştir. Her iterasyonda elde edilen bu parametreler Şekil 4.15.'teki YSA'nın üzerine yerleştirilip ileri yönde hareket yapılıp hata değeri hesaplanacaktır. ABC, bu hata değerini bir sonraki iterasyonda aşağı çekmeye çalışacaktır.

Tablo 4.4. ABC'nin optimize edeceği parametre sayısı

Açıklama	Parametre sayısı
Girdi katmanı ile birinci gizli katman arası	8x41=328
İlk gizli katman ile ikinci gizli katman arası	41x36=1476
İkinci gizli katman ve çıktı katman arası	36x4=144
Gizli katmanlar ve çıktı katmanlardaki eşik değerleri	41+36+4=81
Toplam parametre sayısı	328+1476+144+81=**2029**

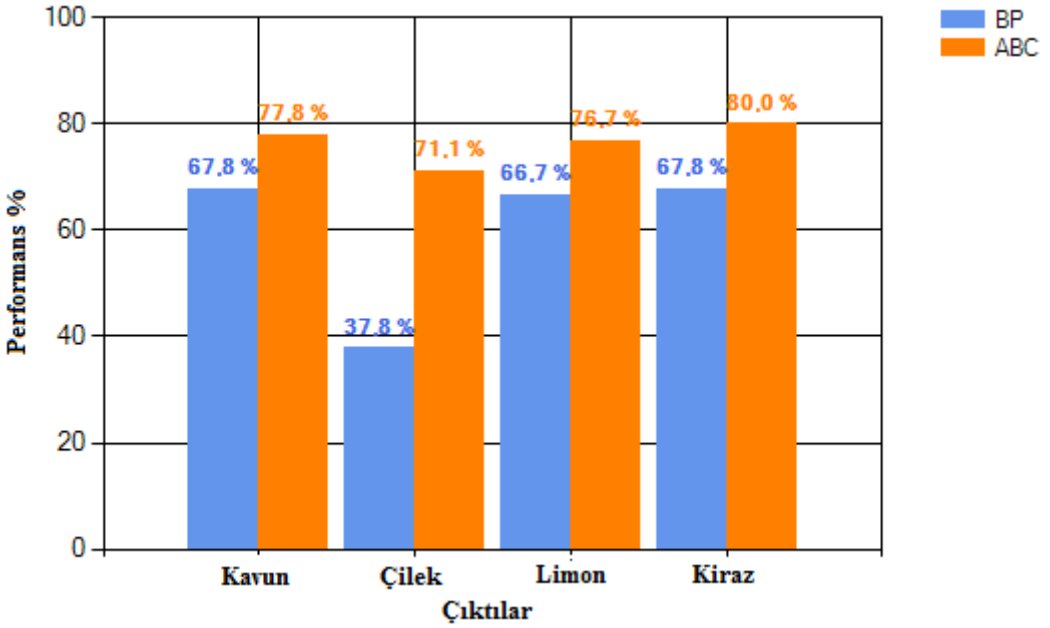
4.6.3. Elde edilen bulgular

Aroma veri seti 36 adet veriden oluşmaktadır. Bu veri seti %66,67 oranında eğitim, %33,33 oranında test verisi olarak ayrılmış ve YSA-BP ile YSA-ABC bu eğitim veri seti ile eğitilmiştir. Eğitim süresince elde edilen MSE grafikleri Şekil 4.17.'de verilmiştir. Bu MSE grafikleri, 30 defa tekrar edilen eğitimin, 10.000 epoch ve iterasyon sonucunda elde edilen grafiklerdir. Yakınlaştırılmış görünümde YSA-ABC'den daha iyi MSE grafiği elde edildiği görülmektedir. YSA'nın BP ile eğitilmesinde elde edilen minimum MSE değeri 0,0000145 iken ABC ile eğitilesinde elde edilen minimum MSE değeri 0,0000019 olmuştur.



Şekil 4.17. YSA-BP ve YSA-ABC için eğitimde elde edilen MSE grafiği

Bu eğitimler sonucunda oluşan ağlara daha önce hiç görmedikleri test verisi verilip, test performansları ölçülmüştür. Performanslar Şekil 4.18.'de görüldüğü üzere ABC için ortalama değer %76,39 iken BP'de bu değer %60 olarak ölçülmüştür. BP için ortalamanın düşük olmasında çilek verisinin (%37.8) düşük performansı büyük rol oynamıştır. ABC dört meyvede de daha başarılı bir performans sergilemiştir.



Şekil 4.18. Test verisi üzerinde BP ve ABC algoritmalarının başarımları

YSA'nın BP ile eğitilmesinde daha iyi sonuç alınır mı? Sorusunu yanıtlamak için farklı tasarımlarda aynı eğitim ve test verisi kullanılmış ve Tablo 4.5.'te ölçülen MSE değerleri listelenmiştir. Hiçbir tasarımın MSE değeri Şekil 4.15.'teki tasarımın MSE değerine erişememiştir.

Tablo 4.5. Farklı YSA-BP yapılarından elde edilen MSE değerleri

Gizli Katman Sayısı	Nöron Sayısı	Min MSE Değeri
1	41	0,00023
1	45	0,00101
2	25 + 10	0,0011
2	41 + 36	0,000032
3	41 + 36 + 15	0,0034

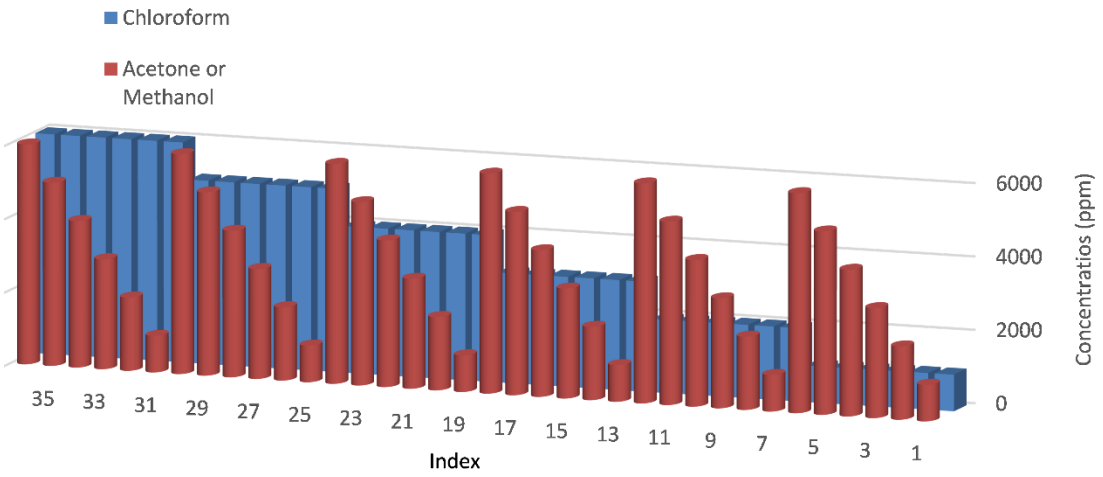
4.7. AC ve MC İkili Gaz Karışımlarının YSA-GA ile Sınıflandırılması

Sanayi endüstrisinde solvent olarak kullanılan uçucu gazlar, insan üzerinde olumsuz etkiler bıraktığı bilinmektedir [124]. Bu gazların belli oranlardaki ikili karışımlarının sınıflandırılması, olumsuz etkilerin büyüklüğünü tespit açısından önemlidir. Bu gazlardan metanol ve asetonun kloroform ile belli oranlardaki ikili karışımları

yapılmış ve laboratuvar ortamında gaz sensörleri yardımıyla gaz sensörlerin verileri elde edilmiştir.

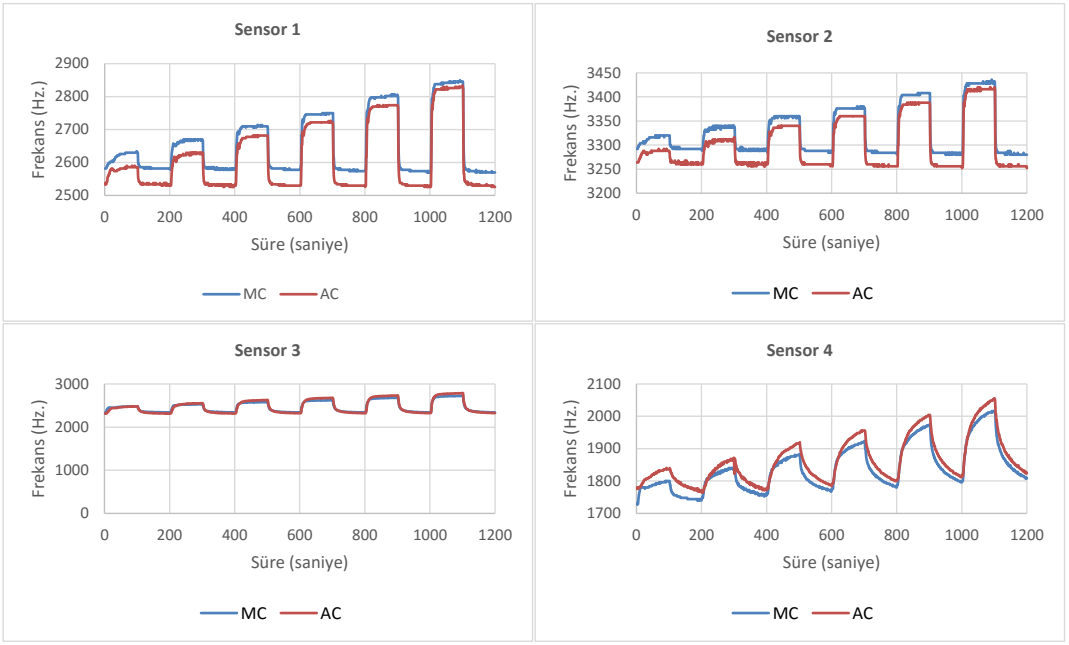
4.7.1. Verinin elde edilmesi

Solvent yapımında sıklıkla kullanılan metanol, aseton ve kloroform gazlarının ikili karışım verileri gaz sensörleri yardımıyla elde edilmiştir. İkili gaz karışım oranlarında kloroform sabit tutularak aseton ve metanol belli oranlarda arttırılmıştır. Karışım oranları Şekil 4.19.'da gösterildiği gibi kloroform sabit tutularak her ölçümde diğer iki gazdan (aseton, metanol) biri 1000 ppm'den başlayarak, 6000 ppm'e kadar 1000'er ppm arttırılmıştır. Cam tüpüm içinde -15° C derecede konulan ikili gaz karışımları, her bir karışım oranında ölçüm 200 saniye sürmüştür. Bu 200 saniyelik sürenin 100 saniyelik kısmında gaz karışımı sensörlere verilmiş, diğer 100 saniyelik kısmında ise hava ile sensörler temizlenmiştir.

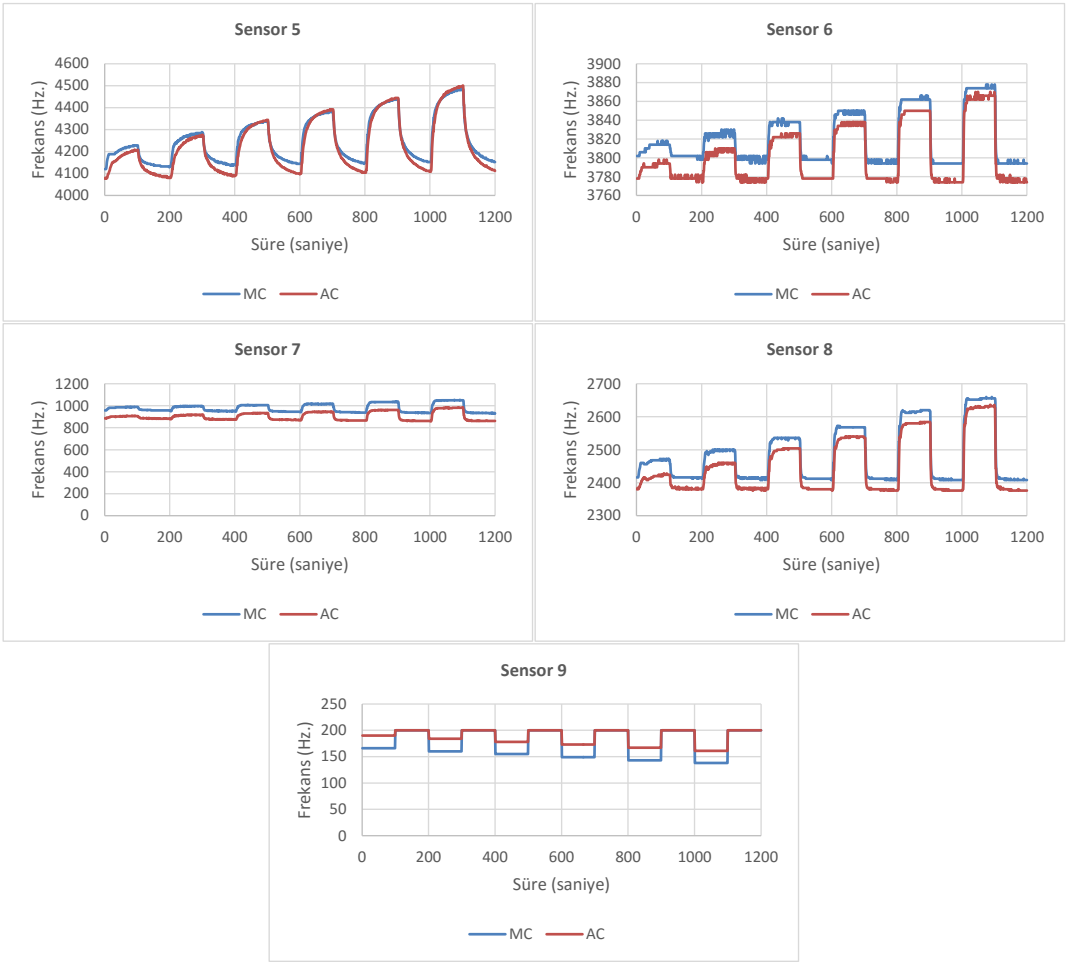


Şekil 4.19. İkili gaz karışım oranları

Kullanılan sensör dizisi 9 adet QCM sensörden oluşmaktadır. Sensörlerden dördüncü sensörün kaplama frekansı 23 KHz, diğer sensörler 22 KHz'dir. Altı farklı karışım oranı kullanıldığı için toplam ölçüm 1200 saniye sürmüştür. 9 adet sensörün karışımlara verdikleri tepkiler Şekil 4.20. ve Şekil 4.21.'de verilmiştir. Kloroformun sabit tutulup, asetonun arttırıldığı ikili gaz karışımından elde edilen veri sınıfının adına AC, Kloroformun sabit tutulup, metanolun arttırıldığı ikili gaz karışımından elde edilen veri sınıfının adına ise MC denilmiştir.



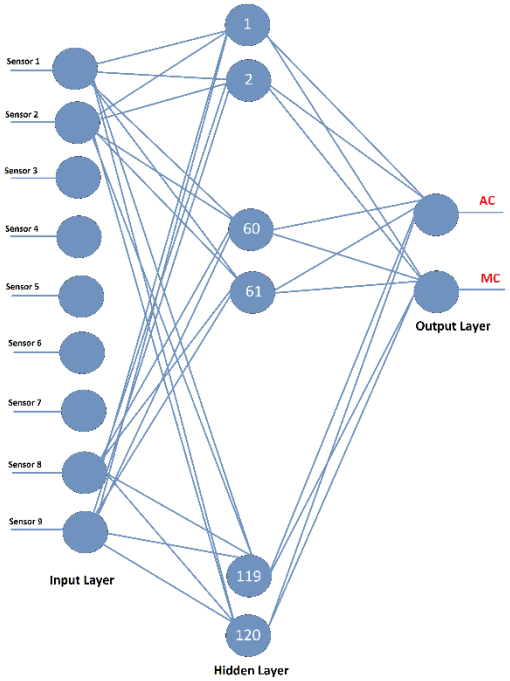
Şekil 4.20. Sensörlerin MC ve AC ikili gazlara verdikleri tepki (sensör 1-4)



Şekil 4.21. Sensörlerin MC ve AC ikili gazlara verdikleri tepki (sensör 5-9)

4.7.2. Tasarlanan yapay sinir ağı

İkili gaz karışımlarının ölçüldüğü düzenekte 9 adet sensör olduğu için tasarlanan YSA'da 9 adet girdi bulunmaktadır. 7 farklı YSA yapısı tasarlanmış olup en iyi sonucun alındığı yapı Şekil 4.22.'de verilmiştir. Bu yapıda bir girdi, bir gizli ve bir de çıktı katmanı bulunmaktadır. Gizli katmanda 120 adet nöron vardır. Çıktı katmanında iki adet sınıf (AC, MC) olduğundan iki adet çıktı bulunmaktadır. Gelen verinin hangi sınıftan olduğu kararı, ilgili çıktının 1, diğer çıktının 0 değerini alması ile belirlenir.



Şekil 4.22. Çalışmada en iyi sonucu veren YSA

Çalışmada kullanılan YSA-BP ve YSA-GA için belirlenen parametreler Tablo 4.6.'da verilmiştir. BP algoritmasında ağırlıkların ve eşik değerlerinin alabilecekleri değer aralıkları [-1, 1] iken, Genetik'te bu değer [-10, 10] arası olmuştur.

Tablo 4.6. BP ve Genetik algoritmalarının parametreleri

BP	Genetic
Öğrenme Katsayısı: 0,2	Mutasyon oranı: 0,001
Momentum: 0,8	Çaprazlama oranı: 0,85
Epoch: 500	Değer aralıkları: [-10, 10]
Ağırlık değer aralıkları: [-1, 1]	Popülasyon boyutu: 1000
Eşik değer aralıkları: [-1, 1]	İterasyon: 200

4.7.3. Elde edilen bulgular

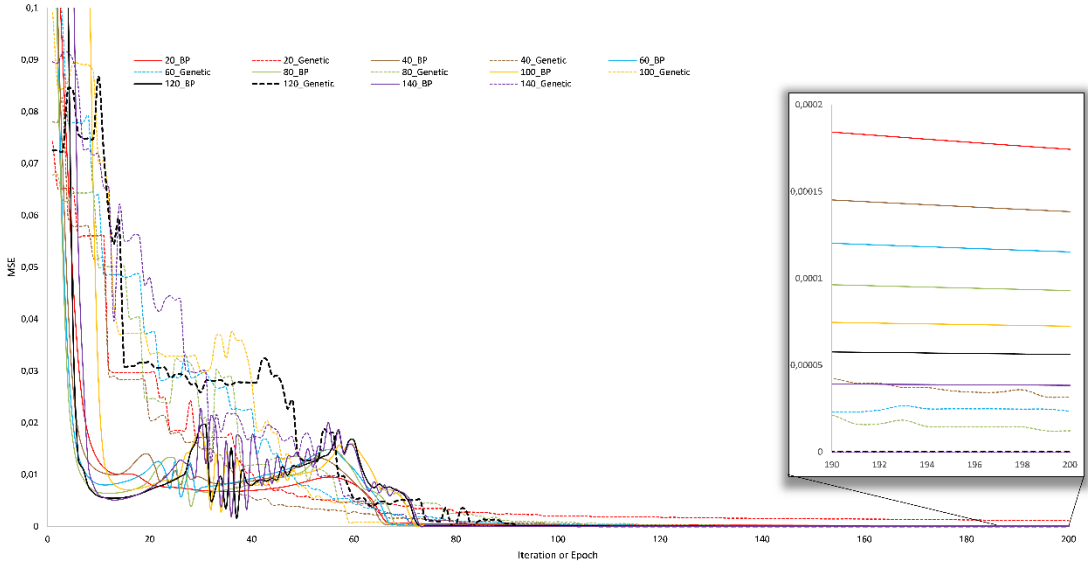
İkili gaz karışımları ölçümünden elde edilen veri seti %75 eğitim, %25 test olmak üzere ikiye ayrılmıştır. Aynı eğitim veri seti kullanılarak, YSA-BP ve YSA-GA ağları kendi yöntemleri ile eğitilmiştir. Aynı eğitim veri seti 7 farklı ağ yapısı kullanılarak eğitim sağlanmış ve elde edilen test verisi MSE hata değerleri karşılaştırılmıştır. Ağ yapılarında gizli katmandaki nöron sayısı 20'den başlayarak 140'a kadar 20'şer arttırılarak testler uygulanmıştır. YSA-BP 500 epoch, YSA-GA ise 200 iterasyon çalıştırılmıştır. Tablo 4.7.'den de görüldüğü üzere 120 nöron sayısından sonra test verisi MSE değerleri artmaya başlamıştır. YSA-BP için en iyi test sonucu 20 ve 100 nöron içeren ağlardan elde edilmiştir.

Tablo 4.7. Farklı ağ yapılarının vermiş oldukları MSE değerleri

Modeller	**MSE YSA-BP**		**MSE YSA-GA**	
Nöron Sayıları	*AC*	*MC*	*AC*	*MC*
20	6,26E-05	**5,87E-05**	1,66E-02	2,68E-03
40	7,87E-05	7,84E-05	3,81E-02	4,33E-03
60	1,15E-04	9,69E-05	2,92E-04	2,80E-02
80	7,81E-05	7,67E-05	6,74E-06	1,22E-03
100	**6,14E-05**	9,33E-05	1,03E-05	2,94E-05
120	7,61E-05	6,17E-05	**2,91E-09**	**2,46E-08**
140	6,27E-05	5,95E-05	1,01E-07	7,80E-06

Genetik ile eğitilen YSA'da en iyi sonuç 120 nöron içeren ağda elde edilmiştir. YSA-GA için 120 nöronda elde edilen test sonucuna, YSA-BP'nin hiçbir senaryosunda erişilememiştir. Yedi farklı senaryonun eğitim aşamasında verdikleri MSE değerleri Şekil 4.23.'de verilmiştir. En iyi senaryo olan 120 nörona sahip ağ koyu siyah çizgili bir şekilde gösterilmiş ve ulaşılan minimum MSE değeri 2,25E-09 olmuştur. Bu

başarıyı test verisinde de gösterip AC için minimum MSE değeri 2,91E-09 ve MC için 2,46E-08 değeri elde edilmiştir. Şekil 4.21.'de hangi çizginin hangi senaryoya ait olduğu gizli katmandaki nöron sayısını ifade eden başındaki sayı ile verilmiştir. Şekil 4.21.'de eğitim aşamasındaki MSE değerleri incelendiğinde diğer hiçbir MSE değerinin en iyi senaryoyu geçemediği görülmüştür.



Şekil 4.23. Farklı senaryolarda eğitimde elde edilen MSE değerleri

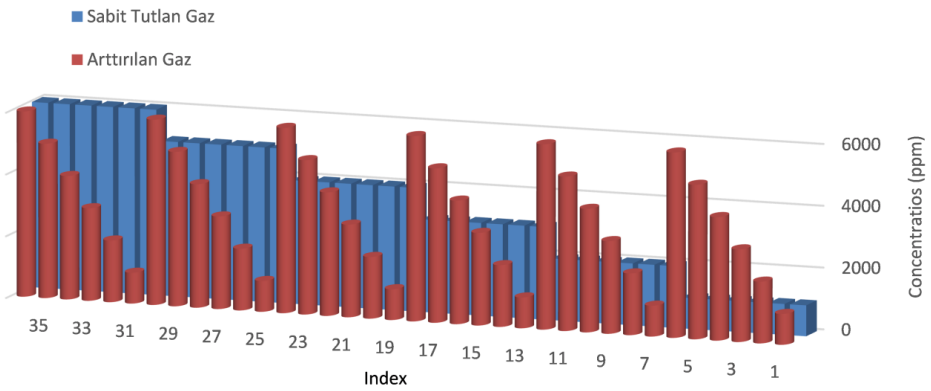
4.8. İkili Gaz Karışımlarının ABC Tabanlı YSA ile Sınıflandırılması

Aseton, kloroform ve metonol'un belli oranlardaki karışımları ile elde edilen ve solvent olarak bilinen gaz maddesi, endüstride üretim sektöründe yağ sökme ve temizlik gibi işlerde kullanılmaktadır. Solventin insan üzerinde olumsuz etki bıraktığı bilinmektedir [124]. İkili gaz karışımlarının, karışım miktarlarının sınıflandırılabilmesi, insan üzerinde bırakacağı etkinin tespiti açısından önemlidir. Bu çalışmada 3 gazın kendi aralarındaki ikili gaz karışımları elde edilip oluşan veri seti üzerinde eğitim gerçekleştirilmiştir.

4.8.1. Verinin elde edilmesi

Türkiye'de Gebze'de bulunan TÜBİTAK'a bağlı Dr. Cihat Taşaltın'ın sorumlusu olduğu gaz sensörleri laboratuvarında, aseton, metanol ve kloroformun belli

oranlarda ikili gaz karışım ölçümleri yapılmıştır. Karışım oranları Şekil 4.24.'te gösterildiği gibi ikili gaz karışımlarından biri sabit tutularak diğeri, 1000 ppm'den başlayarak, 6000 ppm'e kadar 1000'er ppm artırılmıştır.



Şekil 4.24. İkili gazlardaki karışım oranları

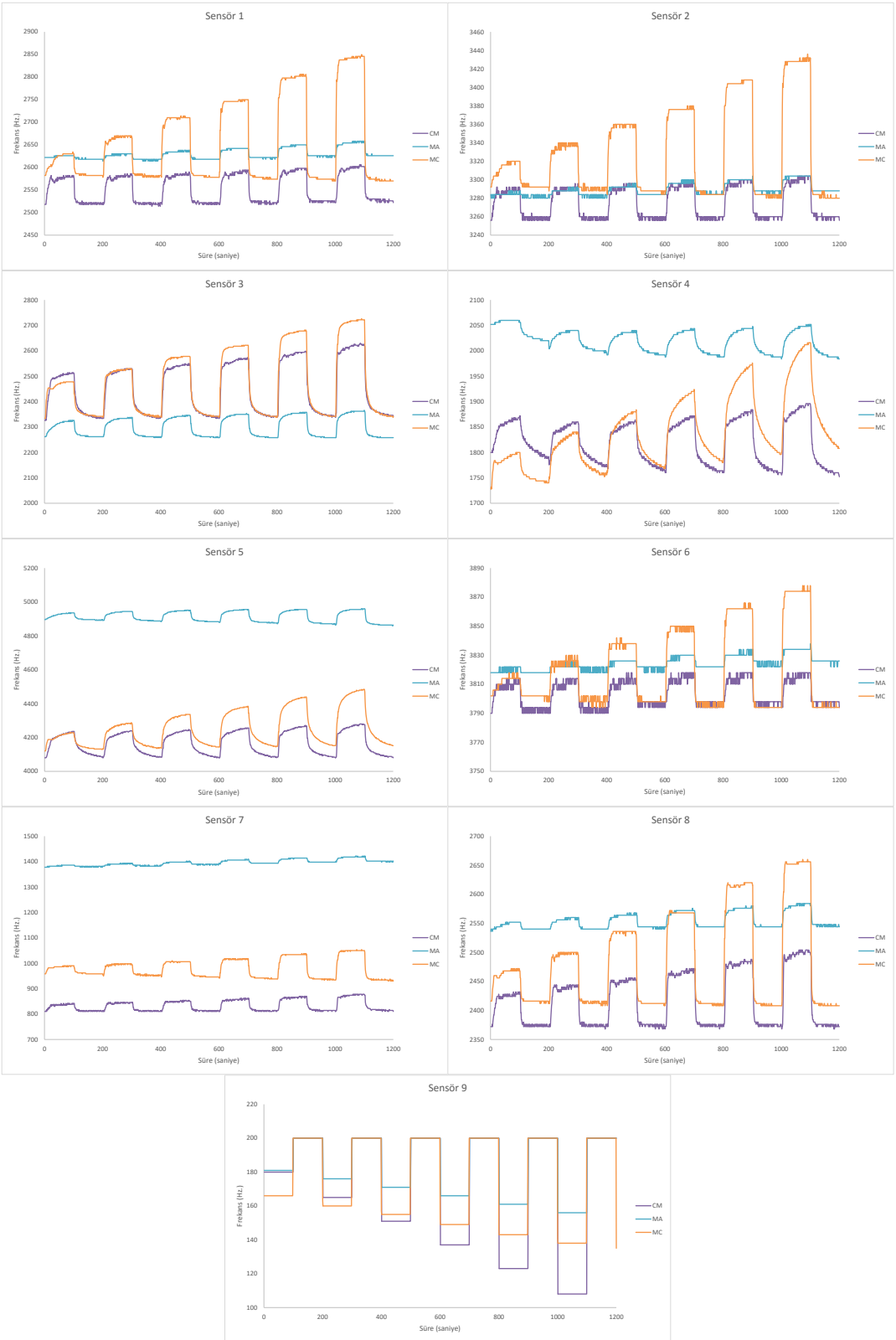
Sıcaklığı -15° C muhafaza edilen cam tüpün içerisine konulan bu ikili gaz karışımı için 200 saniye süren ölçüm yapılmıştır. Bu 200 saniyelik sürecin 100 saniyelik kısmında gaz karışımı sensörlere verilmiş diğer 100 saniyelik kısımda ise saf hava verilerek sensörler temizlenmiştir.

Dokuz adet sensörden oluşan sistem, dördüncü sensör hariç, diğer sensörler 22 kHz kaplama frekansına sahiptir. Dördüncü sensör ise 23 kHz kaplama frekansına sahiptir. Aseton sabit tutulup, metanol ve kloroform arttırıldığı, metanol sabit tutturulup kloroform arttırıldığı için 3 farklı karışım sınıfı oluşmuştur. Toplam 216 adet veri bulunmaktadır. Hangi ikili karışım bu dokuz adet QCM sensöre verilmiş ise ilgili ikili karışımın çıktı değeri 1 diğerleri 0 olmaktadır. İkili gaz karışım sınıfları ve gaz isimleri Tablo 4.8.'de verilmiştir.

Tablo 4.8. İkili gaz karışımlarının içeriği ve sınıf isimleri

Oranı arttırılan gaz	Sabit tutulan gaz	Sınıf adı
Kloroform	Metanol	CM
Metanol	Aseton	MA
Kloroform	Aseton	CA

Dokuz adet sensörün ikili gaz karışımlarına sabit gaz 1000 ppm'de iken verdikleri tepki Şekil 4.25.'te verilmiştir.



Şekil 4.25. Sensörlerin ikili gaz karışımlarına verdikleri tepki

Tasarlanan yapay sinir ağlarında Sigmoid fonksiyonu kullanıldığı için değerlerin [0,1] arası olması gerekmektedir. Bundan dolayı min-max normalizasyonu

uygulanmıştır. Fakat normalizasyon [0, 1] aralığına değil, sayısal taşmaları ve 0 değerinin eğitimdeki yanlış yönlenmesinin önüne geçmek için [0,1 0,9] arasına normalize edilmiştir. Uygulanan min-max normalizasyon hesabı (Denklem 4.18)'de veriliştir.

$$x_{i,0,1-0,9} = 0,8x\left[\frac{x_i - x_{min}}{x_{max} - x_{min}}\right] + 0,1 \tag{4.18}$$

4.8.2. Metot ve senaryolar

İkili gaz karışımlarının sınıflandırılması ve başarı elde edebilmek için daha önce başarılı sonuçlar elde edilmiş YSA-ABC yöntemi kullanılmıştır. Geleneksel YSA-BP yöntemi ile YSA-ABC yöntemi karşılaştırılmış ve analizler yapılmıştır. Hangisinin bu veri setinde en iyi sonuçlar verdiğini görebilmek ve hangi tasarımın daha başarılı olduğunu belirleyebilmek adına tek gizli katmana sahip, farklı epoch ve nöron sayılarındaki ağlarda testler yapılmıştır. Senaryolar için epoch sayısı 3000'den başlayarak sırasıyla 5000, 7000 ve 10000 sayılarında ağlar eğitime tabi tutulmuş ve her epoch denemesinde gizli katmandaki nöron sayısı 20'dan başlayıp 20'şer arttırıp 100 nörona kadar ağlar tasarlanmıştır. Bütün çalışmalarda ortak kullanılan YSA-BP ve YSA-ABC parametreleri Tablo 4.9.'da verilmiştir. BP için momentum katsayısı 0,8 ve öğrenme katsayısı 0,2 seçilmiş iken ABC için, koloni sayısı 50 ve yiyecek sayısı 250 seçilmiştir.

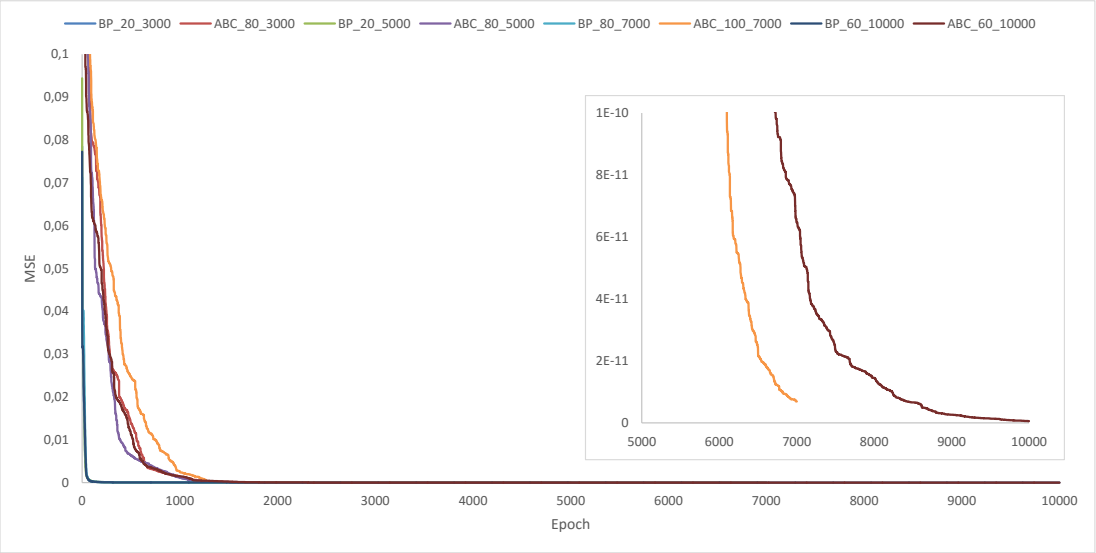
Tablo 4.9. YSA-BP ve YSA-ABC parametreleri

YSA-BP		YSA-ABC	
Parametre	Değer	Parametre	Değer
Ağırlıklar alt limit	-1	Alt sınır	-10
Ağırlıklar üst limit	1	Üst sınır	10
Momentum katsayısı	0,8	Koloni sayısı	50
Öğrenme katsayısı	0,2	Yiyecek kaynağı	250

Her senaryo, ortalama değerleri elde edebilmek için aynı parametreler ve veri setleri ile eğtim ve test çalışması 3 defa tekrar edilmiştir. Veri seti %66,67 eğitim, %33,33 test olarak ayrılmıştır. Her senaryoda belirtilen bu veri setleri rastgele seçilmektedir.

4.8.3. Elde edilen bulgular

Çalışmada toplam 20 senaryo eğitilip test işlemi uygulanmıştır. Bu 20 adet senaryoda en iyi sonucu 3,82E-12 test verisi MSE hata değeri ile YSA-ABC vermiştir. Bu senaryo 7000 epoch'ta çalıştırılan ve gizli katmanında 100 nörona sahip yapay sinir ağıdır. Buna en yakın YSA-BP senaryosu ise 10000 epoch'ta çalıştırılan ve gizli katmanında 60 nöron bulunan yapay sinir ağı olmuştur. Her epoch denemesinin en iyi test sonuçları alınmış ve Şekil 4.26.'daki eğitimde elde edilen MSE hata grafiği oluşmuştur.



Şekil 4.26. Senaryoların en iyi test sonuçlarının eğitimdeki MSE grafikleri

Şekil 4.26.'dan da görüldüğü üzere eğitimdeki en düşük MSE değeri 6,93E-12 ile yine 7000 epoch'ta çalıştırılan ve gizli katmanında 100 nöron bulunan yapay sinir ağına aittr. En yakın takipçisi yine ABC ile eğitilmiş olan 10000 epoch'ta çalıştırılmış ve gizli katmanında 60 nöronu olan YSA'dır. YSA-ABC'nin en iyi senaryosu ile YSA-BP'nin en iyi senaryosunun test sonuçları çıktı sınıfları baz alınarak karşılaştırıldığında Tablo 4.10.'daki tablo elde edilir.

Tablo 4.10. Test çıktı sınıflarına göre YSA-ABC ile YSA-BP algoritmalarının başarım karşılaştırması

Sınıf	YSA-BP	YSA-ABC
AC	2,71E-07	4,59E-12
MC	1,83E-07	3,93E-16
AM	1,46E-07	6,87E-12

Tablo 4.10. incelendiğinde genel ortalaması iyi olan YSA-ABC'nin bu başarısını sınıf bazlı karşılaştırmada da ortaya koyduğu görülmektedir. YSA-ABC'nin en iyi tespit ettiği karışım MC karışımıdır. Diğer iki karışımı hemen hemen aynı oranda sınıflandırmıştır. YSA-BP ise bütün karışımların sınıflandırmasında YSA-ABC'nin çok gerisinde kalmış kendi içinde hepsini hemen hemen aynı hata oranında sınıflandırmıştır.

4.9. Farklı QCM Gaz Sensör Verisinin YSA-ABC ile Sınıflandırılması

Bu çalışmada sadece 5 farklı gazın sınıflandırma işlemi değil, farklı sensör yapılarında sınıflandırma performansını belirlemek te hedeflenmiştir.

4.9.1. Veri setinin oluşturulması

Viyana üniversitesi, gaz sensörleri laboratuvarında, Prof. Dr. Peter Lieberzeit öncülüğünde 5 farklı QCM gaz sensörü kullanılarak, her sensör yapısında 5 farklı gaz (1-octanol, 1-proponal, 2-butanol, 2-propanol ve iso-butanol) ölçülmüştür. Bu QCM sensörler üzerinde 2 farklı kanal bulunmaktadır. Bu kanalların birinde moleküler baskılanmış polimer (MIP), diğerinde ise nanoparçacık (NP) bulunur. MIP ve NP'nin oranları değiştirilerek farklı yapıda QCM sensör elde edilmiştir. Her QCM sensördeki bir gazın ölçümü 120 dakika sürmüştür. Bu süre şu şekilde organize edilmiştir. İlk 30 dakika sensör, saf hava verilerek temizlenmiştir. Gaz örneği 5 farklı ölçekte sensöre verilmiştir. Bu ölçekler Tablo 4.11.'de verilmiştir.

Tablo 4.11. Ölçümde kullanılan gazın hava ile karışım miktarları

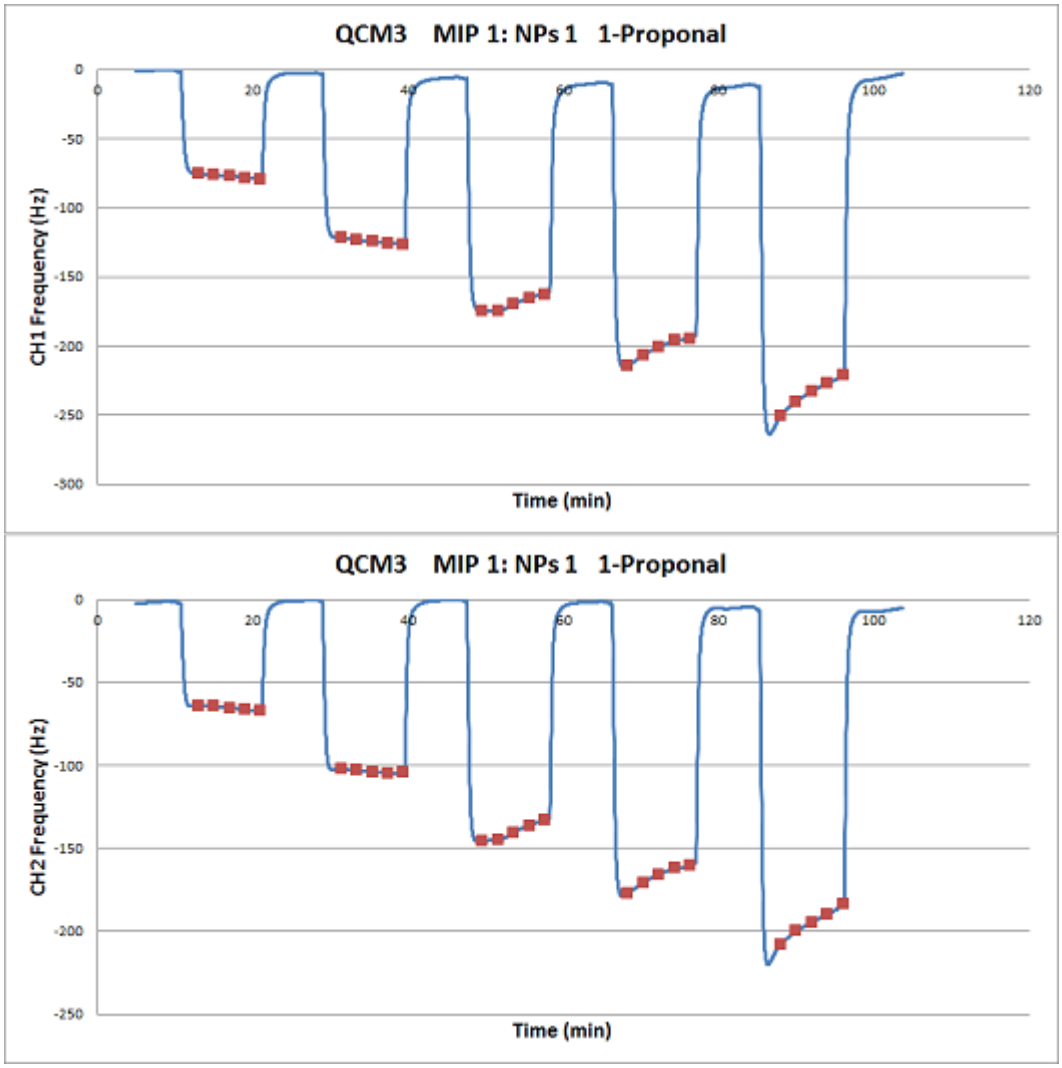
Ölçek Numarası	Hava Oranı (ml)	Gaz Oranı (ml)
1. Ölçek	0,799	0,201
2. Ölçek	0,700	0,300
3. Ölçek	0,600	0,400
4. Ölçek	0,501	0,499
5. Ölçek	0,400	0,600

Tablo 4.11.'den de görüldüğü üzere hava oranı her ölçekte giderek azaltılmıştır. Her ölçek arası 7 dk. sensör saf hava ile temizlenmiştir. Ölçümler 25° C oda sıcaklığında

yapılmıştır. Gaz örnekleri sıvı halde Şekil 4.27.'de görüldüğü gibi 50 ml'lik cam tüpe konulmuştur. Hava yardımıyla sıvı halindeki örnek, gaz halinde sensöre ulaşmakta ve sensörün tepkisi 2 farklı kanal aracılığıyla bilgisayara sayısal veri olarak iletilmektedir.

Şekil 4.27. Örneklerin konulduğu cam tüp

Burada kanal 1 ve 2'nin verileri, Hz cinsinden frekansları hesaplanıp veri seti oluşturulmaktadır. Örnek olarak QCM3 (MIP:1, NP:1) sensörün bir ve ikinci kanalının 1-proponal için üretmiş olduğu frekans değerleri Şekil 4.28.'de verilmiştir. Şekil 4.28.'de görüldüğü üzere her ölçekte 5 örnek alınmıştır. Bu şekilde alınan örnekler ile bütün bir veri seti oluşturulmuştur.



Şekil 4.28. QCM sensörünün 1 ve 2. kanalının 1-proponal örneğine verdiği tepki

Bu çalışmada aktivasyon fonksiyonu olarak daha önceki çalışmalara benzer olarak Sigmoid fonksiyonu kullanılmıştır. Bu fonksiyon girdi olarak sadece [0, 1] aralığında değer kabul ettiği için veri seti [0, 1] aralığına min-max normalizasyonu (Denklem 4.17) kullanılarak normalize edilmiştir.

4.9.2. Sensörlerin yapısı

Bu çalışmada 2 kanaldan oluşan QCM sensör kullanılmıştır. Kanalları, baskılanmış polimer (MIP) ve nanoparçacık (NP) oluşturmaktadır. Polimer veya nano parçacığın oranları değiştirilerek farklı yapıda QCM sensörler elde edilmiştir. Kullanılan QCM sensörlerden biri örnek olarak Şekil 4.29.'da verilmiştir.

Şekil 4.29. Kullanılan iki kanallı QCM Sensör

Şekil 4.29.'a bakıldığında iki adet sarı daire görülecektir. Bunlarda biri kanal 1'i, diğeri ise kanal 2'yi oluşturmaktadır. QCM sensörlerde kullanılan MIP ve NP oranları Tablo 4.12.'de görülmektedir. QCM6 sadece MIP'ten, QCM12 sadece NP'den oluşmaktadır. Diğer sensör yapılarında MIP ve NP'den belli oranlarda karışım bulunmaktadır.

Tablo 4.12. QCM sensörlerde kullanılan MIP ve NP oranları

Sensöre Verilen İsim	MIP Oranı	NP Oranı
QCM3	1	1
QCM6	1	0
QCM7	1	0,5
QCM10	1	2
QCM12	0	1

4.9.3. Metot ve senaryolar

Her QCM sensörden elde edilen verilerin sınıflandırılması için daha önce başarılı sonuçlar elde edilmiş YSA-ABC yöntemi kullanılmıştır. Geleneksel YSA-BP yöntemi ile YSA-ABC yöntemi karşılaştırılmış ve analizler yapılmıştır. Hangisinin bu veri setinde en iyi sonuçlar verdiğini görebilmek ve hangi tasarımın daha başarılı olduğunu belirleyebilmek adına tek gizli katmana sahip, farklı epoch ve nöron sayılarındaki ağlarda testler yapılmıştır. Her farklı QCM yapısı için epoch sayısı 500'den başlayarak sırasıyla 1000, 3000, 5000, 7000 ve 10000 sayılarında ağlar

eğitime tabi tutulmuş ve her epoch denemesinde gizli katmandaki nöron sayısı 10'dan başlayıp 10'ar arttırıp 100 nörona kadar ağlar tasarlanmıştır. Bütün çalışmalarda ortak kullanılan YSA-BP ve YSA-ABC parametreleri Tablo 4.13.'te verilmiştir. BP için momentum katsayısı 0,8 ve öğrenme katsayısı 0,2 seçilmiş iken ABC için, koloni sayısı 50 ve yiyecek sayısı 250 seçilmiştir.

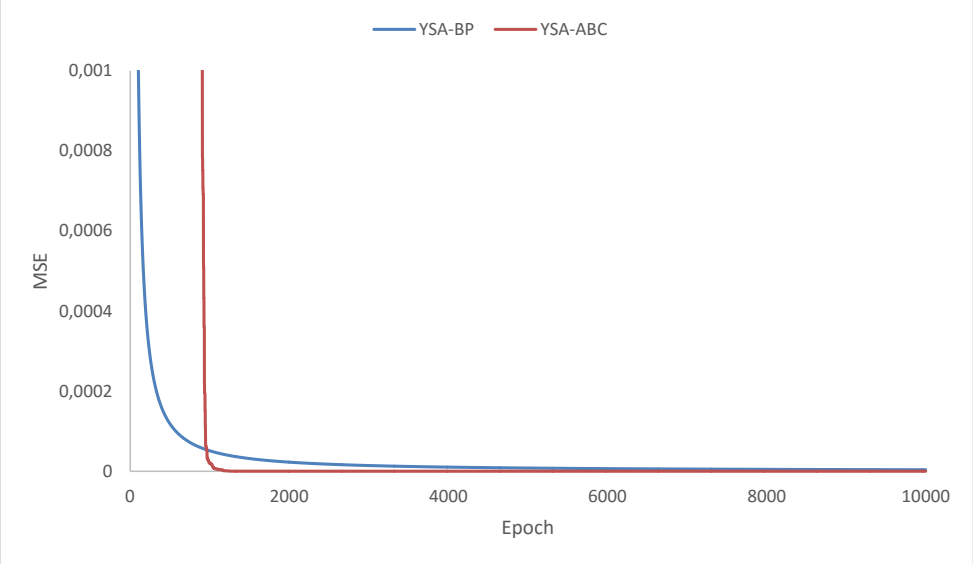
Tablo 4.13. YSA-BP ve YSA-ABC parametreleri

YSA-BP		YSA-ABC	
Parametre	Değer	Parametre	Değer
Ağırlıklar alt limit	-1	Alt sınır	-10
Ağırlıklar üst limit	1	Üst sınır	10
Momentum katsayısı	0,8	Koloni sayısı	50
Öğrenme katsayısı	0,2	Yiyecek kaynağı	250

Her senaryo, ortalama değerleri elde edebilmek için aynı parametreler ve veri setleri ile 3 defa çalıştırılmıştır. Veri seti %60 eğitim, %40 test olarak ayrılmıştır. Her senaryoda belirtilen bu veri setleri rastgele seçilmektedir.

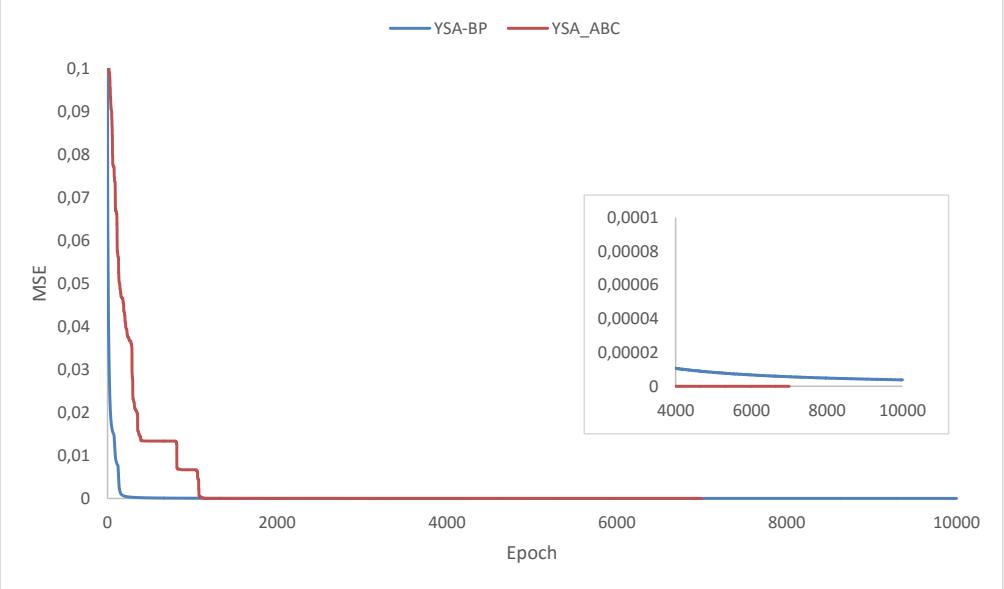
4.9.4. Elde edilen bulgular

Her bir QCM yapısı için toplam 60 senaryo eğitilip test işlemi uygulanmıştır. QCM3 için en iyi senaryo YSA-ABC ile gerçekleşmiş olup test verisinde 2,33E-16 MSE hata değeri elde edilmiştir. Bu hata değeri 5000 epoch ile çalıştırılan ve gizli katmanında 40 nörona sahip yapay sinir ağı ile yakalanmıştır. En yakın YSA-BP senaryosu 10000 epoch ile çalıştırılan 60 nörona sahip MSE hata değeri 4,18E-06 olan yapay sinir ağı olmuştur. Bu iki senaryonun eğitim veri setinin MSE grafiği Şekil 4.30.'da verilmiştir. YSA-BP'nin eğitimde ulaştığı en düşük MSE değeri 3,65E-06 iken YSA-ABC'nin eğitimde ulaştığı en düşük MSE değeri 5,69E-16 olmuştur.



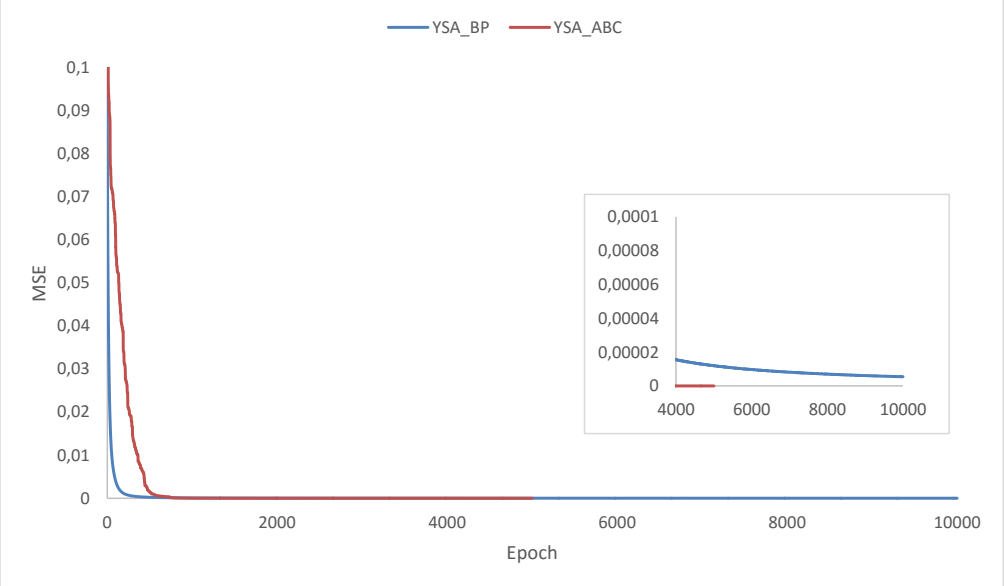
Şekil 4.30. QCM3 için en iyi eğitimin MSE grafiği

QCM6 için en iyi senaryo yine YSA-ABC ile gerçekleşmiş olup test verisinde 1,41E-16 MSE hata değeri elde edilmiştir. Bu hata değeri 5000 epoch ile çalıştırılan ve gizli katmanında 70 nörona sahip yapay sinir ağı ile yakalanmıştır. En yakın YSA-BP senaryosu 10000 epoch ile çalıştırılan 70 nörona sahip yapay sinir ağı, test MSE hata değeri 3,94E-06 olmuştur. Bu iki senaryonun eğitim veri setinin MSE grafiği Şekil 4.31.'de verilmiştir. YSA-BP'nin eğitimde ulaştığı en düşük MSE değeri 3,81E-06 iken YSA-ABC'nin eğitimde ulaştığı en düşük MSE değeri 3,13E-16 olmuştur. Son ulaşılan eğitim hata değerini daha iyi gösterebilmek için Şekil 4.31.'deki son kısım büyütülmüştür.



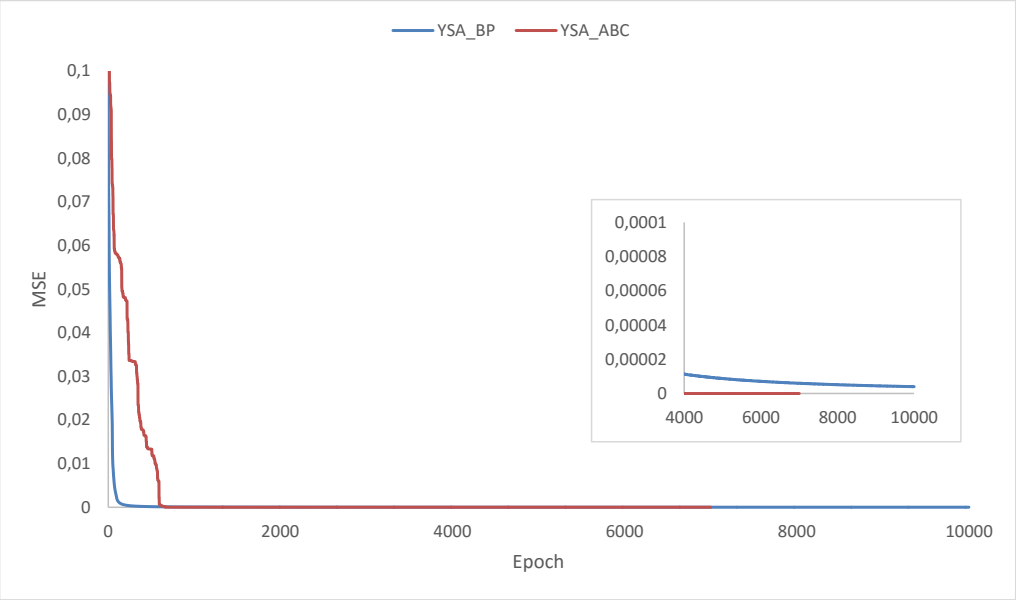
Şekil 4.31. QCM6 için en iyi eğitimin MSE grafiği

QCM7 için en iyi senaryo yine YSA-ABC ile gerçekleşmiş olup test verisinde 1,53E-16 MSE hata değeri elde edilmiştir. Bu hata değeri 5000 epoch ile çalıştırılan ve gizli katmanında 30 nörona sahip yapay sinir ağı ile yakalanmıştır. En yakın YSA-BP senaryosu 10000 epoch ile çalıştırılan 40 nörona sahip yapay sinir ağı, test MSE hata değeri 3,23E-06 olmuştur. Bu iki senaryonun eğitim veri setinin MSE grafiği Şekil 4.32.'de verilmiştir. YSA-BP'nin eğitimde ulaştığı en düşük MSE değeri 5,43E-06 iken YSA-ABC'nin eğitimde ulaştığı en düşük MSE değeri 8,77E-16 olmuştur.



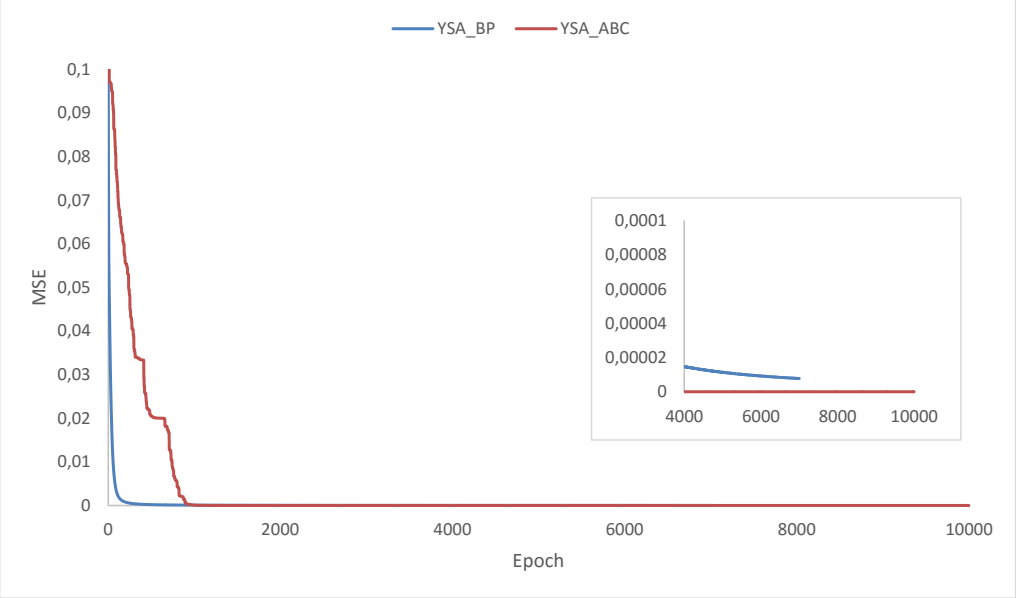
Şekil 4.32. QCM7 için en iyi eğitimin MSE grafiği

QCM10 için en iyi senaryo yine YSA-ABC ile gerçekleşmiş olup test verisinde 1,74E-16 MSE hata değeri elde edilmiştir. Bu hata değeri 7000 epoch ile çalıştırılan ve gizli katmanında 40 nörona sahip yapay sinir ağı ile yakalanmıştır. En yakın YSA-BP senaryosu 10000 epoch ile çalıştırılan 80 nörona sahip yapay sinir ağı, test MSE hata değeri 4,05E-06 olmuştur. Bu iki senaryonun eğitim veri setinin MSE grafiği Şekil 4.33.'te verilmiştir. YSA-BP'nin eğitimde ulaştığı en düşük MSE değeri 4,01E-06 iken YSA-ABC'nin eğitimde ulaştığı en düşük MSE değeri 8,18E-16 olmuştur.



Şekil 4.33. QCM10 için en iyi eğitimin MSE grafiği

QCM12 için en iyi senaryo yine YSA-ABC ile gerçekleşmiş olup test verisinde 4,20E-16 MSE hata değeri elde edilmiştir. Bu hata değeri 10000 epoch ile çalıştırılan ve gizli katmanında 70 nörona sahip yapay sinir ağı ile yakalanmıştır. En yakın YSA-BP senaryosu 7000 epoch ile çalıştırılan 50 nörona sahip yapay sinir ağı, test MSE hata değeri 5,52E-06 olmuştur. Bu iki senaryonun eğitim veri setinin MSE grafiği Şekil 4.34.'te verilmiştir. YSA-BP'nin eğitimde ulaştığı en düşük MSE değeri 7,73E-06 iken YSA-ABC'nin eğitimde ulaştığı en düşük MSE değeri 4,97E-16 olmuştur.



Şekil 4.34. QCM12 için en iyi eğitimin MSE grafiği

QCM sensör yapısının değiştirilmesi sınıflandırmadaki başarıyı etkilememiş ve 5 farklı QCM yapısında da gazların sınıflandırma düzeyi YSA-ABC için E-16 gibi başarılı bir düzeyde seyretmiştir.

BÖLÜM 5. BULGULAR VE TARTIŞMA

Bu tez kapsamında gaz sensörleri kullanılarak 4 farklı çalışma yapılmıştır. Elde edilen koku verilerinin sınıflandırılması için literatürde bu alanda sıklıkla kullanılan yapay sinir ağları kullanılmıştır. Yapay sinir ağları geleneksel geri yayılım (BP) algoritması ile eğitildiğinde lokal minimuma takılma ve veri setini ezberleme gibi zayıf yönleri bulunmaktadır [1], [40]. Bundan dolayı yapay sinir ağları, yapay arı koloni algoritması (ABC) ve Genetik algoritma (GA) kullanılarak YSA'nın eğitim aşaması optimize edilmiştir. Daha önceki bölümlerde bu çalışmalardan elde edilen sonuçlar verilmiş olup performans karşılaştırmaları yapılmıştır. Bu bölümde ise YSA-ABC hibrit algoritması, YSA-GA hibrit algoritması ile karşılaştırması yapılıp performans analizleri irdelenecektir.

5.1. ABC, GA ve BP'nin Eğitim Süresi Başarım Karşılaştırması

Bu tez kapsamında yapılan çalışmalarda eğitim her ne kadar online eğitim olmasa ve eğitim bir kez yapılsa bile eğitimin ne kadar zaman aldığı algoritma karmaşıklığı açısından önemlidir. Yapılan 4 çalışmanın zaman analizleri bu bölümde verilmiştir. Elde edilen zamanlar aynı özellikteki bilgisayarda ve algoritmanın birkaç kez çalıştırılması ve ortlamaların alınması yöntemiyle bulunmuştur.

5.1.1. Dört farklı aroma verisinin sınıflandırılmasındaki eğitimde geçen zaman analizi

Bu tezde yapılan çalışmalardan biri olan, dört farklı aroma türünden elektronik burun yardımıyla elde edilen koku verisinin, sınıflandırma için yapılan eğitimlerin ve bu eğitimlerin ne kadar zaman sürdüğü Tablo 5.1.'de verilmiştir. Bu sürelerin hesaplandığı bilgisayarın özelliği ise Tablo 5.2.'de verilmiştir.

Tablo 5.1. YSA-BP, YSA-ABC ve YSA-GA'nın eğitim süreleri

Senaryolar	Eğitim Süresi		
	YSA-BP	YSA-GA	YSA-ABC
10000 Epoch 41 Nöron	1 dk.	3 saat	15 dk.
10000 Epoch 45 Nöron	1,03 dk.	3 saat 1 dk.	15,4 dk.
10000 Epoch 25+10 Nöron	1,2 dk.	3 saat 5 dk.	15,8 dk.
10000 Epoch 41+36 Nöron	1,5 dk.	7 saat 12 dk.	42 dk.
10000 Epoch 41+36+15 Nöron	1,6 dk.	8 saat 47 dk.	53 dk.

Tablo 5.1.'e bakıldığı zaman bütün senaryolarda en hızlı hesaplama geleneksel yöntem olan BP'de yapılmaktadır. BP'yi ABC takip etmekte ve en yavaş GA sürmektedir. BP ile ABC arasındaki zaman farkı belki kabul edilebilir düzeyde olduğu söylenebilir. Fakat GA'nın hesaplama süresi diğer iki algoritmaya göre çok yavaş kalmaktadır.

Tablo 5.2. Algoritmaların karşılaştırıldığı bilgisayarın özellikleri

Bileşenler	Özellik
İşlemci	İntel i7 6700K 4GHz
RAM-Bellek	8GB DDR4 2666 MHz
Disk	Samsung 850 PRO 512GB SSD Disk

5.1.2. AC ve MC ikili gaz karışımlarının sınıflandırılmasındaki eğitimde geçen zaman analizi

Bu çalışmanın zaman analizleri, Tablo 5.2.'de özellikleri verilen bilgisayarda yapılmıştır. Koloroformun sabit tutulup, asetonun artırıldığı ve yine kloroformun sabit tutulup metanolün arttırıldığı ikili gaz karışım çalışmasında AC ve MC isimlerinin verildiği ikili gaz karışımlarının sınıflandırılması yapılmıştır. Bu çalışmada algoritmaların eğitim yaparken ne kadar zaman sürdüğü Tablo 5.3.'te verilmiştir.

Tablo 5.3. YSA-BP, YSA-ABC ve YSA-GA algoritmaları için ikili gaz karışım çalışmasındaki eğitim süreleri

Senaryolar	Eğitim Süresi		
	YSA-BP	YSA-GA	YSA-ABC
20 Nöron	1 sn.	3 dk. 10 sn.	20 sn.
40 Nöron	1 sn.	6 dk. 3 sn.	35 sn.
60 Nöron	2 sn.	8 dk. 30 sn.	51 sn.
80 Nöron	3 sn.	11 dk. 1 sn.	1 dk. 6 sn.
100 Nöron	4 sn.	13 dk. 28 sn.	1 dk. 22 sn.
120 Nöron	5 sn.	16 dk. 6 sn.	1 dk. 39 sn.
140 Nöron	6 sn.	18 dk. 53 sn.	1 dk. 55 sn.

Tablo 5.3.'te verilen senaryolar tek gizli katmana sahip yapay sinir ağlarıdır. BP 500 epoch çalışıtırılken, GA ve ABC 200 iterasyon çalıştırılmıştır. BP 500 epoch çalıştırılmasına rağmen diğer çalışmaya benzer olarak en hızlı algoritma olmuştur. ABC en uzun 1 dk. 22 sn. sürerken, GA 13 dk. 28 sn. sürmüştür.

5.1.3. İkili gaz karışımlarının sınıflandırılmasındaki eğitimde geçen zaman analizi

Bu çalışmada asetonun sabit tutulup, metanol ve kloroformun arttırıldığı, metanolun sabit tutulup kloroformun arttırıldığı ikili gaz karışım çalışmasında çok fazla sayıda senaryo çalıştırılmış ve sonuçlar daha önceki bölümlerde sunulmuştur. Zaman analizi olarak bütün senaryolar analiz edilmemiş sadece her epoch denemesindeki en düşük ve en yüksek nöron sayısına sahip ağlar seçilmiştir. Bahsedilen bu ağlarda yapılan zaman analizi Tablo 5.4.'te listelenmiştir.

Tablo 5.4. YSA-BP, YSA-ABC ve YSA-GA algoritmaları için ikili gaz karışım çalışmasındaki eğitim süreleri

Senaryolar	Eğitim Süresi		
	YSA-BP	YSA-GA	YSA-ABC
3000 epoch, 20 Nöron	18 sn.	2 saat 15 dk.	4 dk. 11 sn.
3000 epoch, 100 Nöron	1 dk. 16 sn.	9 saat 13 dk.	14 dk. 5 sn.
5000 epoch, 20 Nöron	32 sn.	3 saat 45 dk.	6 dk. 44 sn.
5000 epoch, 100 Nöron	2 dk. 20 sn.	13 saat 5 dk.	24 dk. 14 sn.
7000 epoch, 20 Nöron	44 sn.	4 saat 52 dk.	9 dk. 47 sn.
7000 epoch, 100 Nöron	3 dk. 19 sn.	17 saat 10 dk.	33 dk. 35 sn.
10000 epoch, 20 Nöron	1 dk. 5 sn.	5 saat 34 dk.	13 dk. 29 sn.
10000 epoch, 100 Nöron	4 dk. 10 sn.	21 saat 48 dk.	47 dk. 59 sn.

5.1.4. Farklı QCM gaz sensörlerinden elde edilen verilerin sınıflandırılmasındaki eğitimde geçen zaman analizi

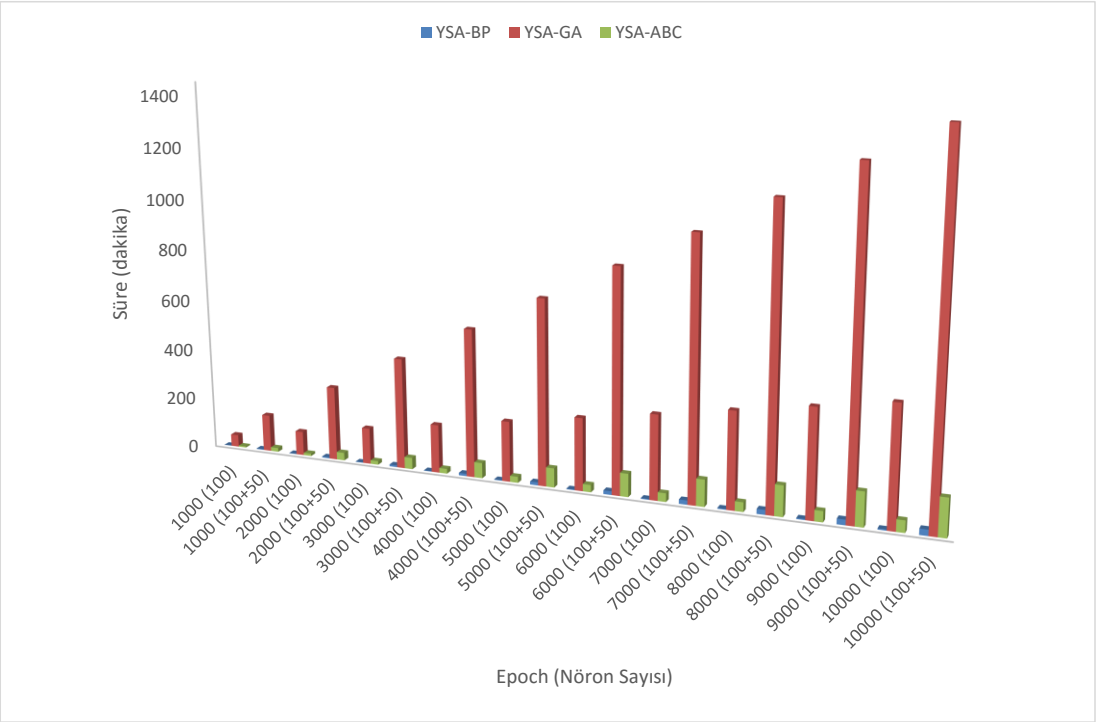
Çok fazla senaryoya sahip olan bu çalışmada, 5 farklı QCM yapısında 5 farklı gaz ölçümleri yapılmış ve elde edilen veri setlerinde toplamda 300 senaryo çalıştırılmıştır. Veri seti boyutu her farklı QCM sensör yapısında aynı olduğu için sadece QCM3 sensör yapısındaki senaryolar zaman bakımından analiz edilmiştir. Bu senaryolardan da sadece en büyük ağ yapısına ait olanlar alınmıştır. Gizli katmanında 100 nöron içeren yapay sinir ağlarının eğitim süreleri değişen epoch sayılarına göre Tablo 5.5.'te verilmiştir.

Tablo 5.5. YSA-BP, YSA-ABC ve YSA-GA algoritmaları için QCM gaz sensör çalışmasındaki eğitim süreleri

Senaryolar	Eğitim Süresi		
	YSA-BP	YSA-GA	YSA-ABC
500 epoch, 100 nöron	3 sn.	3 dk. 14 sn.	26 sn.
1000 epoch, 100 nöron	5 sn.	6 dk. 40 sn.	50 sn.
3000 epoch, 100 nöron	16 sn.	19 dk. 32 sn.	2 dk. 32 sn.
5000 epoch, 100 nöron	28 sn.	32 dk. 5 sn.	4 dk. 29 sn.
7000 epoch, 100 nöron	32 sn.	42 dk. 11 sn.	6 dk. 17 sn.
10000 epoch, 100 nöron	58 sn.	1 saat 7 dk.	8 dk. 59 sn.

Tablo 5.5.'te de görüldüğü üzere bu veri seti için YSA-BP saniyeler içerisinde hesaplarken, YSA-ABC birkaç dakika içersinde, YSA-GA ise 10000 iterasyonu 1 saatte hesaplamaktadır.

Epoch ve nöron sayısına bağlı olarak eğitim zaman grafiği karşılaştırma amaçlı olarak Şekil 5.1.'de verilmiştir. Büyük bir veri seti olması için ikili gaz karışımları veri seti seçilmiştir. Epoch sayısı 1000'den başlayarak 1000'er artırılıp 10000 epoch'a kadar zaman analizi yapılmıştır. Her epoch denemesinde iki farklı yapay sinir ağı analiz edilmiştir. Birinci yapay sinir ağı tek gizli katmanda 100 nörona sahip yapay sinir ağı iken, ikinci yapay sinir ağında iki gizli katman ve sırasıyla 100 ve 50 nöron bulunmaktadır. Katman sayısı arttığı zaman özellikle YSA-GA'da zaman astronomik bir şekilde katlanarak artmaktadır. YSA-ABC'de ise yine gözle görülür bir artış olurken YSA-GA'ya göre hesaplama zamanı çok daha hızlıdır.



Şekil 5.1. YSA-BP, YSA-GA ve YSA-ABC algoritmalarınn eğitim sürelerinin karşılaştırılması

Yapılan eğitim zaman analizlerinin vermiş olduğu sonuçlar göstermektedir ki, YSA'nın eğitiminde en hızlı algoritma geleneksel olan geri yayılım (BP) algoritmasıdır. Bundan sonra gelen en hızlı algoritma ABC ve en yavaşı ise GA'dır. Fakat bu algoritmalar test verisinde aynı sırada başarı sergilememektedirler. Test verisi üzerinde en iyi başarıyı Bölüm 5.2.'de de bahsedildiği üzere ABC, daha sonra GA ve en son BP göstermektedir. Bu durumda eğitimdeki harcanan zaman göz ardı edilebilir. Çünkü eğitim bir kez yapıldığı için uzun bir zaman da olsa bu zaman bir kez harcanacaktır. Online eğitimin söz konusu olduğu durumlarda test performansı da bir arada düşünülecek ise GA yerine ABC kullanılabilir. Fakat çok hızlı diye BP seçilmesi durumunda test verisindeki üstün performans kaybedilmiş olacaktır.

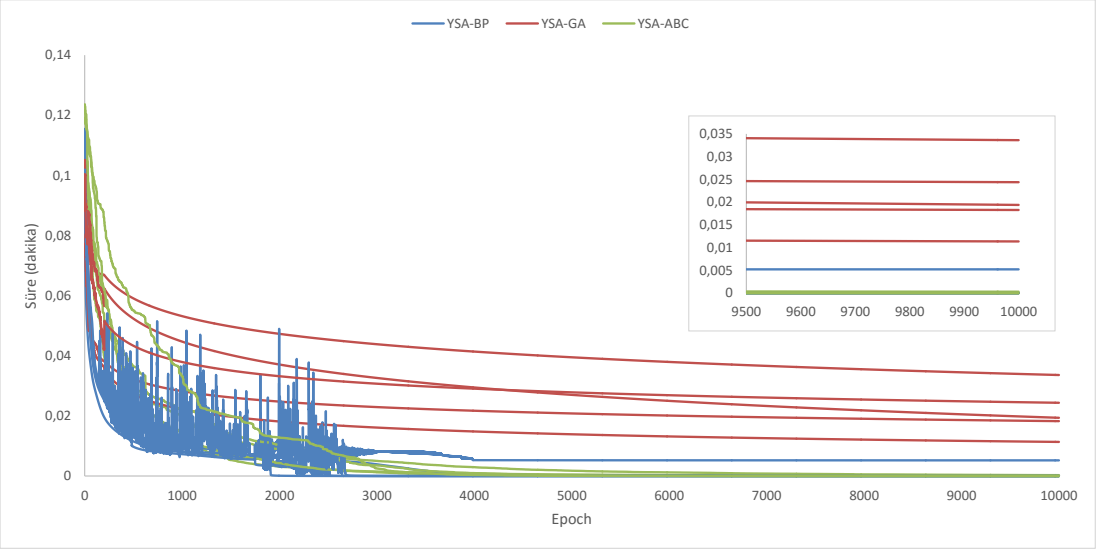
5.2. Yapay Sinir Ağının Eğitildiği, ABC, GA ve BP'nin Eğitim ve Test Başarım Karşılaştırmaları

Bu tez kapsamında yapılan 4 çalışmanın üçünde YSA-ABC kullanılmış ve YSA-BP ile karşılaştırılmıştır. Sadece birinde YSA-GA kullanılmış ve YSA-BP ile

karşılaştırılmıştır. Bu bölümde ise 4 çalışmada da ABC, GA ve BP kullanılıp karşılaştırma yapılacak ve grafik yardımıyla başarımları analiz edilecektir.

5.2.1. Dört farklı aroma verisinin sınıflandırılmasında eğitim ve test başarım analizi

Bu tezde yapılan çalışmalardan biri olan, dört farklı aroma türünden elektronik burun yardımıyla elde edilen koku verisinin, sınıflandırmasında YSA-ABC ile başarı elde edilmişti. Aynı çalışma YSA-GA ile yapıldığında eğitimdeki MSE grafikleri Şekil 5.2.'deki gibi olmaktadır.



Şekil 5.2. YSA-BP, YSA-GA ve YSA-ABC algoritmalarının eğitimdeki MSE grafik karşılaştırması

Şekil 5.2.'ye bakıldığında mavi renkli grafik YSA-BP'yi, kırmızı renkli grafik YSA-GA'yı ve yeşil renkli grafik ise YSA-ABC'yi göstermektedir. Kırmızı grafik ile gösterilen YSA-GA, diğer eğitim modelleri olan YSA-BP ve YSA-ABC kadar düşük hata değerleri verememiştir. YSA-BP ise kararsız bir eğitim grafiği göstermiştir. YSA-BP eğitiminde erişilen en düşük hata değeri 2,98E-06 iken, YSA-GA'da 1,14E-02 ve YSA-ABC'de bu değer 3,15E-08 olmuştur. Bu sonuçlar, eğitimde en düşük hata değerine, YSA-ABC ile ulaşıldığını göstermiştir. Eğitilen bu ağların test verisindeki başarımları Tablo 5.6.'da verilmiştir. Tablodan görüldüğü üzere test verisinde en düşük hata değerine YSA-ABC daha sonra YSA-BP ve en son YSA-GA erişmiştir. YSA-ABC'nin en düşük hata değerine eriştiği senaryo, iki gizli katmana

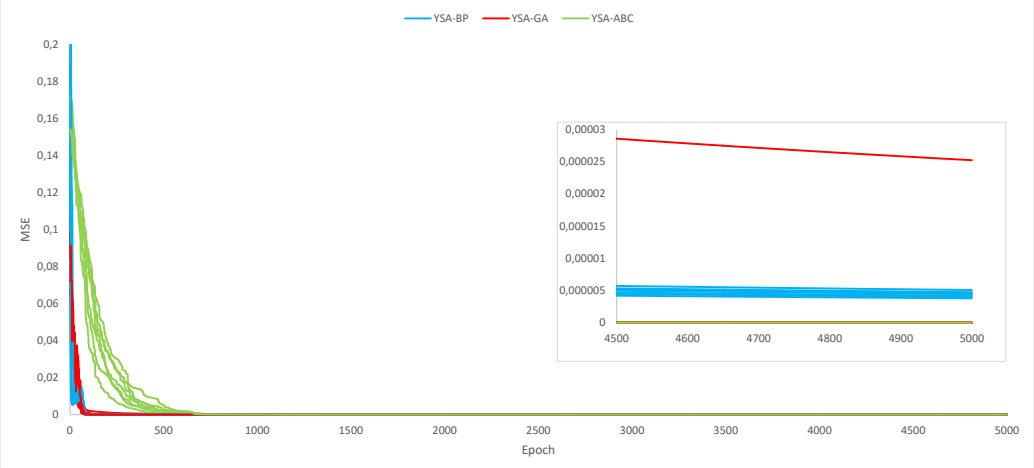
sahip ve gizli katmanlarında sırasıyla 41 ve 36 nöron bulunan yapay sinir ağıdır. İkinci en iyi senaryoda yine YSA-ABC'ye aittir.

Tablo 5.6. YSA-BP, YSA-ABC ve YSA-GA algoritmalarının 4 farklı türü aroma sınıflandırılmasındaki test verisi sonuçları

Gizli Katmandaki Nöron Sayısı	Test Verisi MSE Hata Değeri		
	YSA-BP	YSA-GA	YSA-ABC
41	2,71E-02	8,01E-02	4,52E-04
45	7,68E-05	1,33E-01	1,92E-04
25+10	4,88E-03	6,95E-02	6,00E-03
41+36	3,85E-04	6,48E-02	**1,29E-08**
41+36+15	7,06E-06	8,67E-02	4,55E-07

5.2.2. AC ve MC ikili gaz karışımlarının sınıflandırılmasında eğitim ve test başarım analizi

Kloroformun sabit tutulup aseton ve metanolün artırıldığı ikili gaz karışımı çalışmasında BP, GA ve ABC algoritmaları 7 farklı senaryoda eğitilmiştir. Eğitimlerindeki MSE hata grafiği Şekil 5.3.'te verilmiş olup mavi renkler YSA-BP, kırmızı renkler YSA-GA ve yeşil renkler YSA-ABC senaryolarını ifade etmektedir. Şekil 5.3.'te son 500 epoch daha iyi anlaşılması için yakınlaştırılıp gösterilmiştir. Eğitimde en iyi MSE hata değerini veren 1,98E-237 değeri ile YSA-GA olmuştur. Daha sonra en iyi değeri 9,51E-16 ile YSA-ABC vermiştir. YSA-BP ise ulaşabildiği en düşük değer 3,78E-06 olmuştur.



Şekil 5.3. YSA-BP, YSA-GA ve YSA-ABC algoritmalarının eğitimdeki MSE grafik karşılaştırması

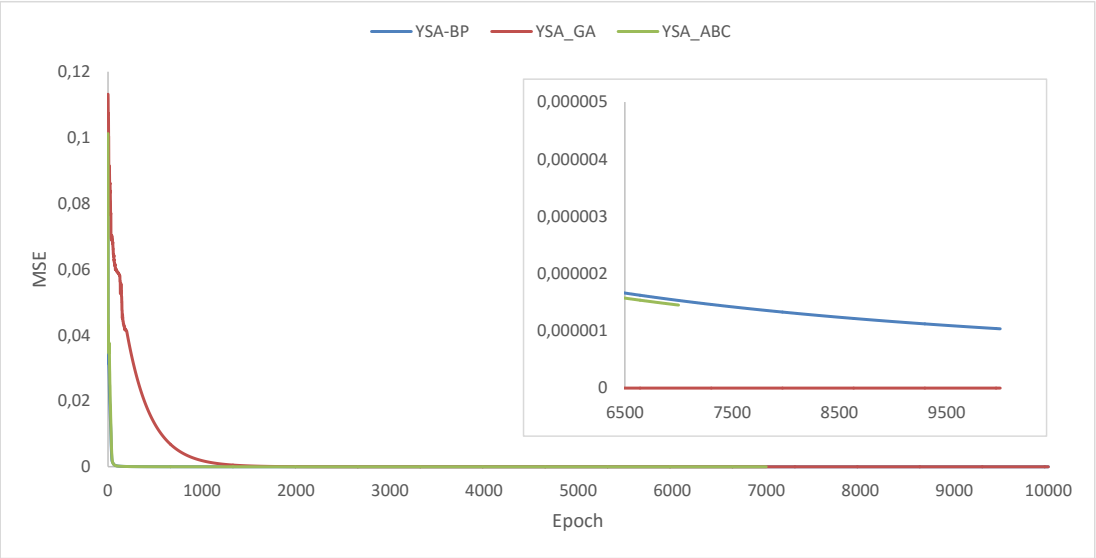
YSA-GA'nın eğitiminin başarılı olduğunu söyleyebilmek için test verisinde göstereceği başarısı da diğer iki algoritmayı geçmesi gerekmektedir. Bundan dolayı eğitilmiş ağlara uygulanan test işlemi Tablo 5.7.'deki MSE hata değerlerini ortaya çıkarmıştır. Bu hata değerlerinde en iyi performansı yine YSA-GA vermiştir. YSA-ABC'de çok yakın hata değeri performansı sergilemiştir. YSA-BP test veri performansında da en kötü algoritma olmuştur. YSA-GA, AC ikili gaz karışımını MC'ye göre daha iyi tahmin etmiştir. MC'nin tahmin edilme performansında YSA-ABC sınıflandırması, YSA-GA'ya çok yakın sonuç vermiştir.

Tablo 5.7. YSA-BP, YSA-ABC ve YSA-GA algoritmaları için ikili gaz karışım çalışmasındaki test verisi başarım sonuçları

Gizli Katmandaki Nöron Sayısı	Test Verisi MSE Hata Değeri					
	YSA-BP		YSA-GA		YSA-ABC	
	AC	MC	AC	MC	AC	MC
20	**5,56E-06**	**5,35E-06**	1,66E-02	2,68E-03	5,03E-07	8,19E-07
40	6,53E-06	6,52E-06	3,81E-02	4,33E-03	7,06E-07	1,27E-07
60	7,02E-06	7,29E-06	2,92E-04	2,80E-02	8,89E-07	9,79E-07
80	8,39E-06	8,59E-06	6,74E-06	1,22E-03	2,67E-07	1,95E-07
100	7,64E-06	8,04E-06	1,03E-05	2,94E-05	7,73E-07	2,88E-07
120	8,24E-06	7,02E-06	**2,91E-09**	**2,46E-08**	1,39E-05	6,90E-6
140	7,25E-06	7,02E-06	1,01E-07	7,80E-06	**2,81E-08**	**9,51E-08**

5.2.3. İkili gaz karışımlarının sınıflandırılmasında eğitim ve test başarım analizi

Bu çalışmada asetonun sabit tutulup, metanol ve kloroformun arttırıldığı, metanolun sabit tutulup kloroformun arttırıldığı ikili gaz karışım çalışmasında çok fazla sayıda senaryo çalıştırılmış ve sonuçlar elde edilmiştir. YSA-GA'nın nasıl bir performans sergilediğini görmek adına aynı senaryolar YSA-GA için çalıştırılmıştır. Üç algoritmanın en iyi test performansı veren senaryolarının eğitim grafiği seçilmiş ve Şekil 5.4.'te verilmiştir. Seçilen senaryolar YSA-BP için gizli katmanında 60 nöron olan ve 10000 epoch çalıştırılan yapay sinir ağı, YSA-GA için gizli katmanında 40 nöron olan ve 10000 epoch çalıştırılan yapay sinir ağı ve YSA-ABC için gizli katmanında 100 nöron olan ve 7000 epoch çalıştırılan yapay sinir ağıdır.



Şekil 5.4. YSA-BP, YSA-GA ve YSA-ABC algoritmalarının eğitimdeki MSE grafik karşılaştırması

Şekil 5.4.'ten de görüleceği üzere en düşük MSE değerini 1,54E-18'lik bir değer ile YSA-GA ulaşmıştır. Fakat aynı performansı Tablo 5.8.'den görüleceği üzere test verisinde gösterememiştir. Burada veriyi ezberlediği yorumu yapılabilir. Tablo 5.8.'de bütün senaryolardaki BP, GA ve ABC algoritmalarının test verisine karşı verdikleri hata değerleri görülmektedir. Test verisinde en iyi hata değerini 3,82E-12 ile YSA-ABC göstermiştir. En yakın hata değerleri yine YSA-ABC vermesine karşılık en yakın farklı algoritma, 1,27E-09 ile YSA-GA olmuştur. Sonuncu algoritma ise 2,00E-07 ile YSA-BP olmuştur. Bütün senaryolarda YSA-GA ile YSA-BP hemen hemen birbirlerine yakın performans göstermelerine karşılık YSA-ABC ile aralarındaki fark fazladır.

Tablo 5.8. YSA-BP, YSA-ABC ve YSA-GA algoritmaları için ikili gaz karışım çalışmasındaki test verisi başarım sonuçları

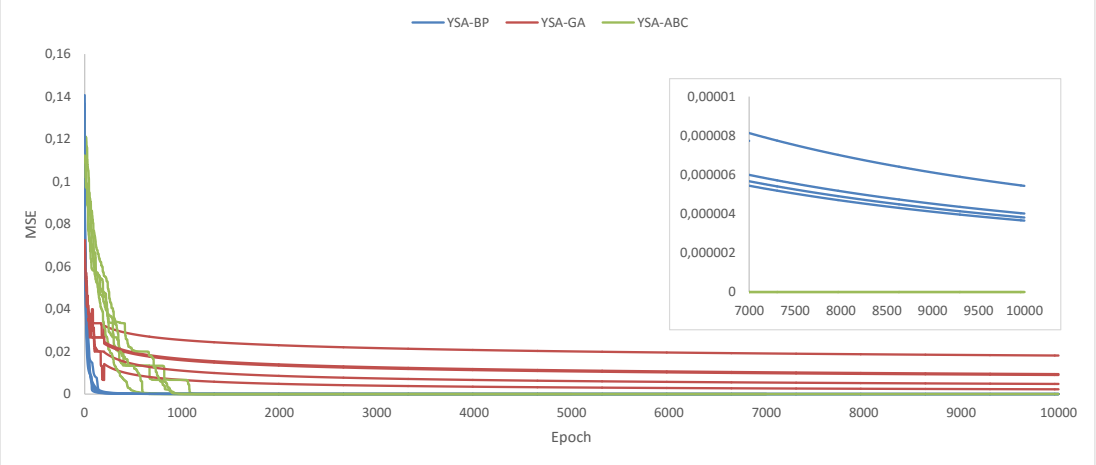
Epoch ve Gizli Katmandaki Nöron Sayısı		Test Verisi MSE Hata Değeri		
		YSA-BP	YSA-GA	YSA-ABC
3000	20	2,59E-06	2,65E-04	5,24E-04
	40	2,78E-05	2,29E-04	3,21E-04
	60	6,12E-06	6,50E-02	4,79E-06
	80	4,00E-06	1,13E-01	7,23E-07
	100	5,63E-06	5,60E-02	1,03E-04
5000	20	6,27E-07	1,29E-06	1,47E-07
	40	6,78E-06	5,62E-06	3,53E-07
	60	3,38E-05	4,38E-03	1,11E-07
	80	8,61E-07	2,94E-04	6,85E-09
	100	2,11E-06	3,12E-05	1,00E-07

Tablo 5.8. (Devamı)

Epoch ve Gizli Katmandaki Nöron Sayısı		Test Verisi MSE Hata Değeri		
		YSA-BP	YSA-GA	YSA-ABC
7000	20	1,43E-04	6,19E-06	1,93E-05
	40	2,01E-05	3,19E-07	2,70E-09
	60	2,06E-05	7,14E-07	1,01E-07
	80	4,90E-07	5,32E-06	8,74E-11
	100	9,59E-07	8,13E-07	**3,82E-12**
10000	20	1,34E-05	2,16E-07	1,41E-06
	40	3,55E-06	**1,27E-09**	5,08E-11
	60	**2,00E-07**	6,38E-07	4,32E-12
	80	2,28E-05	7,54E-07	6,34E-11
	100	5,08E-06	3,91E-07	3,40E-10

5.2.4. Farklı QCM gaz sensörlerinden elde edilen verilerin sınıflandırılmasında eğitim ve test başarım analizi

Çok fazla senaryoya sahip olan bu çalışmada, 5 farklı QCM yapısında 5 farklı gaz ölçümleri yapılmış ve elde edilen veri setlerinde toplamda 300 senaryo çalıştırılmıştır. YSA-ABC ile başarılı sonuçların alındığı bu çalışmada YSA-GA nasıl sonuç vereceğini görebilmek adına YSA-BP ve YSA-ABC'deki en başarılı tasarımlar YSA-GA'da çalıştırılmıştır. Beş farklı QCM sensör yapısındaki veri setleri üzerine yapılan eğitimlerden elde edilen MSE hata grafikleri Şekil 5.5.'te verilmiştir.



Şekil 5.5. YSA-BP, YSA-GA ve YSA-ABC algoritmalarının eğitimdeki MSE grafik karşılaştırması

Şekil 5.5.'ten görüldüğü üzere en düşük MSE hata değerlerine YSA-ABC ile erişilmiştir. YSA-GA, YSA-BP'yi geçememiş ve MSE değerleri yüksek seviyelerde kalmıştır. YSA-ABC ile eğitilen senaryolarda en düşük 3,16E-16 hata değeri elde

edilirken, YSA-BP ile yapılan eğitim YSA-GA'dan daha iyi eğitim grafiği göstermiş ve 3,65E-06 hata değeri elde edilmiştir. Bu çalışmadaki en kötü performans YSA-GA'ya ait olup 0,00224 hata değeri elde edilmiştir. QCM3 sensör çalışmasında YSA-ABC'de en düşük MSE değeri 3,16E-16, YSA-GA'da 0,0181 ve YSA-BP'de 3,65E-06 olmuştur. QCM6'da eğitimde erişilen en düşük MSE, YSA-ABC'de 3,91E-16, YSA-GA'da 0,0022 ve YSA-BP'de 3,81E-06 olmuştur. QCM7 sensör çalışmasında eğitimde erişilen en düşük MSE hata değerleri, YSA-ABC için 8,77E-16, YSA-GA için 0,0048 ve YSA-BP için 5,43E-06 olmuştur. QMC10'da eğitimde erişilen en düşük MSE hata değeri YSA-ABC için 8,18E-16, YSA-GA için 0,0090 ve YSA-BP için 4,02E-06 olmuştur. Son olarak QCM12'de eğitimde erişilen en düşük MSE hata değeri YSA-ABC için 4,98E-16, YSA-GA için 0,0094 ve YSA-BP için 7,73E-06 olmuştur.

Eğitilen bu ağlar üzerinde test verisi uygulanmış ve Tablo 5.9.'daki MSE değerleri elde edilmiştir. Tablo 5.9.'dan görüleceği üzere bütün QCM sensör yapılarındaki test verisinde en başarılı eğitim algoritması YSA-ABC olmuştur. Daha sonra YSA-BP ve en başarısız eğitim algoritması YSA-GA olmuştur.

Tablo 5.9. YSA-BP, YSA-ABC ve YSA-GA algoritmalarının QCM gaz sensör çalışmasındaki test verisi başarım sonuçları

	Epoch ve Gizli Katmandaki Nöron Sayısı	Test Verisi MSE Hata Değeri		
		YSA-BP	YSA-GA	YSA-ABC
QCM3	10000 60	4,18E-06	0,00367	7,467E-16
	5000 40	1,01E-05	0,00618	2,33E-16
QCM6	5000 70	7,89E-06	0,00795	1,41E-16
	10000 70	3,94E-06	0,00351	2,09E-16
QCM7	5000 30	7,56E-06	0,00637	1,53E-16
	10000 40	3,23E-06	0,00699	4,33E-16
QCM10	7000 40	8,14E-06	0,00462	1,74E-16
	10000 80	4,05E-06	0,00391	7,55E-15

Tablo 5.9. (Devamı)

	Epoch ve Gizli Katmandaki Nöron Sayısı	Test Verisi MSE Hata Değeri		
		YSA-BP	YSA-GA	YSA-ABC
QCM12	7000 50	5,52E-06	0,00307	2,44E-15
	10000 70	8,04E-06	0,00283	4,20E-16

Yapılan 4 çalışmanın üçünde YSA-ABC en iyi performansı verirken, sadece 1 çalışmada YSA-GA daha iyi performans vermiştir. Fakat bu çalışmada da YSA-ABC, YSA-GA'ya göre çok yakın değerler üretebilmiştir. YSA-GA'nın eğitimindeki sürenin aşırı uzunluğu ve test verisindeki düşük performansı, eğitim için YSA-ABC'nin seçilmesini daha cazip kılmaktadır.

BÖLÜM 6. SONUÇLAR VE ÖNERİLER

Elektronik burun, sağlık alanından, endüstriye, savunma sanayinden, yiyecek kalitesine kadar birçok alanda kullanılmaktadır. Elektronik burunda kullanılan koku sensörlerinin sadece belli kokular için kullanılabildiğinden piyasada çok fazla sayıda ve çeşitlilikte elektronik burun bulunmaktadır. Bu tez kapsamında incelenen elektronik burun çeşitlerine bakıldığında, her biri özellikle bir alanda daha çok kullanıldığı görülecektir. Bu tez kapsamında yapılan 4 farklı çalışmada MOSES II, dokuz ve tek QCM sensöre sahip farklı elektronik burunlar kullanılmıştır.

Elektronik burundan elde edilen ham verilerden belli aralıklarla veya tepe nokta değerleri alınıp veri seti oluşturulmuştur. Tasarlanan YSA'larde aktivasyon fonksiyonu Sigmoid kullanıldığı için veri setlerindeki değerler [0 1] aralığına normalize edilmiş fakat bazı çalışmalarda verideki sıfır ve bir değeri eğitimin yanlış yakınsamasına neden olabileceği düşünüldüğü için, [0,1 0,9] aralığına normalize edilmiş ve normalizsyon tekniği olarak min-max normalizasyonu kullanılmıştır. Bu şekilde bazı koku veri setlerinde daha başarılı sonuçlar alındığı görülmüştür.

Bu tez kapsamında Microsoft Visual C# kullanılarak bir yazılım geliştirilmiştir. Bu yazılım sayesinde programa okutulan koku verisinin, hangi parametrelerinin girdi veya çıktı oldukları ve çıkarılacak nitelikler belirtildikten sonra YSA'nın ve eğitim modelinin tasarlanmasına geçilebilir. Yazılım sayesinde YSA'da kaç gizli katman ve her gizli katmanda kaç adet nöron olacağı ayarlanabilir. YSA'nın tasarımı görsel olarak kullanıcıya sunulduğu için kullanıcı nasıl bir ağ tasarladığını kolaylıkla anlayabilmektedir. Geliştirilen yazılımda YSA'nın eğitimi istenirse ABC, GA veya her ikisi ile de yapılabilir. Program otomatik olarak eğitim modelini BP ile karşılaştıracaktır. Literatürdeki diğer çalışmalardan farklı olarak ABC ve GA ile yapılan eğitim sırasında baz alınacak hata hesaplama modeli MSE, MAE veya R^2

olabilmektedir. Bazı koku veri setlerinde MAE ile eğitimin yapılması MSE veya R^2'ye göre daha başarılı sonuçlar alınmasını sağlamıştır.

Yapılan 4 çalışmanın ilkinde MOSES II elektronik burunu kullanılarak 4 farklı aroma türü YSA-ABC eğitim modeli kullanılarak başarılı bir şekilde sınıflandırılmıştır. YSA-ABC'nin ulaştığı düşük MSE hata değeri ve test verisindeki %76,39 ortalama performansı ile YSA-BP ve YSA-GA'yı geride bırakmıştır. YSA-BP'nin bir aromadaki çok düşük performansına karşılık YSA-ABC'de 4 aromada da yüksek performans elde edilmiştir.

Yapılan ikinci çalışmada AC ve MC ikili gaz karışımları sınıflandırılmış, eğitim ve test verisinde en iyi performansı YSA-GA göstermiştir. Eğitimde ulaştığı çok düşük MSE değerine test verisinde ulaşamamış fakat YSA-ABC'den daha iyi sonuç vermiştir. YSA-GA test verisinde AC ikili gaz karışımı için 2,91E-09 ve MC için 2,41E-08 MSE hata değerlerine ulaşmıştır.

Yapılan üçüncü çalışmada MA, CA ve CM ikili gaz karışımları sınıflandırılmış ve eğitim sırasında en düşük MSE hata değerine YSA-GA ulaşmıştır. Fakat aynı başarıyı test verisinde gösterememiş ve YSA-ABC test verisini daha iyi sınıflandırmıştır. YSA-ABC'nin test verisinde ulaştığı en düşük MSE değeri 3,82E-12 iken YSA-GA 1,27E-09 ve YSA-BP ise 2,00E-07 olmuştur. Bu YSA-GA'nın aslında eğitim verisini ezberlediği göstermektedir.

Yapılan dördüncü çalışmada 5 farklı gaz kullanılan 5 farklı QCM sensör üzerinde ayrı ayrı ölçümü yapılmış, eğitimler ve test verisinde en iyi başarıyı YSA-ABC göstermiştir. Bu sonuç 5 farklı QCM sensör yapısında da aynı olmuştur. YSA-ABC test verisinde genelde E-16 hata seviyelerine erişmiş, bu değer YSA-GA için E-03 ve YSA-BP için E-06 seviyelerinde kalmıştır. QCM sensör yapısının değiştirilmesinin YSA-ABC'deki yüksek performansı düşürmediği, bilakis her sensör yapısında aynı üstün başarıyı gösterdiği gözlemlenmiştir.

Yapılan çalışmalardan Sadece AC ve MC ikili gaz karışımlarının sınıflandırıldığı çalışmada YSA-GA başarılı olmuş, bir çalışmada ise eğitimde en iyi olmasına rağmen testte aynı başarıyı gösterememiştir. 2 çalışmada YSA-ABC en iyi eğitim ve test performansını göstermiş, AC ve MC ikili gaz karışım çalışması hariç diğer 3 çalışmada ise en iyi test performansını sergilemiştir. Yapay sinir ağlarında asıl önemli olan, ağın hiç görmediği test verisine karşı gösterdiği başarıdır. Bundan dolayı 4 çalışmanın üçünde diğer eğitim modellerine göre başarılı sonuç vermiştir.

Tez kapsamında yapılan 4 çalışmada ağların eğitim süreleri analiz edilmiş. Bütün çalışmalarda en hızlı eğitim algoritması YSA-BP olmuştur. Senaryoların genelinde saniyeler içerisinde hesaplamalar yapılmıştır. İkinci en hızlı eğitim algoritması YSA-ABC olmuştur. Eğitim süresi genelde dakikalar düzeyinde sürmüştür. En yavaş eğitim algoritması YSA-GA olmuştur. Saatlerce süren eğitim neticesinde kabul edilebilir hata değerlerine ulaşabilmiştir. Koku verisinin sınıflandırılmasında YSA-BP hızlı olmasına karşılık kabul edilebilir hata değerlerine ulaşamamıştır. Diğer tarafta makul sürede eğitim yapabilen YSA-ABC 4 çalışmanın 3'ünde gösterdiği yüksek performans ile koku sınıflandırmasında başarılı olduğunu göstermiştir.

Çalışmalarda çok fazla senaryo denenmiş bu denemelerde YSA'da gizli katman sayısının belli oranda arttırılması veri setine bağlı olarak YSA-ABC'de daha başarılı sonuçlar alınmasını sağlamıştır. Aynı durum YSA-BP ve YSA-GA'da görülmemiştir. Yine ağırlıkların alabileceği alt ve üst limitlere farklı değerler verilmesinin yapılan senaryolarda YSA-ABC ve YSA-GA için çok önemli olduğu görülmüştür. Elektronik burun verisinin sınıflandırılmasında YSA-ABC ve YSA-GA eğitimlerinde ağırlıklar [-1 1] aralığında çok iyi sonuçlar veremezken, [-10 10] aralığında başarılı sonuçlar verdiği görülmüştür. YSA-GA'da popülasyon, YSA-ABC'de yiyecek kaynak sayısının büyüklüğü sonuca daha hızlı varmada etkili olabilmektedir. Fakat aşırı derecede bu sayıları arttırmak eğitim süresinin çok uzun sürmesine neden olmaktadır.

İleriye dönük önerilen çalışmalar olarak:

- Eğitimde PSO, BMO ve özellikle SVM gibi farklı algoritmaların denenmesi ve bu süreçlerin geliştirilen yazılıma entegre edilmesi,
- Geliştirilen yazılımın, koku veri setine göre tasarlanacak yapay sinir ağında gizli katman ağ ve nöron sayısını önerebilmesi,
- Yazılımın, seçilen bir elektronik burun ile kurulumunun yapılıp entegre çalışmasının sağlanması,

gibi çalışmalar denenebilir.

KAYNAKLAR

[1] Bishop, C. M., Pattern Recognition and Machine Learning. 1st. Ed., Springer India, Singapore, 738 , 2006.

[2] Pearce, T. C., Schiffman, S. S., Nagle, H. T., Gardner, J. W., Handbook of Machine Olfaction: Electronic Nose Technology. 624 , 2006.

[3] Wilson, A. D., Baietto, M., Applications and Advances in Electronic-Nose Technologies. Sensors, 9 (7), 5099–5148, 2009.

[4] Saraoglu, H. M., Elektronik Burun Teknolojisi ve Uygulama Alanları. Akademik Bilişim, , 419–427, 2008.

[5] Chiu, S.-W., Tang, K.-T., Towards a Chemiresistive Sensor-Integrated Electronic Nose: A Review. Sensors, 13 (10), 14214–14247, 2013.

[6] Peris, M., Escuder-Gilabert, L., A 21st century technique for food control: electronic noses. Analytica chimica acta, 638 (1), 1–15, 2009.

[7] Wilson, A., Diverse Applications of Electronic-Nose Technologies in Agriculture and Forestry. Sensors, 13 (2), 2295–2348, 2013.

[8] Baietto, M., Wilson, A., Electronic-Nose Applications for Fruit Identification, Ripeness and Quality Grading. Sensors, 15 (1), 899–931, 2015.

[9] Ampuero, S., Bosset, J. O., The electronic nose applied to dairy products: a review. Sensors and Actuators B: Chemical, 94 (1), 1–12, 2003.

[10] Padilla, M., Montoliu, I., Pardo, A., Perera, A., Marco, S., Feature extraction on three way enose signals. Sensors and Actuators B: Chemical, 116 (1)–(2), 145–150, 2006.

[11] Ozmen, A., Dogan, E., Design of a Portable E-Nose Instrument for Gas Classifications. IEEE Transactions on Instrumentation and Measurement, 58 (10), 3609–3618, 2009.

[12] Soh, A. C., Chow, K. K., Mohammad Yusuf, U. K., Ishak, A. J., Hassan, M. K., Khamis, S., Development of neural network-based electronic nose for herbs recognition. International Journal on Smart Sensing and Intelligent Systems, 7 (2), 584–609, 2014.

[13] Omatu, S., Yano, M., E-nose system by using neural networks. Neurocomputing, 172, 394–398, 2016.

[14] Patel, H. K., The Electronic Nose: Artificial Olfaction Technology. Springer India, New Delhi, 247 , 2014.

[15] Cho, J. H., Kurup, P. U., Decision tree approach for classification and dimensionality reduction of electronic nose data. Sensors and Actuators B: Chemical, 160 (1), 542–548, 2011.

[16] Xiao, Z., Yu, D., Niu, Y., Chen, F., Song, S., Zhu, J., Zhu, G., Characterization of aroma compounds of Chinese famous liquors by gas chromatography-mass spectrometry and flash GC electronic-nose. Journal of chromatography. B, Analytical technologies in the biomedical and life sciences, 945–946, 92–100, 2014.

[17] Gupta, S., Variyar, P. S., Sharma, A., Application of mass spectrometry based electronic nose and chemometrics for fingerprinting radiation treatment. Radiation Physics and Chemistry, 106, 348–354, 2015.

[18] Versari, A., Laurie, V. F., Ricci, A., Laghi, L., Parpinello, G. P., Progress in authentication, typification and traceability of grapes and wines by chemometric approaches. Food Research International, 60, 2–18, 2014.

[19] Zakaria, A., Ali, A. Y., Adom, A. H., Ahmad, M. N., Masnan, M. J., Aziz, A. H. A., Fikri, N. A., Abdullah, A. H., Kamarudin, L. M., Improved classification of Orthosiphon stamineus by data fusion of electronic nose and tongue sensors. Sensors, 10 (10), 8782–8796, 2010.

[20] Lebrun, M., Plotto, A., Goodner, K., Ducamp, M.-N., Baldwin, E., Discrimination of mango fruit maturity by volatiles using the electronic nose and gas chromatography. Postharvest Biology and Technology, 48 (1), 122–131, 2008.

[21] Dymerski, T., Gębicki, J., Wardencki, W., Namieśnik, J., Application of an Electronic Nose Instrument to Fast Classification of Polish Honey Types. Sensors, 14 (6), 10709–10724, 2014.

[22] Ghasemi-varnamkhasti, M., Mohtasebi, S. S., Razavi, S. H., Ahmadi, H., Dicko, A., Discriminatory Power Assessment of the Sensor Array of an Electronic Nose System for the Detection of Non Alcoholic Beer Aging. Czech Journal of Food Science, 30 (3), 236–240, 2012.

[23] Dutta, R., Hines, E. L., Gardner, J. W., Boilot, P., Bacteria classification using Cyranose 320 electronic nose. BioMedical Engineering OnLine, 1 (4), 1–7, 2002.

[24] Tran, V. H., Thurston, M., Jackson, P., Lewis, C., Yates, D., Bell, G., Thomas, P. S., Breath Analysis of Lung Cancer Patients Using an Electronic Nose Detection System. IEEE Sensors Journal, 10 (9), 1514–1518, 2010.

[25] Zhou, B., Wang, J., Discrimination of different types damage of rice plants by electronic nose. Biosystems Engineering, 109, 250–257, 2011.

[26] Suh, C. P.-C., Ding, N., Lan, Y., Using an Electronic Nose to Rapidly Assess Grandlure Content in Boll Weevil Pheromone Lures. Journal of Bionic Engineering, 8 (4), 449–454, 2011.

[27] Pizzoni, D., Compagnone, D., Di Natale, C., D'Alessandro, N., Pittia, P., Evaluation of aroma release of gummy candies added with strawberry flavours by gas-chromatography/mass-spectrometry and gas sensors arrays. Journal of Food Engineering, 167, 77–86, 2015.

[28] Qin, Z., Pang, X., Chen, D., Cheng, H., Hu, X., Wu, J., Evaluation of Chinese tea by the electronic nose and gas chromatography–mass spectrometry: Correlation with sensory properties and classification according to grade level. Food Research International, 53 (2), 864–874, 2013.

[29] Li, Y., Lei, J., Liang, D., Identification of Fake Green Tea by Sensory Assessment and Electronic Tongue. Food Science and Technology Research, 21 (2), 207–212, 2015.

[30] Breijo, E. G., Guarrasi, V., Peris, R. M., Fillol, M. A., Pinatti, C. O., Odour sampling system with modifiable parameters applied to fruit classification. Journal of Food Engineering, 116 (2), 277–285, 2013.

[31] Cui, S., Wang, J., Yang, L., Wu, J., Wang, X., Qualitative and quantitative analysis on aroma characteristics of ginseng at different ages using E-nose and GC-MS combined with chemometrics. Journal of pharmaceutical and biomedical analysis, 102, 64–77, 2015.

[32] Zheng, X., Lan, Y., Zhu, J., Westbrook, J., Hoffmann, W. C., Lacey, R. E., Rapid Identification of Rice Samples Using an Electronic Nose. Journal of Bionic Engineering, 6 (3), 290–297, 2009.

[33] Rodriguez, S. D., Monge, M. E., Olivieri, A. C., Negri, R. M., Bernik, D. L., Time dependence of the aroma pattern emitted by an encapsulated essence studied by means of electronic noses and chemometric analysis. Food Research International, 43 (3), 797–804, 2010.

[34] Deshmukh, S., Jana, A., Bhattacharyya, N., Bandyopadhyay, R., Pandey, R. a., Quantitative determination of pulp and paper industry emissions and associated odor intensity in methyl mercaptan equivalent using electronic nose. Atmospheric Environment, 82, 401–409, 2014.

[35] Musatov, V. Y., Sysoev, V. V., Sommer, M., Kiselev, I., Assessment of meat freshness with metal oxide sensor microarray electronic nose: A practical approach. Sensors and Actuators B: Chemical, 144 (1), 99–103, 2010.

[36] Liu, M., Han, X., Tu, K., Pan, L., Tu, J., Tang, L., Liu, P., Zhan, G., Zhong, Q., Xiong, Z., Application of electronic nose in Chinese spirits quality control and flavour assessment. Food Control, 26 (2), 564–570, 2012.

[37] Singh, H., Raj, V. B., Kumar, J., Mittal, U., Mishra, M., Nimal, A. T., Sharma, M. U., Gupta, V., Metal oxide SAW E-nose employing PCA and ANN for the identification of binary mixture of DMMP and methanol. Sensors and Actuators B: Chemical, 200, 147–156, 2014.

[38] Wei, G., An, W., Zhu, Z., Gas Mixture Quantification Based on Hilbert–Huang Transform and Neural Network by a Single Sensor. International Journal of Pattern Recognition and Artificial Intelligence, 25 (06), 927–942, 2011.

[39] Gulbag, A., Temurtas, F., Yusubov, I., Quantitative discrimination of the binary gas mixtures using a combinational structure of the probabilistic and multilayer neural networks. Sensors and Actuators B: Chemical, 131 (1), 196–204, 2008.

[40] Haykin, S., Neural Networks: A Comprehensive Foundation. 2. Ed., Prentice Hall PTR, , 1999.

[41] Al-Bastaki, Y., An Artificial Neural Networks-Based on-Line Monitoring Odor Sensing System. Journal of Computer Science, 5 (11), 878–882, 2009.

[42] Bhattacharya, N., Tudu, B., Jana, A., Ghosh, D., Bandhopadhyaya, R., Bhuyan, M., Preemptive identification of optimum fermentation time for black tea using electronic nose. Sensors and Actuators B: Chemical, 131 (1), 110–116, 2008.

[43] Cevoli, C., Cerretani, L., Gori, A., Caboni, M. F., Gallina Toschi, T., Fabbri, A., Classification of Pecorino cheeses using electronic nose combined with artificial neural network and comparison with GC–MS analysis of volatile compounds. Food Chemistry, 129 (3), 1315–1319, 2011.

[44] Morita, A., Araki, T., Ikegami, S., Okaue, M., Sumi, M., Ueda, R., Sagara, Y., Coupled Stepwise PLS-VIP and ANN Modeling for Identifying and Ranking Aroma Components Contributing to the Palatability of Cheddar Cheese. Food Science and Technology Research, 21 (2), 175–186, 2015.

[45] Gursoy, O., Somervuo, P., Alatossava, T., Preliminary study of ion mobility based electronic nose MGD-1 for discrimination of hard cheeses. Journal of Food Engineering, 92 (2), 202–207, 2009.

[46] Pavlou, A. K., Magan, N., Jones, J. M., Brown, J., Klatser, P., Turner, A. P. F., Detection of Mycobacterium tuberculosis (TB) in vitro and in situ using an electronic nose in combination with a neural network system. Biosensors & bioelectronics, 20 (3), 538–44, 2004.

[47] Saraoglu, H. M., Edin, B., E-Nose System for Anesthetic Dose Level Detection using Artificial Neural Network. Journal of Medical Systems, 31 (6), 475–482, 2007.

[48] Vasant, P. M., Meta-Heuristics Optimization Algorithms in Engineering, Business, Economics, and Finance. 1. Ed., IGI Global, 734 , 2013.

[49] Dhanarajan, G., Mandal, M., Sen, R., A combined artificial neural network modeling–particle swarm optimization strategy for improved production of marine bacterial lipopeptide from food waste. Biochemical Engineering Journal, 84, 59–65, 2014.

[50] Ghaedi, M., Ansari, A., Bahari, F., Ghaedi, A. M., Vafaei, A., A hybrid artificial neural network and particle swarm optimization for prediction of removal of hazardous dye brilliant green from aqueous solution using zinc sulfide nanoparticle loaded on activated carbon. Spectrochimica acta. Part A, Molecular and biomolecular spectroscopy, 137, 1004–15, 2015.

[51] Ghaedi, M., Ghaedi, A. M., Ansari, A., Mohammadi, F., Vafaei, A., Artificial neural network and particle swarm optimization for removal of methyl orange by gold nanoparticles loaded on activated carbon and Tamarisk. Spectrochimica acta. Part A, Molecular and biomolecular spectroscopy, 132, 639–54, 2014.

[52] Yan, J., Tian, F., Feng, J., Jia, P., He, Q., Shen, Y., A PSO-SVM Method for Parameters and Sensor Array Optimization in Wound Infection Detection based on Electronic Nose. Journal of Computers, 7 (11), 2012.

[53] Vapnik, V. N., The Nature of Statistical Learning Theory. Springer New York, New York, NY, , 2000.

[54] Haddi, Z., Amari, a., Alami, H., El Bari, N., Llobet, E., Bouchikhi, B., A portable electronic nose system for the identification of cannabis-based drugs. Sensors and Actuators B: Chemical, 155 (2), 456–463, 2011.

[55] Qiu, S., Gao, L., Wang, J., Classification and regression of ELM, LVQ and SVM for E-nose data of strawberry juice. Journal of Food Engineering, 144, 77–85, 2015.

[56] Brudzewski, K., Classification of milk by means of an electronic nose and SVM neural network. Sensors and Actuators B: Chemical, 98 (2)–(3), 291–298, 2004.

[57] Brudzewski, K., Osowski, S., Golembiecka, a., Differential electronic nose and support vector machine for fast recognition of tobacco. Expert Systems with Applications, 39 (10), 9886–9891, 2012.

[58] Wang, D., Wang, X., Liu, T., Liu, Y., Prediction of total viable counts on chilled pork using an electronic nose combined with support vector machine. Meat science, 90 (2), 373–7, 2012.

[59] Zhang, L., Tian, F., Nie, H., Dang, L., Li, G., Ye, Q., Kadri, C., Classification of multiple indoor air contaminants by an electronic nose and a hybrid support vector machine. Sensors and Actuators B: Chemical, 174, 114–125, 2012.

[60] Guney, S., Atasoy, A., Multiclass classification of n-butanol concentrations with k-nearest neighbor algorithm and support vector machine in an electronic nose. Sensors and Actuators B: Chemical, 166–167, 721–725, 2012.

[61] Pardo, M., Sberveglieri, G., Classification of electronic nose data with support vector machines. Sensors and Actuators B: Chemical, 107 (2), 730–737, 2005.

[62] Narendra, Fukunaga, A Branch and Bound Algorithm for Feature Subset Selection. IEEE Transactions on Computers, C–26 (9), 917–922, 1977.

[63] Alshamlan, H. M., Badr, G. H., Alohali, Y. A., Genetic Bee Colony (GBC) algorithm: A new gene selection method for microarray cancer classification. Computational biology and chemistry, 56, 49–60, 2015.

[64] Nasimi, R., Irani, R., Moradi, B., An Improved Ant Colony Algorithm–Based ANN for Bottom Hole Pressure Prediction in Underbalanced Drilling. Petroleum Science and Technology, 30 (13), 1307–1316, 2012.

[65] Askarzadeh, A., Rezazadeh, A., Artificial neural network training using a new efficient optimization algorithm. Applied Soft Computing Journal, 13 (2), 1206–1213, 2013.

[66] Torrecilla, J. S., Otero, L., Sanz, P. D., Optimization of an artificial neural network for thermal/pressure food processing: Evaluation of training algorithms. Computers and Electronics in Agriculture, 56 (2), 101–110, 2007.

[67] Chen, D., Lu, R., Zou, F., Li, S., Teaching-learning-based optimization with variable-population scheme and its application for ANN and global optimization. Neurocomputing, 173, 1096–1111, 2016.

[68] Goldberg, D. E., Holland, J. H., Genetic Algorithms and Machine Learning. Machine Learning, 3 (2/3), 95–99, 1988.

[69] Mahajan, R., Kaur, G., Neural Networks using Genetic Algorithms. International Journal of Computer Applications, 77 (14), 6–11, 2013.

[70] Sajan, K. S., Tyagi, B., Kumar, V., Genetic algorithm based artificial neural network model for voltage stability monitoring. 18th National Power Systems Conference, NPSC, , 1–5, 2014.

[71] Zaji, A. H., Bonakdari, H., Application of artificial neural network and genetic programming models for estimating the longitudinal velocity field in open channel junctions. Flow Measurement and Instrumentation, 41, 81–89, 2015.

[72] Mellit, A., ANN-based GA for generating the sizing curve of stand-alone photovoltaic systems. Advances in Engineering Software, 41 (5), 687–693, 2010.

[73] Jin Ma, Bing-Shu Wang, Yong-Guang Ma, ANN-based real-time parameter optimization via GA for superheater model in power plant simulator. 2008 International Conference on Machine Learning and Cybernetics, , 2269–2273, 2008.

[74] Kim, E., Lee, J. H., Shin, B. J., Lee, S., Byun, Y. T., Kim, J. H., Kim, H. S., Lee, T., An Odor Monitoring System Based on Differentiated Pattern Recognition Implemented a Semiconductor Gas Sensor Array. Advanced Science Letters, 19 (10), 2901–2904, 2013.

[75] Li, H., Wang, D., Zhang, Y., Knowledge-based genetic algorithms data fusion and its application in mine mixed-gas detection. Journal of Software, 7 (2), 303–307, 2012.

[76] Nezhadali, A., Sadeghzadeh, S., Optimization of stripping voltammetric sensor by mixture design-artificial neural network-genetic algorithm for determination of trace copper(II) based on iodoquinol-carbon nanotube modified carbon paste electrode. Sensors and Actuators B: Chemical, 224, 134–142, 2016.

[77] Papes Filho, A. C., Maciel Filho, R., Hybrid training approach for artificial neural networks using genetic algorithms for rate of reaction estimation: Application to industrial methanol oxidation to formaldehyde on silver catalyst. Chemical Engineering Journal, 157 (2)–(3), 501–508, 2010.

[78] Baklacioglu, T., Modeling the fuel flow-rate of transport aircraft during flight phases using genetic algorithm-optimized neural networks. Aerospace Science and Technology, 49, 52–62, 2016.

[79] Shi, B., Zhao, L., Zhi, R., Xi, X., Optimization of electronic nose sensor array by genetic algorithms in Xihu-Longjing Tea quality analysis. Mathematical and Computer Modelling, 58 (3)–(4), 752–758, 2013.

[80] A, M., R, R., A Comparison of Artificial Bee Colony algorithm and Genetic Algorithm to Minimize the Makespan for Job Shop Scheduling. Procedia Engineering, 97, 1745–1754, 2014.

[81] TIAN, F., Jia, Y., XU, S., FENG, J., HE, Q., SHEN, Y., JIA, P., KADRI, C., Classification of electronic nose data on wound infection detection using support vector machine combined GA. Journal of Computational Information Systems, 8 (8), 3349–3357, 2012.

[82] Karaboga, D., An idea based on Honey Bee Swarm for Numerical Optimization. Technical Report TR06, Erciyes University, , 10, 2005.

[83] Bullinaria, J. A., AlYahya, K., Artificial Bee Colony training of neural networks: comparison with back-propagation. Memetic Computing, 6 (3), 171–182, 2014.

[84] Karaboga, D., Akay, B., Artificial Bee Colony (ABC) Algorithm on Training Artificial Neural Networks. 2007 IEEE 15th Signal Processing and Communications Applications, , 1–4, 2007.

[85] Uzlu, E., Akpinar, A., Ozturk, H. T., Nacar, S., Kankal, M., Estimates of hydroelectric generation using neural networks with the artificial bee colony algorithm for Turkey. Energy, 69, 638–647, 2014.

[86] Ozkan, C., Ozturk, C., Sunar, F., Karaboga, D., The artificial bee colony algorithm in training artificial neural network for oil spill detection. Neural Network World, 21 (6), 473–492, 2011.

[87] Ebrahimi, E., Monjezi, M., Khalesi, M. R., Armaghani, D. J., Prediction and optimization of back-break and rock fragmentation using an artificial neural network and a bee colony algorithm. Bulletin of Engineering Geology and the Environment, 75 (1), 27–36, 2016.

[88] Garro, B. A., Rodríguez, K., Vázquez, R. A., Classification of DNA microarrays using artificial neural networks and ABC algorithm. Applied Soft Computing, 38, 548–560, 2016.

[89] Anuar, S., Selamat, A., Sallehuddin, R., Hybrid Artificial Neural Network with Artificial Bee Colony Algorithm for Crime Classification. , 31–40, 2015.

[90] Adak, M., Yumusak, N., Classification of E-Nose Aroma Data of Four Fruit Types by ABC-Based Neural Network. Sensors, 16 (3), 304, 2016.

[91] Gardner, J. W., Bartlett, P. N., Electronic Noses: Principles and Applications. 1st. Ed., OXFORD, 264 , 1999.

[92] Adak, M. F., Yumusak, N., Electronic Noses that are used in Academic Studies. 2nd International Symposium on Innovative Technologies in Engineering and Science, , 1–12, 2014.

[93] Poprawski, J., Boilot, P., Tetelin, F., Counterfeiting and quantification using an electronic nose in the perfumed cleaner industry. Sensors and Actuators B: Chemical, 116 (1)–(2), 156–160, 2006.

[94] Song, S., Zhang, X., Hayat, K., Jia, C., Xia, S., Zhong, F., Xiao, Z., Tian, H., Niu, Y., Correlating chemical parameters of controlled oxidation tallow to gas chromatography-mass spectrometry profiles and e-nose responses using partial least squares regression analysis. Sensors and Actuators, B: Chemical, 147, 660–668, 2010.

[95] Pardo, M., Sberveglieri, G., Gardini, S., Dalcanale, E., A hierarchical classification scheme for an Electronic Nose. Sensors and Actuators B: Chemical, 69 (3), 359–365, 2000.

[96] Falasconi, M., Gobbi, E., Pardo, M., Della Torre, M., Bresciani, A., Sberveglieri, G., Detection of toxigenic strains of Fusarium verticillioides in corn by electronic olfactory system. Sensors and Actuators, B: Chemical, 108, 250–257, 2005.

[97] Gibson, T. D., Prosser, O., Hulbert, J. N., Marshall, R. W., Corcoran, P., Lowery, P., Ruck-Keene, E. A., Heron, S., Detection and simultaneous identification of microorganisms from headspace samples using an electronic nose. Sensors and Actuators B: Chemical, 44 (1)–(3), 413–422, 1997.

[98] Pavlou, A. K., Magan, N., McNulty, C., Jones, J., Sharp, D., Brown, J., Turner, A. P. F., Use of an electronic nose system for diagnoses of urinary tract infections. Biosensors & bioelectronics, 17 (10), 893–9, 2002.

[99] Dragonieri, S., Brinkman, P., Mouw, E., Zwinderman, A. H., Carratú, P., Resta, O., Sterk, P. J., Jonkers, R. E., An electronic nose discriminates exhaled breath of patients with untreated pulmonary sarcoidosis from controls. Respiratory medicine, 107 (7), 1073–8, 2013.

[100] Plaza, V., Crespo, A., Giner, J., Merino, J. L., Ramos-Barbón, D., Mateus, E. F., Torrego, A., Cosio, B. G., Agustí, A., Sibila, O., Inflammatory asthma phenotype discrimination using an electronic nose breath analyzer. Journal of Investigational Allergology and Clinical Immunology, 25 (6), 431–437, 2015.

[101] Dragonieri, S., Annema, J. T., Schot, R., van der Schee, M. P. C., Spanevello, A., Carratú, P., Resta, O., Rabe, K. F., Sterk, P. J., An electronic nose in the discrimination of patients with non-small cell lung cancer and COPD. Lung cancer (Amsterdam, Netherlands), 64 (2), 166–70, 2009.

[102] Dragonieri, S., Schot, R., Mertens, B. J. a, Le Cessie, S., Gauw, S. a, Spanevello, A., Resta, O., Willard, N. P., Vink, T. J., Rabe, K. F., Bel, E. H., Sterk, P. J., An electronic nose in the discrimination of patients with asthma and controls. The Journal of allergy and clinical immunology, 120 (4), 856–62, 2007.

[103] Zhang, H., Wang, J., Ye, S., Predictions of acidity, soluble solids and firmness of pear using electronic nose technique. Journal of Food Engineering, 86, 370–378, 2008.

[104] Zhang, H., Rapid and On-Line Instrumentation for Food Quality Assurance. Rapid and On-Line Instrumentation for Food Quality Assurance, Elsevier, 324-338 , 2003.

[105] Olafsdottir, G., Nesvadba, P., Di Natale, C., Careche, M., Oehlenschläger, J., Tryggvadóttir, S. V, Schubring, R., Kroeger, M., Heia, K., Esaiassen, M., Macagnano, A., Jørgensen, B. M., Multisensor for fish quality determination. Trends in Food Science & Technology, 15 (2), 86–93, 2004.

[106] Natale, C. Di, Olafsdottir, G., Einarsson, S., Martinelli, E., Paolesse, R., D'Amico, A., Comparison and integration of different electronic noses for freshness evaluation of cod-fish fillets. Sensors and Actuators B: Chemical, 77 (1)–(2), 572–578, 2001.

[107] Garcia, M., Aleixandre, M., Gutierrez, J., Horrillo, M. C., Electronic nose for ham discrimination. Sensors and Actuators, B: Chemical, 114, 418–422, 2006.

[108] Popov, T. A., Human exhaled breath analysis. Annals of Allergy, Asthma and Immunology, 106, 451–456, 2011.

[109] Van Deventer, D., Mallikarjunan, P., Optimizing an electronic nose for analysis of volatiles from printing inks on assorted plastic films. Innovative Food Science and Emerging Technologies, 3, 93–99, 2002.

[110] Horner, G., Keil, E. M., Chemosensory system for rapid automated quality control. Journal of Chromatography A, 845, 85–92, 1999.

[111] Stuetz, R., Integrated Analytical Systems. Comprehensive Analytical Chemistry, Elsevier, 513-539 , 2003.

[112] Bourgeois, W., Stuetz, R. M., Use of a chemical sensor array for detecting pollutants in domestic wastewater. Water Research, 36, 4505–4512, 2002.

[113] Bourgeois, W., Gardey, G., Servieres, M., Stuetz, R. M., A chemical sensor array based system for protecting wastewater treatment plants. Sensors and Actuators, B: Chemical, 91, 109–116, 2003.

[114] Kuhlmann, J., Bartsch, I., Willbold, E., Schuchardt, S., Holz, O., Hort, N., Hoche, D., Heineman, W. R., Witte, F., Fast escape of hydrogen from gas cavities around corroding magnesium implants. Acta Biomaterialia, 9, 8714–8721, 2013.

[115] Negri, R. M., 6 . 3 . Electronic Noses in Perfume Analysis. Analysis of Cosmetic Products, Elsevier Ltd, , 276–290, 2007.

[116] Haddad, R., Carmel, L., Harel, D., A feature extraction algorithm for multi-peak signals in electronic noses. Sensors and Actuators B: Chemical, 120 (2), 467–472, 2007.

[117] Carmel, L., Sever, N., Harel, D., On predicting responses to mixtures in quartz microbalance sensors. Sensors and Actuators B: Chemical, 106 (1), 128–135, 2005.

[118] Carmel, L., Sever, N., Lancet, D., Harel, D., An eNose algorithm for identifying chemicals and determining their concentration. Sensors and Actuators B: Chemical, 93 (1)–(3), 77–83, 2003.

[119] Samardzic, R., Sussitz, H. F., Jongkon, N., Lieberzeit, P. A., Quartz Crystal Microbalance In-Line Sensing of Escherichia Coli in a Bioreactor Using Molecularly Imprinted Polymers. Sensor Letters, 12 (6), 1152–1155, 2014.

[120] Wackerlig, J., Lieberzeit, P. A., Molecularly imprinted polymer nanoparticles in chemical sensing – Synthesis, characterisation and application. Sensors and Actuators B: Chemical, 207, 144–157, 2015.

[121] Naklua, W., Suedee, R., Lieberzeit, P. A., Dopaminergic receptor–ligand binding assays based on molecularly imprinted polymers on quartz crystal microbalance sensors. Biosensors and Bioelectronics, 81, 117–124, 2016.

[122] Adak, M. F., Yumusak, N., Developing a Software that Train Neural Networks by Artificial Bee Colony Algorithm (ABC). 3rd International Symposium on Innovative Technologies in Engineering and Science, Valencia/Spain, 145–150, 2015.

[123] Schalkoff, R. J., Artificial Neural Networks. 1. Ed., McGraw-Hill Companies, 448 , 1997.

[124] Ho, M. H., Guilbault, G. G., Rietz, B., Continuous detection of toluene in ambient air with a coated piezoelectric crystal. Analytical Chemistry, 52 (9), 1489–1492, 1980.

EKLER

EK A: ANN-GA C# Program Kodu

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MFA_ANN.Genetic
{
    class Genetic
    {
        private double crossoverRate;
        private double mutationRate;
        private Data[] VeriSeti;
        private int GirdiSayisi;
        private int CiktiSayisi;
        private HiddenLyr[] araKatmanlar;
        private ActivationFunction AktivasyonFonksiyonu;
        private double? stepEsik;
        private double aralikMinDeger;
        private double aralikMaxDeger;

        public Genetic(Data[] veriSeti, int girdiSayisi, int ciktiSayisi,
HiddenLyr[] araKatmanlar, ActivationFunction AktivasyonFonksiyonu, double?
stepEsik, double caprazlamaOrani, double mutationRate, double aralikMin,
double aralikMax)
        {
            VeriSeti = veriSeti;
            GirdiSayisi = girdiSayisi;
            CiktiSayisi = ciktiSayisi;
            this.araKatmanlar = araKatmanlar;
            this.AktivasyonFonksiyonu = AktivasyonFonksiyonu;
            this.stepEsik = stepEsik;

            crossoverRate = caprazlamaOrani;
            this.mutationRate = mutationRate;
            this.aralikMinDeger = aralikMin;
            this.aralikMaxDeger = aralikMax;
        }

        public Population PopulasyonuIyilestir(Population populasyon)
        {
            if (populasyon.size() < 4) return populasyon;

            Population yeniPopulasyon = new
Population(populasyon.size(),0,mutationRate, false, VeriSeti, GirdiSayisi,
```

```
CiktiSayisi, araKatmanlar, AktivasyonFonksiyonu, stepEsik,null,
aralikMinDeger, aralikMaxDeger);

            Population CaprazlamayaGirmeyecek;
            Population CaprazlamayaGirecek = EniyiSecimi(populasyon,out
CaprazlamayaGirmeyecek);
            Chromosome[] tumKromozomlar = new
Chromosome[CaprazlamayaGirecek.size()*2];

            int son = tumKromozomlar.Length - 1;

            for (int i = 0; i < CaprazlamayaGirecek.size(); i+=2, son-=2)
            {
                Chromosome[] yeniKromozomlar =
crossover(CaprazlamayaGirecek.getChromosome(i),
CaprazlamayaGirecek.getChromosome(i + 1));
                if (yeniKromozomlar != null)
                {
                    tumKromozomlar[i] = CaprazlamayaGirecek.getChromosome(i);
                    tumKromozomlar[i + 1] =
CaprazlamayaGirecek.getChromosome(i + 1);
                    tumKromozomlar[son - 1] = yeniKromozomlar[0];
                    tumKromozomlar[son] = yeniKromozomlar[1];
                }
                else
                {
                    tumKromozomlar[i] = CaprazlamayaGirecek.getChromosome(i);
                    tumKromozomlar[i + 1] =
CaprazlamayaGirecek.getChromosome(i + 1);
                    tumKromozomlar[son - 1] = null;
                    tumKromozomlar[son] = null;
                }
            }


            int uzunluk=0;
            for (int i = 0; i < tumKromozomlar.Length; i++)
            {
                if (tumKromozomlar[i] != null) uzunluk++;
            }
            Chromosome[] sonKromozomlar = new Chromosome[uzunluk];
            for (int i = 0, j=0; i < tumKromozomlar.Length; i++)
            {
                if (tumKromozomlar[i] != null)
                {
                    sonKromozomlar[j] = tumKromozomlar[i];
                    j++;
                }
            }


            HeapTree h = new HeapTree(sonKromozomlar.Length);
            foreach (Chromosome kro in sonKromozomlar)
            {
                h.Ekle(kro);
            }

            int indeks = 0;
```

```
            for(;CaprazlamayaGirmeyecek != null && indeks <
CaprazlamayaGirmeyecek.size(); indeks++)
            {
                yeniPopulasyon.saveChromosome(indeks,
CaprazlamayaGirmeyecek.getChromosome(indeks));
            }


            for (int i=0; i < CaprazlamayaGirecek.size(); indeks++, i++)
            {
                yeniPopulasyon.saveChromosome(indeks,h.EnKucuguCikar());
            }


            for (int i = 0; i < yeniPopulasyon.size(); i++)
            {
                yeniPopulasyon.getChromosome(i).mutate();
            }

            return yeniPopulasyon;
        }

        private Chromosome[] crossover(Chromosome kromozom1, Chromosome
kromozom2)
        {
            if (Rand.Rnd.NextDouble() > crossoverRate)
                return null;

            Chromosome[] yeniKromozomlar = new Chromosome[2];
            yeniKromozomlar[0] = new Chromosome(kromozom1.size(),mutationRate,
VeriSeti, GirdiSayisi, CiktiSayisi, araKatmanlar, AktivasyonFonksiyonu
,stepEsik, aralikMinDeger, aralikMaxDeger);
            yeniKromozomlar[1] = new Chromosome(kromozom1.size(),mutationRate,
VeriSeti, GirdiSayisi, CiktiSayisi, araKatmanlar, AktivasyonFonksiyonu,
stepEsik, aralikMinDeger, aralikMaxDeger);

            int rastgeleNokta = Rand.Rnd.Next(1, kromozom1.size());
            for (int i = 0; i < rastgeleNokta; i++)
            {
                yeniKromozomlar[0].setGene(i, kromozom1.getGene(i));
                yeniKromozomlar[1].setGene(i, kromozom2.getGene(i));
            }
            for(int i= rastgeleNokta;i< kromozom2.size(); i++)
            {
                yeniKromozomlar[0].setGene(i, kromozom2.getGene(i));
                yeniKromozomlar[1].setGene(i, kromozom1.getGene(i));
            }
            return yeniKromozomlar;
        }

        private Population EniyiSecimi(Population populasyon,out Population
caprazlamayaGirmeyecek)
        {
            populasyon.sort();

            int size = populasyon.size();
            if (size % 2 != 0)
            {
                size--;
            }
```

```
            else size -= 2;
            int indeks = 0;
            if (size != populasyon.size())
            {
                caprazlamayaGirmeyecek = new Population(populasyon.size() -
size, 0, mutationRate, false, VeriSeti, GirdiSayisi, CiktiSayisi,
araKatmanlar, AktivasyonFonksiyonu, stepEsik,null, aralikMinDeger,
aralikMaxDeger);
                for (; indeks < populasyon.size() - size; indeks++)
                    caprazlamayaGirmeyecek.saveChromosome(indeks,
populasyon.getChromosome(indeks));
            }
            else
            {
                caprazlamayaGirmeyecek = null;
            }

            Population caprazlamayaGirecek = new Population(size, 0,
mutationRate,false, VeriSeti, GirdiSayisi, CiktiSayisi, araKatmanlar,
AktivasyonFonksiyonu, stepEsik,null, aralikMinDeger, aralikMaxDeger);
            for (int i=0 ; i < size; indeks++, i++)
            {
                caprazlamayaGirecek.saveChromosome(i,
populasyon.getChromosome(indeks));
            }

            return caprazlamayaGirecek;
        }
    }
}


using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MFA_ANN.Genetic
{
    class Population
    {
        private Chromosome[] Kromozomlar;
        double mutationRate;
        private Data[] VeriSeti;
        private int GirdiSayisi;
        private int CiktiSayisi;
        private HiddenLyr[] araKatmanlar;
        private ActivationFunction AktivasyonFonksiyonu;
        private double? stepEsik;

        public Population(int populasyonSayisi, int genSayisi, double
mutationRate, bool ilkOlusum, Data[] VeriSeti, int GirdiSayisi, int
CiktiSayisi, HiddenLyr[] araKatmanlar, ActivationFunction
AktivasyonFonksiyonu, double? stepEsik,ArrayList kromozomDegerleri, double
aralikMin, double aralikMax)
        {
            Kromozomlar = new Chromosome[populasyonSayisi];
            this.mutationRate = mutationRate;
```

```
            this.VeriSeti = VeriSeti;
            this.GirdiSayisi = GirdiSayisi;
            this.CiktiSayisi = CiktiSayisi;
            this.araKatmanlar = araKatmanlar;
            this.AktivasyonFonksiyonu = AktivasyonFonksiyonu;
            this.stepEsik = stepEsik;

            for (int i = 0; i < size(); i++)
            {
                Chromosome kromozom = new Chromosome(genSayisi,
mutationRate,VeriSeti, GirdiSayisi, CiktiSayisi, araKatmanlar,
AktivasyonFonksiyonu, stepEsik,aralikMin, aralikMax);
                if (ilkOlusum)
                {
                    if(kromozomDegerleri != null)
                    {
                        double?[] kroDegerler = kromozomDegerleri[i] as
double?[];
                        kromozom.generateChromosome(kroDegerler);
                    }
                    else kromozom.generateChromosome(null);
                }
                saveChromosome(i, kromozom);
            }
        }

        public int size()
        {
            return Kromozomlar.Length;
        }

        public Chromosome getChromosome(int index)
        {
            return Kromozomlar[index];
        }

        public void saveChromosome(int index, Chromosome kromozom)
        {
            Kromozomlar[index] = kromozom;
        }

        public Chromosome getFittest()
        {
            Chromosome fittest = Kromozomlar[0];

            for (int i = 1; i < size(); i++)
            {
                if (fittest.getFitness() > getChromosome(i).getFitness())
                {
                    fittest = getChromosome(i);
                }
            }
            return fittest;
        }

        public void sort()
        {
            HeapTree heap = new HeapTree(size());
            Chromosome[] siraliKromozomlar = new Chromosome[size()];
            int index = 0;
```

```
            foreach(Chromosome kromozom in Kromozomlar)
            {
                heap.Ekle(kromozom);
            }

            while (!heap.Bosmu())
            {
                siraliKromozomlar[index++] = heap.EnKucuguCikar();
            }
            Kromozomlar = siraliKromozomlar;
        }

        private int getFittestIndex()
        {
            Chromosome fittest = Kromozomlar[0];
            int index = 0;

            for (int i = 1; i < size(); i++)
            {
                if (fittest.getFitness() > getChromosome(i).getFitness())
                {
                    fittest = getChromosome(i);
                    index = i;
                }
            }
            return index;
        }
    }
}
```

EK B: ANN-ABC C# Program Kodu

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MFA_ANN
{
    class BeeColony
    {
        private int NP;
        private int FoodNumber;
        private int limit;
        private int maxCycle;

        private int D;
        private double []lb;
        private double []ub;

        private int runtime;

        private double[,] Foods;
        private double[] f;
        private double[] fitness;
        private double[] trial;
        private double[] prob;
        private double[] solution;

        private double ObjValSol;
        private double FitnessSol;
        private int neighbour;
        private int param2change;

        public double GlobalMin;
        public double[] GlobalParams;
        public double[] GlobalMins;

        private double r;

        private ActivationFunction AktivasyonFonksiyonu;
        private double? stepEsik;

        public Data[] VeriSeti;
        private int GirdiSayisi;
        private int CiktiSayisi;
        double[,] trin;
        double[] trout;
        private HiddenLyr[] araKatmanlar;

        private HataTuru hataTuru;
        private double[] Tort;
        private int r2icinBagimsizDegiskenSayisi;
        private bool gercekCiktilaraDonustur;
        RealOutput[] ciktiGercekMinMax;
```

```
        double normMin, normMax;

        public BeeColony(Data[] veriSeti, int girdiSayisi, int ciktiSayisi,
int NP, int limit, int maxCycle, double lb, double ub, int runtime,
ActivationFunction AktivasyonFonksiyonu, double? stepEsik, HiddenLyr[]
araKatmanlar, HataTuru htTuru, bool gercekCiktilaraDonustur, RealOutput[]
ciktiGercekMinMax, double normMin, double normMax)
        {
            this.NP = NP;
            this.FoodNumber = NP / 2;
            this.limit = limit;
            this.maxCycle = maxCycle;

            int geciciParametreSayisi = girdiSayisi *
araKatmanlar[0].thresholds.Length;
            geciciParametreSayisi += araKatmanlar[araKatmanlar.Length-
1].thresholds.Length * ciktiSayisi;
            for (int i = 0; i < araKatmanlar.Length; i++)
            {
                geciciParametreSayisi += araKatmanlar[i].thresholds.Length;
                if (i + 1 < araKatmanlar.Length)
                {
                    geciciParametreSayisi +=
araKatmanlar[i].thresholds.Length *  araKatmanlar[i+1].thresholds.Length;
                }
            }

            geciciParametreSayisi += ciktiSayisi;
            this.D = geciciParametreSayisi;

            this.lb = new double[D];
            this.ub = new double[D];
            for (int i = 0; i < D; i++) this.lb[i] = lb;
            for (int i = 0; i < D; i++) this.ub[i] = ub;

            this.runtime = runtime;
            Foods = new double[FoodNumber, D];
            f = new double[FoodNumber];
            fitness = new double[FoodNumber];
            trial = new double[FoodNumber];
            prob = new double[FoodNumber];
            solution = new double[D];
            GlobalParams = new double[D];
            GlobalMins = new double[runtime];
            this.AktivasyonFonksiyonu = AktivasyonFonksiyonu;
            this.stepEsik = stepEsik;

            VeriSeti = veriSeti;
            GirdiSayisi = girdiSayisi;
            CiktiSayisi = ciktiSayisi;

            Tort = new double[CiktiSayisi];
            trin = new double[VeriSeti.Length, GirdiSayisi];

            for (int i = 0; i < VeriSeti.Length; i++)
            {
                double[] tmp = VeriSeti[i].inputs;
                for (int j = 0; j < GirdiSayisi; j++)
                {
                    trin[i, j] = tmp[j];
```

```
                }
            }


            trout = new double[VeriSeti.Length];
            for (int i = 0; i < VeriSeti.Length; i++)
            {
                double[] tmp = VeriSeti[i].outputs;
                for (int j = 0; j < CiktiSayisi; j++)
                {
                    trout[i] = tmp[j];
                }
            }

            hataTuru = htTuru;

            double r2TargetToplam = 0;
            for(int i =0;i< CiktiSayisi; i++)
            {
                for(int j=0;j< VeriSeti.Length; j++)
                {
                    double[] tmp = VeriSeti[j].outputs;
                    r2TargetToplam += tmp[i];
                }
                Tort[i] = r2TargetToplam / VeriSeti.Length;
            }

            this.araKatmanlar = new HiddenLyr[araKatmanlar.Length];
            for(int i=0;i<araKatmanlar.Length;i++) this.araKatmanlar[i] =
araKatmanlar[i];

            r2icinBagimsizDegiskenSayisi = 7;
            this.gercekCiktilaraDonustur = gercekCiktilaraDonustur;
            this.ciktiGercekMinMax = ciktiGercekMinMax;
            this.normMin = normMin;
            this.normMax = normMax - normMin;
        }

        private double CalculateFitness(double fun)
        {
            double result = 0;
            if (fun >= 0)
            {
                result = 1 / (fun + 1);
            }
            else
            {

                result = 1 + Math.Abs(fun);
            }
            return result;
        }
        public void MemorizeBestSource()
        {
            for (int i = 0; i < FoodNumber; i++)
            {
                if (f[i] < GlobalMin)
                {
                    GlobalMin = f[i];
                    for (int j = 0; j < D; j++)
```

```
                        GlobalParams[j] = Foods[i, j];
                }
            }
        }
        private void init(int index)
        {
            for (int j = 0; j < D; j++)
            {
                r = ((double)Rand.Rnd.NextDouble() * 32767 / ((double)32767 +
(double)(1)));
                Foods[index, j] = r * (ub[j] - lb[j]) + lb[j];
                solution[j] = Foods[index, j];
            }
            f[index] = calculateFunction(solution);
            fitness[index] = CalculateFitness(f[index]);
            trial[index] = 0;
        }
        public void initial()
        {
            for (int i = 0; i < FoodNumber; i++)
            {
                init(i);
            }
            GlobalMin = f[0];
            for (int i = 0; i < D; i++)
            {
                GlobalParams[i] = Foods[0, i];
            }
        }
        public void SendEmployedBees()
        {

            for (int i = 0; i < FoodNumber; i++)
            {

                r = ((double)Rand.Rnd.NextDouble() * 32767 / ((double)(32767)
+ (double)(1)));
                param2change = (int)(r * D);


                r = ((double)Rand.Rnd.NextDouble() * 32767 / ((double)(32767)
+ (double)(1)));
                neighbour = (int)(r * FoodNumber);


                while(neighbour==i)
                {
                    r = ((double)Rand.Rnd.NextDouble() * 32767 /
((double)(32767) + (double)(1)));
                    neighbour = (int)(r * FoodNumber);
                }

                for (int j = 0; j < D; j++)
                {
                    solution[j] = Foods[i, j];
                }


                r = ((double)Rand.Rnd.NextDouble() * 32767 / ((double)(32767)
+ (double)(1)));
```

```
                solution[param2change] = Foods[i, param2change] + (Foods[i,
param2change] - Foods[neighbour, param2change]) * (r - 0.5) * 2;

                for (int ind = 0; ind < D; ind++) {
                    if (solution[ind] < lb[ind]) solution[ind] = lb[ind];
                    if (solution[ind] > ub[ind]) solution[ind] = ub[ind];
                }
                ObjValSol = calculateFunction(solution);
                FitnessSol = CalculateFitness(ObjValSol);

                if (FitnessSol > fitness[i])
                {
                    trial[i] = 0;
                    for (int j = 0; j < D; j++)
                    {
                        Foods[i, j] = solution[j];
                    }
                    f[i] = ObjValSol;
                    fitness[i] = FitnessSol;
                }
                else
                {
                    trial[i] = trial[i] + 1;
                }
            }
        }
        public void CalculateProbabilities()
        {
            double maxfit;
            maxfit = fitness[0];
            for (int i = 1; i < FoodNumber; i++)
            {
                if (fitness[i] > maxfit)
                    maxfit = fitness[i];
            }

            for (int i = 0; i < FoodNumber; i++)
            {
                prob[i] = (0.9 * (fitness[i] / maxfit)) + 0.1;
            }
        }
        public void SendOnlookerBees()
        {
            int i, t;
            i = 0;
            t = 0;

            while (t < FoodNumber)
            {

                r = ((double)Rand.Rnd.NextDouble() * 32767 / ((double)(32767)
+ (double)(1)));
                if (r < prob[i])
                {
                    t++;

                    r = ((double)Rand.Rnd.NextDouble() * 32767 /
((double)(32767) + (double)(1)));
                    param2change = (int)(r * D);
```

```
                    r = ((double)Rand.Rnd.NextDouble() * 32767 /
((double)(32767) + (double)(1)));
                    neighbour = (int)(r * FoodNumber);

                    while (neighbour == i)
                    {
                        r = ((double)Rand.Rnd.NextDouble() * 32767 /
((double)(32767) + (double)(1)));
                        neighbour = (int)(r * FoodNumber);
                    }
                    for (int j = 0; j < D; j++)
                    {
                        solution[j] = Foods[i, j];
                    }

                    r = ((double)Rand.Rnd.NextDouble() * 32767 /
((double)(32767) + (double)(1)));
                    solution[param2change] = Foods[i, param2change] +
(Foods[i, param2change] - Foods[neighbour, param2change]) * (r - 0.5) * 2;

                    for (int ind = 0; ind < D; ind++)
                    {
                        if (solution[ind] < lb[ind]) solution[ind] = lb[ind];
                        if (solution[ind] > ub[ind]) solution[ind] = ub[ind];
                    }

                    ObjValSol = calculateFunction(solution);
                    FitnessSol = CalculateFitness(ObjValSol);

                    if (FitnessSol > fitness[i])
                    {
                        trial[i] = 0;
                        for (int j = 0; j < D; j++)
                        {
                            Foods[i, j] = solution[j];
                        }
                        f[i] = ObjValSol;
                        fitness[i] = FitnessSol;
                    }
                    else
                    {
                        trial[i] = trial[i] + 1;
                    }
                }
                i++;
                if (i == FoodNumber)
                    i = 0;
            }
        }
        public void SendScoutBees()
        {
            int maxtrialindex;
            maxtrialindex = 0;
            for (int i = 1; i < FoodNumber; i++)
            {
                if (trial[i] > trial[maxtrialindex])
                    maxtrialindex = i;
            }
            if (trial[maxtrialindex] >= limit)
            {
```

```
                init(maxtrialindex);
            }
        }
        private double calculateFunction(double[] sol)
        {
            return YSATrain(sol);
        }
        private double YSATrain(double[] sol)
        {
            ArrayList Agirliklar = new ArrayList();
            ArrayList EsikDegerler = new ArrayList();
            ArrayList Ciktilar = new ArrayList();

            int solIndeks=0;

            double [,] tmpAgirliklar = new double[GirdiSayisi,
araKatmanlar[0].thresholds.Length];
            for (int i = 0; i < GirdiSayisi; i++)
            {
                for (int j = 0; j < araKatmanlar[0].thresholds.Length; j++,
solIndeks++) tmpAgirliklar[i, j] = sol[solIndeks];
            }
            Agirliklar.Add(tmpAgirliklar);

            for (int i = 0; i < araKatmanlar.Length; i++)
            {
                double[] tmpEsikDegerler = new
double[araKatmanlar[i].thresholds.Length];
                double[] ciktilar = new
double[araKatmanlar[i].thresholds.Length];
                for(int j=0;j<tmpEsikDegerler.Length;j++,solIndeks++)
tmpEsikDegerler[j] = sol[solIndeks];
                EsikDegerler.Add(tmpEsikDegerler);
                Ciktilar.Add(ciktilar);

                if(i+1 < araKatmanlar.Length){
                    tmpAgirliklar = new
double[araKatmanlar[i].thresholds.Length,
araKatmanlar[i+1].thresholds.Length];
                    for (int j = 0; j < araKatmanlar[i].thresholds.Length;
j++)
                    {
                        for (int k = 0; k < araKatmanlar[i +
1].thresholds.Length; k++, solIndeks++) tmpAgirliklar[j, k] = sol[solIndeks];
                    }
                    Agirliklar.Add(tmpAgirliklar);
                }
            }

            tmpAgirliklar = new double[araKatmanlar[araKatmanlar.Length -
1].thresholds.Length, CiktiSayisi];
            for (int i = 0; i < araKatmanlar[araKatmanlar.Length -
1].thresholds.Length; i++)
            {
                for (int j = 0; j < CiktiSayisi; j++, solIndeks++)
                {
                    tmpAgirliklar[i, j] = sol[solIndeks];
                }
            }
            Agirliklar.Add(tmpAgirliklar);
```

```
            double[] tempEsikDegerler = new double[CiktiSayisi];
            double[] tmpciktilar = new double[CiktiSayisi];
            for (int j = 0; j < CiktiSayisi; j++, solIndeks++)
tempEsikDegerler[j] = sol[solIndeks];
            EsikDegerler.Add(tempEsikDegerler);
            Ciktilar.Add(tmpciktilar);

           double htt = 0;
            double r2Payda = 0;

           for (int indeks = 0; indeks < VeriSeti.Length; indeks++)
            {
                for (int i = 0; i < Ciktilar.Count; i++)
                {
                    double[] girdiler;
                    if (i == 0) girdiler = VeriSeti[indeks].inputs;
                    else girdiler = Ciktilar[i - 1] as double[];

                    double[] ciktilar = Ciktilar[i] as double[];
                    double[] netDegerler = new double[ciktilar.Length];
                    double[,] agirliklar = Agirliklar[i] as double[,];
                    double[] esikDegerler = EsikDegerler[i] as double[];

                    for (int j = 0; j < ciktilar.Length; j++)
                    {
                        double dugumEsikDeger = esikDegerler[j];

                        netDegerler[j] = Net(girdiler, agirliklar, j,
dugumEsikDeger);
                    }

                    for (int k = 0; k < ciktilar.Length; k++)
                    {
                        double cikti;
                        if (AktivasyonFonksiyonu ==
ActivationFunction.SIGMOID) cikti = Sigmoid(netDegerler[k]);
                        else if (AktivasyonFonksiyonu ==
ActivationFunction.TANJANT) cikti = TanjantHiperbolik(netDegerler[k]);
                        else cikti = StepActivation(netDegerler[k]);

                        ciktilar[k] = cikti;
                    }
                }
                double[] aginCiktilari = Ciktilar[Ciktilar.Count - 1] as
double[];
                double toplamHata = 0;
                double iterasyonPaydaToplam=0;
                for (int cki = 0; cki < CiktiSayisi; cki++)
                {
                    if (hataTuru == HataTuru.MSE) toplamHata +=
MSEHesapla(aginCiktilari[cki], VeriSeti[indeks].outputs[cki], cki);
                    else if (hataTuru == HataTuru.MAPE) toplamHata +=
MAPEHesapla(aginCiktilari[cki], VeriSeti[indeks].outputs[cki], cki);
                    else
                    {
                        double payda;
                        toplamHata += R2Hesapla(aginCiktilari[cki],
VeriSeti[indeks].outputs[cki],out payda, cki);
                        iterasyonPaydaToplam += payda;
```

```
                    }
                }
                if (hataTuru == HataTuru.MSE) htt += (toplamHata / (2 *
CiktiSayisi));
                else if (hataTuru == HataTuru.MAPE) htt += (toplamHata /
CiktiSayisi);
                else
                {
                    htt += (toplamHata / CiktiSayisi);
                    r2Payda += (iterasyonPaydaToplam / CiktiSayisi);
                }
            }
           if(hataTuru == HataTuru.MSE) return htt / VeriSeti.Length;
           else if (hataTuru == HataTuru.MAPE) return (htt / VeriSeti.Length);
            else
            {
                double r2 = 1 - ((1 - (htt / r2Payda)) * ((VeriSeti.Length -
1) / (VeriSeti.Length - r2icinBagimsizDegiskenSayisi - 1)));
                if (r2 < 0) return 0;
                else return r2;
            }
        }

        private double GercekCikti(double normalizeCikti, int ciktiIndeks)
        {
           return (((normalizeCikti - normMin) / normMax) *
(ciktiGercekMinMax[ciktiIndeks].max - ciktiGercekMinMax[ciktiIndeks].min)) +
ciktiGercekMinMax[ciktiIndeks].min;
        }

        private double MSEHesapla(double cikti,double hedef, int ciktiIndeks)
        {
            if(gercekCiktilaraDonustur)
            {
                cikti = GercekCikti(cikti, ciktiIndeks);
                hedef = Math.Round(GercekCikti(hedef, ciktiIndeks),0);
            }
            return Math.Pow((hedef-cikti),2);
        }
        private double MAPEHesapla(double cikti, double hedef, int
ciktiIndeks)
        {
            if (gercekCiktilaraDonustur)
            {
                cikti = GercekCikti(cikti, ciktiIndeks);
                hedef = Math.Round(GercekCikti(hedef, ciktiIndeks), 0);
            }

            return Math.Abs(hedef-cikti)/hedef;
        }
        private double R2Hesapla(double cikti, double hedef, out double
paydaDeger,int ciktiIndeks)
        {
            double ort = Tort[ciktiIndeks];
            if (gercekCiktilaraDonustur)
            {
                cikti = GercekCikti(cikti, ciktiIndeks);
                hedef = Math.Round(GercekCikti(hedef, ciktiIndeks), 0);
                ort = GercekCikti(Tort[ciktiIndeks], ciktiIndeks);
            }
```

```
            paydaDeger = Math.Pow((hedef - ort), 2);
            return Math.Pow((hedef - cikti), 2);
        }

        public double[,] YSATest(Data[] TestVerisi)
        {
            ArrayList Agirliklar = new ArrayList();
            ArrayList EsikDegerler = new ArrayList();
            ArrayList Ciktilar = new ArrayList();
            double[,] sonuclar = new double[TestVerisi.Length, CiktiSayisi];

            int solIndeks = 0;

            double[,] tmpAgirliklar = new double[GirdiSayisi,
araKatmanlar[0].thresholds.Length];
            for (int i = 0; i < GirdiSayisi; i++)
            {
                for (int j = 0; j < araKatmanlar[0].thresholds.Length; j++,
solIndeks++) tmpAgirliklar[i, j] = GlobalParams[solIndeks];
            }
            Agirliklar.Add(tmpAgirliklar);

            for (int i = 0; i < araKatmanlar.Length; i++)
            {
                double[] tmpEsikDegerler = new
double[araKatmanlar[i].thresholds.Length];
                double[] ciktilar = new
double[araKatmanlar[i].thresholds.Length];
                for (int j = 0; j < tmpEsikDegerler.Length; j++, solIndeks++)
tmpEsikDegerler[j] = GlobalParams[solIndeks];
                EsikDegerler.Add(tmpEsikDegerler);
                Ciktilar.Add(ciktilar);

                if (i + 1 < araKatmanlar.Length)
                {
                    tmpAgirliklar = new
double[araKatmanlar[i].thresholds.Length, araKatmanlar[i +
1].thresholds.Length];
                    for (int j = 0; j < araKatmanlar[i].thresholds.Length;
j++)
                    {
                        for (int k = 0; k < araKatmanlar[i +
1].thresholds.Length; k++, solIndeks++) tmpAgirliklar[j, k] =
GlobalParams[solIndeks];
                    }
                    Agirliklar.Add(tmpAgirliklar);
                }
            }

            tmpAgirliklar = new double[araKatmanlar[araKatmanlar.Length -
1].thresholds.Length, CiktiSayisi];
            for (int i = 0; i < araKatmanlar[araKatmanlar.Length -
1].thresholds.Length; i++)
            {
                for (int j = 0; j < CiktiSayisi; j++, solIndeks++)
                {
                    tmpAgirliklar[i, j] = GlobalParams[solIndeks];
                }
            }
```

```
            Agirliklar.Add(tmpAgirliklar);

            double[] tempEsikDegerler = new double[CiktiSayisi];
            double[] tmpciktilar = new double[CiktiSayisi];
            for (int j = 0; j < CiktiSayisi; j++, solIndeks++)
tempEsikDegerler[j] = GlobalParams[solIndeks];
            EsikDegerler.Add(tempEsikDegerler);
            Ciktilar.Add(tmpciktilar);

            double[] dogruSonuclar = new double[CiktiSayisi];
            for (int indeks = 0; indeks < TestVerisi.Length; indeks++)
            {
                for (int i = 0; i < Ciktilar.Count; i++)
                {
                    double[] girdiler;
                    if (i == 0) girdiler = TestVerisi[indeks].inputs;
                    else girdiler = Ciktilar[i - 1] as double[];

                    double[] ciktilar = Ciktilar[i] as double[];
                    double[] netDegerler = new double[ciktilar.Length];
                    double[,] agirliklar = Agirliklar[i] as double[,];
                    double[] esikDegerler = EsikDegerler[i] as double[];

                    for (int j = 0; j < ciktilar.Length; j++)
                    {
                        double dugumEsikDeger = esikDegerler[j];

                        netDegerler[j] = Net(girdiler, agirliklar, j,
dugumEsikDeger);
                    }

                    for (int k = 0; k < ciktilar.Length; k++)
                    {
                        double cikti;
                        if (AktivasyonFonksiyonu ==
ActivationFunction.SIGMOID) cikti = Sigmoid(netDegerler[k]);
                        else if (AktivasyonFonksiyonu ==
ActivationFunction.TANJANT) cikti = TanjantHiperbolik(netDegerler[k]);
                        else cikti = StepActivation(netDegerler[k]);

                        ciktilar[k] = cikti;
                    }
                }
                double[] aginCiktilari = Ciktilar[Ciktilar.Count - 1] as
double[];
                for (int cki = 0; cki < CiktiSayisi; cki++)
                {
                    if(gercekCiktilaraDonustur) sonuclar[indeks, cki] =
GercekCikti(aginCiktilari[cki],cki);
                    else sonuclar[indeks, cki] = aginCiktilari[cki];
                }
            }

            return sonuclar;
        }
        private int SonucIndeks(double[] ciktilar)
        {
            double? fark1 = null;
            double? fark2 = null;
```

```
            if (ciktilar.Length == 1) return 0;
            if (ciktilar.Length == 2)
            {
                if (ciktilar[0] >= ciktilar[1]) return 0;
                else return 1;
            }
            if (ciktilar[0] != ciktilar[1])
            {
                fark1 = ciktilar[0];
                fark2 = ciktilar[1];
            }
            else fark1 = ciktilar[0];
            for (int i = 2; i < ciktilar.Length; i++)
            {
                if (fark2 == null && ciktilar[i] != fark1) return i;
                else if (fark2 != null)
                {
                    if (ciktilar[i] != fark1) return 0;
                    else return 1;
                }
            }
            return -1;
        }
        private double Net(double[] girdiler, double[,] agirliklar, int
netDugumIndeks, double dugumEsikDeger)
        {
            double net = 0;
            for (int i = 0; i < girdiler.Length; i++)
            {
                net += girdiler[i] * agirliklar[i, netDugumIndeks];
            }
            net += 1 * dugumEsikDeger;
            return net;
        }
        private double[,] CopyToMultArray(double[] array, int startIndex, int
length, int x, int y)
        {
            double[,] copiedArray = new double[x,y];
            for (int i = 0; i < x; i++)
            {
                for (int j = 0; j < y; j++,startIndex++)
                {
                    copiedArray[i,j] = array[startIndex];
                }
            }
            return copiedArray;
        }

        private double Sigmoid(double net)
        {
            double cikti = (1 / (1 + Math.Pow(Math.E, -net)));
            return cikti;
        }

        private double TanjantHiperbolik(double net)
        {
            double cikti = (Math.Pow(Math.E, net) + Math.Pow(Math.E, -net)) /
(Math.Pow(Math.E, net) - Math.Pow(Math.E, -net));
            return cikti;
        }
```

```
        private double StepActivation(double net)
        {
            double cikti;
            if (net > stepEsik) cikti = 1;
            else cikti = 0;

            return cikti;
        }

        public int Runtime
        {
            get
            {
                return runtime;
            }
        }
        public int MaxCycle
        {
            get
            {
                return maxCycle;
            }
        }
        public int Dprm
        {
            get
            {
                return D;
            }
        }
    }
}
```

ÖZGEÇMİŞ

1984 yılında Gölbaşı'nda doğdu, 1995 yılında Mardin Cumhuriyet İlkokulu'ndan, 1999 yılında Van Özel Serhat Koleji'nden ve 2002 yılında İstanbul Davutpaşa Süper Lisesi'nden mezun oldu. 2004 yılında İzmir Dokuz Eylül Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü'ne başladı. Buradan 2009 yılında Bilgisayar Mühendisi olarak mezun oldu. 2010 yılında Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü'nde yüksek lisans eğitimine başladı. 2012 yılında buradan yüksek mühendis olarak mezun oldu. Aynı zamanda 2010 yılında Sakarya Üniversitesi Bilgisayar Araştırma ve Uygulama Merkezi'nde (BAUM) yazılım mühendisi olarak çalışmaya başladı. 2012 yılında buradan ayrıldı. 2012 yılının şubat ayında Sakarya Üniversitesi, Bilgisayar ve Bilişim Bilimleri Fakültesi, Bilgisayar Mühendisliği Bölümü'nde Araştırma Görevlisi olarak işe başladı. Aynı yılda yine Sakarya Üniversitesi, Bilgisayar ve Bilişim Mühendisliği Anabilim dalında Doktora eğitimine başladı, çalıştığı tez konusu ile ilgili bilimsel makaleler ve sempozyum bildirileri, yardımcı olduğu dersler ile ilgili laboratuvar notları ve basılmış ders kitabı bulunmaktadır. Evli ve iki çocuk babasıdır.