EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(DOKTORA TEZİ)

YAPAY SİNİR AĞLARI ÜZERİNDE KURAL ÇIKARMA İŞLEMİ İÇİN BAĞLANTICI MELEZ BİR ZEKİ SİSTEM GELİŞTİRİLMESİ

Ahmet Cumhur KINACI

Tez Danışmanı: Doç. Dr. Aybars UĞUR

Bilgisayar Mühendisliği Anabilim Dalı

Bilim Dalı Kodu: 619.01.00
Sunuş Tarihi: 17.05.2013

Bornova-İZMİR
2013

Ahmet Cumhur KINACI tarafından **DOKTORA TEZİ** olarak sunulan "**YAPAY SİNİR AĞLARI ÜZERİNDE KURAL ÇIKARMA İŞLEMİ İÇİN BAĞLANTICI MELEZ BİR ZEKİ SİSTEM GELİŞTİRİLMESİ**" başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve **17.05.2013** tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:	**İmza**
Jüri Başkanı: Doç. Dr. Aybars UĞUR	.....
Raportör Üye: Prof. Dr. Serdar KORUKOĞLU	.....
Üye: Yrd. Doç. Dr. Cengiz GÜNGÖR	.....
Üye: Yrd. Doç. Dr. Hasan BULUT	.....
Üye: Yrd. Doç. Dr. Korhan KARABULUT	.....

ÖZET

YAPAY SİNİR AĞLARI ÜZERİNDE KURAL ÇIKARMA İŞLEMİ İÇİN BAĞLANTICI MELEZ BİR ZEKİ SİSTEM GELİŞTİRİLMESİ

KINACI, Ahmet Cumhur

Doktora Tezi, Bilgisayar Mühendisliği Anabilim Dalı
Tez Danışmanı: Doç. Dr. Aybars UĞUR
17.05.2013, 84 sayfa

Yapay sinir ağları (YSA) bir makine öğrenmesi yöntemi olarak bir çok problemin çözümünde kullanılmaktadır. Öğrenebilme yeteneği sayesinde, eldeki verileri temsil edecek modele geçiş kolaylıkla sağlanabilmektedir. Bu özelliğine karşın en büyük dezavantajı kapalı kutu özelliği taşımasıdır. Bu nedenle doğru sonuçlar vermesine rağmen neden sonuç ilişkilerini doğrudan gösterememesi, yöntemin güvenilirliğini azaltmaktadır. Bu çalışmada eğitilmiş bir yapay sinir ağının kapalı kutu özelliğini ortadan kaldıracak bir eğitimsel, bağlantıcı ve melez bir yaklaşım önerilmektedir. Önerilen yaklaşım YSA'yı kapalı kutu olarak kabul etmekte ve sadece girdi çıktı değerlerini ele almaktadır. Bağlantıcı bir algoritma olan Growing Neural Gas (GNG) kullanılarak, eğitilmiş ağ tarafından temsil edilen kural karar bölgelerinin topolojileri bulunmaktadır. GNG tarafından bulunan, topolojileri ifade eden çizge kullanılarak kural kümesi oluşturulmaktadır. GNG algoritmasının parametrelerinin problem veri kümesine göre eniyileştirilmesi için genetik algoritmalardan faydalanılmakta ve böylece melez bir zeki sistem oluşturulmaktadır.

Geliştirilen yöntemin pratikteki başarısının ölçülmesi için çok katmanlı algılayıcı yapay sinir ağı modeli ile Iris, Ecoli ve Wisconsin Diagnostic Breast Cancer veri setleri üzerinde k-kat çapraz geçerleme yaklaşımı kullanılarak performans ölçümleri yapılmıştır. Ölçüm değerleri olarak kural sayısı, kural kümesi doğruluğu ve kural kümesinin aslına uygunluğu kriterleri kullanılmış ve deneysel çalışmalarda başarılı sonuçlar elde edilmiştir.

Anahtar Sözcükler: Yapay Sinir Ağları, Kural Çıkarma, Melez Zeki Sistemler, Topoloji Öğrenme, GNG, Genetik Algoritmalar

ABSTRACT

DEVELOPING A CONNECTIONIST HYBRID INTELLIGENT SYSTEM FOR EXTRACTING RULES FROM ARTIFICIAL NEURAL NETWORKS

KINACI, Ahmet Cumhur

Ph.D. in Computer Engineering
Supervisor: Assoc. Prof. Dr. Aybars UGUR
17.05.2013, 84 pages

As a machine learning algorithm artificial neural networks (ANN) are used for solving very wide range of problems. Because of the learning ability of ANN, the model which represents data can be build easily. However, the main disadvantage of ANN is that it has black box property. Even it gives correct results, the reliability of this method has some problems, because of not being able to explain the reasoning process. In this work a pedagogical, connectionist and hybrid approach for removing the black box property is presented. This approach takes ANN as a black box and evaluates only input and output relations of ANN. A connectionist method, Growing Neural Gas (GNG) is used to find the topologies of the decision regions which are represented by the trained network. The graph which is constructed by GNG and represents these topologies, is used to extract rules. Genetic algorithms is used in order to optimize GNG parameters for the target problem data set and as a result the method becomes a hybrid intelligent system.

In order to measure the performance of the presented method; it is applied to Iris, Ecoli and Wisconsin Diagnostic Breast Cancer data sets with MLP model and by using k-fold cross-validation technique. Rule set size, accuracy and fidelity are measured in the experiments and successful results are obtained.

Keywords: Artificial Neural Networks, Rule Extraction, Hybrid Intelligent System, Topology Learning, GNG, Genetic Algorithms

TEŞEKKÜR

Doktora süreci boyunca desteklerini benden esirgemeyen anneme ve babama çok teşekkür ederim.

İÇİNDEKİLER

İÇİNDEKİLER (Devam)

İÇİNDEKİLER (Devam)

İÇİNDEKİLER (Devam)

İÇİNDEKİLER (Devam)

ŞEKİLLER DİZİNİ

ŞEKİLLER DİZİNİ (Devam)

ÇİZELGELER DİZİNİ

ALGORİTMALAR DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltma	Açılım
ADT	Andrews-Diederich-Tickle
ANN	Artificial Neural Networks
CEBP	Constrained Error Backpropagation
FCA	Formal Concept Analysis
GA	Genetik Algoritmalar
GNG	Growing Neural Gas
KBCNN	Knowledge-based Conceptual Neural Network
MLP	Multilayer Perceptron
MVL	Multiple Valued Logic
RBF	Radial Basis Function
RNN	Recurrent Neural Network
SOM	Self Organizing Map
SVM	Support Vector Machine
VIA	Validity Interval Analysis
YSA	Yapay Sinir Ağları

1 GİRİŞ

Zeka'nın çeşitli tanımları bulunmakla beraber bilinen en önemli özelliği örneklerden öğrenebilme ve genelleştirme yeteneğidir. Bilimsel anlamda bu konuda bir çok çalışma yapılmıştır. Bu çalışmalar sonucu geliştirilen yöntemler ve algoritmalar makine öğrenmesi olarak isimlendirilmiş ve araç sürüş sistemi, DNA analizi gibi birçok gerçek hayat problemine uygulanmıştır. Öğrenme yöntemlerini kullanmanın en temel nedeni, bir problemi çözecek programın doğrudan nasıl yazılacağını bilmemek veya doğrudan gerçekleştiriminin çok zor olmasıdır.

Diğer bir neden ise eldeki veriyi modelleyecek bir yapıya sahip olma gereksinimidir. Bu sayede verilerin daha iyi yorumlanmasını, açıklanmasını ve yeni bilgiler bulunmasını sağlayacak bir model oluşturulmuş olur.

Bir çok sistem için olduğu gibi öğrenmeyle oluşturulan sistemleri değerlendirmek için de doğruluk ve anlaşılabilirlik kriterleri kullanılır. Doğruluk, sistemin öğrenmesi sırasında görmediği örnekleri başarılı olarak yorumlayabilmesini ifade eder. Açıklanabilirlik ise oluşturulan modelin ne derecede anlaşılabildiği ve incelenebildiğini ifade eder. Yapay sinir ağları doğruluğu yüksek modeller oluşturmasının yanında açıklanabilirliği çok düşüktür. Birçok problemde doğruluğun yüksek olmasının yanında modelin açıklanabilir olması da gerekmektedir. Bu tezle amaçlanan, yapay sinir ağları için açıklanabilirliği sağlayacak bir yöntem geliştirmektir.

1.1 Yapay Sinir Ağları (YSA)

Yapay sinir ağları (YSA), örneklerden öğrenme, bilgiyi genelleştirebilme, paralel çalışma gibi özellikleri sayesinde veri işleme aracı olarak günümüzde bir çok alanda kullanılmaktadır. Yapay sinir ağları bir bilgi işleme teknolojisidir. Yapay sinir ağları günümüzde kullanılan bilgi işleme yaklaşımlarından tamamen farklı bir yaklaşımdır. Bu hesaplama tekniğinin, örüntü tanıma, robot kontrolü ve bilgi kazancı gibi çeşitli adresleme problemlerinde daha etkin olduğu kanıtlanmıştır.(Kohonen, 1988)

Yapay sinir ağları, insan beyninin çalışma mantığının bilgisayar ortamına aktarılmasını ve böylece geleneksel programlama yöntemleriyle gerçekleştirimi oldukça zor veya mümkün olmayan biyolojik sinir ağlarının yeteneklerinin kullanılmasını sağlar (Fausett, 1994).

İnsan beyninin çalışma prensipleri ışığında ortaya çıkan yapay sinir ağları, gücünü insan beyninin iki temel özelliği olan; paralel dağıtık yapısından ve öğrenme sonucunda genelleyebilme yeteneğinden alır.

Teknik olarak, bir yapay sinir ağının en temel görevi, kendisine gösterilen bir girdi kümesine karşılık gelebilecek bir çıktı kümesi belirlemektir. Bunu yapabilmesi için ağ, ilgili olayın örnekleri ile eğitilerek (*öğrenme*) genelleme yapabilecek yeteneğe kavuşturulur. Genelleme sayesinde benzer girdilere karşılık gelen çıktı kümeleri belirlenir. Ağı oluşturan temel elemanların bilgileri işleme yetenekleri ve birbirleri ile bağlantılarının şekilleri değişik modelleri oluşturmaktadır (Haykin, 1999).

YSA'nın temel işlevi gözlemlerden dayalı olarak bir fonksiyon oluşturmak olarak düşünülebilir. Verinin karmaşık olduğu durumlarda veya bu veriyi temsil edecek fonksiyonun elle oluşturulmasının zor veya imkansız olduğu durumlarda kullanılabilir. Bu bağlamda kullanım alanları üç ana başlıkta toplanabilir :

1. Fonksiyon tahminleme (regresyon analizi, zaman serileri tahminlemesi, v.b.)
2. Sınıflandırma (örüntü tanıma, v.b.)
3. Veri işleme (filtreleme, kümeleme, sıkıştırma v.b.)

1.1.1 YSA tarihçesi

YSA konusunun temellerinin McCulloch ve Pitts'in 1943 yılındaki makaleleriyle (McCulloch and Pitts, 1943) atıldığı kabul edilir. Bu makalede sinir ağlarının birçok matematiksel fonksiyonu hesaplayabileceğini göstermişlerdir. Makale, bir çok araştırmacıyı konuya çekecek bir etki yaratmıştır. John von Neumann başta olmak üzere diğer araştırmacılar insan beyni gibi çalışan bilgisayarlar veya makineler tasarlanmasıyla ilgili yayınlar (Von Neumann, 1951; Von Neumann, 1956) yapmışlardır.

Donald Hebb'in 1949'da yazmış olduğu kitabında (Hebb, 1949) biyolojik sinir bağlantılarının öğrenmesiyle ilgili bir kural önermiştir. Bu kitap daha çok fizyoloji konusu üzerine olsa da bir çok araştırmacı için yapay sinir ağları için öğrenme yöntemleri geliştirmek anlamında ilham kaynağı olmuştur.

F. Rosenblatt yazdığı "Principles of Neurocomputing" kitabı (Rosenblatt, 1962) ve bulduğu Perceptron (Algılayıcı) ile YSA konusunun oluşmasında

önemli katkıda bulunmuştur. Daha sonraları, Perceptron öğrenmesinden farklı bir öğrenmeye sahip olan ADALINE (Widrow and Hoff, 1960), B. Widrow tarafında geliştirilmiş ve birçok alanda başarılı bir şekilde uygulanmıştır.

1969 yılında Minsky ve Papert'in yazdığı "Perceptrons" kitabında (Minsky and Seymour, 1969) XOR mantıksal fonksiyonunun Perceptron tarafından temsil edilemeyeceğini matematiksel olarak ispatlamışlardır. Bu sonuç konu üzerine olan ilgiyi olumsuz yönde etkilemiş ve araştırmacıların uzun bir süre konu üzerinde çalışmamasına neden olmuştur.

John Hopfield'ın 1983 ve 1986 yılında yapmış olduğu yayınlar (Hopfield, 1982; Hopfield, 1984) konuya ilgiyi tekrar canlandırmıştır. Rumelhart ve McClelland'ın 1986 yılındaki makaleleri (Rumelhart and McClelland, 1986) ile daha önce Minsky ve Papert'in algılayıcılar için bulduğu kısıtları ortadan kaldıran bir yaklaşım sunmuşlardır. Bu sayede öğrenme kurallarının geliştirilmesinin önünü açmışlardır ve konunun tekrar canlanmasında büyük bir etkide bulunmuşlardır.

1.2 YSA'dan Kural Çıkarma

Birçok konudaki başarılı uygulamalarına rağmen, YSA'lar sembolik makine öğrenme sistemlerindeki gibi bir açıklama mekanizmasına ihtiyaç duyarlar. Eğitilmiş bir YSA'nın ürettiği sonuca nasıl karar verdiğini anlamak bir problem olarak varlığını sürdürmektedir. Sembolik makine öğrenme sistemlerindeki neden sonuç ilişkisinin YSA'da doğrudan gözlemlenememesi bu yöntemin kararlı ve güvenilir olmasına gölge düşürmektedir. Bağlantıcı yapı üzerinde temel işlem elemanları arasındaki bağlantı ağırlıkları sayesinde ifade edilen öğrenilmiş bilgi, aynı zamanda karar sürecinin temel elemanları olduğu bilinmesine rağmen, bu dağıtık olarak ifade edilen bilginin sembolik olarak nasıl ifade edileceği kesin olarak bilinmemektedir. YSA'da gömülü olarak bulunan bilginin sembolik yorumlanması YSA'nın ***kara kutu*** problemini ortadan kaldırmaya yönelik bir yaklaşımdır. YSA üzerindeki bilginin tekrar düzenlenmesi ve sembolik olarak tekrar oluşturulması *kural çıkarma* olarak ifade edilir.

Özellikle hayati öneme sahip konulardaki uygulamalarında, YSA'nın davranışının açıklanması gereksinimi çeşitli kural çıkarma yaklaşımlarını ortaya çıkarmıştır. Bu yaklaşımlardan bir kısmı YSA'nın iç yapısını doğrudan ele alarak çözüm getirmeye çalışmaktadır. Bu tarzdaki yaklaşım çözümlemeli (decompositional) olarak adlandırılır. Bir diğer yaklaşımda ise YSA'nın girdilere karşılık verdiği çıktılar gözlemlenerek kurallar oluşturulmaya çalışılmaktadır.

YSA'nın sadece bir tahmin edici olarak kullanıldığı bu yaklaşıma eğitimsel (pedagogical) adı verilir. Çözümlemeli ve eğitimsel yaklaşımın bir arada kullanıldığı yöntemler ise derleme (eclectic) olarak adlandırılır.

YSA ve kural çıkarma birlikte kullanıldığında, özellikle verinin gürültülü olduğu durumlarda C4.5(Quinlan, 1993) gibi diğer kural öğrenme tekniklerinden daha başarılı olabilmektedir (Fu, 1994). Bu gibi sonuçlar konunun önemini göstermektedir.

YSA'dan kural çıkarmanın öncelikli amacı, eğitim verisini doğru sınıflandıran kurallar üretmek değil, YSA'yı en iyi şekilde temsil edecek ve açıklayabilecek kuralları oluşturmaktır.

1.3 Bağlantıcı (Connectionist) Sistemler

Bağlantıcı sistemler, çok sayıda basit ama birbirleriyle bağlı birimlerden oluşan ağlardır. Her bir birime gerçek değerler ulaşır. Genellikle bu birimler gelen bu değerleri basitçe toplamaktan çok, bir fonksiyona bağlı olarak kendi durumunu değiştirir. Her bağlantı taşıdığı değeri "ağırlığı" ile değiştirerek iletir (Fodor and Pylyshyn, 1988). Bağlantıcı sistemlerin en çok kullanılan biçimi yapay sinir ağlarıdır.

1.4 Melez Zeki Sistem

Melez bir sistem oluşturmanın başlıca nedenleri şu şekilde sıralanabilir (Goonatilake and Khebbal, 1994) :

- Yöntem iyileştirme; farklı yöntemlerin birbirlerinin zayıf yönlerini kapatacak şekilde birleştirilmesidir.
- Alt problemleri çözecek tek bir yöntem olmaması durumunda.
- Çoklu işlevsellik kazandırma; tek bir mimaride birden fazla bilgi işleme yeteneğine sahip bir sistem oluşturmak.

Melez zeki sistemler, birden fazla yapay zeka sistemini veya yöntemini yukarıdaki nedenlerden dolayı bünyesinde barındıran sistemler olarak tanımlanabilir.

1.5 Tezin Kapsamı

Bu tezde yapay sinir ağları için eğitimsel yaklaşımla bir kural çıkarma yöntemi geliştirilmiştir. Geliştirilen yöntemin literatüre katkısı temel olarak üç başlıkta ifade edilebilir:

1. YSA'nın sahip olduğu sınıflandırma bölgelerinin topolojik olarak ifade edilebileceği yaklaşımı literatüre kazandırılmıştır. Bunların başarılı bir şekilde kurallara dönüştürülebildiği gösterilmiştir.
2. Kural çıkarma yönteminin bağlantıcı, melez ve zeki bir sistem olması.
3. Kural çıkarma süreci, kullanıcı müdahalesini en aza indirecek şekilde tasarlanmıştır.

Geliştirilen sistemin temel mantığı, eğitilmiş bir YSA'yı tahminleyici olarak kullanıp girdi çıktı ilişkileri kullanılarak kuralların oluşturulması şeklindedir. Literatürde ağın sadece girdi çıktı ilişkilerinin dikkate alındığı kural çıkarma yaklaşımları **eğitimsel** olarak adlandırılır (Tickle et al., 1998).

Önerilen yaklaşımda girdi çıktı ilişkileri topolojik olarak ele alınmıştır. Girdi vektörlerinin ait oldukları sınıfları belirleyen ilişkiler eğer YSA tarafından öğrenilmiş ise, bu YSA'dan o sınıfları ayırmayı sağlayan kural bölgelerinin topolojileri öğrenilip bu topolojileri kapsayan kurallar oluşturulabilir. Bu topolojilerin bulunması için yine bir YSA modeli olan Growing Neural Gas (GNG) kullanılmıştır.

1.6 Tez Düzeni

Tez toplam 6 bölümden oluşmaktadır. Bölüm 2'de, yapay sinir ağlarından kural çıkarma konusu hakkında detaylı bilgi ve literatür çalışması verilmiştir. Bölüm 3'te, geliştirilen yönteme temel oluşturan teorik konulardan bahsedilmiştir. Bölüm 4'te, geliştirilen kural çıkarma yöntemi anlatılmıştır. Bölüm 5, geliştirilen yöntemin çeşitli problemler üzerinde uygulanmasına ve sonuçlarına ilişkin bilgileri içerir. Bölüm 6'da, tezle ilgili sonuçlar genel çerçevede yorumlanmıştır.

2 YSA'DAN KURAL ÇIKARMAYLA İLGİLİ ÇALIŞMALAR

Bu bölümde çeşitli problemlerin çözümü için eğitilmiş YSA'ların, neden kural çıkarma yöntemlerine ihtiyaç duyduğu açıklanmış ve bir kural çıkarma mekanizmasının YSA'lara kazandıracağı faydalardan bahsedilmiştir. Kural çıkarma yöntemlerinde takip edilen yaklaşımlar ve bu kural çıkarma yöntemlerini değerlendirmek için geliştirilmiş literatürde kabul görmüş bir taksonomi olan ADT'ye değinilmiştir. YSA'dan kural çıkarma yöntemlerinin kronolojik sırada literatür özeti bu bölümde yer almaktadır. Bu tez kapsamında önerilen kural çıkarma yöntemiyle aynı yaklaşımdaki (eğitimsel yaklaşım) belli başlı kural çıkarma yöntemleri hakkında detaylı bilgi verilmiştir.

2.1 Kural Çıkarma Gereksinimi

Yapay sinir ağları bir çok alanda başarıyla uygulanmaktadır. Ancak ulaştığı sonuçları açıklayamaması nedeniyle **kara kutu** (Şekil 2.1) olarak adlandırılmaktadır. Tutarlı açıklamaların yapılabilmesi özellikle bazı uygulama alanları için önemlidir. Örneğin; kredi kartı onay sistemi veya hastalık teşhis sistemi. Bu nedenle kural çıkarma algoritmaları önem kazanmıştır. Kara kutu özelliğinin getirdiği temel dezavantajlar şöyle sıralanabilir :

- Sistematik çıkarsama olmaması
- Doğrudan sebep-sonuç ilişkisi kurulamaması
- Kritik alanlarda (sağlık v.b.) kara kutu olması nedeniyle kabul edilemez riskler taşıması

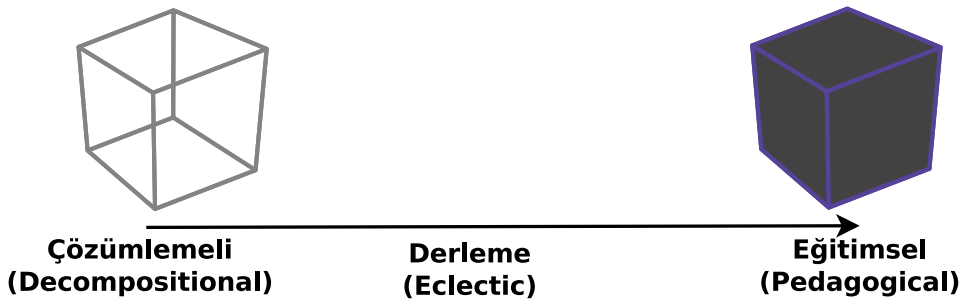
Şekil 2.1: Kara Kutu Sistem

Andrews ve diğ. YSA'dan kural çıkarmanın getireceği avantajları 6 başlıkta toplamışlardır (Andrews et al., 1995) :

1. “Kullanıcıya açıklama yapabilme” yeteneği katması
2. “Hayati önem taşıyan” alanlarda YSA’nın kullanılabilmesi
3. Yazılım sistemlerindeki YSA bileşenlerinin doğrulama ve hata ayıklama sürecine dahil edilebilmesi
4. YSA’nın yaptığı genelleme hatalarının açıklanabilmesi ve genelleştirmenin iyileştirilmesi
5. Veri incelemesi ve bilimsel teorilerin tümevarımı
6. Sembolik Yapay Zeka sistemleri için bilgi edinme

Bu kapsamda eğitilmiş sinir ağlarından bilgi çıkartma önemli bir yapay zeka konusu haline gelmiştir. Eğitilmiş bir sinir ağından sembolik bilgi temelde üç yaklaşımla çıkartılır (Tickle et al., 1998) (Şekil 2.2 (Andrews, 2003)’deki gösterim kullanılmıştır) :

- **Ayrıştırımsal** : Sinir ağının her elemanı incelenir. Bu seviyede çıkarılan bilgi birleştirilerek tüm ağın bilgi tabanı oluşturulur.
- **Eğitimsel** : Sadece ağın girdi/çıktı davranışı gözlemlenir.
- **Derleme** : İki yöntemin birlikte kullanıldığı bir yaklaşımdır.



Şekil 2.2: Ayrıştırımsal-Derleme-Eğitimsel Kural Çıkarma Yaklaşımları

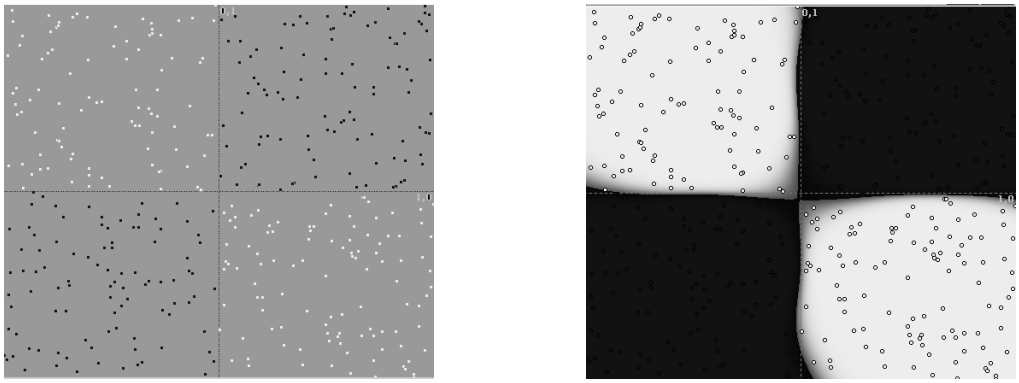
2.2 Kural Çıkarma Sürecine Genel Bakış

Bir sınıflandırma problemi için eğitilmiş bir YSA, girdi verilerinden genellikle daha fazlasını kapsayacak şekilde veriyi sınıflandırmayı öğrenir. Bu aynı zamanda YSA’nın genelleme yapabilmesinin temel nedenidir. Bu durumu görsel

olarak ifade etmek için Şekil 2.3'de sürekli değerler için XOR fonksiyonuna (Tablo 2.1) ait noktalar eğitim verisi olarak kullanılmış ve bir YSA bu iki sınıfı ayırması için eğitilmiştir. Eğitilmiş ağ kullanılarak, resimdeki her nokta için YSA'nın yaptığı sınıflandırma sonuçları, girdi verisine uygun olacak şekilde görselleştirilmiştir (Şekil 2.3).

Tablo 2.1: Sürekli XOR için Doğruluk Tablosu

A	B	Çıktı
$-1 < A < 0$	$-1 < B < 0$	0
$0 < A < 1$	$-1 < B < 0$	1
$-1 < A < 0$	$0 < B < 1$	1
$0 < A < 1$	$0 < B < 1$	0



a b

Şekil 2.3: a) Sürekli XOR Girdi Verisi b) Eğitilmiş YSA Karar Bölgeleri

YSA'nın ayırdığı sınıf noktaları ile girdi verileri arasındaki farklılık rahatlıkla görülebilmektedir. YSA girdi verisini doğru sınıflandırmakta ve bunun yanında genelleştirerek daha fazla noktayı sınıflandırabilmektedir.

Görsel ve az boyutlu bir örnek olması nedeniyle YSA'nın sahip olduğu sınıflandırma kuralları görselleştirilebilmiştir. Ancak bu her problem ve veri için mümkün değildir. Bu nedenle her boyutta veri için eğitilmiş ağlardan kural çıkarabilmek önemlidir.

Bu problem için çıkarılabilecek kural biçimi şu şekilde olabilir :

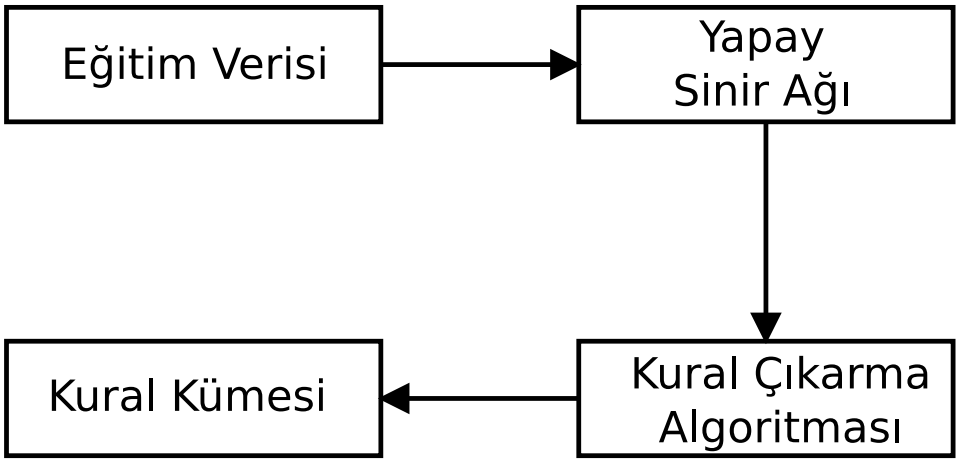
$$\begin{array}{ll} IF & \forall 1 \leq i \leq n \ : \ x_i \in \left[x_{i_{en\,küçük}}, x_{i_{en\,büyük}}\right] \\ THEN & girdi\ S_k\ sınıfına\ ait \end{array}$$

n girdi boyutunu ve k sınıf sayısını temsil eder. Bu problem için çıkarılabilecek örnek bir kural Tablo 2.2'de verilmiştir.

Tablo 2.2: Sürekli XOR Verisi için Eğitilmiş YSA'dan çıkarılabilecek örnek bir kural

IF	$-0.94 < A < -0.03$
AND	$-0.96 < B < -0.02$
THEN	0

Kural çıkarma süreci (Şekil 2.4) genel olarak şu şekildedir : Öncelikle probleme ait veriler ile istenilen hata seviyesinde YSA eğitimi yapılır. Bu eğitilmiş YSA, kural çıkarma algoritması tarafından kullanılarak kural kümesi oluşturulur. Algoritma, YSA'yı yaklaşım yöntemine göre iç yapısını ve sahip olduğu ağırlıkları kullanarak kural çıkarma yapabileceği gibi girdilere karşılık ürettiği cevapları da kullanabilir. Kural kümesi üzerinde sadeleştirme ve budama işlemleri yapılabilir.



Şekil 2.4: YSA'dan Kural Çıkarma Sürecinin Genel Akışı

2.3 Andrews-Diederich-Tickle (ADT) Sınıflandırması

Andrews, Diederich, ve Tickle YSA'dan kural çıkarma yöntemleri için ADT sınıflandırmasını tanımlamışlardır.(Andrews et al., 1995) Buna göre kriterler şunlardır :

- Çıkarılan kuralların *ifade gücü (expressive power)* (veya *kural biçimleri*)
- Çıkarılan kuralların *kalitesi (quality)*
- Ağın kural çıkarma tekniği ile yansıtılma *saydamlığı (translucency)*

- Kural çıkarma algoritmasının *karmaşıklığı (complexity)*
- Algoritmanın farklı ağlarda uygulanabilmesi; *taşınabilirlik (portability)*

2.3.1 İfade gücü

ADT sınıflandırması ile Tickle ve diğ.(Tickle et al., 1998) ifade gücü (veya kural biçimleri) kriteri için kuralları 4 temel gruba ayırmışlardır :

1. Önermeler mantığı (Propositional logic) (örnek: if . . . then . . . else)
2. Klasik olmayan mantık (Nonconventional logic) (örnek: fuzzy logic)
3. Birinci dereceden mantık (First-order logic)
4. Sonlu durum makineleri (Finite state machines)

2.3.2 Kalite

Towell ve Shavlik'in çalışmalarını(Towell and Shavlik, 1993) temel alarak , Andrews ve diğ. (Andrews et al., 1995) kalite için dört ölçüt belirlemişlerdir :

1. Kural kesinliği veya doğruluğu (accuracy) : Kural kümesinin, probleme ait daha önce kullanılmamış örnekleri doğru olarak sınıflandırmasıdır.
2. Aslına uygunluk (fidelity) : YSA'dan çıkarılan kural kümesinin, bu ağın davranışını ne kadar iyi yansıttığını ifade eder.
3. Tutarlılık (consistency) : Farklı eğitimlerden sonra YSA'dan üretilen kural kümelerinin, daha önce kullanılmamış örnekler için aynı sınıflandırmayı yapması.
4. Kural anlaşılabilirliği (comprehensibility) : Literatürde genellikle kuralların okunurluğu veya kural kümesinin büyüklüğü olarak ifade edilir. Ancak büyüklükle anlaşılabilirlik arasında birebir ilişki olduğunu söylemek doğru olmaz (Johansson et al., 2010). Anlaşılabilirliği ölçmek için yöntemler kısıtlıdır. Bu nedenle bu özellik pratikte daha çok ikili bir değer (anlaşılabilir veya anlaşılamaz) gibi düşünülüp öznel olarak veya sadece kural kümesi büyüklüğüne bakılarak belirlenir. Ve kural çıkarma yöntemlerinin birbirleriyle yüzeysel bir karşılaştırmasını yapmak için kullanılır.

2.3.3 Saydamlık

Saydamlık kriteri, üzerinde çalışılan YSA'nın nasıl ele alındığıyla ilgilidir. Çıkarılan kurallar ve YSA iç yapısı arasındaki ilişkiyi ifade eder. Decompositional, pedagogical ve eclectic olmak üzere üç grupta değerlendirilir.

1. Çözümleyici algoritmalar : Kurallar sinir ağ hücreleri seviyesinde oluşturulur. Ağın yapısındaki en temel elemanlar (aktivasyon fonksiyonları, bağlantı ağırlıkları v.b.) kullanılır.
2. Eğitimsel algoritmalar : İncelenen ağı kapalı kutu olarak kullanırlar.
3. Derleme algoritmalar : Diğer iki yaklaşım birlikte kullanılır.

2.3.4 Karmaşıklık

Kural çıkarma algoritmasının karmaşıklığı algoritmaların birbirleriyle karşılaştırılmalarında kullanmak için faydalı bir değerlendirme kriteridir. Ancak literatürdeki bir çok algoritmanın karmaşıklığı hakkında yazarların bilgi vermemesinden dolayı pratikte bu karşılaştırmalar tam anlamıyla yapılamamaktadır.

Ayrıca Golea (Golea, 1996) kural çıkarmanın NP-hard bir problem olabileceğini göstermiştir. Yaklaşımların, zaman ve yer karmaşıklığını nasıl etkilediği net değildir. Jacobsson (Jacobsson, 2005) RNN (Recurrent Neural Network) için kural çıkarma yöntemlerinin karmaşıklığı üzerine çalışılmadığını, bu incelemenin karmaşıklığı etkileyen birçok etken olmasından dolayı başlı başına karmaşık bir iş olduğunu belirtmiştir.

2.3.5 Taşınabilirlik

Son olarak taşınılabilirlik kriteri, kural çıkarma yönteminin farklı YSA mimarilerinde ve eğitim düzenlerinde uygulanabilirliğini ifade etmektedir. Literatürde sadece belli kriterlerdeki YSA üzerinde çalışan özel amaçlı algoritmalar bulunduğundan, bu kriterle kural çıkarma yöntemleri arasındaki bu açıdan farklılıkları ifade edilebilir.

Taşınabilirlik ile saydamlık kriteri arasında bir ilişki olduğunu söylemek

yanlış olmaz. Saydamlık kriterine bağlı olarak yöntemin taşınabilirliği de değişmektedir. Çözümlemeli yaklaşımlar belli bir YSA mimarisine ve/veya eğitim yöntemine bağlı oldukları için taşınabilirlikleri genelde mümkün değildir. Buna karşın eğitimsel yaklaşımların taşınabilirliğinin olduğunu söylemek yanlış olmaz. Hatta sadece YSA için değil farklı kapalı kutu sistemler için de uygulanabilir.

2.3.6 ADT için genel değerlendirme

Yukarıda açıklanan ADT sınıflandırması kural çıkarma yöntemlerini karşılaştırmak açısından önemlidir. Ancak pratikte bu sınıflandırma yöntemine ait kriterlerin ölçülmesinde problemler mevcuttur. Bu sınıflandırma kriterleri de zamanla değişmeye açıktır.

Farklı araştırmacılar benzer kriterler belirlemişlerdir. Örneğin Thrun iyi bir kural çıkarma algoritmasının dört temel kritere sahip olması gerektiğini yazmıştır (Thrun, 1993) :

1. Belirli bir YSA mimarisi gereksinimi olmaması
2. Özel bir eğitim gereksinimi olmaması
3. Doğruluk
4. Yüksek ifade gücü

2.4 Literatür Özeti

Bu bölümde literatürdeki çeşitli kural çıkarma yöntemleri hakkında genel bilgi verilmektedir.

YSA tarafında temsil edilen bilgiyi anlamaya yönelik ilk çalışma 1988 yılında Gallant tarafından önerilmiştir (Gallant, 1988). YSA'ları uzman sistemlerin bilgi tabanı olarak kullanan bir yöntem geliştirilmiştir.

1992'de McMillan ve diğ. bir YSA olan RuleNet'i tanımlamışlardır (Mcmillan et al., 1992). Çözümlemeli bir yaklaşımla eğitim sırasında kural çıkaran bir ağ modeli önerilmiştir. Ancak uygulama alanının sınırlı olması ve ayrıca kendine has eğitim yöntemi içerdiği için diğer problemlerde kullanımı için genelleştirmek zordur.

1993'de Towell ve Shavlik, KBANN (Towell et al., 1990) sistemler için Subset ve MofN algoritmalarını önermişlerdir (Towell and Shavlik, 1993).

Thrun VIA (Validity Interval Analysis) yaklaşımı ile bir kural çıkarma yöntemi önermiştir (Thrun, 1993). Bu yöntem, etkinleştirmelerin (activation) olduğu tüm aralıkları ileri ve geri yönde yayma fikrine dayanır. Ağ topolojisi ve eğitim yöntemine dair herhangi bir varsayım bulunmadığından çözümlemeli bir yaklaşım değildir.

Fu, sembolik ve bağlantıcı çıkarsamaları birleştiren melez bir sistem sunmuştur (Fu, 1993). KBCNN (Knowledge-based conceptual neural network) olarak isimlendirdiği bu modelde, faydalı görülen alan (domain) özellikleri ve kavramları tanımlanıp başlangıç alan bilgisi olarak sisteme bağlanır. Anlamsal bilgiyi koruyacak şekilde bu bağlantılara uygun ağırlık değerleri atanır. Daha sonra bu temel yapı emprik hatayı azaltacak şekilde kendi kendine uyum sağlayarak gelişir. KBCNN öğrenme modeli, öğrenilen bilgilerin sembolik kurallarla ifade edilmesini sağlar.

Daha sonra Fu, çözümlemeli KT algoritmasını önermiştir (Fu, 1994). Bu algoritma sezgisel olarak kural uzayında arama yapar. Bu kural uzayı girdi özelliklerinin kombinasyonları şeklinde genişletilmiştir. Girdi özellikleri pozitif ve negatif olmak üzere (özelliğin ağırlık değerinin pozitif veya negatif olmasına bağlı olarak) iki grup halinde ele alınır. Kurallar oluşturulurken bu grupların kendi içerisindeki kombinasyonları ele alınır daha sonra diğer gruptaki özelliklerle birleştirilerek devam edilir. KT algoritması, her bir gizli veya çıktı birimine karşılık gelen kavram için kurallar oluşturur. Her kavram için KT ilgili özelliklerin kombinasyonlarını bir arama ağacında ele alır. Bu ağaçtaki her bir düğüm özelliklerin bir kombinasyonunu ifade eder. KT ayrıca arama uzayını budamak için bazı sezgisel yöntemler uygular.

1994'te Craven ve Shavlik YSA'ların sorgulanabilir olmasını incelemişler ve kural çıkarma sürecini öğrenme problemi olarak değerlendirmişlerdir (Craven and Shavlik, 1994). Kural öğrenme sürecinde, birisi eğitilmiş YSA'yı temel alan diğeri ise SUBSET algoritmasına dayalı iki tane tahminleyici (oracle) kullanmışlardır. Bu iki tahminleyici sorgulanarak kurallar oluşturulmaktadır.

Sethi ve Yoo bağlantı ağırlıklarını sembolik gösterime çeviren bir yöntem önermişlerdir (Sethi and Yoo, 1994).

RULEX (Andrews and Geva, 1994) algoritması Radial Basis Function

(RBF) ağlara benzeyen Constrained Error Back-Propagation (CEBP) ağlardan kural çıkarmaya yarayan bir algoritmadır. CEBP ağındaki her bir düğüm eğitim örnekleri üzerindeki ayrık bölgeler ile sınırlandırılır. Bu algoritmanın temel özelliği, arama uzayını kendi kullandığı ağ ile kontrol etmesidir. Diğer yaklaşımlarda genellikle bu sezgisel yöntemlerle yapılmaktadır.

1995'de Alexander ve Mozer, McMillan ve diğ. 1992'deki (Mcmillan et al., 1992) çalışmalarındaki template-matching yaklaşımını genişleterek MofN kuralları çıkaran bir algoritma önermişlerdir(Alexander and Mozer, 1995).

1995'de Narazaki ve diğ. YSA'da dağınık olarak bulunan bilginin yeniden düzenlenmesini ve bulanık sınıflandırma kurallarının çıkarılmasını sağlayan bir yöntem önermişlerdir. (Narazaki et al., 1995)

Craven ve Shavlik, karar ağaçları çıkarmaya yarayan TREPAN algoritmasını önermişlerdir (Craven and Shavlik, 1995). Geleneksel karar ağacı oluşturma algoritmalarına benzemekle birlikte temel olarak YSA'yı tahminleyici olarak kabul edip sorgulayarak bu karar ağacını oluşturur.

Sethi ve Yoo 1994'deki bağlantı ağırlıklarına bağlı kural çıkarma algoritmalarını (Sethi and Yoo, 1994) multiple-valued logic (MVL) gösterimine dönüştürecek hale getirmişlerdir (Sethi and Yoo, 1996). Bu yöntemde tüm eksi değerdeki ağırlıklar artı değerlere dönüştürülüp daha sonra geri izlemeli (backtracking) arama yapılarak kurallar oluşturulur.

Herrmann, mantıksal YSA'lardan (logical neural nets) kural çıkarma algoritması tanımlamıştır (Herrmann and Thier, 1996). Yapay sinir hücrelerinin mantıksal operasyonları girdilerine uygulayabildiği bu ağlar için geriye yayılma öğrenme algoritmasını uyarlanmıştır. Bu sayede doğrudan mantıksal kurallar çıkarılabilinecek bir ağ oluşturulabilmektedir.

Taha ve Gosh, üç kural çıkarma algoritması ve çıkarılan kuralları değerlendirmek için bir metodoloji geliştirmişlerdir (Taha and Ghosh, 1996; Taha and Ghosh, 1999). İlk yöntem *Binarized Input-Output Rule Extraction (BIO-RE)*, ikili (binary) girdilerle eğitilmiş YSA'dan ikili kurallar çıkaran eğitimsel bir tekniktir. İkinci yöntem olan Partial-RE, gizli ve çıktı katmanı hücrelerine gelen ağırlıkları sıralama ve tek tek çıktıya etkisini deneyerek çıktıya etki eden ağırlıkları arama ve bunun sonunda kural oluşturma şeklindedir. Üçüncü yöntem Full-RE, bağlantıların farklı kombinasyonlarıyla çıktı hücresindeki etkisini incelemeye dayanır.

Lu ve diğ. gizli katman elemanlarının aktivasyon değerlerinin kümelenmesine dayalı bir kural çıkarma algoritması önermişlerdir (Lu et al., 1996). İstenilen doğruluk oranında eğitilmiş bir YSA üzerinde gereksiz bağlantılar bir ağ budama algoritması ile çıkarılır. Gizli hücrelerin aktivasyon değerleri incelenerek sınıflandırma kuralları çıkarılır.

1997'de Setiono, Liu ve Tan'ın çalışmasını temel alan, öncesinde YSA'daki gereksiz bağlantıları kaldıran daha sonra aktivasyon değerlerini kümeleyerek kural çıkaran bir algoritma önermiştir (Setiono, 1997).

Setiono ve Liu, NeuroLinear yöntemini anlatan makalelerini yayınlamışlardır (Setiono and Liu, 1997). Karar bölgelerinin sınırlarını ifade eden hiper düzlemleri bularak sınıflandırma desenlerini ifade eden kurallar oluşturan bir yöntemdir.

Benítez ve diğ. YSA modellerini bulanık kurallarla göstermeyi sağlayan bir yöntem geliştirmişlerdir (Benitez et al., 1997). Weijters ve diğ. YSA'yı eğiten ve otomatik olarak kural çıkaran bir algoritma geliştirmişlerdir (BP-SOM) (Weijters et al., 1997). Setiono ve Liu, sınıflandırma problemleri için eğitilmiş YSA'lardan oblique karar kuralları çıkaran (Setiono and Liu, 1997). 1998'de Das ve Mozer bir YSA öğrenme algoritması önermişlerdir (Das and Mozer, 1998).

1999'da Duch ve diğ. mantık kuralları çıkaran bir metodoloji önermişlerdir (Duch et al., 1999). Vahed ve Omlin geri dönüşümlü ağlardan kural çıkarmanın *deterministic finite-state automata* davranışı gösterdiğini belirmişlerdir (Vahed and Omlin, 1999). Keedwell ve diğ. YSA girdi uzayında kural arayan genetik algoritma barındıran bir sistem önermişlerdir (Keedwell et al., 2000).

Setiono ve Leow, FERNN algoritmasını önermişlerdir (Setiono and Leow, 2000). Kural çıkarma algoritmalarında ön işlem olarak sıkça kullanılan YSA'nın budanması aşamasında, tekrar eğitimin yapılması gereksinimini ortadan kaldırmayı amaçlayan hızlı bir budama yöntemi sunmuşlardır. Tek gizli katmana sahip ve tam bağlı bir eğitilmiş YSA üzerinde gizli katman birimlerinin bilgi getirisine bakılarak budama gerçekleşir.

Palade ve diğ. Backpropagation YSA'lar için bir kural çıkarma algoritması önermişlerdir (Palade et al., 2000). Bu yöntem ağ üzerinde aralık yayma (interval propagation) mantığına dayanır. Kural çıkarma algoritması, bir sinir ağını tersini çeviren bir süreç kullanır.

2001'de Duch ve diğ. crisp ve fuzzy kural çıkarma metodolojisi önermişlerdir

(Duch et al., 2001).

Garcez ve diğ. monoton olmayan kurallar çıkarmak için tam (complete) ve güvenilir (sound) bir yöntem önermiştir (d'Avila Garcez et al., 2001). Çözümlemeli yaklaşıma sahiptir. Yöntem öncelikle ağdaki ağırlıklar içerisindeki düzenli yapıları bulmaya çalışır. Eğer belli bir düzene sahip gruplar var ise bir dizi budama kuralları uygulanarak arama uzayı daraltılabilmektedir. Bu sayede önerilen kural çıkarma algoritmasının karmaşıklığı azaltılabilmektedir. Çıkarılan kural kümesinin boyutunu azaltmak için bir dizi basitleştirme kuralının uygun olduğu ifade edilmiştir. Önerilen yöntemin tam ve güvenilir olduğu gösterilmiştir.

Palade ve diğ. crisp kurallar çıkarmak için bir yöntem önermiştir (Palade et al., 2001). Snyders ve Omlin sembolic kuralların adaptiv bias olan ve olmayan YSA'lar için karşılaştırmasını yapmıştır. Jiang ve diğ. YSA'ları ve kural öğrenmeyi birleştiren bir yöntem sunmuşlardır (Jiang et al., 2002). Bu yöntemde YSA birliği (neural network ensemble) yaklaşımı kullanılmıştır.

Setiono ve diğ. doğrusal olmayan regresyon problemi için eğitilmiş YSA'da kullanılabilecek bir kural çıkarma yöntemi olan REFANN'ı önermişlerdir (Setiono et al., 2002). Tek gizli katmana sahip ve tek doğrusal çıktı elemanı olan ağlarda kullanılır. Bu yöntemde öncelikle gereksiz gizli katman elemanları ve girdi özellikleri (attribute) budanır. Daha sonra gizli katman aktivasyon fonksiyonu parçalı doğrusal fonksiyonlara çevrilir.

Fan ve Li farklı mekanik problemleri tespit etmek için eğitilmiş YSA için tanılama kuralları çıkaran bir yöntem sunmuşlardır (Fan and Li, 2002). Bu yöntemde gizli işlem birimleri, girdi uzayındaki bölgeleri gösterdiği şeklinde yorumlanmıştır. Eğitim verisi üzerinden oluşturulan başlangıç kuralları bir dizi sadeleştirme ve indirgeme algoritmaları kullanılarak son haline getirilmektedir.

Kolman ve Margolit APFRB (all-permutations fuzzy rule base) adını verdikleri yeni bir bulanık modeli geliştirmişlerdir (Kolman and Margaliot, 2005). Bu modelin matematiksel olarak standart ileri beslemeli sinir ağına eşit olduğunu göstermişlerdir. Makalelerinde bu eşitliğin kullanılmasıyla YSA'dan bilgi çıkarımı ve YSA'ya bilgi eklemenin nasıl yapılabileceğini açıklamışlardır.

Zhang ve diğ. özelliklerin sınıflandırmaya olan etkisini ölçmek için eğitilmiş YSA'nın diferansiyel bilgisini kullanan bir ölçüm yöntemi sunmuşlardır (Zhang et al., 2005). Bu yöntem kesikli ve sürekli verilerin ikisi için de uygulanabilir. Bu yöntem temel alınarak sürekli özelliklere sahip sınıflandırma problemleri için

eğitilmiş YSA için bir kural çıkarma yaklaşımı önerilmiştir.

Bader ve Hölldobler "the core method" diye isimlendirdikleri yöntemi önermişlerdir (Bader and Hölldobler, 2006). Bağlantıcı model üretmek için recurrent (geri dönüşlü) ağları ileri beslemeli çekirdek (core) ile kullanır.

Etchells ve Lisboa ikili veriler için kural çıkarmayı sağlayan genel bir metodoloji geliştirmişlerdir (Etchells and Lisboa, 2006).

Hruschka ve Ebecken sınıflandırma problemleri için eğitilmiş çok katmanlı algılayıcılardan (multilayer perceptrons) kural çıkarmayı sağlayan bir algoritma sunmuşlardır (Hruschka and Ebecken, 2006). Bu algoritma iki temel adımdan oluşmaktadır. İlk adımda, gizli işlem elemanlarının etkinlik değerleri kümelemeli (clustering) genetik algoritma uygulanarak bulunur. Daha sonra, girdi değerleri ile bağlantılı olarak bu kümeleri tanımlayan sınıflandırma kuralları üretilir.

Johansson ve diğ. kural çıkarma algoritmalarının eldeki tüm bilgiyi kullanmadığını ileri sürmüşlerdir (Johansson et al., 2006). Test verileri ile kapalı sistemin bu verilere karşılık ürettiği tahminlerin birlikte kullanılmasının daha başarılı sonuçlar verdiğini deneylerle göstermişlerdir.

Bader ve diğ. ikili eşik (binary threshold) elemanlarından oluşan ileri beslemeli sinir ağları için propositional kurallar çıkaran çözümleyicili bir yaklaşım sunulmuştur (Bader et al., 2007). Temel mantık, ağın çözümlenmesinden sonra uygun bir arama ağacı kullanılarak kurallar oluşturulmasıdır.

Guo ve diğ. sürekli ve kesikli girdi özellikleri ile ilgili problemleri çözmeye yönelik iki fikir sunmuşlardır (Guo et al., 2007). Bu sayede çıkarılan kurallar sadece kesikli veriler için değil aynı zamanda sürekli veriler içinde anlaşılabilir olabilmektedir. Her iki yöntemin birbirlerine göre avantaj ve dezavantajları gösterilmiştir.

Saad ve Wunsch HYPINV isimli bir kural çıkarma algoritması sunmuşlardır (Saad and Wunsch, 2007). Eğitimsel bir algoritmadır ve çıkardığı kurallar hiper düzlemler (hyperplane) şeklindedir. Ağı tersten kullanma (istenen bir çıktıyı üretecek girdiyi hesaplama) mantığına dayanır.

Tan ve diğ. öğrenebilen ve bulanık kurallar çıkarabilen melez bir sistem önermişlerdir (Tan et al., 2007). Sistem "fuzzy ARTMAP" ve "rectangular basis function network (RecBFN)" birleşiminden oluşur.

Huysmans ve diğ. her türlü kara-kutu sistemden kural çıkarmayı sağlayan Minerva algoritmasını önermişlerdir (Huysmans et al., 2008).

Zarate ve diğ. Formal Concept Analysis (FCA) kullanarak eğitilmiş YSA'dan kural çıkaran bir çalışma yapmışlardır (Zárate et al., 2008). FCANN ismini verdikleri bu yeni yaklaşım yapar verikümesi üretmeyi içeren bir aşamayı içerir. Bu yöntem ile parametreler arasındaki ilişkiler dolaylu kurallar ile açıklanır.

Dancey ve diğ. ExTree isimli kural çıkarma algoritmalarını içeren bir yayın yapmışlardır (Dancey et al., 2010). YSA'lardan karar ağacı çıkaran eğitimsel bir yaklaşımdır. Algoritmalarını sağlık veri kümeleri üzerinde uygulamışlardır.

Huynh ve Reggia hata geriye yayılım eğitim sürecini değiştirerek gizli katmanın girdi desenlerini daha etkin bir şekilde ayıracak şekilde eğitilmesini sağlamışlardır (Huynh and Reggia, 2011). Bu sayede daha kural karmaşıklığını arttırmadan ve başarısını azaltmadan daha az kural çıkarmak mümkün olmuştur. Ve bu sayede mevcut kural çıkarma algoritmalarının başarısı da artmıştır.

Mohamed genetik algoritmalarla kural çıkaran bir yöntem sunmuştur (Mohamed, 2011).

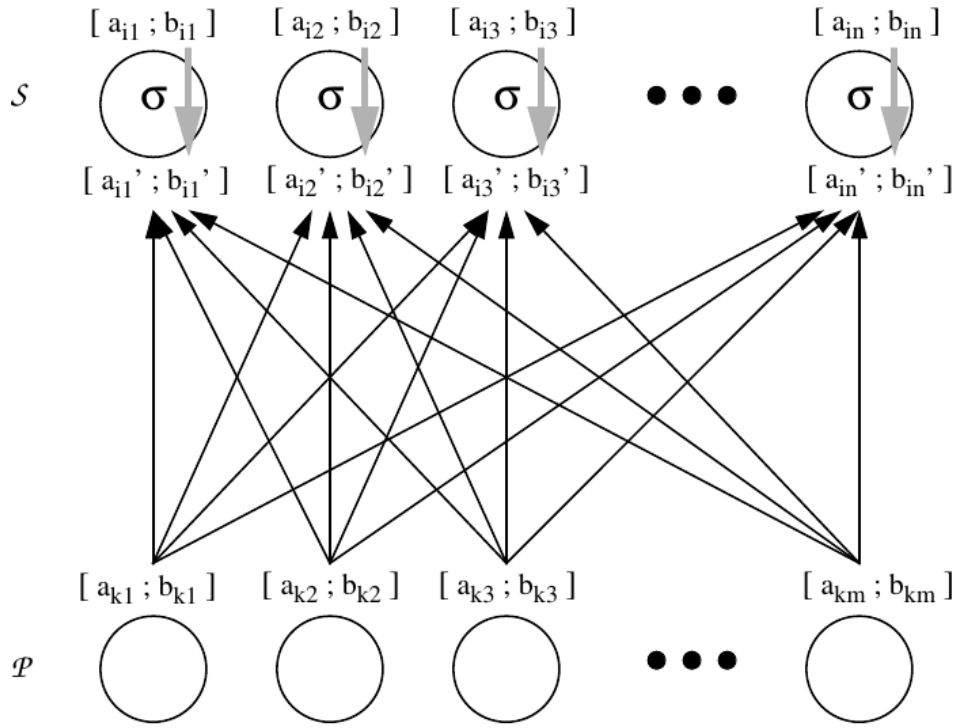
Huynh ve Reggia hata geri yayılımlı eğitim sürecini değiştiren ve bu sayede YSA'nın daha kolay FSM çıkarılabilecek hale gelmesini sağlayan bir yöntem önermişlerdir (Huynh and Reggia, 2012).

2.5 Literatürdeki Eğitimsel Yaklaşımlı Önemli Çalışmalar

2.5.1 VIA (Validity Interval Analysis)

Thrun bu çalışmada kural çıkarımı için özel bir eğitime sahip olmayan standart geri yayılımlı (backpropagation) ağlarda doğrudan girdi ve çıktı ilişkilerine bakarak kural çıkaran bir yöntem önermiştir (Thrun, 1993). Girdi desenlerindeki sistematik değişikliklerin çıktı sınıflandırmasını nasıl değiştirdiğini incelemiştir. Yöntem eğitilmiş ağda birim elemanların etkinlik durumlarını "geçerlilik aralığı" (validity interval) kavramını kullanarak incelemiştir. Bir birimin geçerlilik aralığı etkinlik değeri için en büyük aralığı temsil eder. Kural çıkarma; kural oluşturma ve deneme döngüsü içerisinde yapılır.

P katmanındaki elemanlar S katmanının elemanlarına bağlıdır. Her j ($j \in P \cup S$) elemanı için bir geçerlilik aralığı $[a_j; b_j]$ atanır. Tüm $i \in S$ için geçerlilik aralıklarının izdüşümleri ile girdiler toplamları net_i için $\left[a_j^{'}; b_j^{'}\right]$ aralıkları oluşturulur. Bu aralıklar ve tüm $k \in P$ elemanlarının geçerlilik aralıkları, P katmanındaki x_k aktivasyon değerleri için doğrusal kısıtları oluşturur (Şekil 2.5). Bu aşamadan sonra doğrusal programlama teknikleri uygulanabilmektedir.



Şekil 2.5: Geçerlilik Aralıklarının Ağ Katmanı Elemanlarına Atanması

$$
\begin{aligned}
\forall k \in P: \quad & x_k \geq a_k \\
& x_k \leq b_k \\
\forall i \in S: \quad & net_i \geq a_i^{'} \left(net_i = \sum_{k \in P} w_{ik} x_k + \theta_i \right) \\
& net_i \leq b_i^{'} \left(net_i = \sum_{k \in P} w_{ik} x_k + \theta_i \right)
\end{aligned}
$$

VIA algoritması (Algoritma 2.1) ile kural çıkarma, ileri ve geri olmak üzere iki aşamadan oluşur. Bu her iki aşamada da doğrusal programlama ile işlem birimlerinin aktif olduğu aralıklar bulunur. Doğrusal programlama tekniklerinden Simplex algoritması tercih edilmiştir.

Algoritma 2.1 VIA Algoritması

1 - İleri : $i \in S$ elemanları için a_i ve b_i sınır değerlerini iyileştirmek için doğrusal programlamayla yeni sınırlar bulunur.

$$\hat{a}_i^{'} = \min(net_i)\ ,\ \hat{a}_i = \sigma(\hat{a}_i^{'})$$
$$\hat{b}_i^{'} = \max(net_i)\ ,\ \hat{b}_i = \sigma(\hat{b}_i^{'})$$

min/max işlemleri doğrusal programlama aracılığıyla hesaplanır.

2 - Geri : $k \in P$ elemanları için a_k ve b_k sınır değerlerinin iyileştirmesi yapılır. Yine doğrusal programlama kullanılır.

$$a_k = \min(x_k)$$
$$b_k = \max(x_k)$$

VIA genel amaçlı bir kural çıkarma yöntemidir. Thrun, yöntemi XOR , "Three Monks" , robot kolu kinematiği problemlerinde uygulamıştır. Yöntem, herhangi bir problem türüyle kısıtlı olmamakla beraber Thrun karmaşık problemlerde yöntemin başarısız olduğunu belirtmiştir.

2.5.2 Rule Extraction as Learning

Craven ve Shavlik önerdikleri yöntemde temel fikir olarak kural çıkarmayı bir öğrenme işi olarak ele almışlardır (Craven and Shavlik, 1994). Hedef olarak YSA'nın temsil ettiği fonksiyon ve girdi özellikleri de doğrudan ağın girdi özellikleri olacak şekilde kullanılmıştır. Yöntemin çalışma adımları Algoritma 2.2 ve Algoritma 2.3'te verilmiştir.

Algoritma 2.2 Rule Extraction as Learning

```
for each class c
    R_c := ∅
repeat
    e := EXAMPLES()
    c := classify(e)
    if e not covered by R_c then
        /* learn a new rule */
        r := conjunctive rule formed from e
        for each antecedent r_i of r
            r' := r but with r_i dropped
            if SUBSET(c, r') = true then r := r'
        R_c := R_c ∨ r
until stopping criterion met
```

Algoritma 2.3 Examples

```
/* create a random example */
for each feature e_i with possible values v_i1,...,v_in
    e_i := randomly-select(v_i1,...,v_in)
calculate the total input s to output unit
if s ≥ θ then return e
impose random order on all feature values
/* consider the values in order */
for each value v_ij
    if changing feature e_i's value to v_ij increases s
        e_i := v_ij
    if s ≥ θ then return e
```

Değiştirilen kuralların eğitilmiş ağ ile çelişip çelişmediğini kontrol etmek için Subset adını verdikleri bir fonksiyon kullanmışlardır. Çıkan kurallar M-of-N türündedir. Bu yöntem de ağın eğitimi için herhangi bir özel yöntem kullanılmasını gerektirmemektedir.

2.5.3 Ruleneg

Pop ve diğ. *PAC öğrenme*[1] yöntemine dayalı Ruleneg eğitimsel kural çıkarma yöntemini önermişlerdir (Pop et al., 1994). Bu yöntem eğitim verisindeki her örneği kullanarak kurallar üretir. Kural üretirken örnek desenin her bir özelliğinin önemini test eder. Eğer bir özelliğin yokluğu sonuç sınıfında değişikliğe yol açıyor ise önemli olarak kabul edilir. Kurallar bu şekilde özellik değerlerini toplayarak oluşturulur. Sadece ikili değerler üzerinde çalışıyor olması önemli bir dezavantaj getirir.

2.5.4 TREPAN

TREPAN algoritması (Algoritma 2.4) her türlü öğrenme modelinden karar ağacı çıkaran genel amaçlı bir yöntemdir (Craven and Shavlik, 1995). Bu yöntemde sorgular kullanılarak öğrenme modeli tarafından temsil edilen içeriğe yaklaşan bir karar ağacı üretilir. Yazarlar bu yöntemin yüksek boyut problemlere iyi ölçeklenebildiğini belirtmişlerdir.

TREPAN, doğrudan eğitim verilerini kullanan geleneksel karar ağacı oluşturma yöntemlerine (CART, ID3, C4.5, ID2-of-3) benzer bir yöntemdir. Farklı olduğu noktalar şu şekilde sıralanabilir :

[1] Probably approximately correct learning (Valiant, 1984)

- Tahminleyici kullanımı : Tahminleyici olarak eğitilmiş YSA kullanarak, rastgele veriler üreterek bunların ait olduğu sınıf etiketleri bulunabilir. Karar ağacı oluşturulurken bu bilgi kullanılır.

- Bölme Türü (Split Type) : M-of-N tarzında ifadeler kullanır. Bir M-of-N ifade boole (boolean) bir ifadedir. Bu ifadede m tam sayısı eşik değerini ifade eder ve n tane ikili ifade içeren bir kümeye sahiptir. Bir M-of-N ifadesi n tane ifadeden m tanesi sağlanmış ise sağlanmış kabul edilir.

  Örnek 2.1. a, b, c ikili değerler olsun. $2 - of - \{a, b, c\}$ M-of-N ifadesi mantıksal olarak şu ifadeye eşittir : $(a \wedge b) \vee (a \wedge c) \vee (b \wedge c)$

- Bölme Seçimi (Split Selection) : Geleneksel karar ağacı oluşturma algoritmalarında karar ağacının derinliği arttıkça eldeki eğitim verilerinin sayısı azaldığı için bölme seçimi yetersiz olmaktadır. Buna karşın TREPAN eğitilmiş YSA'yı tahminleyici olarak kullanabildiği için bu noktada istediği kadar veri üreterek buradaki seçimin başarısını arttırabilmektedir.

Algoritma 2.4 TREPAN Algoritması

```
/* Given a net's training set and feature set, induce a tree that models the net.  */
TREPAN(training_examples, features)
{
   for each example E ∈ training_examples    /* obtain the net's labeling of each example */
      class label for E := ORACLE(E)
   return MAKE_SUBTREE(training_examples, features, {})
}

MAKE_SUBTREE(examples, features, constraints)
{
   check stopping criteria using examples and calls to ORACLE(constraints)
   if stopping criteria satisfied
      make new leaf, L
      determine class label for L using examples and calls to ORACLE(constraints)
      return L
   else
      /* use features to build splits;
         use examples and calls to ORACLE(constraints) to evaluate them */
      S := best binary split
      using S as a seed, search for best M-of-N split, S'
      make new node, N, for S'
      for each outcome, I, of S'
         examples_(I) := members of examples with outcome I on split S'
         constraints_(I) := constraints ∪ {S' = I}
         Ith child of N := MAKE_SUBTREE(examples_(I), features, constraints_(I))
      return N
}
```

2.5.5 Symbolic interpretation of artificial neural networks

Bu makalede üç adet kural çıkarma algoritması önerilmiştir (Taha and Ghosh, 1999). Bunlardan Binarized Input-Outpu Rule Extraction (BIO-RE) eğitimsel bir

yaklaşıma sahiptir. Sadece ikili değerler ile eğitilmiş YSA'lar üzerinde çalışabilir. Eğer girdiler ikili düzende değil ise bunlar şu şekilde çevrilir :

$$y_i = \left\{ \begin{array}{cc} 1 & eğer\ x_i \geq \mu_i \\ 0 & değilse \end{array} \right\}$$

Burada x_i, X_i girdisinde bir değerdir. μ_i, X_i'nin ortalama değeridir. y_i ise ikili sisteme çevrilmiş girdi değeridir.

BIO-RE yöntemi eğer girdi verisi ikili düzende ise veya ikili düzene çevrildiği zaman YSA eğitiminin başarısında bir düşüş yaşanmıyor ise uygulanabilir bir yöntemdir.

2.5.6 Orthogonal search-based rule extraction (OSRE) for trained neural networks

Etchells ve Lisboa, RULENEG (Pop et al., 1994) algoritmasını 1'den N'e kadar kodlanmış kategorik ve sıralı veri kümeleri üzerinde çalışabilir şekilde değiştiren bir algoritma önermişlerdir (Etchells and Lisboa, 2006). Girdilerin ikili sistemde kodlanır ve sonuçta çıkarılan ikili kuralların analitik karar yüzeyine uyması sağlanır.

2.5.7 Neural network explanation using inversion (HYPINV)

Saad ve Wunsch sınıflandırma problemleri için kullanılan YSA'lar için bir kural çıkarma algoritması sunmuşlardır (Saad and Wunsch, 2007). HYPINV, hiperdüzlemler kullanarak conjunction (bağlama) ve disjunction (ayrışma) şekillerinde kurallar üretir. YSA'nın karar sınırlarını parçalı olarak bu hiperdüzlemlerle ifade eder. Kurallar için aslına uygunluk ve karmaşıklık ilişkisi kontrol edilebilir. Ağın girdileri ikili veya sürekli değerler olabilir ancak ağın çıktısı sadece ikili değerler alabilir.

2.5.8 FCANN

Zárate ve diğ. Formal Concept Analysis (FCA) kullanarak YSA'dan bilgi çıkaran bir yöntem önermişlerdir. FCA'nın temel işlevlerinden biri de özel diagramlarla bilgi gösterimi yapmasıdır.(Zárate et al., 2008)

FCANN aslında diğer kural çıkarma yöntemleri gibi tek başına bir sınıflandırıcı değildir. Bunun yerine girdi uzayındaki parametreler arasındaki ilişkiler bulan bir bilgi çıkarma yöntemidir. Oluşturulan sentetik veriler ile FCA kullanarak bu ilişkiler diagramlar şeklinde ifade edilir.

2.5.9 Recursive neural network rule extraction for data with mixed attributes

Setiono ve diğ. önerdiği yöntemin getirdiği yenilik kurallar oluşturma sürecinde kesikli verileri sürekli verilerden önce ele almasıdır (Setiono et al., 2008). Algoritma öncelikle kesikli değerlerle ifade edilen özellikler ile YSA'nın sınıflandırma sürecini açıklamaya çalışır. Sadece kesikli değerlerden oluşan bir kural eğer yeterli doğruluğa ulaşmaz ise iki yöntem ile bu kuralı geliştirir :

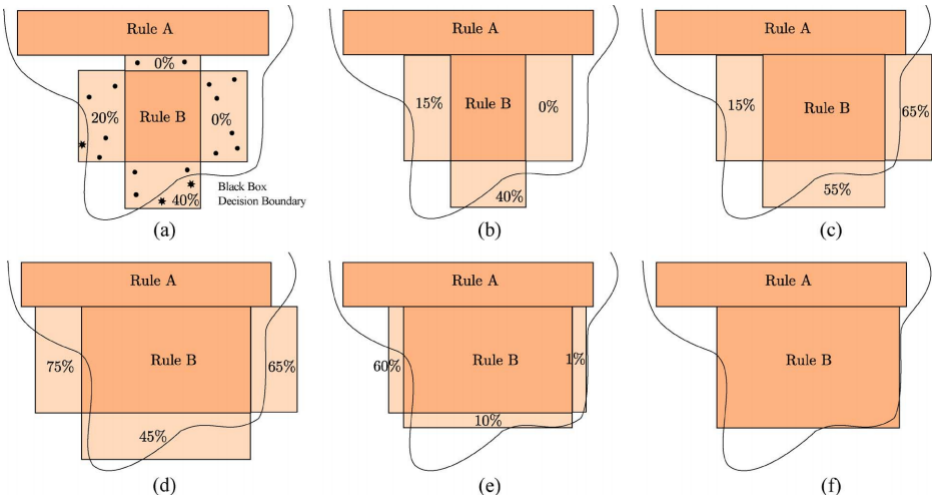
1. Özyinelemeli olarak kuralda bulunmayan kesikli değerlerle gösterilen özellikleri ekleyerek daha fazla kural oluşturma
2. Sadece sürekli değerlerle gösterilen özelliklerden oluşan hiper düzlemler oluşturmak.

Algoritma için başlangıçta kullanılan YSA budama algoritması büyük önem taşır. Bu sayede problemin çözümü için gerekli olmayan girdiler olmadan daha etkin kurallar çıkarılabilmektedir.

2.5.10 Minerva

Minerva eğitimsel yaklaşımla geliştirilmiş bir yöntemdir. Bu nedenle her türlü kapalı kutu sisteme uygulanabilir. Temel mantık olarak tek bir girdi desenini kural olarak kabul edip, bu kuralın sınırlarını kapalı kutu sistemin karar sınırlarına yaklaşacak şekilde genişleterek kurallar kümesini bulmayı amaçlamaktadır. Çakışan kurallardan sakınma ve basit kurallar oluşturma

özelliklerine sahiptir.(Huysmans et al., 2008) Temel algoritma adımları Algoritma 2.5'de gösterilmiştir. Algoritmanın çalışmasına örnek Şekil 2.6'de gösterilmiştir.



Şekil 2.6: Minerva algoritmasının çalışmasına örnek

Algoritma 2.5 Minerva algoritması

```
MINERVA(Examples, Threshold)
1  RuleSet= ∅
2  Initialize UpdateSizes //to default values
3  Rule =LEARN-ONE-RULE(Ruleset,Threshold,Examples,UpdateSizes)
4  While (Not Finished)
5    RuleSet=RuleSet ∪ Rule
6    Examples = Examples - {Examples covered by Rule}
7    Rule= LEARN-ONE-RULE(Ruleset,Threshold,Examples,UpdateSizes)
8  End While
9  Remove Bad Rules from RuleSet //Bad Rule = covers less than 3 training observations
                                    or training accuracy below 50%
10 Ruleset=RuleSet ∪ Default Rule //Class of Default Rule = Majority of
                                    not covered training examples
11 Return RuleSet
```

```
   Learn-One-Rule(OtherRules, Threshold, Examples, UpdateSizes)
1    Rule= Convert an Example into a very specific rule
2    niterations=5
3    for i=1 to niterations
4      while cangeneralize
5        Rule= GeneralizeRule(Rule,OtherRules,Threshold,UpdateSizes)
6      end
7      UpdateSizes=UpdateSizes/5
8    end
9    Rule=PruneRule(Rule,OtherRules,Examples)
```

```
   GeneralizeRule(Rule, OtherRules, Threshold, UpdateSizes)
1  BestPerformanceDiff=+Inf
2  for each attribute i
3      //try to decrease lower bound
4      NewRule=Rule
5      UpdateStep(i)=min{UpdateSizes(i),AllowedUpdate(i)}
6      NewRule.Upperbound(i)=Rule.Lowerbound(i)
7      NewRule.Lowerbound(i)=Rule.Lowerbound(i)-Updatestep(i)
8      PerformanceDifference=EvaluateRule(NewRule,Rule.Prediction)
9      if PerformanceDifference<BestPerformanceDiff
10         BestPerformanceDiff=PerformanceDifference
11         GeneralizedRule=Rule
12         GeneralizedRule.Lowerbound(i)=NewRule.Lowerbound(i)
13     end
14     //try to increase upper bound
15     ...(similar)
16   end
17 end //end for each attribute
18
19 if BestPerformanceDiff>Threshold
20     //rule can not be generalized, return original rule
21     Return Rule
22 else
23     Return GeneralizedRule
24 end
```

```
   EvaluateRule(Rule,OriginalRulePrediction)
1    SampledObservations= CreateSamples(Rule, NumberofSamples)
2    SamplePredictions= Query(BlackBox, SampledObservations)
3    if ProblemType='classification'
4      NumberSimilar=Number of samples for which SamplePredictions=OriginalRulePrediction
5      PerformanceDiff=1-NumberSimilar/NumberofSamples
6    elseif ProblemType='regression'
7      PerformanceDiff=|Mean(SamplePredictions)-OriginalRulePrediction|
8    end
```

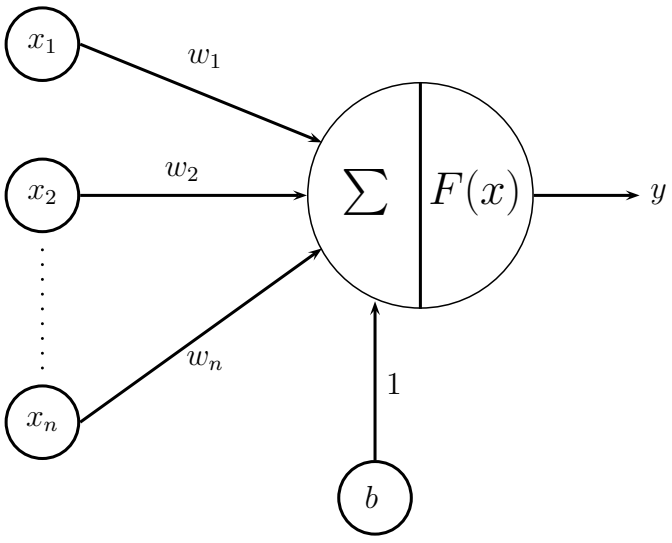
3 ÖNBİLGİLER

Bu bölüm, tez kapsamında önerilen yöntem için teorik altyapıyı oluşturan konular hakkında genel bilgiler içermektedir. Sırasıyla topoloji kavramından genel olarak bahsedilmiştir. Belli başlı topolojik tanımlar ve bu tez kapsamında kullanılan özellikleri hakkında bilgi verilmiştir. Bu bağlamda topoloji öğrenme kavramı üzerinde durulmuştur.

Topoloji öğrenme yeteneğine sahip ve önerilen yöntemde kullanılan Growing Neural Gas (GNG) algoritması hakkında detaylı bilgi verilmiştir. GNG paratrelerinin veri kümesine göre eniyileştirilmesi için kullanılan genetik algoritmalardan da bu bölümde bahsedilmiştir.

3.1 YSA Teorisi

Bu bölümde bir YSA'nın matematiksel olarak nasıl ifade edildiği genel hatlarıyla gösterilmektedir. En basit işlem birimi olan yapay sinir hücresi (Şekil 3.1), kendisine gelen girdiyi çıktıya dönüştüren bir yapıya sahiptir. Bu yapıda girdiler ağırlıklarıyla çarpılarak bir toplam değeri oluşturulur. Daha sonra bu toplam değeri, hücreye özgü aktivasyon fonksiyonu ile çıktının oluşturulmasını sağlar.



Şekil 3.1: Yapay Sinir Hücresi

Bir YSA temelde bir yönlü çizgedir. k birim elemanı t anında şu değerlere sahiptir:

1. $x_i(t)$ girdi vektörünün t anındaki değerini temsil eder.
2. $w_i(t)$ ağırlık vektörünün t anındaki değerini temsil eder.
3. b sapma miktarı (bias)
4. F aktivasyon fonksiyonu
5. $y(t)$, t anındaki çıktı değeridir.

$$y(t) = F(\sum_{i=1}^{n} w_i(t) \cdot x_i(t) + b)$$

Aktivasyon fonksiyonu sinir hücresinin temel özelliğini belirler. Herhangi bir matematiksel fonksiyon olabilir. Kullanılan aktivasyon fonksiyonu squashing fonksiyonu türünden olmalıdır. Yaygın olarak kullanılan aktivasyon fonksiyonları: Adım fonksiyonu, doğrusal fonksiyon ve doğrusal olmayan (sigmoid) fonksiyonlardır.

Birim elemanlar yönlü ve ağırlıklı bağlantılarla birbirlerine bağlıdırlar. j elemanından k elamnına olan bağlantı w_{kj} olarak gösterilebilir.

Tanım 3.1. Squashing Fonksiyonu

Bir $f : \mathbb{R} \to \mathbb{R}$ fonksiyonu yalnız ve yalnız sabit olmayan, sürekli, monoton artan ve sınırlı bir fonksiyonsa squashing fonksiyondur.

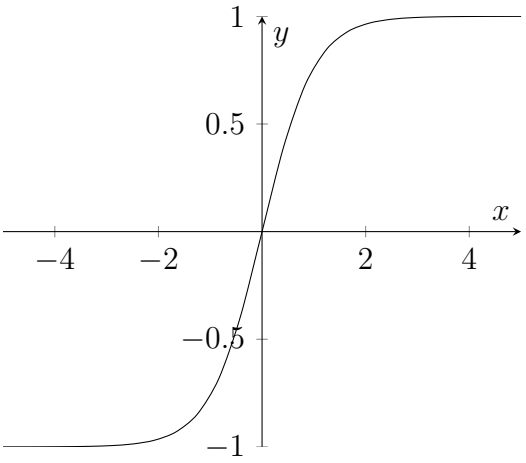
Tanım 3.2. Sigmoid Fonksiyon

$p, h, s \in \mathbb{R}$ olmak üzere sigmoid fonksiyonların parametreli hali şu şekilde tanımlanır:

$$\frac{h}{1 + e^{-s(x-p)}}$$

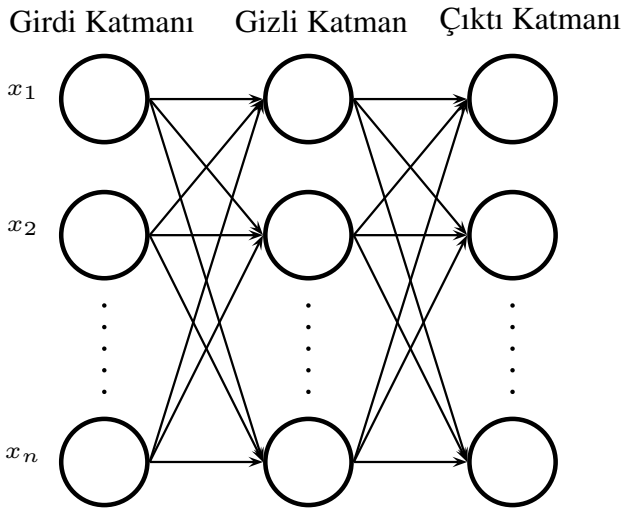
Tanım 3.3. Hiperbolik Tanjant (Şekil **3.2**)

$$\frac{2}{1 + e^{-2x}} - 1$$

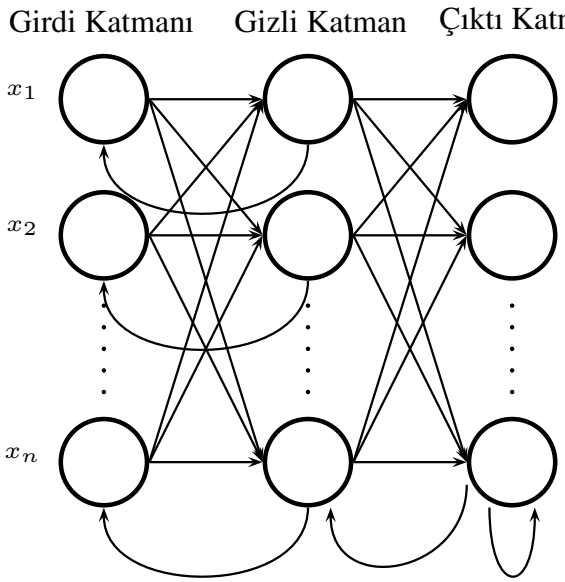


Şekil 3.2: Hiperbolik Tanjant Fonksiyonu

Aralarındaki bağlantılar ve oluşturdukları katmanlara göre değişik mimarilere (Şekil 3.3 ve 3.4) sahip YSA'lar bulunmaktadır.



Şekil 3.3: İleri Beslemeli Ağ Modeli

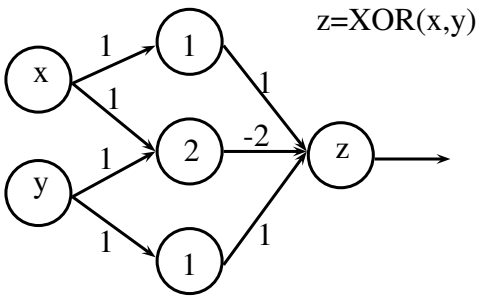


Şekil 3.4: Recurrent Ağ Modeli

Tek katmanlı algılayıcılar ile çözülemeyen doğrusal olmayan XOR (Tablo 3.1) mantıksal fonksiyonu çok katmanlı algılayıcılar kullanılarak temsil edilebilir (Şekil 3.5).

Tablo 3.1: XOR için Doğruluk Tablosu

x	y	z (çıktı)
0	0	0
1	0	1
0	1	1
1	1	0



Şekil 3.5: XOR Probleminin Çok katmanlı Algılayıcılarla Örnek Çözümü

Teorem 3.4. Kolmogorov'un Sinir Ağı Dönüşüm Varlığı Teoremi (Kolmogorov's Mapping Neural Network Existence Theorem)

Verilen sürekli bir $f : [0,1]^n \rightarrow R^m, f(x) = y$ *fonksiyonu için,* f *girdi katmanında* n *işlem elemanı, gizli katmanında* $(2n+1)$ *işlem elemanı ve çıktı katmanında* m *işlem elemanı olan üç katmanlı ileri beslemeli bir sinir ağı tarafından tam olarak gerçekleştirilebilir.*

Teoremin ispatı (Hecht-Nielsen, 1987) yayınında bulunabilir. Bu teorem herhangi bir fonksiyonun bir YSA ile temsil edilebileceğini göstermekle beraber bu ağın nasıl oluşturulacağı hakkında kesin sonuca ulaştıran bir yöntem bilinmemektedir.

Funahashi, bu teoremi şu şekilde tekrar formüle etmiştir (Funahashi, 1989) :

Teorem 3.5. $\phi : \mathbb{R} \rightarrow \mathbb{R}$ *sabit olmayan, sınırlı, monoton artan sürekli bir fonksiyon olsun.* $K \subseteq \mathbb{R}^n$ *kompakt,* $f : K \rightarrow \mathbb{R}$ *sürekli bir eşleme (mapping) ve* $\epsilon > 0$ *olsun.* ψ *squashing fonksiyonuna sahip, girdi-çıktı eşlemesi* $\overline{f} : K \rightarrow \mathbb{R}$ *şeklinde ve* $\max_{x \in K} d\left(f(x), \overline{f}(x)\right) < \epsilon$*'yi sağlayan bir 3 katmanlı ileri beslemeli ağ vardır.* d*,* $\mathbb{R}$ *topolojisini indükleyen bir ölçüttür.*

Bu iki teoremin gösterdiği bakış açısıyla bu araştırmada 3 katmanlı ileri beslemeli ağlar üzerinde çalışılmıştır.

3.2 YSA'da Öğrenme

YSA öğrenmesi üç ana başlıkta toplanır: Eğiticili (Supervised), Eğiticisiz (Unsupervised) ve Destekleyicili (Reinforcement) öğrenme.

- Eğiticili Öğrenme
- Eğiticisiz Öğrenme
- Destekleyicili Öğrenme

Literatürde, geliştirilen öğrenme algoritmalarına temel olan belli başlı öğrenme yaklaşımları hakkında bilgi verilmiştir.

3.2.1 Hebbian öğrenmesi

Bu yöntem Donald Hebb tarafından sunulan öneriyi temel alır (Hebb, 1949): Buradaki temel fikir, iki sinir hücresi arasındaki pozitif etkileşimin ikisi arasındaki bağlantının ağırlığını kuvvetlendireceği ya da negatif etkilişim sonucu bu ağırlığın zayıflayacağı şeklindedir. Klasik tanımıyla gösterilecek olursa:

k ve j iki işlem elemanı olsun ve w_{kj} ise j'den k'ya olan bağlantının ağırlığı olsun. t anındaki j'nin değeri $v_j(t)$ ve k'nın değeri $v_k(t)$ ile gösterilsin. j ile k arasındaki ağırlığın değişim miktarı şu şekildedir:

$\Delta w_{kj}(t) = F(v_j(t), v_k(t))$, burdaki F herhangi bir fonksiyondur. Bu formül genellikle şu şekilde kullanılır:

$\Delta w_{kj}(t) = \alpha(v_j(t))(v_k(t))$, buradaki α sabit bir değerdir ve genellikle "öğrenme oranı" olarak tanımlanır.

Son olarak ağırlık değeri şu şekilde güncellenir :

$w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj}(t)$

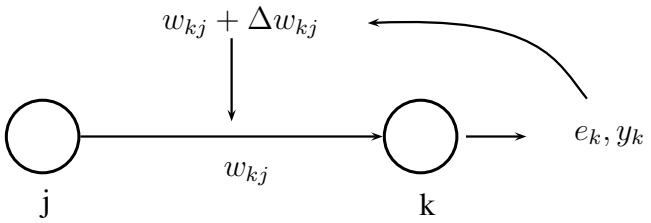
Hebbian öğrenmesi genellikle eğiticisiz öğrenme yöntemlerine temel oluşturur.

3.2.2 Hata düzeltmeli öğrenme

$b_k(t)$, k elemanı için t anındaki beklenen çıktı, $y_k(t)$ ise gerçek çıktı olsun. $y_k(t)$ çıktı değeri $y_j(t)$ vektörünün k elemanına girdi olarak uygulanmasıyla elde edilir. Beklenen ve gerçek çıktı değerleri arasındaki fark $e_k(t) = b_k(t) - y_k(t)$, $\Delta w_{kj}(t) = \alpha e_k(t) y_j(t)$

$$w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj}(t)$$

Hata düzeltmeli öğrenme (Şekil 3.6) eğiticili öğrenmede kullanılan temel yaklaşımlardan biridir.



Şekil 3.6: Hata Düzeltmeli Öğrenme

3.2.3 Rekabetçi öğrenme ve Kohonen katmanı

Kohonen katmanı ve öğrenme kuralı en temel rakabetçi öğrenme algoritmasıdır. Kohonen katmanı, her biri n girdi $v_1, \ldots, v_n$ alan N elemandan oluşur. Bir v_j girdisini Kohonen elemanı i'ye bağlayan w_{ij} ağırlığı vardır. $\mathbf{w_i}$ ağırlık vektörünü $\mathbf{v}$ ise girdi vektörünü temsil eder.

Her bir Kohonen birimi i için girdiye olan uzaklığı $D_i(\mathbf{w_i}, \mathbf{v})$ hesaplanır. Genellikle bu uzaklık hesaplamasında Öklid uzaklığı kullanılır. Kohonen katmanındaki her eleman için bu uzaklık değerleri hesaplandıktan sonra en küçük uzaklık değerine sahip eleman **kazanan** olarak belirlenir. Bu kazanan elemanın ağırlık vektörü girdi vektörüne yakınsayacak şekilde güncellenir.

3.3 Topoloji ve Topoloji Öğrenme (Topology Learning)

Tanım 3.6. X bir küme olsun. X üzerindeki bir τ ***topoloji***si, X'in alt kümelerinden oluşan bir derlemedir, öyle ki :

1. $\emptyset$ ve X açık kümedir
2. τ'nin elemanlarının her türlü birleşimi yine τ'nin elemanıdır
3. τ'nin sonlu sayıdaki her türlü elemanının kesişimi yine τ'nin elemanıdır

X ve kendi üzerinde tanımlı τ topolojisi birlikte bir ***topolojik uzay*** oluşturur.

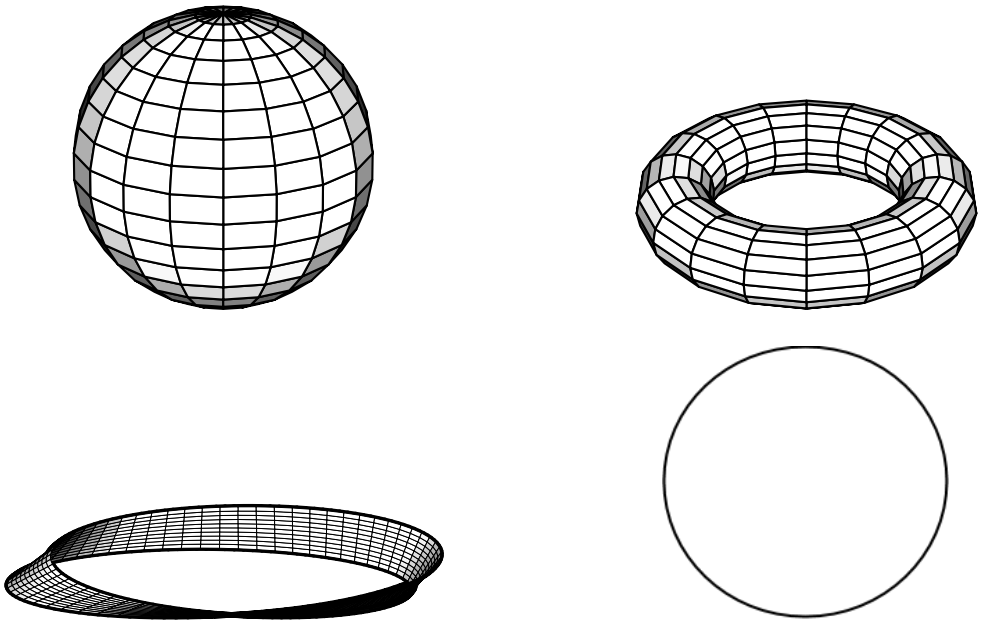
Topolojik uzay (X, τ) şeklinde ifade edilmekle birlikte genellikle sadece X şeklinde de gösterilebilir.

Her türlü metrik uzay, topolojik uzay olarak değerlendirilebilir. X bir metrik uzay olsun ve d uzaklık fonksiyonuna sahip olsun. X'in bir alt kümesi

V açık kümedir sadece ve sadece V'de bulunan herhangi bir v noktası için $\{x \in X : d(x, v) < \delta\} \subset V$ koşulunu sağlayan $\delta > 0$ değeri vardır. Boş küme $\emptyset$ ve tüm X açık kümedir ve diğer iki önerme de sağlanmaktadır. Metrik uzay, topolojik uzay olma önermelerinin hepsini sağlar. Bu tür açık kümelerden oluşan derlemeye, d uzaklık fonksiyonunun X üzerinde oluşturduğu topoloji denir.

Önerme 3.7. *n-boyutlu Öklid uzayı $\mathbb{R}^n$'de bulunan her X alt kümesi bir topolojik uzaydır. Ayrıca $\mathbb{R}^n$ kendi başına bir topolojik uzaydır. Bu uzay, Öklid uzaklık fonksiyonu ile elde edilmiştir.*

Şekil 3.7'de dört adet topolojik uzay örneği verilmiştir.



Şekil 3.7: Topolojik uzaylara örnekler : Küre, Torus, Mobius ve Çember

Topoloji öğrenme; verilen bir çok boyutlu veri dağılımının $P(\xi)$ topolojisini yansıtan topolojik yapıyı bulmak olarak tanımlanabilir (Fritzke, 1995). Çeşitli yayınlarda, topoloji öğrenme kavramı ile aynı veya yakın anlam taşıyan topoloji modelleme (topology modelling) veya topoloji temsili (topology representing) gibi ifadeler de kullanılmaktadır.

Elde bulunan bu çok boyutlu Öklid uzayına ait noktalar kümesinin topolojik özelliklerinin ölçülmesi ile bu veri kümesi hakkındaki bilgi arttırılabilir.

Bir noktalar kümesine topolojik açıdan temelde iki şekilde yaklaşılır:

1. Topoloji korumalı (topology preserving) yaklaşım

2. Topoloji öğrenme (topology learning) yaklaşımı

Topoloji korumalı yaklaşımda, verinin daha aşağı seviyeli boyutlu ve sınırlı bir topolojiye sahip uzaya doğrusal olmayan yansıması ile (nonlinear projection) görselleştirmeye imkan vermesi amaçlanır (Tibshirani, 1992; Bishop et al., 1998; Tipping and Bishop, 1999; Lee et al., 2002; Silva and Tenenbaum, 2003). Bu konuda yapılan başlıca çalışmalar; Generative Topographic Mapping (Bishop et al., 1998) , Multi-Dimensional Scaling (Lee et al., 2002; Silva and Tenenbaum, 2003), Principal Curves (Tibshirani, 1992). Bu çalışmaların hepsinde elde edilecek modelin boyutu görselleştirmeye imkan verecek şekilde önceden sabitlenir ancak bu da topolojinin sınırlanmasına neden olur. Ayrıca SOM (Kohonen, 1990) yaklaşımı da Kohonen katmanının topolojisinin önceden belirlenmesi gerektiği için bu kategoride değerlendirilir.

Topoloji öğrenmede ise topolojisi sınırlı olmayan ve veriyi temsil edebilen bir yapı oluşturmak amaçtır (Martinetz and Schulten, 1994; De Silva and Carlsson, 2004).

Karmaşık bir fonksiyonun daha basit temel fonksiyonların birleşimi şeklinde ifade edilmesi gibi, karmaşık bir topoloji uzayının basit ve temel topolojik yapılar halinde gösterilmesi mümkündür. Bir **simpleksler kompleksi** (simplical complex) bu tarzda, simplekslerin birleşmesinden oluşan bir modeldir. Her bir simpleks kendi boyutuna sahiptir. 1-simpleks doğru parçasını, 2-simpleks üçgeni, ... k-simpleks $k+1$ noktadan oluşan dışbükey (convex) yapıyı temsil eder. Bir simpleksler kompleksinde, simpleksler köşeleri veya yüzleri ile doğrudan bağlantılıdır. Bu şekildeki bir yapıdan topolojik bilgi çıkarmak mümkündür (De Silva and Carlsson, 2004). Delaunay kompleksi bir simpleksler kompleksidir. Delaunay çizgesi, Delaunay kompleksinin köşeleri ve kenarlarından oluşur (Okabe et al., 2009).

Topoloji öğrenme ile ilgi çalışmalar Edelsbrunner ve Shah'ın çalışmalarına dayanır (Edelsbrunner and Shah, 1997). Bu çalışmada, verilen bir $\mathcal{M} \subset \mathbb{R}^D$ manifoldu ve $\mathcal{M}$'in yakınındaki bir küme N_0 vektör prototipleri $\underline{w} \in \left(\mathbb{R}^D\right)^{N_0}$ için $\mathcal{M}$ ile aynı topolojiye sahip bir simpleksler kompleksinin olduğunu ve bunun da $\underline{w}$'nin Delaunay kompleksi olduğunu belli koşullara bağlı olara ispatlamışlardır .

Mevcut problemde $\mathcal{M}$ manifoldu bilinmemekle birlikte ancak M veri noktası $\underline{v} \in \mathcal{M}^M$ sayesinde bilinebilir. Bu problemi çözmek için Martinetz ve Schulten "Rekabetçi Hebbian Öğrenmesi" (Competitive Hebbian Learning - CHL) olarak isimlendirdikleri ve prototiplerden oluşan bir çizge oluşturan bir algoritma önermişlerdir (Martinetz and Schulten, 1994). Yaklaşımları De Silva ve Carlsson

tarafından simpleksler kompleksine uyarlanmıştır (De Silva and Carlsson, 2004). Her iki çalışma için de $\mathcal{M}$ manifoldu için Edelsbrunner ve Shah'ın ispatındaki koşullar zayıflatılmış ve bu sayede sonlu sayıda $\mathcal{M}$'ye ait $\underline{v}$ ile doğrulanabilmiştir. Böylece $\underline{w}$ üzerinde oluşturulan çizge veya simpleksler kompleksi, eğer $\underline{v}$ $\mathcal{M}$'in yeterli yoğunluktaki bir örneklemi ise $\mathcal{M}$ ile aynı topolojiye sahip olduğu ispatlanmıştır.

CHL, $\underline{w}$'de bulunan iki prototipi eğer $\underline{v}$ noktasında en yakın (yakınlık Öklid uzaklığı olarak) birinci ve ikinci komşular ise bu iki prototipi birbirine bağlar. Bu şekilde topolojiyi temsil eden çizge oluşur.

3.4 Growing Neural Gas (GNG)

Growing Neural Gas (GNG) algoritması Bernd Fritzke tarafından yayınlanmış bir eğiticisiz kümeleme algoritmasıdır. Verilen R^n için tanımlı girdi dağılımı için GNG büyüyen bir çizge veya düğüm ağı oluşturur. Her bir düğümün R^n'de bir konumu vardır. GNG, kümelerdeki kod vektörleri bulunarak vektör kuantizasyonu için kullanılabilir. Ayrıca girdi dağılımını temsil eden topolojik yapıyı bulmak için de kullanılabilir. (Fritzke, 1995)

Algoritma iki düğümle başlayarak, düğümlerin kenarla bağlı olduklarında komşu kabul edildikleri bir çizge oluşturur. Komşuluk bilgisi rekabetçi Hebbian öğrenmesinin bir türeviyle bulunur :

Her girdi sinyali $\overline{x}$ için buna en yakın Öklid uzaklığına sahip iki düğüm arasına bir kenar eklenir.

GNG sadece zamana bağlı değişmeyen aynı zamanda sabit değerlere sahip parametreler de kullanır. Ayrıca ne kadar düğüm kullanılacağı da önceden belirlenmek zorunda değildir. Düğümler çalışma zamanında eklenir. Yeni düğüm eklenmesi ağın performans kriterine veya en fazla düğüm sayısı değeriyle sınırlanabilir.

GNG algoritması her k düğümünün aşağıdakilere sahip olduğunu kabul eder:

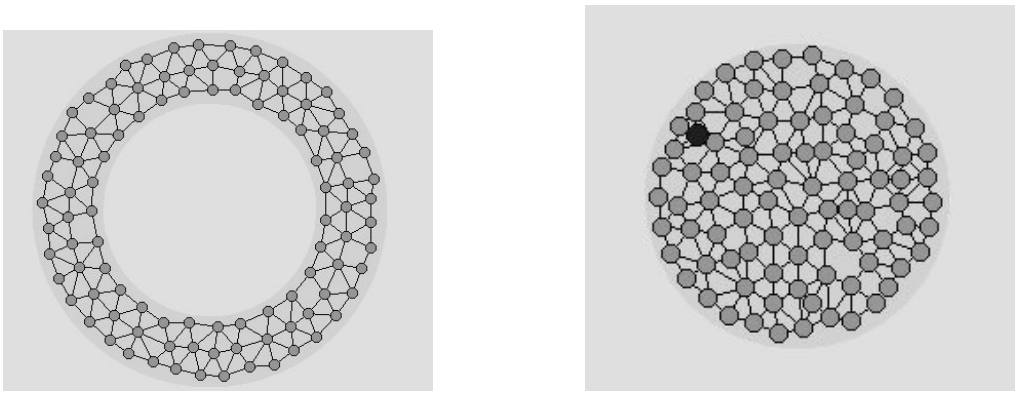
- $\overline{x}$: R^n için tanımlı referans vektörü.
- $error_k$: yerel toplam hata değişkeni.
- k düğümünün topolojik komşularını tanımlayan kenar kümesi.

Referans vektörü, düğümün girdi uzayındaki konumu olarak düşünülebilir. Yerel toplam hata, yeni düğümlerin eklenmesinde kullanılan bir istatistiksel ölçümdür. Her kenar **yaş** değişkenine sahiptir. Bu sayede yaşlanmış kenarların ne zaman kaldırılacağına karar verilebilir ve topoloji güncel tutulmuş olur.

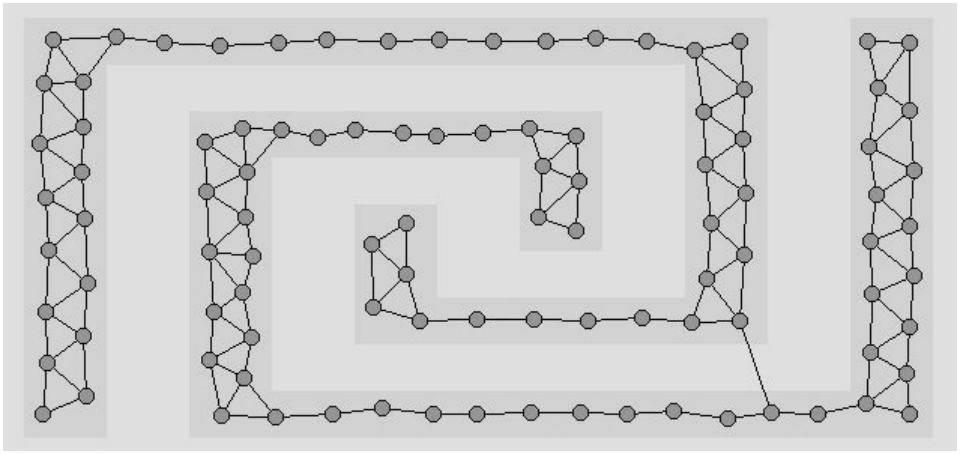
GNG algoritması Algoritma 3.1'de verilmiştir.

3.4.1 GNG ile topoloji bulma örnekleri

GNG algoritması topoloji bulma ve temsil etme özelliğini görsel olarak ifade eden örnekler şu şekilde incelenebilir. Şekil 3.8'de halka ve daire , Şekil 3.9'da iç içe geçmiş 2 spiral, ve Şekil 3.10'da kesikli veri kümeleri için GNG'nin başarılı şekilde topolojileri bulduğu görülebilir[2].



Şekil 3.8: GNG - Halka ve Daire Örneği



Şekil 3.9: GNG - Spiraller Örneği

[2]GNG ile ilgili şekiller DemoGNG uygulaması kullanılarak elde edilmiştir. http://sund.de/netze/applets/gng/full/GNG.html

Algoritma 3.1 GNG Algoritması

1: **İlklendirme** : Rastgele konuma sahip iki düğüm yarat. Birbirlerine sıfır yaşa sahip kenarla bağla. Ve düğümlerin hata değerini sıfıra eşitle.

2: Girdi vektörü $\overline{x}$'ı oluştur.

3: $\overline{x}$ vektörüne en yakın iki düğüm s ve t'yi bul. Tüm k düğümleri için, $\overline{w_s}$ ve $\overline{w_t}$ referans vektörlerine sahip iki düğüm için $\|\overline{w}_s - \overline{x}\|^2$ en küçük ve $\|\overline{w}_t - \overline{x}\|^2$ ikinci en küçük değer olacak şekilde bulunur.

4:Kazanan s düğümünün yerel hata değeri güncellenir.

$$error_s \leftarrow error_s + \|\overline{w}_s - \overline{x}\|^2$$

5: s ve topolojik komşularını (s düğümüne bir kenarla bağlı tüm düğümler) $\overline{x}$ vektörüne, uzaklığın e_w ve e_n oranlarında yaklaştır. $e_w, e_n \in [0, 1]$

$$\overline{w}_s \leftarrow \overline{w}_s + e_w(\overline{x} - \overline{w}_s)$$

$$\overline{w}_n \leftarrow \overline{w}_n + e_n(\overline{x} - \overline{w}_n), \forall n \in Komşu(s)$$

6: s düğümüyle komşuluğundaki tüm düğümlerle olan kenarların yaş değerini arttır.

7: Eğer s ve t bir kenarla bağlıysalar kenarın yaş değerini 0 yap. Bağlı değillerse bir kenarla bağla.

8: Yaş değeri a_{max}'dan büyük olan kenarlar varsa bunları kaldır. Kenar kaldırmalardan sonra hiç bir kenarla bağlı olmayan düğüm varsa o düğümleri de kaldır.

9: Tekrar sayısı λ'nın bir tam katıysa ve en fazla düğüm sayısına daha ulaşılmamışsa yeni bir düğüm ekle.

Yeni bir düğüm r eklenmesi şu şekilde yapılır:
9_1: En büyük hataya ($error$) sahip u düğümünü bul.

9_2: u komşuluğundaki en büyük hataya sahip v düğümünü bul.

9_3: Yeni düğüm r'ı u ve v düğümleri arasına şu şekilde yerleştir:

$$\overline{w}_r \leftarrow \frac{(\overline{w}_u + \overline{w}_v)}{2}$$

9_4: u ile r ve v ile r arasında kenarları oluştur, u ile v arasındaki kenarı kaldır.

9_5: u ve v'nin hata değerlerini azalt, r'nin hata değerini ata.

$$error_u \leftarrow \alpha \times error_u$$

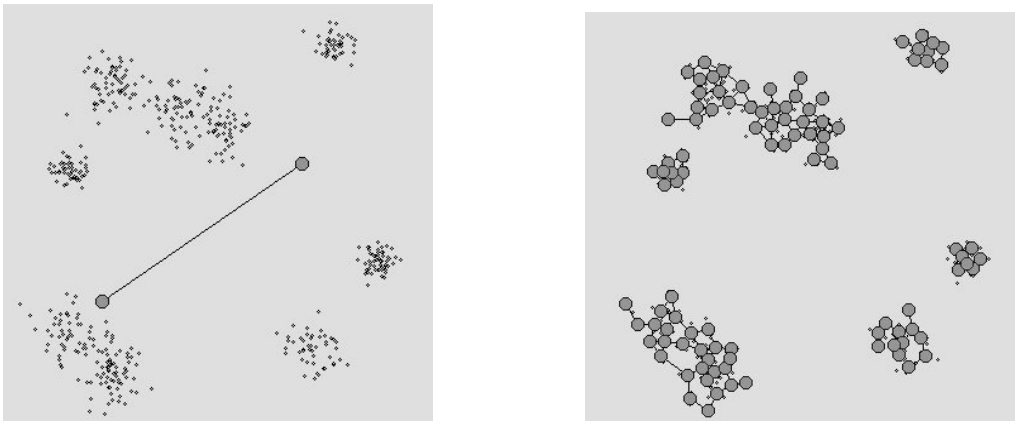
$$error_v \leftarrow \alpha \times error_v$$

$$error_r \leftarrow error_u$$

10: Tüm düğümlerin hata değerlerini azalt:

$$error_j \leftarrow error_j - \beta \times error_j$$

11: Durma kriteri sağlanmadıysa 2. adıma git.



Şekil 3.10: GNG - Kesikli Veriler Örneği

3.5 Genetik Algoritmalar (GA)

Genetik algoritmalar (GA), doğadaki genetik yapının gelişimi benzeri bir yaklaşım kullanan bir rastlantısal arama ve eniyileştirme tekniğidir (Goldberg, 1989). GA karmaşık ve büyük arama uzayında uygunluk fonksiyonunun ölçümüne göre optimal çözüme yakın çözümler üretebilir. Arama uzayı kromozom denilen yapılarda kodlanmış halde tutulur. Bir dizi kromozomun biraraya gelmesiyle populasyonlar yaratılır.

Uygunluk fonksiyonu (fitness function) her bir kromozomun istenilen çözüme ne kadar yakın olduğunu ölçer. Biyolojideki kavramlar olan çaprazlama ve mutasyon operatör olarak kullanılır ve yeni koromozomlar yani nesiller üretilir. Bu üretim istenilen uygunluk fonksiyonu ölçümüne ulaşılana kadar veya başka durma koşulu sağlanana kadar devam eder.

3.5.1 Kromozomların kodlanması

Her kromozom, problemin bir çözümüdür. Bu çözümü temsil etmek için GA'da kullanılabilecek biçimde temsil etmek gerekir. İkili kodlama veya permütasyon kodlama en sık kullanılan yöntemlerdir.

3.5.2 Çaprazlama (Crossover)

Çaprazlama birden fazla çözüm kromozomunu kullanarak yeni bir kromozom oluşturma işlemidir. Kromozomların bu iş için seçilmesi için çeşitli yöntemler

vardır:

- Roulette Wheel
- Boltzmann
- Tournament
- Rank
- Steady state
- Truncation

Çaprazlama bir kaç türlü yapılabilmektedir:

- Tek nokta çaprazlama : Seçilen iki kromozom üzerinde tek bir nokta belirlenerek, bu noktanın ayırdığı parçaların yer değiştirilmesi şeklindedir.
- İki nokta çaprazlama : Seçilen iki kromozom üzerinde iki nokta belirlenerek, bu noktalar arasında kalan parçaların yer değiştirilmesi ile gerçekleşir.
- Kes ve ekle : Kromozomlar üzerinden farklı noktalar seçilere parça değişimi yapılır. Bu nedenle yeni kormozomların uzunlukları farklı olur.
- Uniform Çaprazlama: Kromozomlar arasında sırayla her bir parçanın değiştirilme olasılığına göre yer değiştirilmesi şeklinde yapılır.

3.5.3 Mutasyon

Bir kromozom üzerindeki rastgele bir parçanın önceki durumundan farklı bir duruma değiştirilmesidir. Mutasyon, yerel minimumlardan kaçınmayı sağlaması açısından önemlidir.

3.6 YSA'nın Tahminleyici Olarak Kullanılması

Önerilen yöntemin YSA ile etkileşimi sadece üyelik sorgularıyla (membership query) (Angluin, 1988) olmaktadır. Üyelik sorgusu, tahminleyiciye yöneltilen öğrenicinin örnek uzayına ait bir örnek sorudur. Verilen bir üyelik sorgusuna karşılık tahminleyici örneğin ait olduğu sınıf etiketini cevap olarak verir.

Bu bağlamda, önerilen yöntem ile YSA tarafından temsil edilen fonksiyon veya karar bölgelerinin keşfedilmesi amaçlanır. Eğitilmiş bir YSA, tahminleyici olarak kullanılır ve üyelik sorguları ile örneklerin sınıflandırılması sağlanır.

Üyelik sorgulaması YSA'nın eğitim verileri için yapılabileceği gibi sonradan üretilen rastgele örnekler için de yapılabilir. Eğitim verileri için YSA'nın cevap olarak döndüğü sınıf etiketleri gerçekte doğru olan sınıfı göstermeyebilir. YSA'dan kural çıkarmanın amacı eğitim verisini doğru sınıflandıran kurallar üretmek değil YSA'yı en iyi şekilde temsil edecek ve açıklayabilecek kuralları oluşturmaktır. Bu bağlamda YSA'nın verdiği cevaplar doğru olarak kabul edilir.

3.7 Geliştirilen İlk Yöntem

Önerilen kural çıkarma yönteminin geliştirimi aşamasında oluşturulan ilk yönteme, tez kapsamında önerilen yöntemin gelişim aşamaları hakkında fikir vermesi açısından bu bölümde kısaca değinilmiştir.

GNG'nin topoloji bulma özelliği, kural sınırlarının bulunmasında kullanılır. GNG, girdi verisinin her boyutu için ayrı ayrı çalıştırılır. Yani girdi verisinin aynı anda sadece tek boyutundaki değerler değişken yapılarak GNG'ye girdi verisi olarak verilir. Başlangıç olarak kullanacağımız girdi verisi rastgele türetilebileceği gibi, eğitim verisi içerisinden de seçilebilir. Aradaki fark, sadece doğru sınıflandırılmış bir girdi verisi üretmek için harcanacak zamandır. GNG için girdi verileri türetilecek başlangıç girdi verisi referans noktası olarak adlandırılabilir.

3.7.1 Geliştirilen ilk yöntemin detayları

Eğitilmiş bir YSA'dan, bu yöntemle kural çıkarmak için temel adımlar şunlardır:

Rastgele üretilen bir veri (n boyutlu) üzerinde GNG algoritması her bir boyut için çalıştırılır. İlk üretilen verinin YSA'nın verdiği sınıf etiketinden farklı bir veri gelirse bu sınıf için yeni bir GNG ilklendirilir ve bundan sonra bu sınıf için gelen veriler ait olduğu GNG'ye girdi olarak verilir.

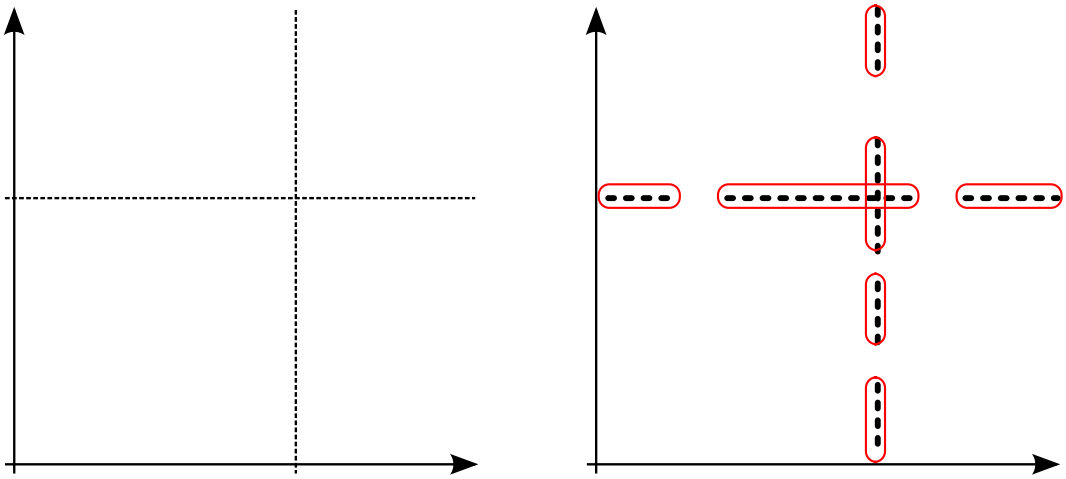
Bu sayede problem uzayı üzerinde eksenlere paralel kesitler alarak Sınıflandırma bölgeleriyle bu kesitlerin kesişimiyle oluşan aralıklar GNG'ler tarafından temsil edilir.

Bu ilk yöntem şu aşamalardan oluşur:

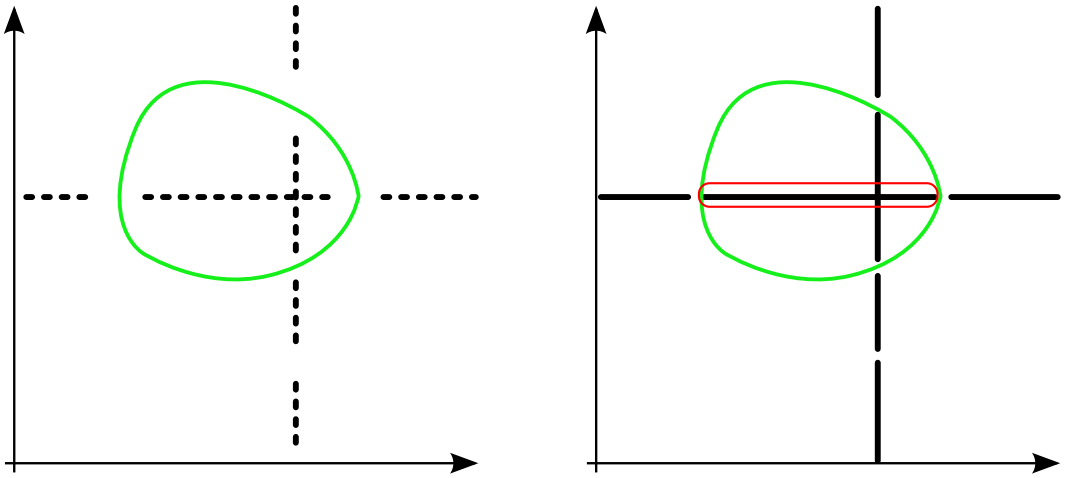
1. Eğitim verilerinin en küçük ve en büyük değerleri aralığında kural kümesinin kapsamadığı rastgele bir nokta yarat.
2. Bu noktayı kullanarak her boyut için noktanın diğer boyutlarındaki değerlerini değiştirmeden rastgele noktalar oluştur.
3. Her boyut için o boyut için oluşturulan noktalar ve bu noktalara karşılık eğitilmiş YSA'dan gelen çıktı değerlerini kullanarak GNG çalıştır.
4. GNG sonucu oluşan topolojik olarak birbirinden ayrı parçaları bul. Bu parçaları, kural bölgesinin sınırlarına genişleyecek şekilde büyüt.
5. Bu parçalardan referans noktası ile kesişenler içerisinden en uzun olanı al. Kesişmeyen parçaları bir listede tut.
6. Bulunan en uzun parçayı bölme oranına gore parçalara böl. Parçanın orta noktasını ata.
7. Her parça için orta nokta kullanılarak kendi ve son boyut haricindeki boyutlar için sırasıyla rastgele noktalar üreterek GNG çalıştır. Özyinelemeli olarak 4-5-6-7 adımlarını tekrarla. En son boyuta gelindiğinde sadece 4 ve 5 adımlarını çalıştır.
8. En son boyuta gelindiğinde kendi dahil bu adıma kadar her boyut için üretilen parçaların en düşük ve en yüksek değerlerini kullanarak kural oluştur ve kural kümesine ekle.
9. Durma kriteri için performans ölçümü yap: Kural kümesinin kapsadığı eğitim noktaları eğitim kümesinden çıkar.
10. Durma kriteri sağlanmadıysa 1. adıma git.

3.7.2 Bir örnek üzerinde açıklama

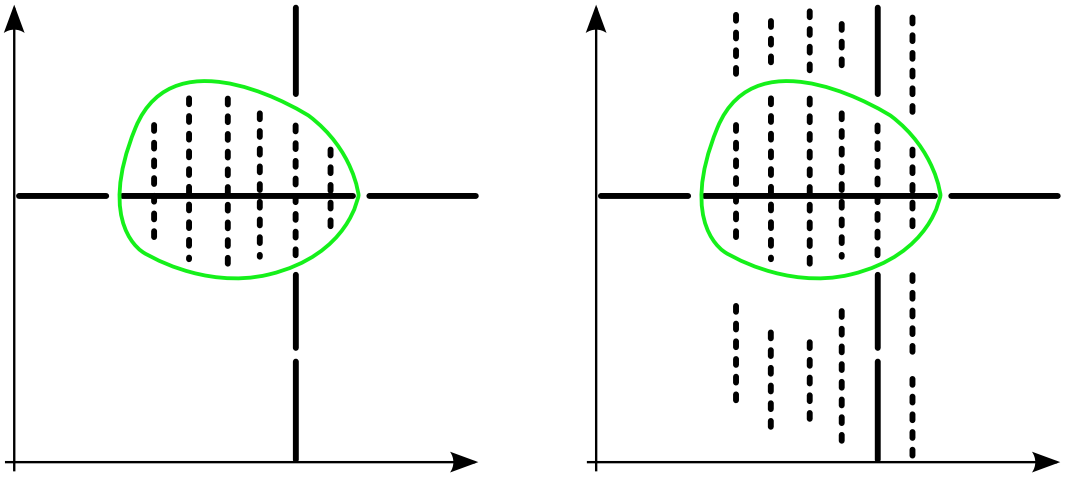
YSA'nın doğru olarak sınıflandırdığı bir nokta al. Bu noktadan geçen koordinat eksenlerine paralel doğrular üzerinde rastgele noktalar al. Bu noktalar ile her doğru üzerinde ayrı ayrı GNG çalıştır. GNG ile birbirinden kopan parçaları bul.



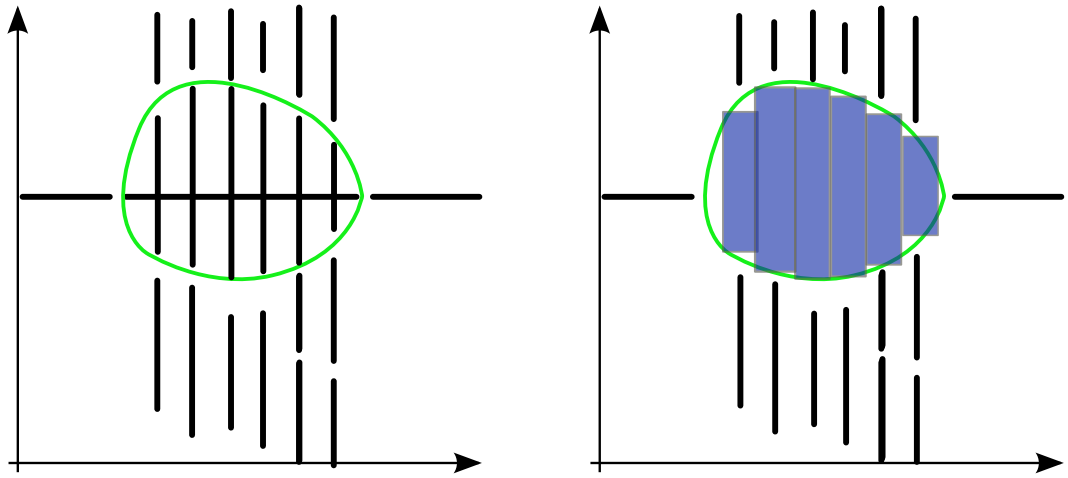
Bu parçaları bölge sınır değerlerine yaklaşacak şekilde genişlet ve uzunluk öncelik kuyruğuna at. Uzunluk öncelik kuyruğundaki ilk elemanı al.



Seçilen parçayı eşit parçalara böl. Bölünen parçalar başlangıç noktası olarak kabul edilerek diğer boyut veya boyutlarda GNG çalıştır.



Bulunan parçaları genişlet. Başlangıç parçasıyla kesişenleri kural çıkarmak için ayır. Diğerlerini uzunluk öncelik kuyruğuna at. Durma kriteri gerçekleşene kadar öncelik kuyruğundaki parçalarla devam et.



3.7.3 Kullanılan parametreler

Geliştirilen yöntem için 3 parametre belirlenmiştir. Bunlardan ikisi algoritma çalışması sırasında bir sınıf bölgesine ait parçanın kural oluşturma işlemi için kullanılırken hangi oranda parçalara bölüneceğini ve en küçük parça yüzdesini ifade etmektedir. Bu parametreler yüzde ile ifade edilip eğitim verisinin girdilerinin her bir boyutu için aldığı en yüksek ve en düşük değer aralığının uzunluğuna oranını ifade eder.

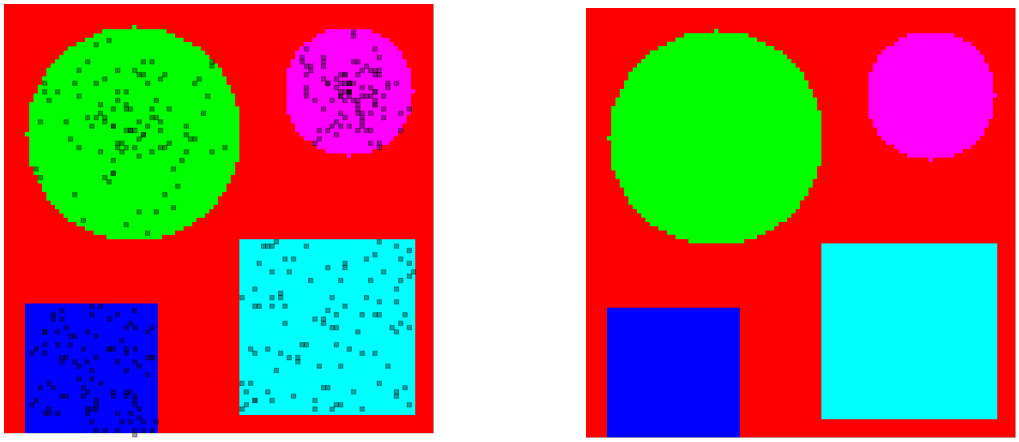
Son parametre, algoritmanın sonlanma kriteridir. Algoritmanın o ana kadar çıkardığı kuralların eğitim verilerinin yüzdesel olarak ne kadarını kapsadığında durması gerektiğini ifade eder.

3.7.4 Geliştirilen ilk yöntem İçin yapılan deneyler ve sonuçları

Önerilen yöntemin uygulandığı problemlerden ilki için veriler bilgisayar ortamında üretilmiştir. İlk deney için parametreleri belirlenen daireler ve kareler içerisinden rastgele noktalar alınmış ve böylece eğitim veri seti oluşturulmuştur. İkinci deney için Fisher'ın Iris veri seti kullanılmıştır. Veriler kullanılmadan önce normalizasyon işleminden geçirilmiştir.

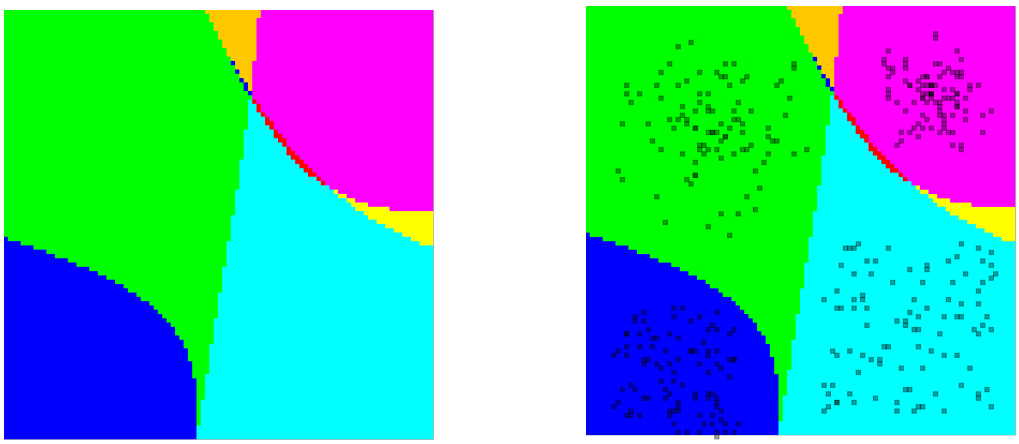
Daireler ve Kareler :

Bu problem için R^2'de $[0, 1]$ aralığında değerlerden oluşan (x, y) noktaları 2 adet daire ve 2 adet kare alandan seçilmiştir(Şekil 3.11). Her sınıftan rastgele 100 tane nokta alınarak 400 noktadan oluşan eğitim seti üretilmiştir. Ayrıca yine aynı sayıda nokta içeren doğrulama veri seti de üretilmiştir.



Şekil 3.11: Eğitim Verileri ve Ait Olduğu Sınıf Bölgeleri

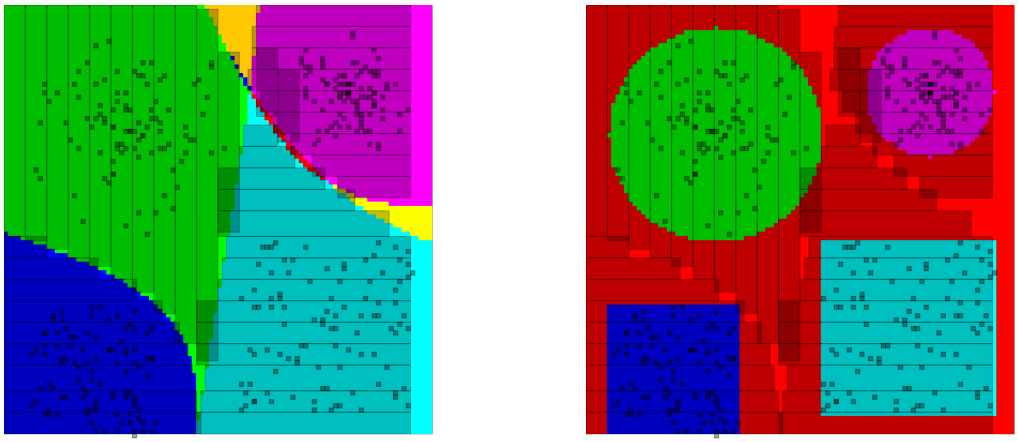
Bu veriler ile 0.01 hata oranına sahip bir MLP eğitilmiştir. Bu problemin nispeten boyut ve veri sayısı olarak basit olması nedeniyle YSA'nın gerçekte nasıl kural bölgeleri oluşturduğu hesaplanabilmiştir. Bu hesaplama için 0.01 adım boyutu kullanılarak 1000 tane nokta, eğitilmiş YSA tarafından sınıflandırılmış ve bu sonuçlar Şekil 3.12'de görselleştirilmiştir



Şekil 3.12: Eğitilmiş YSA'nın oluşturduğu kural bölgeleri ve eğitim verileriyle görselleştirilmesi

Son aşama olarak önerilen kural çıkarma algoritması 0.15 bölme yüzdesi 0.50 en düşük parça yüzdesi ve 0.99 kapsama oranı ile çalıştırılmıştır. Çıkarılan 45 kural ile %99.5 oranında doğru sınıflandırma başarısı sağlanmıştır. Bu kuralların YSA

tarafından gerçekte temsil ettiği kural bölgelerini ve gerçek sınıf bölgelerini nasıl kapsadığı Şekil 3.13'de gösterilmiştir.



Şekil 3.13: Çıkarılan kuralların YSA'nın oluşturduğu kural bölgelerini kapsaması

Iris Veri Seti[3] **:**

Bu sınıflandırma problemi veri setinde 3 sınıf ve her sınıf için 50 veri bulunmaktadır.

Bu problem için 0.01 hata oranına ulaşan bir YSA eğitilmiştir. Bu eğitilmiş YSA için önerilen kural çıkarma algoritması 0.25 bölme oranı, 0.50 en küçük parça boyutu ve 0.90 eğitim verisi kapsama durma kriteri ile eğitilmiştir. Elde edilen kural kümesi 48 kural içermektedir ve eğitim verisinin %82'sini başarılı olarak sınıflandırmıştır.

[3] **http://archive.ics.uci.edu/ml/datasets/Iris**

4 YSA ÜZERİNDEN KURAL ÇIKARAN BAĞLANTICI MELEZ ZEKİ SİSTEM

Önerilen sistemin geliştirilmesinde eğitimsel bir yaklaşım izlenmiştir. Bu yöntem sınıflandırma yapması için eğitilmiş YSA'larda kullanılabilir. Ayrıca bu amaçla oluşturulmuş başka kapalı sistemlere de uyarlanabilir. Sınıflandırma problemleri için eğitilen YSA'dan kural çıkarma problemine *topoloji öğrenme problemi* olarak yaklaşılmıştır. Yöntem kısaca, YSA'nın oluşturduğu sınıflandırma bölgelerinin bulunup, kurallara dönüştürülmesi süreci olarak özetlenebilir. Bu amaçla topolojilerin bulunmasında bağlantıcı yaklaşım (GNG algoritması) kullanılmıştır. Genetik algoritmalar ile GNG parametreleri eniyileştirmesi yapılarak iki yöntemi birlikte kullananan melez bir yapı oluşturulmuştur .

Yöntemin çıkardığı kurallar hiperdikdörtgenler (Tanım 4.1 ve 4.2) şeklindedir.

Tanım 4.1. Hiperdikdörtgen (Hyperrectangle veya orthotope), dikdörtgenin yüksek boyutlar için genelleştirilmiş halidir. Aralıkların, kartezyen çarpımı şeklinde ifade edilir.

Tanım 4.2. Hiperdikdörtgen Kural (HK) : m boyutlu bir hiperdikdörtgen (HR) eğer kapsadığı tüm noktalar YSA tarafından aynı sınıfa atanıyor ise bir kurala karşılık gelir.

$$HK = \{x_i| \wedge a_j \leq x_{ij} \leq b_j \rightarrow S_k, 1 \leq j \leq m\}$$

Bu tanımda $\mathbf{x}_i = [x_{i1}, \ldots, x_{im}]$ girdi vektörünü ve a_j ile b_j alt ve üst sınırları ifade eder. S_k sınıf etiketine ve m ise girdi uzayının boyutuna karşılık gelir.

Problemlerde kullanılan ağların yapısı 3 katmanlı ileri beslemeli türünde ağlardır. Eğitim için bu ağ yapısına uygun herhangi bir eğitim algoritması kullanılabilir. Geri yayılım ve bunun türevleri olan çeşitli eğitim algoritmaları vardır. İstenilen bir öğrenme algoritması kullanılabilir.

3 katmanlı ağ seçilmesinin nedeni Kolmogorov teoremi ve bunun YSA'lara uyarlanmasıyla ilgilidir. Birden fazla gizli katmana sahip olan ileri beslemeli ağların bazı fonksiyonları temsil edemediği bilinmektedir. Ancak tek gizli katman, bir girdi ve bir çıktı katmanına sahip ağlar Teorem 3.4'e göre her türlü fonksiyonu temsil edebilirler.

Geliştirilen kural çıkarma algoritması öncelikli olarak sınıflandırma problemleri üzerine uygulanmıştır. Diğer problem türleri (fonksiyon tahminleme, veri işleme ve diğ.) bu tez kapsamında incelenmemiştir.

YSA teorik olarak her türlü veri üzerinde çalışabilmekle birlikte pratikte daha hızlı ve iyi bir eğitim için başta normalizasyon olmak üzere veri üzerinde bazı önişlemler yapılması yararlı olmaktadır. Eksik verilerin eğitim kümesinden çıkarılması da yararlı olacaktır.

Sınıflandırma problemi için eğitim verisinde beklenen çıktıların kodlanması için ikili kodlama (binary encoding) kullanılmıştır. Çıktı katman eleman sayısı bu kodlamaya bağlıdır. İkili kodlamanın beklenen sınıf sayısını temsil edecek aralıkta olması yeterli olmakla birlikte, bu aralığın arttırılmasının YSA öğrenmesine ve çıkarılan kurallara etkisi bu tez kapsamında incelenmemiştir.

YSA eğitimlerinde hedeflenen hata oranı ile genelleme özelliği arasında doğrusal olmayan bir ilişki vardır. Pratikte bu hata oranları genel olarak $0.1, 0.05$ ve 0.01 değerleri olarak kullanılır. Hedeflenen hata oranı ile eğitim sonucu YSA'nın genelleme yeteneği arasındaki ilişki bu tezin kapsamı dışındadır.

Bir sınıflandırma problemini çözen ileri beslemeli YSA eğitim sırasında sinir hücresi elemanlarının bağlantı ağırlıklarını eğitim verisine göre değiştirerek (uyum sağlayarak) girdi değerleri için sınıfları ayıran bölgeler oluşturur. Eğitilmiş ağa bir girdi verisi verilip sınıflandırması istendiği zaman sinir ağı bağlantılar arası ağırlıklar, aktivasyon ve transfer fonksiyonlarını kullanarak girdinin hangi çıktıya karşılık geldiğini tahmin eder. Bir diğer deyişle oluşturduğu sınıf bölgelerinden hangisine dahil olduğuna karar verir.

Eğitimsel kural çıkarma yaklaşımı doğrultusunda eğitilmiş YSA bir tahminleyici olarak kullanılır ve girdi verilerine karşılık alınan çıktı değerlerini kullanarak sınıflandırma için YSA'nın temsil ettiği kurallar oluşturulur. Yapılan deneylerde literatürde birçok algoritmanın başarısının test edilmesi için kullanılan standartlaşmış problem verileri kullanılmıştır.

Önerilen kural çıkarma yöntemi bir sonraki Bölüm 4.1'den başlayarak detaylı olarak anlatılmıştır.

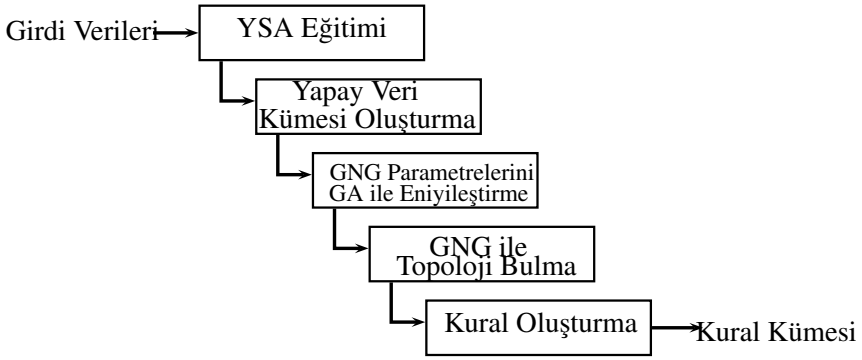
4.1 Önerilen Kural Çıkarma Yöntemi

Önerilen yöntem birbirini takip eden bir dizi aşamadan oluşmaktadır. Problemi çözen YSA'nın eğitilmesiyle başlayan ve sonuçta kural kümesi çıktısına kadar geçen aşamalar sonraki bölümlerde detaylarıyla anlatılmıştır.

Önerilen yöntem beş temel adımdan oluşmaktadır (Tablo 4.1). Her aşamanın çıktısı bir sonraki aşamada girdi olarak kullanılmaktadır (Şekil 4.1).

Tablo 4.1: Önerilen yöntemin aşamaları

1. Aşama	Bir YSA'yı istenilen hata oranına erişene kadar eğitme.
2. Aşama	Eğitilmiş YSA'yı tahminleyici olarak kullanarak yapay veri kümesi üretme.
3. Aşama	Problem için GNG parametrelerini GA kullanarak eniyileştirme.
4. Aşama	GNG ile verilerin topolojisini bulma.
5. Aşama	GNG'nin ürettiği çizgeyi kullanarak kurallar oluşturma.



Şekil 4.1: Önerilen Yöntemin İşleyiş Süreci

4.2 YSA Eğitimi

Öğrenme, bir hipotezi bilinen örnekler üzerinden tanılama ve genelleme olarak tanımlanabilir. Eğiticili YSA öğrenmesi bu tarzda bir öğrenme sağlar. Eğiticili öğrenmede önceden sınıflandırılmış eğitim verileri üzerinde öğrenme gerçekleşir. Eğitim verisindeki her bir örnek belli sayıdaki özellik için değerler içerir ve her bir örnek için doğru çıktı değerini içerir. Amaç bu eğitim verileri üzerinde bir açıklama oluşturacak YSA'yı oluşturmak ve daha önce görülmeyen örnekler üzerinde yüksek doğruluk derecesinde sonuçlar vermesidir.

YSA eğitimi için ağın sahip olduğu eleman sayısı da öğrenmeyi etkiler. Gereğinden daha az elemana sahip ağlar eğitim verisini temsil edebilecek

kapasitede olmayabilir veya gereğinden daha fazla elemana sahip ağlar genelleme özelliğini gösteremeyebilirler.

Eğiticili öğrenmede durma koşulu hatanın hedeflenen değere ulaşmasıyla önceden belirlenir. Hedef hata oranının yüksek tutulması öğrenmenin gerçekleşmemesine neden olur. Çok düşük tutulması da ağın veriyi ezberlemesi durumunun ortaya çıkarır. Bu da genelleme yapamayan yani daha önce görmediği örnekleri düşük doğruluk oranında sınıflandırması anlamına gelir. Pratikte bu iki durumu dengede tutacak oranlar genellikle $0.10, 0.05, 0.01$ şeklindedir. Bu bir kural olmamakla beraber pratikte işe yarayan oranlardır.

Eğiticili öğrenme için standartlaşmış geriye yayılım algoritması sıklıkla tercih edilir. Bunun yanında bu algoritmayla aynı mantıkta daha hızlı algoritmalar da mevcuttur. Bu tez kapsamında, tercih edilecek eğitim algoritmasıyla ilgili bir kısıt bulunmamaktadır. İstenilen hedef hata oranına ulaşılmasını sağlayacak herhangi bir yöntem yeterlidir.

Öğrenme sürecini etkileyebilecek etkenlerden biri de girdi verilerinin ön işlemden geçirilmesidir. Teorik olarak YSA'nın aldığı girdilerin her türlü sayısal değere sahip olması mümkündür. Buna karşın örneğin normalizasyon yapılmış veriler ile eğitim daha başarılı olabilmektedir. Bu tez kapsamında eğitim sırasında kullanılan veriler üzerinde normalizasyon işlemi başta olmak üzere gerekli görülen yerlerde önişlem uygulanmıştır.

Tek gizli katman, bir girdi ve bir çıktı katmanına sahip ağlar Teorem 3.4'e göre her türlü fonksiyonu temsil edebilirler. Bu nedenle tez kapsamında eğitilen ağlar bu yapıdadır.

Tez kapsamında YSA oluşturmak ve eğitmek için Java[4] ortamındaki Encog[5] yazılım kütüphanesi kullanılmıştır.

4.3 Yapay Veri Kümesi Üretme

Eğitilen YSA, sınıflandırma işlemi yapacak hale geldiği için üretilecek her türlü girdi için bir sınıf etiketini cevap olarak verecektir. Bu sayede YSA'nın daha önce görmediği rastgele veriler üretip bu veriler için sınıflandırma yapması istenebilir. Oluşturulan yapay veri kümesi sadece YSA'nın cevaplarından oluşur ve

[4]http://www.java.com/

[5]https://code.google.com/p/encog-java/

eğitim verilerini içermez. Bu yapay veri kümesiyle hedeflenen YSA'nın ürettiği sonuçlar ile bu YSA'yı temsil edebilecek kurallar oluşturmaktır. Bu kuralların öncelikli amacı YSA'yı yansıtmaktır. YSA'nın çözdüğü probleme daha iyi veya yeni bir çözüm önermek değildir. Örneğin; YSA'nın A sınıfına atadığı bir veri gerçekte B sınıfına ait olabilir. Ancak YSA'nın verdiği cevap mutlak doğru olarak kabul edilir ve A sınıfı bilgisi kullanılır. Yapay veriler ile kural bölgelerinin bulunmasında kolaylık sağlanması amaçlanmıştır.

Bu adımda üretilen veri kümesi, kural çıkarma sürecinde olumlu etkiye sahip olabileceği gibi, örneğin eğitim verilerinin bulunmadığı değer aralıklarından üretilen verilerden oluşan yapay veri kümeleri olumsuz etkide bulunabilirler. YSA'lar her türlü girdiye karşılık bir çıktı üretmesine karşın pratikte daha önce görmediği değer aralıkları için tahminlemede bulunması sağlıklı sonuçlar vermez. Bu nedenle bu adımı, önerilen yöntemde seçimlik bir adım olarak belirtmek daha doğru olur.

4.4 Genetik Algoritma Kullanarak GNG Parametrelerini Eniyileştirme

GNG topoloji öğrenme konusundan başarılı olmakla beraber başarısı sahip olduğu parametrelerin uygun değerlere sahip olmasına da bağlıdır. Bu değerler algoritmanın uygulandığı verinin karakteristiğine göre farklılık gösterebilir. Parametrelerin uygun aralıklarda kullanılması için genetik algoritmalar kullanılmıştır.

Her parametre için belirlenen değerlerin hangi kombinasyonunun iyi bir sonuç verdiğinin bulunması için bir eniyileştirme yöntemi olan genetik algoritmalar şu şekilde kullanılmıştır:

Kromozomların oluşturulması: Her parametre için pratikte kullanılabilecek uygun değerler sıralı bir şekilde dizilerde tutulur. Her parametre için bu değerlerin dizideki konumu ikili ifade şeklinde kodlanır. Örneğin; 4 değer içeren dizi için kodlama şu şekilde olur: 00,01,10,10 . Her biri 4 değer içeren dizilere sahip 3 parametre için örnek bir kromozom şu şekilde olur : 10|01|11

Tez kapsamında yapılan deneylerde GNG parametrelerinin herbiri için 8 değer belirlenerek yukarıda anlatıldığı gibi kodlama yapılmış ve GA için kromozomlar oluşturulmuştur. GNG parametrelerinin açıklamaları ve seçilen değerler Tablo 4.2'de gösterilmiştir.

Tablo 4.2: GNG parametrelerinin açıklamaları ve seçilen değerler

Parametre	**Açıklama**	**Değerler**
α	İki düğüm arasına yeni bir düğüm eklenirken bu iki düğümün hatalarının güncellenmesinde kullanılan katsayı	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8
β	Tüm düğümlerin hata değerini azaltırken kullanılan katsayı	0.0001, 0.0003, 0.0005, 0.0007, 0.0009, 0.0010, 0.0012, 0.0015
e_w	Girdiye karşılık en yakın uzaklığa sahip s düğümünün ağırlığının güncellenme oranı	0.01, 0.03, 0.05, 0.07, 0.09, 0.10, 0.12, 0.15
e_n	s düğümünün komşularının ağırlıklarının güncellenme oranı	0.0001, 0.0003, 0.0005, 0.0007, 0.0009, 0.0010, 0.0012, 0.0015
a_{max}	Bir kenarın silinmesi için gereken eşik yaş değeri	10, 20, 30, 40, 50, 60, 70, 80
λ	Yeni düğüm eklemek için döngü sayısının katı olması gereken değer	100, 200, 300, 400, 500, 600, 700, 800
$node_{max}$	Ağın sahip olabileceği en fazla düğüm sayısı	10, 15, 20, 25, 30, 50, 80, 100

Uygunluk fonksiyonu: GNG parametrelerinin uygunluğunu ölçmek için kural çıkarma basamağı uygulanır. Uygunluk fonksiyonunda çıkarılan kural sayısı ve kural kümesi doğruluğu uygunluk değerini hesaplamak için kullanılır. Uygunluk fonksiyonunun genel ifadesi Tablo 4.3'te gösterilmektedir.

Tablo 4.3: Kullanılan uygunluk fonksiyonunun genel ifadesi

$$f(s, d) = k_1 s + k_2 d$$

Bu uygunluk fonksiyonundaki katsayılar k_1 ve k_2, kural sayısı (s) ve kural doğruluğunun (d) önemini ifade eden katsayılardır. Bu tez kapsamında bu katsayıların değerleri $k_1 = 0.5$ ve $k_2 = 0.5$ olarak kullanılmıştır.

Bu aşamada kullanılacak verinin seçimi tüm eğitim verisini kullanmak şeklinde olabileceği gibi verinin belli bir yüzdesini kullanmak şeklinde de olabilir.

Elde edilen en iyi parametre kombinasyonu bir sonraki aşamada GNG'nin

tüm veriler için çalıştırılmasında kullanılır.

GA yazılım kütüphanesi olarak Jenetics[6] kullanılmıştır.

4.5 GNG ile Topoloji Öğrenme

GNG, detayları Bölüm 3.4'de verildiği gibi topoloji öğrenme konusunda başarılı bir yöntemdir. Topoloji öğrenme dışında, kümeleme ve boyut indirgeme amaçlı da kullanılabilir. Önerilen yöntemde elde bulunan veri kümesinin topolojik yapısını bulmak ve bu sayede kurallar oluşturabilmektir.

Bu sayede elimizdeki verilerin birbirleriyle olan Öklid uzaklığına bağlı komşuluk ilişkileri ile verileri temsil edecek topolojinin bulunması sağlanmış olur. Bulunan topolojik yapı YSA'nın temsil ettiği kural bölgelerinin bir nevi iskeleti olarak düşünülebilir. Oluşturulan kurallar ile bu iskelet etrafı doldurularak kural bölgelerinin en iyi şekilde temsil edilmesi sağlanır.

YSA'dan veya başka kapalı sistemlerden kural çıkarma yöntemlerinde topoloji öğrenme yaklaşımına sahip başka bir yöntem veya algoritmaya bu araştırma süresince literatürde başka bir çalışmada rastlanmamıştır.

GNG sayesinde veri kümesini en iyi temsil eden topolojiyi temsil eden çizge oluşmuş olur. Bu çizge bir sonraki aşamada kural çıkarma işleminde kullanılacaktır.

GNG algoritması için kullanılan parametreler bu topolojinin ne kadar iyi öğrenildiğine etki eder. Özellikle GNG ile oluşan çizgenin en fazla kaç düğümden oluşacağı sayısı önemlidir. Gereğinden az olması topolojinin eksik veya yanlış öğrenilmesine neden olur. Ayrıca bu düğüm sayısı bir sonraki aşamadaki oluşturulacak kural kümesinin büyüklüğünü de doğrudan etkiler.

Tez sürecinde GNG algoritmasının Scala[7] programlama dilinde gerçekleştirimi yapılmıştır.

Bir çizge (V, E), köşeleri $V = \{v_1, v_2, \ldots\}$ ve kenarlarıyla $E = \{\{v_i, v_j\}, \{v_k, v_l\}, \ldots\}$ tanımlanır. Bir çizge aynı zamanda çizge topolojisi denilen bir doğal topolojidir. Her kenarı $\{v_i, v_j\}$ birim aralıkta $I = [0, 1]$ tanımlayıp bu kenarları uygun köşelere birleştirerek oluşturulur.

[6] http://jenetics.sourceforge.net/
[7] http://www.scala-lang.org/

Bu yapı kolaylıkla simpleksler kompleksi çerçevesinde ifade edilebilir. Bir simpleksler kompleksi şu şekilde oluşturulabilir : $G = \{\{v\} \mid v \in V\} \cup E$. Çizgenin istenilen topolojik gerçekleştirimi ise G'nin geometrik gerçekleştirimi olan $|G|$'dir.

4.6 Kural Kümesi Oluşturma

GNG ile veri kümesinin topolojisini temsil eden çizge bulunduktan sonra bu çizge kural kümesini oluşturmak için kullanılır. Kural kümesi oluşturma algoritmasının detayları Algoritma 4.1'de verilmiştir. Algoritmanın sözde kodu Algoritma 4.2'de gösterilmiştir.

Öncelik kuyruğu sayesinde en çok bağlantılı düğümler öncelikli olarak kurallara çevrilerek, genelden özele gidecek şekilde bir kural çıkarma işlemi yapılması hedeflenmiştir.

Kural çıkarma süreci öncelik kuyruğundaki elemanlar bitene kadar devam eder. Kuyruktan alınan düğüm kendisi, kendisine bağlı olan tüm düğümleri ve bu düğümlere bağlı veri kümesinden gelen düğümleri kapsayacak şekilde bir hiperdikdörtgen oluşturulur.

Kendisi de dahil olmak üzere kendisine bağlı tüm düğümler YSA tarafından aynı sınıf etiketine atanırlar. Bu durum çizge üzerinde yapılan budama işlemi sonrasında sağlanmış olur.

Oluşturulan hiperdikdörtgen sınıf etiketi atandıktan sonra kural kümesine kural olarak eklenir. Daha sonra bu son eklenen kuralın kapsadığı tüm bağlantılar bulunur ve bu bağlantılar silinir. Bağlantı silinmesi bağlantıyı sağlayan düğümlerin de silinmesi anlamına gelmez. Bir bağlantının kural tarafından kapsanıyor olması demek bağlantıyı oluşturan her iki düğümün de hiperdikdörtgen kural tarafından kapsanıyor olması demektir.

Oluşturulan kuralların sayısı GNG ile oluşturulan çizgedeki düğüm sayısından daha fazla olmaz. Bu sayede kural kümesi büyüklüğü de kontrol edilebilmektedir.

Algoritma 4.1 Kural Çıkarma Algoritması

1. Çizge üzerindeki her bir düğüm için kendine bağlı olan düğüm sayısına göre sıralı bir öncelik kuyruğu oluşturulur. A_{v_i}, v_i köşesini içeren kenarların sayısı olsun. $A_{v_i} \geq A_{v_{i+1}}$ olacak şekilde öncelik kuyruğu $Q(V)$ oluşturulur.

2. Veri kümesindeki tüm örnekler $\forall s_j \in K$ için, G çizgesinde kendisine Öklid uzaklığı olarak en yakın düğüme $v_m \in V \mid d_{min} = uzaklık(s_j, v_m)$ bağlanır $P = \{\{s_j, v_m\}, \ldots\}$, $V^* = V \cup K$ ve $E^* = E \cup P$. Yeni çizgeyi $G^* = (V^*, E^*)$ olarak gösterelim.

3. Çizge üzerinde bağlantı (kenar) budama işlemi yapılır. Bu budama işlemi e_k^* bağlantısını sağlayan iki düğümün eğitilmiş YSA tarafından aynı sınıf etiketine $e_k^* = \{v_t, v_p\} \in E^* \mid f_{YSA}(v_t) \neq f_{YSA}(v_p)$ atanmaması durumunda bu bağlantının silinmesi şeklinde $E^* - \{e_k^*\}$ yapılır.

4. Öncelik kuyruğundaki sıradaki düğüm alınır v^q. Eğer en az bir tane $\{s_j, v^q\} \in E^*$ koşulunu sağlayan $s_j \in K$ varsa devam edilir. Yok ise bu adım tekrarlanır.

5. v^q düğümünün G'deki komşuları $N_{v^q}^G = \{\{v_k\} | v_k \in G \wedge \exists\{v_q, v_k\} \in G\}$, komşularının veri kümesindeki komşuları $\forall v_k, N_{v_k}^K = \{\{s_m\} | s_m \in K \wedge \exists\{s_m, v_k\} \in G^*\}$ ve kendisine ait veri kümesindeki komşuları $N_{v^q}^K = \{\{s_r\} | s_r \in K \wedge \exists\{s_r, v^q\} \in G^*\}$ bulunur. $N_{v^q} = N_{v^q}^G \cup N_{v_k}^K \cup N_{v^q}^K$

6. N_{v^q}'daki tüm düğümleri kapsayacak hiperdikdörtgen $HR_{v^q} = \{\forall x_i | \wedge a_j \leq x_{ij} \leq b_j, 1 \leq i \leq z \wedge 1 \leq j \leq t\}$ oluşturulur. x_i, v_i'nin koordinat vektörü ve $v_i \in N_{v^q}$.

7. Hiperdikdörtgenin sınıf etiketi $YSA(x^q) \rightarrow S_q$ atanarak hiperdikdörtgen kural haline getirilip kural kümesine eklenir.

8. G'deki kenarlardan çıkarılan son kural tarafından kapsananlar $e_k = \{v_t, v_p\} \in E \mid x_t \in HR_{v^q} \wedge x_p \in HR_{v^q}$ silinir (x_t ve x_p, v_t ile v_p koordinat vektörlerini ifade eder). K veri kümesinden gelen örnek düğümlerinden kural tarafından kapsananlar $\forall s_j \in K \wedge s_j \in HR_{v^q}$, G^*'dan silinir.

9. Öncelik kuyruğu boşalana kadar 4. adımdan devam edilir.

Algoritma 4.2 Kural Çıkarma Algoritması Sözde Kodu

```
1:  oncelikKuyrugu ← OncelikKuyruguOlustur(cizge)          ▷ Algoritma Adım 1
2:  CizgeyeDugumleriEkle(cizge,veriKumesi)                 ▷ Algoritma Adım 2
3:  BaglantilariBuda(cizge,ysa)                            ▷ Algoritma Adım 3
4:  while !oncelikKuyrugu.isEmpty() do
5:      dugum ← oncelikKuyrugu.dequeue()                   ▷ Algoritma Adım 4
6:      yeniKural ← HiperDikdortgenOlustur(dugum)          ▷ Algoritma Adım 5 ve 6
7:      KuralSinifEtiketiAta(yeniKural,ysa,dugum)          ▷ Algoritma Adım 7
8:      kuralKumesi ← kuralKumesi + yeniKural              ▷ Algoritma Adım 7 (devam)
9:      DugumVeBaglantiBuda(cizge,yeniKural)               ▷ Algoritma Adım 8
10: end while
```

4.7 Yöntemin Kısıtları

Yöntemin sınıflandırma problemlerini çözen YSA'lar için geliştirilmiştir. Bunun dışındaki problemleri (örneğin; regresyon) çözen YSA'lar için uygulanabilirliği denenmemiştir.

Girdi vektöründe bulunan değerlerin sürekli değerler olması gerekmektedir. Kesikli veya nominal değerler üzerinde topoloji öğrenme yaklaşımının uygun olup olmadığı denenmemiştir. Literatürde kesikli veriler ve nominal değerleri içeren girdilerle kural çıkarmak için değişik yaklaşımlar vardır. Örneğin nominal değerlerin ikili değerlere çevrilerek YSA eğitimi yapılması sonucu bu ikili girdiler bir karar ağacı oluşturmak için doğrudan kullanılabilir. Bir başka deyişle ikili değerler karar bölgelerini ikiye ayırır ve bu ayrık iki bölge ayrı ayrı topoloji öğrenme yaklaşımı ile kurallar çıkarılabilir. Sonuçta çıkan kurallar bir karar ağacı aracılığıyla kullanılabilir.

4.8 Bölüm Özeti

Bu bölümde beş aşamadan oluşan kural çıkarma yönteminin aşamaları hakkında bilgi verilmiştir. Yöntem bu aşamaların birbirini takip edecek sırada işletilmesi sonucu kural kümesi oluşturan bir yapıdadır.

5 DENEYSEL ÇALIŞMALAR

Geliştirilen yöntemin pratikteki başarısının ölçülmesi için çeşitli deney verileri üzerinde performans ölçümleri yapılmıştır. Bu performans ölçümlerinde k-fold cross validation yaklaşımı kullanılmıştır. Ölçüm değerleri olarak kural sayısı, kural kümesi doğruluğu ve kural kümesinin aslına uygunluk kriterleri kullanılmıştır.

5.1 Değerlendirme Kriterleri

Öncelikli değerlendirme kriteri tahminleme doğruluğudur (predictive accuracy). Kural kümesinin daha önce görmediği verileri ne kadarını doğru sınıflandırdığı anlamına gelir.

Ayrıca bir kural çıkarma yöntemi şu başlıklarda da değerlendirilebilir:(Andrews et al., 1995)

- Anlaşılabilirlik : Çıkarılan kuralların insan tarafından anlaşılır olması.
- Aslına Uygunluk : Çıkarılan kuralların çıkarıldığı ağı tam olarak modelleyebilmesi.
- Ölçeklenebilirlik (Scalability): Çok boyutlu girdi uzayına, çok sayıda işlem elemanına ve bağlantıya sahip ağlara ölçeklenebilir olmalıdır.
- Genellik (Generality): Ağ eğitimi ve mimarisi hakkında kısıtlar olmamalıdır.

Anlaşılabilirlik kriterini ölçmek sorunlu bir konudur. Kuralların gösterim şekline bağlı olduğu gibi kural sayısına da bağlı olabilen bir kriterdir. Literatürde bu kriter için ölçüm yapılmasını sağlayacak bir yöntem bulunamamıştır. Bu kriter öznel bir kriter olma karakteristiği ağır basmakta olduğundan tez kapsamında bu açıdan değerlendirme yapılmamıştır.

Aslına uygunluk için deneylerde, çıkarılan kural kümeleri için hem doğruluk hem de aslına uygunluk ayrı ayrı ölçülmüştür. Doğruluk, kural kümesinin test veri kümesini yüzde olarak ne kadar doğru sınıflandırdığını göstermektedir. Buna karşın, aslına uygunluk ise kural kümesinin yüzde olarak YSA ile aynı girdilere karşılık verdiği aynı sonuçların oranını göstermektedir. Ölçümler için kullanılan test veri kümelerinin önemli iki özelliği vardır:

1. Kural kümesi çıkarılmasında veya sinir ağı eğitimi sırasında kullanılmayan örneklerdir.

2. Bu veriler gerçek veri kümesinden ayrılmıştır, sonradan üretilmemiştir.

Birincisi aslına uygunluk tahmininin yansız (unbiased) olduğu anlamına gelir. Diğer özellik ise ölçümlerin gerçekte problem alanında olan verilere dayanılarak yapıldığını gösterir.

Ölçeklenebilirlik doğrudan ölçülmemekle beraber, bu kriter farklı alanlardaki problemlere uygulanarak dolaylı yoldan ölçülmüştür. Önerilen kural çıkarma yönteminde YSA'nın iç yapısını dikkate almadan kurallar çıkarıldığı için çözümlemeli yaklaşımdaki gibi ağın eleman sayısı veya büyüklüğünün doğrudan etkisi olmadığı söylenebilir. Probleme ait veri kümesinin boyutundan bağımsız şekilde sürekli değerlere sahip her türlü veriler için sınıflandırma kuralları çıkarabilen bir yöntemdir.

Önerilen yöntem sınıflandırma problemlerini çözen YSA'lar (veya başka kapalı sistemler) için geliştirilmiş olması dışında mimari ve eğitim yöntemi açısından bir kısıtı yoktur. Buna karşın sınıflandırma problemlerinin çözümünde çok katmanlı algılayıcılar (MLP) standartlaşmış olarak kullanılan mimaridir. Bu nedenle deneylere kullanılan YSA'lar MLP türündendir.

Yapılan performans ölçümleri ile ilgili formuller şu şekildedir :

n : Örnek sayısı

$n_{YSADogru}$: YSA'nın doğru olarak sınıflandırdığı veri sayısı

$n_{KuralKapsanan}$: Kural kümesi tarafından kapsanan veri sayısı

$n_{KuralDogru}$: Kural kümesi tarafından doğru sınıflandırılan veri sayısı

$n_{AslınaUygunSonuç}$: Kural kümesi ile YSA'nın aynı sonucu verdiği veri sayısı

$$YSA\ Doğruluk\ Oranı = \frac{n_{YSADogru}}{n}$$

$$Kural\ Kümesi\ Kapsama\ Oranı = \frac{n_{KuralKapsanan}}{n}$$

$$Kural\ Kümesi\ Doğruluk\ Oranı = \frac{n_{KuralDogru}}{n_{KuralKapsanan}}$$

$$Kural\ Kümesi\ Aslına\ Uygunluk\ Oranı = \frac{n_{AslınaUygunSonuç}}{n_{KuralKapsanan}}$$

5.2 Kullanılan Test Verileri

Kullanılan veriler UCI Machine Learning Repository[8]'de bulunan çeşitli sınıflandırma problemlerine aittir.

5.2.1 Ecoli

Bu veri kümesi Ecoli'nin genetik yapısında bulunan ORF ismli bir gen ile ilgili veriler içerir. Dizilimi, diğer genlere benzerliği ve yapısal bilgi. 8 sınıfa ait 336 gözlem verisi bulunmaktadır. Her bir girdi 7 özelliğe sahiptir.

Sınıf dağılımı: cp (cytoplasm) (143), im (inner membrane without signal sequence) (77), pp (perisplasm) (52), imU (inner membrane, uncleavable signal sequence) (35), om (outer mem-brane) (20), omL (outer membrane lipoprotein) (5), imL (inner membrane lipoprotein) (2), imS (inner membrane, cleavable signal sequence) (2).

5.2.2 Iris

Bu veri kümesi çok bilinen ve kullanılan Fisher'ın Iris (zambak) veri kümesidir. Üç farklı zambak türüne ait dört boyutlu 150 desen içerir.

Sınıfların dağılımı şu şekildedir : Iris Setosa (50), Iris Versicolour (50), Iris Virginica (50)

[8]http://archive.ics.uci.edu/ml/

5.2.3 Wisconsin Diagnostic Breast Cancer (WDBC)

Wisconsin Üniversitesi Hastanesi'nde onkolojistler tarafından yapılmış çeşitli göğüs biopsisi ölçümlerini içeren veri kümesidir. Amaç, hastadan alınan bir doku örneğinin *malignant* veya *benign* olup olmadığını tespit etmektir. İki sınıf ve 10 sayısal özellikten oluşan 569 veri içerir. Sınıf dağılımı şu şekildedir: Benign (357), Malignant (212)

5.3 Önerilen Melez Yöntem İçin Yapılan Deneyler ve Sonuçları

Kullanılan veriler normalizasyon işleminden geçirilerek kullanılmıştır. Sonuç olarak çıkan kurallar da bu normalleştirilmiş verilere göre elde edilmiştir.

Deneylerin güvenilirliğini arttırmak için k-fold cross-validation yöntemi uygulanmıştır.

5.3.1 K-fold Cross-Validation

Tüm deneyler k-fold cross-validation yöntemine göre yapılmıştır. Bu yöntemde veriler k eşit parçaya bölünür. Bir parça test verisi olarak ayrılırken $k-1$ parça eğitim verisi olarak kullanılır. Tüm parçalar test verisi olarak kullanılacak şekilde deneyler k defa tekrarlanır. Çıkan sonuçlar birleştirilir.

Deneylerde tüm veriler 10 parçaya bölünerek uygulama yapılmıştır. Çıkan sonuçların ortalamaları ve standart sapma değerleri hesaplanıp tablo halinde sunulmuştur.

5.3.2 Deneysel Sonuçlar

Eğitilen 3 katmanlı ileri beslemeli yapay sinir ağları, 0.01 hata oranına ulaşılana kadar geri yayılım algoritması kullanılarak eğitilmiştir.

GA ile bulunan, probleme en uygun GNG parametreleri Tablo 5.1'de gösterilmiştir. GA aşağıda belirtilen özellikler ile çalıştırılmıştır. Her bir kromozom için hesaplanan uygunluk değeri kural kümesi oluşturma sürecini de içerdiğinden, popülasyon büyüklüğü GA'nın çalışma süresini doğrudan etkileyen bir faktördür.

Popülasyon büyüklüğü seçilirken bu durum göz önünde bulundurulmuştur.

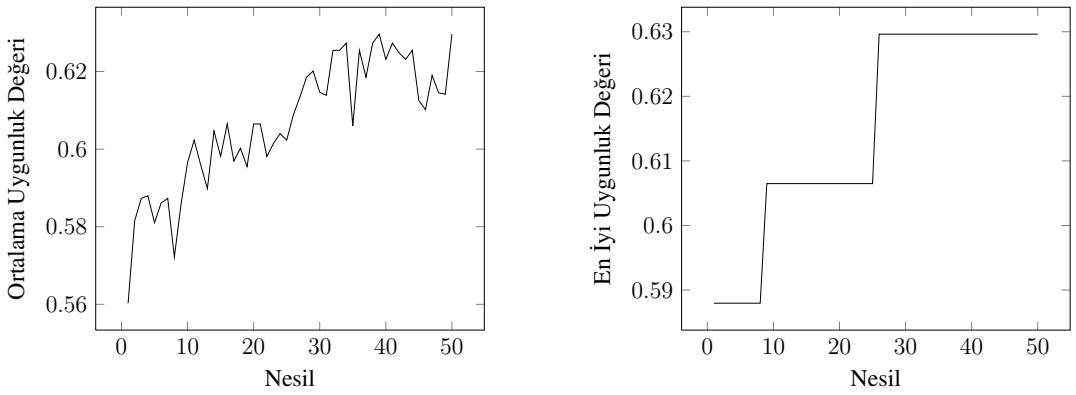
- Popülasyon büyüklüğü : 10
- Tournament Seçimi
- P_m (Mutasyon olasılığı) : 0.10
- Tek nokta çaprazlama - P_c (Çaprazlama olasılığı) : 0.15
- Nesil sayısı : 50

Deneylerde GA ile GNG parametrelerinin eniyileştirilmesi aşamasında, eğitim verisinin $1/5$'lik kısmı rastgele seçilerek kullanılmıştır.

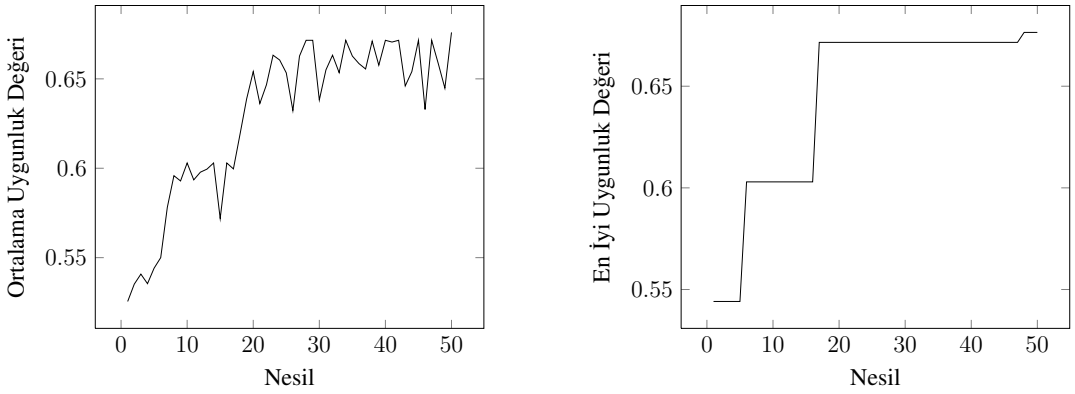
Tablo 5.1: GA ile bulunan GNG Parametreleri

	Parametreler						
Veri Kümesi	α	β	e_w	e_n	a_{max}	λ	$node_{max}$
Iris	0.1	0.001	70	100	0.4	0.0012	10
Wisconsin Breast Cancer	0.03	0.0015	20	500	0.2	0.0015	10
Ecoli	0.05	0.0012	30	400	0.6	0.0012	15

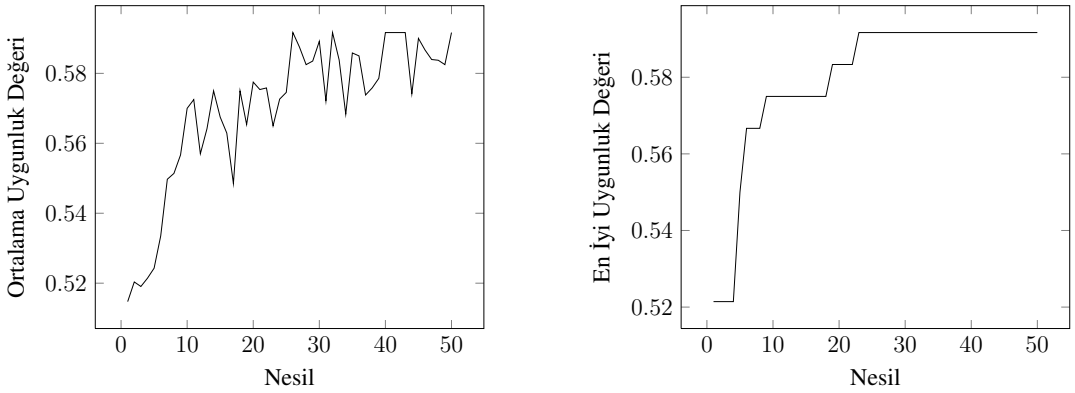
Veri setleri üzerinde GA ile bulunan GNG parametrelerinin nesiller boyunca uygunluk değerlerinin değişimi Şekil 5.1, 5.2 ve 5.3'te verilmiştir.



Şekil 5.1: Iris veri seti üzerinde GA performansı



Şekil 5.2: Wisconsin Diagnostic Breast Cancer veri seti üzerinde GA performansı



Şekil 5.3: Ecoli veri seti üzerinde GA performansı

Kural kümelerinin performans sonuçları k-fold cross validation yöntemi ile yapılmıştır. K bölme sayısını gösterir ve bu deneyler için 10 olarak kullanılmıştır. Her iki tabloda (Tablo 5.2 ve 5.3) da değerler bu 10 farklı durumun ortalamalarını ve standart sapma değerlerini içerir.

Tablo 5.2'de her problem için 10 farklı eğitim verisi üzerinde eğitilen 10 farklı YSA ve bu her YSA için çıkarılan 10 farklı kural kümesinin eğitildikleri veriler üzerindeki performans sonuçlarını göstermektedir.

Tablo 5.3'de bir önceki tablodaki aynı eğitim verileri ve eğitilmiş YSA'lar için oluşturulan aynı kural kümelerinin herbir eğitim verisi için ayrılan birbirinden farklı 10 test verisi üzerindeki performans sonuçlarını göstermektedir.

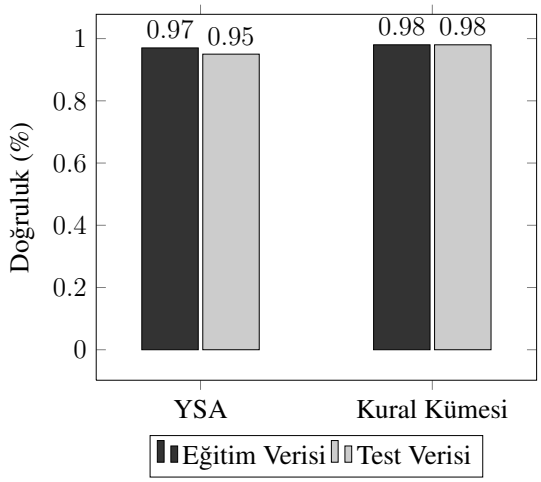
Ayrıca eğitim ve test verisi kümeleri üzerindeki doğruluk oranları Şekil 5.4, 5.5 ve 5.6'da gösterilmektedir. Deneysel çalışmalar sırasında çıkarılan örnek kural kümeleri Ek A'da verilmiştir.

Tablo 5.2: Eğitim Verileri Üzerindeki Sonuçlar

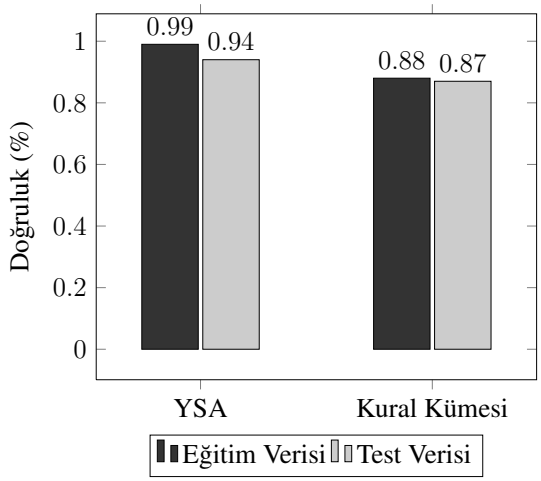
	Eğitim Verisi				
Veri Kümesi	Kural Sayısı Ortalama (St.Sap.)	Kural Kümesi Kapsama Oranı Ortalama (St.Sap.) %	Kural Kümesi Tahminleme Doğruluğu Ortalama (St.Sap.) %	YSA Tahminleme Doğruluğu Ortalama (St.Sap.) %	Kural Kümesi Aslına Uygunluk Ortalama (St.Sap.) %
Iris	3.4 (0.52)	99.11 (0.89)	97.69 (2.26)	96.74 (0.83)	97.69 (2.26)
Wisconsin Breast Cancer	3.0 (0.47)	97.27 (1.41)	87.75 (2.59)	99.04 (0.23)	87.75 (2.59)
Ecoli	5.7 (0.48)	91.58 (2.26)	94.08 (4.61)	89.77 (0.83)	94.08 (4.61)

Tablo 5.3: Test Verileri Üzerindeki Sonuçlar

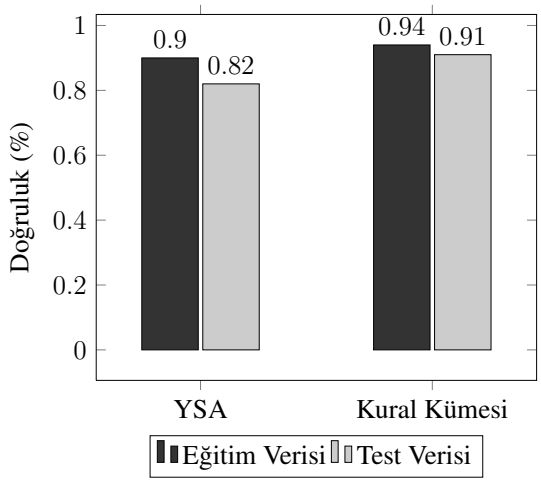
	Test Verisi				
Veri Kümesi	Kural Sayısı Ortalama (St.Sap.)	Kural Kümesi Kapsama Oranı Ortalama (St.Sap.) %	Kural Kümesi Tahminleme Doğruluğu Ortalama (St.Sap.) %	YSA Tahminleme Doğruluğu Ortalama (St.Sap.) %	Kural Kümesi Aslına Uygunluk Ortalama (St.Sap.) %
Iris	3.4 (0.52)	86.67 (7.78)	98.45 (3.43)	94.67 (5.21)	98.45 (3.43)
Wisconsin Breast Cancer	3.0 (0.47)	91.43 (2.53)	86.66 (4.70)	93.93 (2.68)	86.66 (4.70)
Ecoli	5.7 (0.48)	79.09 (4.97)	91.41 (6.47)	81.82 (5.87)	91.41 (6.47)



Şekil 5.4: Iris veri kümesi üzerindeki ortalama doğruluk oranlarının gösterimi



Şekil 5.5: Wisconsin Diagnostic Breast Cancer veri kümesi üzerindeki ortalama doğruluk oranlarının gösterimi



Şekil 5.6: Ecoli veri kümesi üzerindeki ortalama doğruluk oranlarının gösterimi

Ecoli veri kümesi için kural sayısı ortalaması veri kümesindeki sınıf sayısından azdır. Bunun muhtemel nedeni bu veri kümesindeki iki sınıfa ait sadece ikişer örnek olmasıdır.

Orjinal veri kümesinden oluşturulan 10 farklı eğitim ve test verisi grupları için 10 farklı kural kümesi oluşturulmuştur. Aynı kural kümesinin eğitim verileri ve test verileri üzerindeki sonuçları tablolar şeklinde gösterilmiştir.

YSA'nın daha önce görmediği test verileri üzerinde çıkardığı sonuçların doğruluğunun, eğitim verilerine göre daha düşük olması beklenen bir durumdur. Bu durumun kural kümesi performans ölçümleri üzerindeki etkisi daha çok kapsama oranının düşmesi şeklinde yansımıştır.

Çıkarılan kural kümelerinin kapsadığı bölgeler içinde YSA ile aynı sonuçlar vermesi çıkarılan kuralların aslına uygunluk değerinin yüksek olmasını sağlamıştır. Bu da çıkarılan kuralların YSA'nın davranışını ne oranda yansıttığını göstermektedir.

6 SONUÇ

Yapay sinir ağları, bir bilgi işleme yaklaşımı olarak teorik ve pratik anlamda uzun süredir üzerinde çalışılan ve bir çok problemin çözümünde kullanılan bir yöntemdir. Veri kümesinden modele doğrudan geçişi sağlama, gürültülü verilerle çalışma ve genelleyebilme yeteneği gibi avantajlarına karşılık kara kutu bir sistem olması en büyük eksikliği olarak ifade edilebilir.

Eğitilmiş bir YSA'nın karar verme mekanizmasını doğrudan gözlemlemek mümkün değildir. YSA'lar bilgiyi, bağlantıcı yapı üzerinde dağıtık bir şekilde bulundurmaktadır. Kural çıkarma, bu dağıtık bilgiyi yeniden düzenlemek ve yeniden ifade etmek olarak adlandırılabilir. Bu sayede kara kutu olan YSA'ların karar verme süreci açıklanabilir olur.

Literatürde üç temel kural çıkarma yaklaşımı vardır:

- Çözümlemeli
- Eğitimsel
- Derleme

Bu tez kapsamında önerilen yöntem, eğitimsel yaklaşıma sahip olmasının yanında melez ve zeki bir sistemdir.

6.1 Tezin Literatüre Katkıları

Geliştirilen yöntemin literatüre katkısı temel olarak üç başlıkta ifade edilebilir:

1. YSA'nın sahip olduğu sınıflandırma bölgelerinin topolojik olarak ifade edilebileceği yaklaşımı literatüre kazandırılmıştır. Bunların başarılı bir şekilde kurallara dönüştürülebildiği gösterilmiştir.
2. Kural çıkarma yönteminin bağlantıcı, melez ve zeki bir sistem olması.
3. Kural çıkarma süreci, kullanıcı müdahalesini en aza indirecek şekilde tasarlanmıştır.

Yöntem temel olarak, girdilerin oluşturduğu topolojilerin öğrenilmesi ve eğitilmiş YSA tarafından aynı sınıfa atanan verileri temsil eden topolojilere dönüştürülmesi sayesinde kural kümesi oluşturulması fikrine dayanır.

Bu yöntem sınıflandırma yapması için eğitilmiş YSA'larda kullanılabilir. Ayrıca bu amaçla oluşturulmuş başka kapalı sistemlere de uyarlanabilir.

Yöntemin melez zeki bir sistem olarak geliştirilmesinin temel nedeni yöntem iyileştirmek olarak ifade edilebilir. Genetik algoritmalar kullanılarak, GNG için uygun parametrelerin belirlenmesi hem başarıyı arttırır hem de yöntemin otomatikleşmesini sağlar.

Yöntemin çıkardığı kurallar hiperdikdörtgen kurallar şeklindedir.

6.2 Gelecek Çalışmalar

Yöntemin öncelikle sınıflandırma dışındaki diğer makine öğrenme problem türlerine de aktarılması hedeflenmektedir.

Ayrıca yöntemin sadece sürekli veriler için değil kesikli veriler üzerinde de çalışması için geliştirme çalışmaları yapılması düşünülmektedir.

Eğitimsel yaklaşımından dolayı, YSA gibi diğer kapalı kutu sistemler (örneğin; SVM) için de uygulanabilirliği denenecektir.

KAYNAKLAR DİZİNİ

Alexander, J. A. and Mozer, M. C.: 1995, Template-based algorithms for connectionist rule extraction, *Advances in neural information processing systems* 609–616 p.

Andrews, R.: 2003, *An automated rule refinement system*, Thesis

Andrews, R., Diederich, J., and Tickle, A. B.: 1995, Survey and critique of techniques for extracting rules from trained artificial neural networks, *Knowledge-Based Systems* **8(6)**, 373

Andrews, R. and Geva, S.: 1994, Rule extraction from a constrained error back propagation MLP, in *Proc. 5th Australian Conference on Neural Networks*, 9–12 p.

Angluin, D.: 1988, Queries and concept learning, *Machine learning* **2(4)**, 319–342

Bader, S. and Hölldobler, S.: 2006, The core method: Connectionist model generation, *Artificial Neural Networks–ICANN 2006* 1–13 p.

Bader, S., Hölldobler, S., and Mayer-Eichberger, V.: 2007, Extracting propositional rules from feed-forward neural networks—a new decompositional approach, in *Proceedings of the 3rd International Workshop on Neural-Symbolic Learning and Reasoning. Hyderabad, India: IJCAI*, 1–6 p.

Benitez, J. M., Castro, J. L., and Requena, I.: 1997, Are artificial neural networks black boxes?, *Neural Networks, IEEE Transactions on* **8(5)**, 1156–1164

Bishop, C. M., Svensén, M., and Williams, C. K.: 1998, GTM: the generative topographic mapping, *Neural computation* **10(1)**, 215–234

Craven, M. W. and Shavlik, J. W.: 1994, Using sampling and queries to extract rules from trained neural networks, in *Proceedings of the 11th International Conference on Machine Learning*, 37–45 p.

Craven, M. W. and Shavlik, J. W.: 1995, Extracting comprehensible concept representations from trained neural networks, *Working Notes on the IJCAI* **95**, 61–75

Dancey, D., Bandar, Z. A., and McLean, D.: 2010, Rule extraction from neural networks for medical domains, IEEE, 1–8 pp

Das, S. and Mozer, M.: 1998, Dynamic on-line clustering and state extraction: An approach to symbolic learning, *Neural Networks* **11(1)**, 53

d'Avila Garcez, A., Broda, K., and Gabbay, D.: 2001, Symbolic knowledge extraction from trained neural networks: A sound approach, *Artificial*

Intelligence **125(1-2)**, 155

De Silva, V. and Carlsson, G.: 2004, Topological estimation using witness complexes, in *Proceedings of the First Eurographics conference on Point-Based Graphics*, 157–166 p.

Duch, W., Adamczak, R., and Grabczewski, K.: 2001, A new methodology of extraction, optimization and application of crisp and fuzzy logical rules, *Neural Networks, IEEE Transactions on* **12(2)**, 277–306

Duch, W., Adamczak, R., Grabczewski, K., and Zal, G.: 1999, Hybrid neural-global minimization method of logical rule extraction, *Journal of Advanced Computational Intelligence* 3

Edelsbrunner, H. and Shah, N. R.: 1997, Triangulating topological spaces, *International Journal of Computational Geometry & Applications* **7(04)**, 365–378

Etchells, T. A. and Lisboa, P. J. G.: 2006, Orthogonal search-based rule extraction (OSRE) for trained neural networks: a practical and efficient approach, *Neural Networks, IEEE Transactions on* **17(2)**, 374–384

Fan, Y. and Li, C.: 2002, Diagnostic rule extraction from trained feedforward neural networks, *Mechanical Systems and Signal Processing* **16(6)**, 1073

Fausett, L.: 1994, *Fundamentals of neural networks: architectures, algorithms, and applications*, Prentice-Hall Englewood Cliffs, NJ

Fodor, J. A. and Pylyshyn, Z. W.: 1988, Connectionism and cognitive architecture: A critical analysis, *Cognition* **28(1)**, 3–71

Fritzke, B.: 1995, A growing neural gas network learns topologies, *Advances in neural information processing systems* **7**, 625–632

Fu, L.: 1993, Knowledge-based connectionism for revising domain theories, *IEEE Transactions on Systems, Man and Cybernetics* **23(1)**, 173

Fu, L.: 1994, Rule generation from neural networks, *IEEE Transactions on Systems, Man and Cybernetics* **24(8)**, 1114

Funahashi, K.-I.: 1989, On the approximate realization of continuous mappings by neural networks, *Neural Networks* **2(3)**, 183

Gallant, S. I.: 1988, Connectionist expert systems, *Commun. ACM* **31(2)**, 152–169

Goldberg, D.: 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional

Golea, M.: 1996, On the complexity of rule extraction from neural networks and network-querying (tech. rep.), *Canberra, Australia: Department of Systems*

Engineering, Australian National University

Goonatilake, S. and Khebbal, S. (eds.): 1994, *Intelligent Hybrid Systems*, John Wiley & Sons, Inc., New York, NY, USA

Guo, P., Chen, J., and Sun, S.: 2007, Two approaches to extract rules from neural network with mixed attributes, *Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques* 56–63 p.

Haykin, S.: 1999, *Neural networks: a comprehensive foundation*, Prentice Hall

Hebb, D. O.: 1949, *The organization of behavior: a neuropsychological theory*, Wiley

Hecht-Nielsen, R.: 1987, Kolmogorov's mapping neural network existence theorem, in *Proceedings of the international conference on Neural Networks*, Vol. 3

Herrmann, C. and Thier, A.: 1996, *Backpropagation for Neural DNF-and CNF-Networks*, Technical report, Citeseer

Hopfield, J. J.: 1982, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the national academy of sciences* **79(8)**, 2554–2558

Hopfield, J. J.: 1984, Neurons with graded response have collective computational properties like those of two-state neurons, *Proceedings of the national academy of sciences* **81(10)**, 3088–3092

Hruschka, E. R. and Ebecken, N. F. F.: 2006, Extracting rules from multilayer perceptrons in classification problems: a clustering-based approach, *Neurocomputing* **70(1)**, 384–397

Huynh, T. Q. and Reggia, J. A.: 2011, Guiding hidden layer representations for improved rule extraction from neural networks, *Neural Networks, IEEE Transactions on* **22(2)**, 264–275

Huynh, T. Q. and Reggia, J. A.: 2012, Symbolic representation of recurrent neural network dynamics

Huysmans, J., Setiono, R., Baesens, B., and Vanthienen, J.: 2008, Minerva: Sequential covering for rule extraction, *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **38(2)**, 299–309

Jacobsson, H.: 2005, Rule extraction from recurrent neural networks: ATaxonomy and review, *Neural Computation* **17(6)**, 1223–1263

Jiang, Y., Zhou, Z.-H., and Chen, Z.-q.: 2002, Rule learning based on neural

network ensemble, in *Proceedings of the 2002 International Joint Conference on Neural Networks, 2002. IJCNN '02*, Vol. 2, 1416 –1420 pp

Johansson, U., König, R., and Niklasson, L.: 2010, Genetic rule extraction optimizing brier score, in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 1007–1014 p.

Johansson, U., Löfström, T., König, R., and Niklasson, L.: 2006, Why not use an oracle when you got one, *Neural Information Processing-Letters and Reviews* **10(8-9)**, 227–236

Keedwell, E., Narayanan, A., and Savic, D.: 2000, Evolving rules from neural networks trained on continuous data, in *Proceedings of the 2000 Congress on Evolutionary Computation, 2000*, Vol. 1, 639–645 vol.1 pp

Kohonen, T.: 1988, An introduction to neural computing, *Neural networks* **1(1)**, 3–16

Kohonen, T.: 1990, The self-organizing map, *Proceedings of the IEEE* **78(9)**, 1464–1480

Kolman, E. and Margaliot, M.: 2005, Are artificial neural networks white boxes?, *Neural Networks, IEEE Transactions on* **16(4)**, 844–852

Lee, J. A., Lendasse, A., and Verleysen, M.: 2002, Curvilinear distance analysis versus isomap, in *Proceedings of ESANN*, 185–192 p.

Lu, H., Setiono, R., and Liu, H.: 1996, Effective data mining using neural networks, *IEEE Transactions on Knowledge and Data Engineering* **8(6)**, 957

Martinetz, T. and Schulten, K.: 1994, Topology representing networks, *Neural Networks* **7(3)**, 507–522

McCulloch, W. S. and Pitts, W.: 1943, A logical calculus of the ideas immanent in nervous activity, *Bulletin of mathematical biology* **5(4)**, 115–133

Mcmillan, C., Mozer, M. C., and Smolensky, P.: 1992, Rule induction through integrated symbolic and subsymbolic processing, in *Advances in Neural Information Processing Systems 4*, Morgan Kaufmann, 969–976 p.

Minsky, M. and Seymour, P.: 1969, Perceptrons.

Mohamed, M. H.: 2011, Rules extraction from constructively trained neural networks based on genetic algorithms, *Neurocomputing* **74(17)**, 3180–3192

Narazaki, H., Shigaki, T., and Watanabe, T.: 1995, A method for extracting approximate rules from neural network, in *Fuzzy Systems, 1995. International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium., Proceedings of*

1995 IEEE Int, Vol. 4, 1865 –1870 vol.4 pp

Okabe, A., Boots, B., Sugihara, K., and Chiu, S. N.: 2009, *Spatial tessellations: concepts and applications of Voronoi diagrams*, Vol. 501, Wiley

Palade, V., Neagu, D.-C., and Patton, R. J.: 2001, Interpretation of trained neural networks by rule extraction, in B. Reusch (ed.), *Computational Intelligence. Theory and Applications*, No. 2206 in Lecture Notes in Computer Science, Springer Berlin Heidelberg, 152–161 pp

Palade, V., Neagu, D. C., and Puscasu, G.: 2000, Rule extraction from neural networks by interval propagation, in *Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 2000. Proceedings. Fourth International Conference on*, Vol. 1, 217–220 p.

Pop, E., Hayward, R., and Diederich, J.: 1994, RULENEG: extracting rules from a trained ANN by stepwise negation (neurocomputing research centre tech. rep.), *Queensland: Queensland University of Technology*

Quinlan, J. R.: 1993, *C4. 5: programs for machine learning*, Vol. 1, Morgan kaufmann

Rosenblatt, F.: 1962, Principles of neurodynamics

Rumelhart, D. E. and McClelland, J. L.: 1986, Parallel distributed processing: explorations in the microstructure of cognition. volume 1. foundations

Saad, E. W. and Wunsch, D. C.: 2007, Neural network explanation using inversion, *Neural networks* **20(1)**, 78–93

Sethi, I. and Yoo, J.: 1994, Symbolic approximation of feedforward neural networks, in E. S. Gelsema and L. S. Kanal (eds.), *Pattern Recognition in Practice Iv: Multiple Paradigms, Comparative Studies and Hybrid Systems*, Vol. 16, Elsevier Science Publ B V, 313–324 pp, Amsterdam

Sethi, I. and Yoo, J.: 1996, Multivalued logic mapping of neurons in feedforward networks, *Engineering Intelligent Systems* **4(4)**, 243–253

Setiono, R.: 1997, Extracting rules from neural networks by pruning and hidden-unit splitting, *Neural Computation* **9(1)**, 205

Setiono, R., Baesens, B., and Mues, C.: 2008, Recursive neural network rule extraction for data with mixed attributes, *Neural Networks, IEEE Transactions on* **19(2)**, 299–307

Setiono, R. and Leow, W. K.: 2000, FERNN: an algorithm for fast extraction of rules from neural networks, *Applied Intelligence* **12(1)**, 15–25

Setiono, R., Leow, W. K., and Zurada, J. M.: 2002, Extraction of rules from

artificial neural networks for nonlinear regression, *Neural Networks, IEEE Transactions on* **13(3)**, 564–577

Setiono, R. and Liu, H.: 1997, NeuroLinear: from neural networks to oblique decision rules, *Neurocomputing* **17(1)**, 1

Silva, V. d. and Tenenbaum, J. B.: 2003, Global versus local methods in nonlinear dimensionality reduction, *Advances in neural information processing systems* **15**, 705–712

Taha, I. and Ghosh, J.: 1996, Three techniques for extracting rules from feedforward networks, *Intelligent Engineering Systems Through Artificial Neural Networks* **6**, 23–28

Taha, I. A. and Ghosh, J.: 1999, Symbolic interpretation of artificial neural networks, *Knowledge and Data Engineering, IEEE Transactions on* **11(3)**, 448–463

Tan, S. C., Lim, C. P., and Rao, M. V. C.: 2007, A hybrid neural network model for rule generation and its application to process fault detection and diagnosis, *Engineering Applications of Artificial Intelligence* **20(2)**, 203–213

Thrun, S.: 1993, *Extracting provably correct rules from artificial neural networks*, Technical report, Technical Report IAI-TR-93-5, University of Bonn, Institut für Informatik III

Tibshirani, R.: 1992, Principal curves revisited, *Statistics and Computing* **2(4)**, 183–190

Tickle, A. B., Andrews, R., Golea, M., and Diederich, J.: 1998, The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks, *Neural Networks, IEEE Transactions on* **9(6)**, 1057–1068

Tipping, M. E. and Bishop, C. M.: 1999, Mixtures of probabilistic principal component analyzers, *Neural computation* **11(2)**, 443–482

Towell, G., Shavlik, J. W., and Noordewier, M.: 1990, Refinement of approximate domain theories by knowledge-based neural networks, in *Proceedings of the eighth National conference on Artificial intelligence*, 861–866 p.

Towell, G. G. and Shavlik, J. W.: 1993, Extracting refined rules from knowledge-based neural networks, *Machine learning* **13(1)**, 71–101

Vahed, A. and Omlin, C.: 1999, Rule extraction from recurrent neural networks using a symbolic machine learning algorithm, in *6th International Conference on Neural Information Processing, 1999. Proceedings. ICONIP '99*, Vol. 2, 712 –717 vol.2 pp

KAYNAKLAR DİZİNİ (Devam)

Valiant, L. G.: 1984, A theory of the learnable, *Communications of the ACM* **27(11)**, 1134–1142

Von Neumann, J.: 1951, The general and logical theory of automata, *Cerebral mechanisms in behavior* 1–41 p.

Von Neumann, J.: 1956, Probabilistic logics and the synthesis of reliable organisms from unreliable components, *Automata studies* **34**, 43–98

Weijters, T., Bosch, V. D., and Herik, H. J.: 1997, Intelligible neural networks with BP-SOM, *Marcke and Daelemans* 27–36 p.

Widrow, B. and Hoff, M. E.: 1960, Adaptive switching circuits.

Zhang, D., Liu, Y., and Wang, Z.: 2005, Effectively extracting rules from trained neural networks based on the new measurement method of the classification power of attributes, *Fuzzy Systems and Knowledge Discovery* 487–487 p.

Zárate, L. E., Mariano Dias, S., and Junho Song, M. A.: 2008, FCANN: a new approach for extraction and representation of knowledge from ANN trained via formal concept analysis, *Neurocomputing* **71(13)**, 2670–2684

ÖZGEÇMİŞ

Adı Soyadı: Ahmet Cumhur KINACI
Doğum Tarihi: 1980
Doğum Yeri: Sandıklı
Uyruğu: T.C.

EĞİTİM

Yüksek Lisans: Ege Üniversitesi
Bilgisayar Mühendisliği Bölümü, 2006

Lisans: Orta Doğu Teknik Üniversitesi
İstatistik Bölümü, 2003

Lise: Ankara Fethiye-Kemal Mumcu Anadolu Lisesi, 1998

İLGİ ALANLARI

Yapay Zeka, Yapay Sinir Ağları, Örüntü Tanıma

TERİMLER SÖZLÜĞÜ

İNGİLİZCE	TÜRKÇE
accuracy	kesinlik veya doğruluk
activation	etkinleştirme
attribute	özellik
backpropagation	geri yayılımlı
backtracking	geri izlemeli
binary	ikili
black box	kapalı kutu
boolean	boole
clustering	kümeleme
competitive	rekabetçi
comprehensibility	anlaşılabilirlik
complete	tam
complexity	karmaşıklık
conjunction	bağlama
connectionist	bağlantıcı
consistency	tutarlılık
convex	dışbükey
core	çekirdek
crossover	çaprazlama
decompositional	çözümlemeli
disjunction	ayrışma
domain	alan
eclectic	derleme
encoding	kodlama
ensemble	birlik
expressive power	ifade gücü
feed forward	ileri beslemeli
fidelity	aslına uygunluk
finite state machine	sonlu durum makinesi
first-order logic	birinci dereceden mantık
fitness function	uygunluk fonksiyonu
generality	genellik
genetic algorithms	genetik algoritmalar
hyperplane	hiper düzlem
hyperrectangle	hiperdikdörtgen
ikili eşik	binary threshold
interval propagation	aralık yayma

k-fold cross validation	k-kat çapraz geçerleme
linear programming	doğrusal programlama
logical	mantıksal
membership query	üyelik sorgusu
multilayer perceptron	çok katmanlı algılayıcı
nonconventional logic	klasik olmayan mantık
oracle	tahminleyici
pedagogical	eğitimsel
perceptron	algılayıcı
portability	taşınabilirlik
predictive accuracy	tahminleme doğruluğu
propositional logic	propositional logic
quality	kalite
recurrent	geri dönüşlü
scalability	ölçeklenebilirlik
selection	seçme
self organizing map	özörgütlemeli harita
simplical complex	simpleksler kompleksi
sound	güvenilir
split	bölme
template matching	şablon eşleme
topology learning	topoloji öğrenme
translucency	saydamlık
unbiased	yansız
validity interval	geçerlilik aralığı

A ÇIKARILAN KURALLAR

Geliştirilen yöntem kullanılarak Iris, Wisconsin Diagnostic Breast Cancer ve Ecoli veri setleri için çıkarılan kurallar bu bölümde sunulmaktadır. Çıkarılan kurallar k-fold cross validation sırasında üretilen parçalardan ilki için üretilen kuralları içerir. Kurallar orjinal verinin normalize edilmiş hali için uygundur. Kurallardaki her a, b ikilisi verideki özellik için en düşük ve en yüksek değerleri ifade eder.

Iris

Özellik	Kural 1	Kural 2	Kural 3
sepal length in cm	0.361,1.0	0.166,0.75	0.0,0.416
sepal width in cm	0.208,0.75	0.0,0.583	0.125,0.916
petal length in cm	0.644,1.0	0.338,0.779	0.0,0.177
petal width in cm	0.625,1.0	0.375,0.625	0.0,0.208
Sınıf	**Iris Virginica**	**Iris Versicolour**	**Iris Setosa**

Tablodaki değerlerden oluşturulan örnek bir kural :
If (0.361<sepal_length<1.0) and (0.208<sepal_width<0.75) and (0.644<petal_length<1.0) and (0.625<petal_width<1.0) then Iris Virginica

Wisconsin Diagnostic Breast Cancer

Özellik	Kural 1	Kural 2	Kural 3
radius	0.292,1.0	0.0,0.514	0.187,0.435
texture	0.073,1.0	0.0,0.815	0.197,0.484
perimeter	0.287,1.0	0.0,0.489	0.194,0.447
area	0.164,1.0	0.0,0.359	0.092,0.287
smoothness	0.19,0.831	0.0,0.762	0.387,0.811
compactness	0.121,1.0	0.0,0.41	0.258,0.811
concavity	0.125,1.0	0.0,0.309	0.181,0.565
concave points	0.162,1.0	0.0,0.277	0.234,0.522
symmetry	0.125,1.0	0.0,0.85	0.371,0.817
fractal dimension	0.0,0.662	0.039,0.839	0.23,1.0
Sınıf	**Malignant**	**Benign**	**Malignant**

Ecoli

Özellik	Kural 1	Kural 2	Kural 3	Kural 4	Kural 5	Kural 6
mcg	0.044,0.752	0.0,0.719	0.662,0.955	0.741,1.0	0.717,0.865	0.584,0.865
gvh	0.0,0.511	0.214,0.607	0.369,1.0	0.119,0.654	0.357,0.492	0.392,0.857
lip	0.0,0.0	0.0,0.0	0.0,0.0	0.0,0.0	0.797,1.0	0.0,0.0
chg	0.0,0.0	0.0,0.0	0.0,0.0	0.0,0.0	0.0,0.0	0.0,0.0
aac	0.181,0.829	0.352,0.84	0.193,0.659	0.477,0.852	0.42,0.75	0.738,1.0
alm1	0.0,0.536	0.552,0.938	0.288,0.67	0.577,0.917	0.525,0.596	0.34,0.567
alm2	0.0,0.616	0.171,0.949	0.191,0.565	0.626,0.929	0.01,0.436	0.151,0.454
Sınıf	**cp**	**im**	**pp**	**imU**	**omL**	**om**