ENDÜKTİF ÖĞRENME ALGORİTMALARININ KURAL ÜRETME YÖNTEMLERİ VE PERFORMANSLARININ KARŞILAŞTIRILMASI

Ömer AKGÖBEK[1], Ercan ÖZTEMEL[2]

[1] Harran Üniv., Müh. Fak., Endüstri Müh. Bölümü, Şanlıurfa, Tel:414 3128456,oakgobek@harran.edu.tr
[2]Sakarya Üniv., Müh. Fak., End. Müh. Bölümü, Adapazarı.Tel:264 3460353(281),eoztemel@sakarya.edu.tr

ÖZET

"Bilgi Çağı" ve "Bilgi Toplumu" gibi terimlerin sıklıkla kullanıldığı günümüzde, bilginin önemi daha açık bir şekilde ortaya çıkmaktadır. Bilginin önemi arttığı oranda o bilgiye ulaşabilmeyi sağlayan sistemlerin de önemi artmaktadır. Bilgisayar teknolojisindeki büyük gelişmeler sayesinde, dünyanın herhangi bir yerinde üretilen bilginin sayısal hale getirilerek saklanması ve o bilgiye dünyanın herhangi bir yerinden çok kısa sürede erişim mümkün olmaktadır. Bununla birlikte programlama dillerindeki büyük gelişmeler sayesinde bilgiyi işlemek ve istenen bilgiye erişmek de kolaylaşmaktadır. Bu çalışmada, bilgiyi elde etmek amacıyla kullanılan Endüktif Öğrenme teknikleri anlatılacak ve bu alanda geliştirilen algoritmalar karşılaştırılacaktır.

Anahtar Kelimeler - Endüktif öğrenme, Karar ağacı, Kural üretme, Bilgi kazanımı

RULE GENERATION METHODS OF INDUCTIVE LEARNING ALGORITHMS AND COMPARISOIN OF THEIR PERFORMANCES

ABSTRACT

Frequent emphasis on the phrases such as "Information Age" and "Information Society" clearly expresses the importance of knowledge in our daily life. As the importance of knowledge increases, so does the need for tools to reach and retrieve the knowledge. Thanks to great developments in the computer technology, it is possible to store the information generated miles away in an electronic format and retrieve it back quickly at any location in the world, when needed. In addition, rapid developments in programming languages also made it easy to process the information and access it in case of need. In this study, inductive learning techniques which are used to acquire information will be explained and the algorithms developed for such purposes will be compared with each other.

Keywords - Inductive learning, Decision tree, Rule induction, Knowledge acquisition

1. GİRİŞ

Bilgisayar teknolojisinin büyük bir hızla gelişmesi ile beraber yapay zeka alanında çok büyük ve hızlı gelişmeler yaşanmaktadır. Bu gelişmelerle beraber yapay zekanın bir alt dalı olan uzman sistemlerde de büyük ilerlemeler kaydedilmektedir.

Bilgi, genel olarak bir uzmandan, konu ile ilgili kaynaklardan, arşiv bilgilerinden, gözlem veya deneylerden elde edilebilir. Bir uzmandan bilgi elde etme, karşılıklı görüşmeler gerektiren, uzun zaman alan, dikkat isteyen ve sistematik çalışmalar gerektiren bir işlemdir. Uzmanlar, uzmanlık bilgilerini günlük çalışmalarında rahatlıkla kullanabilmelerine rağmen, bunları özetleme ve bir uzman sistemde kullanılabilir hale getirmede aynı başarıyı gösteremeyebilirler. Bu bilgilerin değerlendirilmesi ve bir uzman sistem için bilgi tabanı haline getirilmesi ayrı bir uzmanlık ister. Bilgi tabanının bu

yolla oluşturulması hem uzun zaman almakta hem de yüksek maliyet gerektirmektedir. Çünkü bu işlemleri yerine getirecek uzman kişilere ihtiyaç duyulmakta, bu elemanları bulmak zorlaşmakta ve istihdamı da yüksek maliyet gerektirmektedir.

Bilginin elde edilmesine "bilgi kazanımı" denir. Araştırmacılar bilgi kazanımını, bir uzman sistemin geliştirilmesinde en büyük darboğaz olarak kabul etmektedirler. Feigenbaum [1] bu hususu şöyle dile getirmektedir:"Bilginin gösterimi, kullanılması ve kazanılması ile ilgili çözülmesi gereken çok önemli problemler vardır. Bunlardan bilgi kazanımı problemi en önemli kritik darboğazı oluşturmaktadır".

Bütün bu ve benzeri zorluklar araştırmacıları bu darboğazı aşmak için alternatif teknikler geliştirmeye sevk etmiştir. Bu konuda birçok teknik geliştirilmiş ve geliştirilmeye devam edilmektedir. Geliştirilen bu tekniklerin amacı bilgi kazanımını otomatik hale getirmektir. Bunlardan bir tanesi de 'Endüktif Öğrenme' (Inductive Learning) tekniğidir.

Endüktif öğrenme alanında birçok algoritma geliştirilmiştir. Bu algoritmaların kural üretme yöntemleri arasında farklılıklar olmasına rağmen, sonuçta örnek setini en iyi şekilde temsil eden kuralların üretilmesini sağlarlar.

Bu çalışmanın ikinci bölümünde endüktif öğrenme tekniğinin ana hatları üzerinde durulmuş, üçüncü bölümünde endüktif öğrenme algoritmaları ve bu algoritmaların kural üretme metotları açıklanmış ve dördüncü bölümde ise bu algoritmaların performansları karşılaştırılmıştır.

2. ENDÜKTİF ÖĞRENME

Endüktif öğrenme; gözlem, deney veya bir veritabanından elde edilen örnek setinden genel bilgi çıkarma tekniği ve otomatik bilgi kazanımı için pratik yaklaşımlardan birisi olarak kabul edilir [2].

Son yıllarda araştırmacılar endüktif öğrenebilirliğin teorisi üzerinde çalışmışlar ve temel teorik analizler üzerinde endüktif öğrenmenin mümkün olabilirliğini kanıtlamayı başarmışlardır. İlk olarak Vapnik ve Chervonenkis [3] öğrenebilmenin bir ölçüsü olarak VC-boyutu'nu sunmuşlardır. Daha sonra öğrenebilme alanında birçok teorik çalışma yapılmıştır [4, 5].

Endüktif öğrenmede "öğrenme" kavramı ile eğitme örneklerinden bir kavramı en iyi açıklayan bilgilerin elde edilmesi kastedilmektedir. Burada örneklerden karar kuralları denilen kurallar oluşturulur. Örneklerden genel ifadelere (kurallara) ulaşıldığından endüktif öğrenmeye "tümevarım öğrenme" de denilmektedir [6].

Endüktif öğrenme algoritmalarının tasarımındaki en büyük problem, düzensiz örneklerden aşırı derecede karmaşık tanımlamaları üretmekten sakınmanın nasıl olacağıdır. Bozuk, eksik, düzensiz verilerden öğrenme, çok sayıda verimsiz örnekler tanımlama ve karmaşık karar kurallarının çok sayıda olmasına sebep olabilir. Bu durumda üretilen kavram tanımı genellikle eksik olmaktadır [6].

Genel olarak endüktif öğrenme aşağıdaki gibi tanımlanabilir [7, 8]:

Verilenler:

- Bazı deney veya durumlar hakkında bilgiyi gösteren bir örnek seti (*A*).
- Geçici/kesin olmayan endüktif iddia (boş olabilir).
- Geçmiş bilgi *K*.

Bulunanlar:

- Örnekleri ifade eden ve geçmiş *K* bilgisini temsil eden bir *h* hipotezi. Bunun gösterimi $h \wedge K \Rightarrow A$ şeklindedir. Bu ifade, *h* hipotezi ve geçmiş *K* bilgisinin, A örnek setini ifade ettiğini gösterir. Burada $\wedge$ mantıksal çıktıyı gösterir.

A'nın yapısına bağlı olarak endüktif öğrenmenin iki mümkün durumu vardır. Bunlar: danışmanlı ve danışmansız. *A*'daki örnekler önceden sınıflandırılmışsa bu öğrenmeye danışmanlı, diğer durumda danışmansız olarak adlandırılır.

Bir *A* örnek setinde *e* adet örnek ve *s* adet $\{d_1,d_2,\ldots,d_s\}$ farklı sınıf olsun ve $e(di)$, d_i sınıfına ait bir örneği göstersin. *A*'dan bilgi veya kavram öğrenme;

$$h:\{Des(d_i)\}\ 1 \le i \le s$$

olarak gösterilir. Burada $Des(d_i)$, d_i sınıfı kavramın tanımıdır. Herhangi bir endüktif öğrenme prosedürü tarafından tahmin edilebilen iki şart vardır. Bunlar, eksizlik ve tutarlılık şartıdır. Eksiksizlik:

$$\forall e(d_i) \in A\left(Des(d_i) \Rightarrow e(d_i)\right)$$

Bu şart ile bütün örneklerde, d_i sınıfı $Des(d_i)$ kavramı tarafından kapsanmalıdır. Tutarlılık:

$$\forall e(d_i), e(d_j) \in A\left(e(d_j) \Rightarrow \approx Des(d_i)\right),\ j \neq i$$

Bu şart endüktif öğrenmede çok önemlidir. $Des(d_i)$ kavramı, d_i sınıfına ait olmayan herhangi bir örneği kapsamamalıdır.

Endüktif öğrenme ile bir öğrenme işleminde birkaç mümkün çıktı elde edilmiş olacaktır. *H* bütün mümkün çıktıları içeren bir hipotez uzayı olsun. Endüktif öğrenme, bir *h* hipotezini bulmak için *H* hipotez uzayında bir arama prosedürü olarak açıklanabilir. Arama işlemini yaparken eksiksizlik ve tutarlılık şartlarını göz önünde tutarak aynı zamanda *K* geçmiş bilgisini de tahmin eder [9,10].

Endüktif öğrenmede kullanılacak olan bilgi örnek seti olarak adlandırılır. Bir örnek seti *U* olarak tanımlandığında, *U*'daki örnekler bir şart karakteristik seti $C=\{C_1,C_2,...,C_n\}$ ve bir $D=\{D_1,D_2,...,D_n\}$ karar karakteristiği tarafından tanımlanır. Karakteristiklerin bilgi alanı *V* tarafından verilir ($V=\{V_1,V_2,...,V_n\}$). Herbir $C_i \in C$ olan C_i karakteristiğinin bilgi alanı V_i değer setidir. V_i, $V_i \in V$ olan gözlemlenebilir bir değer setidir.

H'nin boyutu veya mümkün hipotezlerin sayısı;

$$|H| = \sum_{i=1...n} |V_i| + \sum_{\substack{i,j=1...n \\ i \neq j}} |V_i||V_j| + ... + \sum_{\substack{i,j,...k=1...n \\ i \neq j \neq ... \neq k}} |V_i||V_j||V_k| + \prod_{i=1...n} |V_i|$$

olarak hesaplanır. Şart karakteristikleri ikili değerlere sahip ise $|H|$;

$$|H| = \sum_{i=1...n} C_n^i 2^i \qquad \text{olur.}$$

Burada açıkça görülüyor ki, *n* çok büyük bir değer olduğunda bütün hipotez uzayını araştırmak pratik, hatta mümkün değildir.

Arama metodu, endüktif öğrenme prosedürleri için çok önemlidir [10,11]. Arama metoduna alternatif olarak hipotez uzayını mümkün olan en az seviyeye indiren metotlar da geliştirilmiştir [6,12].

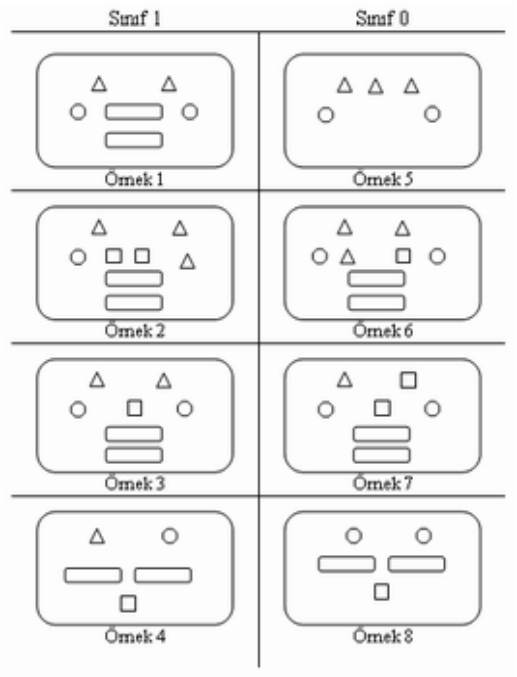
3. ENDÜKTİF ÖĞRENME ALGORİTMALARI

Öğrenme sonuçları ya kesin ya da tahmini sonuçlar olarak bir karar ağacı veya bir kural seti şeklinde sınıflandırılarak gösterilebilir. Karar ağacı şeklinde kural üreten algoritmalar Böl-ve-Fethet yaklaşımını, doğrudan kural üreten algoritmalar ise kapsama yaklaşımını kullanırlar.

Böl-ve-fethet yaklaşımını kullanan endüktif öğrenme algoritmaları bilgiyi bir karar ağacı şeklinde gösterirler. Bu yaklaşım, karar ağacı kullanımından dolayı gereğinden fazla kural tanımlamaları içerebilir. Ayrıca üretilen kurallardaki şart sayılarının fazla olmasından dolayı sonuçlar basitleştirilmiş şekilde değildir. Bir diğer dezavantajı ise, her örnek seti için sadece bir tek karar ağacı oluşturması ve bu ağacın kural setine dönüştürülmesi ile çok sayıda kuralın üretilmesidir. Bunu engellemek için budama yöntemleri kullanılır. Bu durumda ise doğruluk oranında düşüş olur.

Kapsama yaklaşımını kullanan endüktif öğrenme algoritmaları bilgiyi doğrudan bir kural seti şeklinde gösterirler.

Karar ağacı ve kural seti kavramları Şekil 1'de gösterilen iki sınıflı sınıflandırma problemi için açıklanmıştır.



Şekil 1. Örnek seti

Şekil 1'deki örneklerin şekilsel gösterimlerini, endüktif öğrenme prosedürlerindeki gösterimini kolaylaştırmak için Tablo 1'de gösterildiği gibi karakteristik-değer şekline dönüştürülmüştür. Tablo 1'deki örnekler dört şart karakteristiği (C_1, C_2, C_3, C_4) ve bir karar karakteristiği (Sınıf) tarafından tanımlanmıştır. Her bir şart karakteristiği bir geometrik şekle karşılık gelmektedir. Örneğin, C_1 daireyi (⬭), C2 uzun çubuğu (▭), C3 üçgeni (△) ve C4 ise kareyi (□) göstermektedir. Bir şart karakteristiğinin değeri (C_i), ilgili şeklin birkaç elemanını (değer) gösterir [13].

Tablo 1'deki her bir satır bir örneği belirtmektedir. Örneğin, ikinci satır *{C_1=1, C_2=2, C_3=3, C_4=2, Sınıf=1}* Şekil 1'deki ilk sınıfın ikinci örneğini gösterir. Burada; 1 adet daire, 2 adet uzun çubuk, 3 adet üçgen ve 2 adet kare bulunmaktadır.

Tablo 1. Şekil 1 verilen örnek setinin karakteristik-değer biçimi

Örnek No	C_1	C_2	C_3	C_4	Sınıf
1	2	2	2	0	1
2	1	2	3	2	1
3	2	2	2	1	1
4	1	2	1	1	1
5	2	0	3	0	0
6	2	2	3	1	0
7	2	2	1	2	0
8	2	2	0	1	0

Şekil 2, örnek setinden kural seti ve karar ağacı çıkarımını göstermektedir. Kural setindeki ilk kural;

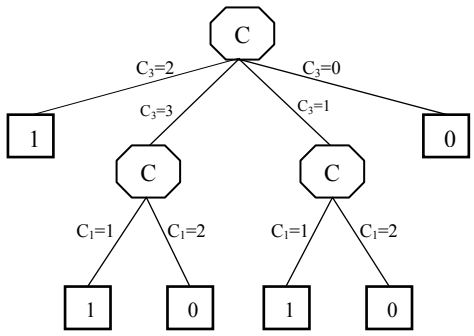
Eğer 2 üçgen var ise 1 nolu sınıfa aittir

şeklinde yorumlanır.

Tablo 1'deki örnek veri seti kullanılarak üretilen kurallar aşağıda verilmiştir (Kural seti REX-1 algoritması ile elde edilmiştir).

Kural 1 : Eğer $C_3 = 2$ ise Sınıf = 1
Kural 2 : Eğer $C_1 = 1$ ise Sınıf = 1
Kural 3 : Eğer $C_1=2$ ve $C_3=3$ ise Sınıf = 0
Kural 4 : Eğer $C_1=2$ ve $C_3=1$ ise Sınıf = 0
Kural 5 : Eğer $C_3=0$ ise Sınıf = 0

Tablo 1'deki örnek seti için üretilen karar ağacı Şekil 2'de verilmiştir (Karar ağacı, C4.5 algoritması kullanılarak elde edilmiştir) [14].



Şekil 2. Karar ağacı.

3.1. Karar Ağacı Tabanlı Algoritmalar

Karar ağacı-tabanlı algoritmalar, genellikle karar ağacını oluşturmak için en büyük bilgi kazancını veren özelliklerden arama yapmak için bilginin entropi ölçüsünü kullanırlar [14,15]. Bu amaçla karar ağacını oluşturmak için örnek setini küçük alt setlere bölerler. Karar ağacı geliştirme prosedürü, eğitme örneklerini kullanıcının belirlediği sonlandırma ölçütüne göre doğru sınıflandırılıncaya kadar devam eder.

Karar ağacı-tabanlı algoritmalar Böl-ve-Fethet metoduna göre işlem yaparak örnekleri alt setlere ayırırlar. Böl-ve-Fethet endüktif öğrenme metodunun ana fikri, bir örnek setini, her bir alt sette sadece tek bir sınıf kalıncaya kadar alt setlere ayırır. Bu metodun sonucu bir karar ağacı şeklindedir. Genel olarak Böl-ve-Fethet endüktif öğrenme prosedürü aşağıdaki gibidir:

1. Verilen *S* örnek seti, $S_{11}, S_{12},...,S_{1n}$ şeklinde alt setlere ayrıştırılır. Bu;

$$S = \bigcup_i S_{1i} \ and \ \bigcap_i S_{1i} = \phi$$ şeklinde ifade edilir.

2. Her bir S_{1i} alt seti için, S_{1i}'de tek bir sınıf varsa ayrıştırma işlemi tamamlanır ve buraya bir sınıf etiketi olarak düğüm konur.

Şekil 3'te ayrıştırma prosedürü gösterilmiştir. Burada, bir sınıfı göstermek üzere düğüm olarak dikdörtgen şekli kullanılmıştır. Prosedürün başlangıç noktası en düşük ayrıştırma seviyesini gösterir. Bir düşük ayrıştırma seviyesinde bir düğüm görülebilir. Bunlar yüksek seviyelerden daha geneldir. Ayrıca aynı ayrıştırma seviyesinde 'çocuk' düğümü ile onların 'ebeveyn' düğümü arasındaki ilişkiler için;

$$S_{(1,i)..(l,j)} = \bigcup_k S_{(1,i)..(l,j)(l+1,k)}$$

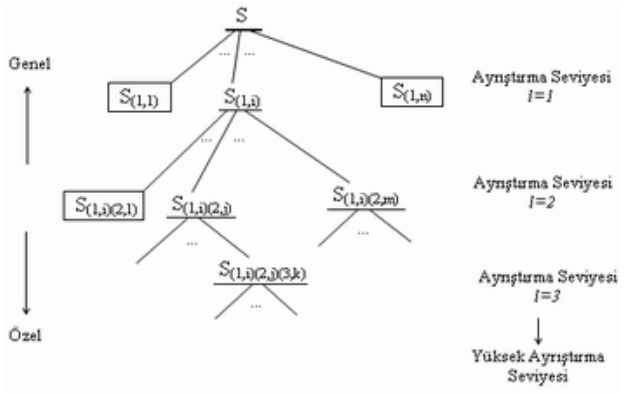
ve

$$\bigcap_k S_{(1,i)...(l,j)(l+1,k)} = \phi$$

ifadeleri yazılabilir. Buradaki *l* ayrıştırma seviyesini gösterir.

S=(C,D,V) bir örnek seti olarak alındığında, bu örnek setindeki elemanlar sırasıyla; *C={C1,C2,...,Cn}* karakteristik setini, *V={V1,V2,...,Vn}* değerler setini ve *D={D1,D2,...,Dn}* ise karar karakteristik (sınıf) setini belirtir. Tablo 1'de verilen örnek seti, Böl-ve-Fethet endüktif öğrenme metoduna göre çok farklı alt setlere ayrıştırılabilir. Bu örnek seti; *C={C1,C2,C3,C4}*, *V={V1,V2,V3,V4}={{1,2}, {0,1,2}, {0,1,2,3}, {0,1,2}}* ve *D={0,1}* bilgilerinden oluşmuştur. Bu bilgilere göre 239 mümkün alt set oluşturulabilir. Bu metot ile her bir düğümdeki ayrıştırmalar optimize edilerek verimli ve kabul edilebilir bir yapı elde edilebilir.

Karar ağaçları oluşturulduktan sonra, ağacın her dalı bir kural oluşturacak şekilde kurallara dönüştürülür. Ancak karar ağacını karar kurallarına çevirmenin bazı dezavantajları vardır [15] ve karar ağaçları ile işlem yapmak oldukça zordur. Her örnek seti için bir tek karar ağacı oluşturulmasından dolayı kök olarak seçilen karakteristik bütün kurallarda yer alacaktır. Bunun sonucu olarak birçok gereksiz kural üretilecek ve üretilen kurallar çok sayıda şart ifadesi içerecektir.



Şekil 3. Böl-ve-Fethet endüktif öğrenme metodunda ayrıştırma prosedürü

Herhangi bir sınıflandırma hakkında bilgi elde etmek bütün ağacın gözden geçirilmesini gerektirir. Bir problemi çözmek, karar ağacını kurallara dönüştürmekle mümkündür. Ancak ağaçlar tarafından gösterilemeyecek kurallar da vardır. Örnek olarak, aşağıdaki kural seti verilsin.

Kural 1: EĞER a = 1 VE b = 1 İSE Sınıf = 1
Kural 2: EĞER c = 1 VE d = 1 İSE Sınıf = 1

Kural-1 ve Kural-2'nin Sınıf 1'i kapsadığını ve diğer tüm kuralların ise Sınıf 2'ye ait olduğu kabul edildiğinde, her iki kural tek bir ağaç tarafından gösterilemez. Çünkü ağacın kökü bir karakteristiğe göre ayrılır ve her iki kurala ait olan bir karakteristik yoktur. Bu durumu göstermek için verilen kurallara bir şartın fazladan eklenmesi gerekecektir. Bu da sonuçta en az bir fazla kuralın çıkmasına sebep olacaktır [16].

Karar ağacını kullanan ilk algoritma, Hunt [17] tarafından geliştirilen CLS (Concept Learning Sytem) algoritmasıdır. CLS serisi algoritmaların toplam sayısı dokuzdur. Bunlardan ilk sekizi sadece iki sınıflı örnek setleri ile çalışabilirken, CLS9 algoritmasında çok sınıflı örnekler çözülebilmektedir. Daha sonra Quinlan [9] tarafından ve bu metodu kullanan ID3 ailesi algoritmaları geliştirilmiştir. (ID3, ASSISTANT ve C4.5). ID3 algoritmasında, ayrıştırmaya rehberlik edecek bir bilgi ölçüsü (entropi) kullanılmıştır. Bilginin entropisi (1) ve (2) eşitlikleri yardımıyla hesaplanır.

$$I(S) = -\sum_{i} p(D = d_i) \log_2 p(D = d_i) \qquad (1)$$

$$E(C_i, S) = \sum_{\substack{C_i = v_{ik} \\ v_{ik} \in V_i}} \frac{|S_j|}{|S|} I(S_j) \qquad (2)$$

I(S), S örnek setinin bilgi değerini gösterir. *E(C_i , S)*, C_i sınıfındaki örneklere ait bilginin entropisini gösterir. Bilgi kazancı ise (3) eşitliği ile hesaplanır.

$$Kazanç(C_i, S) = I(S) - E(C_i, S) \qquad (3)$$

ID3 algoritması bilgi kazancını maksimize eden bir karakteristiği kök düğüm olarak seçer. Her düğüm için ilgili alt setin bilgi kazancı hesaplanarak bu düğüme göre alt setlere ayrıştırma işlemine, her düğümde tek sınıf kalıncaya kadar devam edilir.

Daha sonra, ID3 endüktif öğrenme algoritmalarının bir serisi geliştirildi. Bu seri ID3 ailesi algoritmaları olarak adlandırılır [18]. ID3 algoritmasının versiyonları, ID3-IV [19], GID3 [19], ID4 [20], ID5 [21], ID5R ve ID5R-hat [22] olarak sayılabilir.

ID3 algoritmasının devamı olarak geliştirilen C4.5 algoritması, ID3 algoritmasının endüstriyel versiyonu olarak kabul edilmekte ve bugün birçok uzmanlık alanında yaygın olarak kullanılmaktadır [23]. ID3'ün gelişmiş bir şekli olan C4.5 algoritmasının en büyük özelliği karar ağacının gereksiz dallarını tespit ederek budamasıdır. Bu algoritmanın bir diğer özelliği ise eksik veri ve sayısal değerlerle de çalışabilmesidir.

Karar ağacı üreten bir diğer algoritma ise Breiman tarafından geliştirilmiştir: CART (Classification and Regression Trees) [24]. CART, her düğümündeki tüm karakteristikleri birer birer araştırır. Her bir karakteristik için en çok katkıyı sağlayan en iyi ayrıştırmayı bulur ve *n* adet aday içerisinden en iyi ayrıştırmayı seçer. Sayısal verileri işlemek üzere tasarlanan bu algoritma Böl-ve-Fethet endüktif öğrenme metoduna göre işlem yapmakta ve sadece ikili karar ağacı oluşturmaktaydı.

Daha sonra karar ağacının kısa olmasını sağlamak için Crawford [25] tarafından CART algoritmasının uzantısı olan OC1 algoritması ile GDT-NR ve GDT-RS [26] algoritmaları geliştirilmiştir.

3.2. Kural Tabanlı Algoritmalar

Kural-tabanlı algoritmalar kapsama metodunu kullanırlar. Bu metot verilen örnek setinden daha genel kurallar elde etmek için örnek setini sınıflara ayırır. Kapsama metodu verilen örnek setine dayalı bir kural uzayı hipotezi kurar. Bu kural uzayı hipotezi, arama işlemleri boyunca elimine edilerek en genel kurallar bulunmaya çalışılır. Kural uzayı hipotezi *versiyon uzayı* olarak adlandırılır.

İlk olarak Mitchell [8], kapsama metodunu kullanarak aday-eleme algoritmasını geliştirmiştir. Bu metodu kullanan algoritmalar aşağıdaki mantığa göre işlem yaparlar:

S bir örnek seti ve *H* versiyon uzayı kabul edildiğinde, örnekler sınıflarına göre pozitif veya negatif olarak sınıflandırılırlar. Seçilen örnek ile aynı sınıfta olan örnekler pozitif olarak kabul edilirken, diğer tüm sınıflara ait örnekler negatif olarak dikkate alınır. Başlangıçta *H*, verilen pozitif örneklerdeki tüm mümkün kavramları kapsar. Bu durumda gösterilen örneklerdeki gereksiz aday kavramlar *H*'dan elenerek, sadece bir sınıfa ait kavramların kalması sağlanır. Eleme işlemi şöyle yapılır: Pozitif örneklerde gösterilen ve *H*'yi genelleyen tüm özel kavram tanımlamaları *H*'den silinir. Bu işlem negatif sınıflar üzerinde de tekrarlanarak çok genel kavram tanımlamaları elde edilir. Bu yolla *H*, sadece istenen kavram tanımlamaları kalıncaya kadar kademeli olarak küçültülür. Öğrenilen kavramlar '*Eğer tanım ise karar*' şeklinde kurallar ile gösterilir. Bu algoritma verilen örnek setinden çok sayıda kabul edilebilir kavramı bulabilir. Ancak, çok büyük örnek seti için başlangıçta *H*'ı oluşturmak çok zor, hatta bazı durumlarda imkansızdır.

S bir örnek seti olduğuna göre, pozitif set (S^+) ve negatif set (S^-) olmak üzere bölünebilir. S^+'daki örneklerin hepsi aynı sınıfa aittir ve $S^+ \cup S^- = S$ ve $S^+ \cap S^- = \phi$ olur.

Kapsama metodunu kullanan diğer bir algoritma ise Michalski tarafından geliştirilen AQ ailesi algoritmalarıdır [27]. Bu algoritmalarda kullanılan yöntem aday-eleme algoritmalarındaki ile benzerdir. Farklı olarak başlangıçta oluşturulan *H*, boş tanımlamalar içerir.

Doğrudan kural üreten algoritmalara, AQ Ailesi, CN2, RULES ailesi [28], ILA-1 [29], ILA-2 [30], ve REX Ailesi [31] algoritmaları örnek olarak verilebilir.

4.ENDÜKTİF ÖĞRENME ALGORİTMALARININ PERFORMANSLARINI KARŞILAŞTIRMA

Performans karşılaştırmaları için en yaygın kullanılan algoritmalar ile gerçek hayattan alınmış toplam 11 adet veri seti seçilmiştir [bkz. 32,33]. Bu veri setlerinin kural çıkarma (eğitim seti) ve çıkarılan kuralları test etmek (test seti) için iki farklı seti bulunmaktadır. Bu veri setleri ile ilgili bilgiler Tablo 2'de verilmiştir. Tablo 3, Tablo 4 ve Tablo 5'te verilen REX algoritmalarına ait sonuçlar tarafımızdan, diğer algoritmalara ait sonuçlar ise ilgili referanslardan elde edilmiştir.

4.1. Karar Ağacı Algoritmaları ile Performans Karşılaştırma

Monk1, Monk2 ve Monk3 örnek setleri (bkz. Tablo 2), karar ağacı üreten algoritmalar olan *ID3*, *C4.5*, *C4.5Rules, C4.5Prune, j-Pruned* ile doğrudan kural üreten algoritmalar olan *REX* ailesi algoritmaları kullanılarak üretilen kurallar Tablo 3'te ve doğruluk oranları ise Tablo 4'te gösterilmiştir [bkz. 34,35,36,37]. Bu sonuçlara göre yapılan karşılaştırmalar aşağıda verilmiştir:

- Monk1 problemi için Tablo 3'teki bilgilere göre en az şartla en az kuralı üreten algoritmanın *C4.5Prune* olduğu görülmektedir. Ancak Tablo 4'teki doğruluk oranlarına bakıldığında bu algoritmanın doğruluk oranının %75.7 olduğu ve bu oranın diğer algoritmalara göre düşük olduğu görülmektedir. Monk1 problemi için en yüksek doğruluk oranı değerleri *C4.5* (%93.5) ve REX (%97.2, %97.2, %96.8) ailesi algoritmalarına aittir. Bu algoritmaların kural sayıları ise *C4.5prune* dışındaki diğer algoritmalardan çok daha düşüktür. *C4.5* algoritması 31 adet kural ile %93.5 doğruluk oranına ulaşırken, *REX-1* algoritması 21 kural ile %97.2 doğruluk oranına ulaşmıştır.

Tablo 2. Eğitim ve test veri setlerine ait özellikler [32, 33]

Örnek Seti	Eğitim Seti Örnek Sayısı	Test Seti Örnek Sayısı	Karakteristik Sayısı
Vote	300	135	16
Monk1	124	432	6
Monk2	169	432	6
Monk3	122	432	6
Zoo	67	34	16
Lenses	16	8	4
Parity5+5	100	1024	10
Tic-Tac-Toe	638	320	9
Splice	700	3170	60
Promoter	106	40	57
Iris	70	80	4

- Monk2 problemi için Tablo 3'teki bilgilere göre en az kural üreten algoritmalar sırasıyla *j-pruned (22)* ve *C4.5 Prune (55)* algoritmalarıdır. Tablo 4'teki bilgilere göre bu algoritmaların doğruluk oranları ise *j-pruned* için %55.7 ve *C4.5prune* için %65.0'dır. Oysa, *REX-1* algoritması 78, *REX-2* algoritması 83 ve *REX-3* algoritması ise 79 adet kural üretmiştir. Kural sayılarının diğer iki algoritmadan yüksek olmasına karşılık, doğruluk oranları *REX-1* için %76.2, *REX-2* için %75.7 ve *REX-3* için ise %76.9'dur. Bu bilgiler doğrudan kural üreten *REX* ailesi algoritmalarının doğruluk oranlarının çok daha yüksek olduğunu göstermektedir.
- Monk3 problemine göre en az kuralı üreten *j-Pruned* ve *C4.5Prune* algoritmalarıdır. Bu algoritmalar tarafından üretilen kural sayıları sırasıyla 13 ve 20'dir. Monk3 veri setini, *j-Pruned* %90.9 ve *C4.5 prune* %97.2 doğrulukla sınıflandırmıştır. *REX-1* algoritması ise 26 kural ile %93.5 doğrulukla sınıflandırabilmiştir. Bu bilgiler, Monk3 problemi için *C4.5prune* algoritmasının çok daha iyi bir sonuç ürettiğini göstermektedir.

Tablo 3. Monk problemleri için elde edilen kural sayıları

Algoritma	Monk1	Monk2	Monk3
ID3	53	105	30
C4.5	60	113	27
C4.5 Prune	9	55	20
C4.5 Rules	31	97	23
j-Pruned	15	22	13
REX-1	21	78	26
REX-2	21	83	24
REX-3	22	79	26

Endüktif öğrenme algoritmaları arasında en iyi algoritmalar olarak C4.5 algoritmaları kabul edildiğinden, bu sonuçlara göre REX ailesi algoritmalar ile elde edilen sonuçların çok iyi olduğu rahatlıkla söylenebilir. Çünkü bu algoritmalar verilen eğitim setinden daha az sayıda kurallar üretmekte ve test verilerine göre yüksek doğruluk oranında sonuç vermektedirler.

Tablo 4. Test veri setleri kullanılarak elde edilen doğruluk oranları

Algoritma	Monk1	Monk2	Monk3	Ortalama
ID3	81.00%	69.90%	91.70%	80.87%
C4.5	82.40%	69.70%	90.30%	80.80%
C4.5 Prune	75.70%	65.00%	97.20%	79.30%
C4.5 Rules	93.50%	66.20%	96.30%	85.33%
j-Pruned	67.80%	55.70%	90.90%	71.47%
REX-1	97.20%	76.40%	93.50%	89.03%
REX-2	97.20%	72.20%	89.40%	86.27%
REX-3	96.80%	76.90%	91.00%	88.23%

4.2.Gerçek Veri Setleri ile Performans Karşılaştırma

Tablo 2'deki verileri kullanarak, algoritmalar tarafından her veri seti için elde edilen doğruluk oranları Tablo 5'te verilmiştir [bkz. 29,30,35,37,38]. Elde edilen bu değerler, ±%1 doğruluk oranlarına göre sıralandıklarında en yüksek doğruluk oranını sağlayan algoritma olarak *REX-1* algoritması görülmektedir. Tablo 5'teki koyu (bold) yazılar en yüksek değerleri göstermektedir. Buna göre *C4.5Rules* algoritması 3, *C4.5Prune* algoritması 2, *ID3* algoritması 3, *ILA* algoritması 3, *ILA-2* algoritması 4, *OC1* algoritması 2, *CN2* algoritması 3, *REX-1* algoritması 6, *REX-2* algoritması 4 ve *REX-3* algoritması 3 örnekte en yüksek doğruluk oranlarında kural üretmişlerdir.

Tablo 5'teki örnek setlerinin ortalama doğruluk oranlarına göre en yüksek doğruluk oranı %86.92 ile *REX-1*, ikinci en yüksek doğruluk oranı %85.42 ile *REX-3*, üçüncü en yüksek doğruluk oranı %84.07 ile *CN2* ve dördüncü en yüksek doğruluk oranı ise %84.03 ile *REX-2* algoritmasına aittir.

5. SONUÇLAR

Herhangi bir problemi çözmek için kullanılacak algoritmanın seçimi çok önemlidir. Dikkat edilecek temel iki kavram vardır: Birincisi; örnek setini en iyi sınıflandırabilen başka bir ifade ile doğruluk oranı en yüksek olan algoritmanın, diğeri ise; en az sayıda kural üreten algoritmanın seçilmesidir. Çünkü büyük bir örnek setinden üretilen kuralların çok sayıda olması kaçınılmazdır. Bir örnek verinin bu kurallara göre değerlendirilerek bir sonuç elde edilmesi tüm kuralların ve bu kurallarda yer alan şartların kontrolüne göre yapılır. Bundan dolayı hızlı sonuç üretilmesini gerektiren durumlarda kural sayısının az olması büyük önem taşır.

Endüktif öğrenme alanındaki algoritmalar iki farklı yaklaşım kullanarak kural çıkarma işlemini gerçekleştirmektedir. Karar ağacı yaklaşımını kullanan algoritmalar sadece bir karar ağacına göre kural üretmelerinden dolayı fazla sayıda kural çıkmasına sebep olmaktadır. Bunu engellemek için az öneme sahip olan kurallar budanmaktadır [bkz. 39]. Doğrudan kural üreten algoritmalar ise bilginin değerini ölçmeden kural ürettiklerinden, gereksiz bilgi içeren kuralların çıkmasına sebep olmaktadır. CN2, C4.5 ailesi ve REX ailesi algoritmalarının özellikle entropi ölçüsüne göre kural üretmelerinden dolayı daha iyi sonuçlar ürettikleri görülmektedir.

REX ailesi algoritmalarının sonuçları dikkate alındığında, geliştirilecek yeni endüktif öğrenme algoritmalarında sadece arama yöntemlerini kullanmak yerine, bilgi ölçüsünü kullanmak ve buna göre kurallar elde etmek algoritmanın daha az sayıda kural üretmesi ve daha yüksek doğruluk oranına sahip olması bakımından yararlı olacaktır.

Tablo 5. Algoritmalar tarafından elde edilen doğruluk oranları (%)

Örnek Seti	C4.5 Rules	C4.5 Prune	ID3	ILA	ILA-2	OC1	CN2	REX-1	REX-2	REX-3	*Ortalama*
Vote	95.6	**97.0**	94.1	94.8	**97.0**	**96.3**	95.6	**97.0**	89.6	89.6	*94.66*
Parity5+5	50.0	50.0	50.8	51.2	50.0	52.4	53.0	**64.7**	**63.8**	**64.7**	*55.06*
Monk1	93.5	75.7	81.0	**100**	**100**	91.2	98.6	97.2	97.2	96.8	*93.12*
Monk2	66.2	65.0	69.9	78.5	59.7	**96.3**	75.4	76.4	72.2	76.9	*73.65*
Monk3	96.3	97.2	91.7	88.2	**100**	94.2	90.7	93.5	89.4	91.0	*93.22*
Zoo	85.3	85.3	**97.1**	91.2	88.2	73.5	82.4	**97.1**	85.3	88.2	*87.36*
Lenses	**62.5**	**62.5**	**62.5**	50.0	**62.5**	37.5	**62.5**	**62.5**	**62.5**	**62.5**	*58.75*
Tic-Tac-Toe	**98.1**	82.2	80.9	**98.1**	84.1	85.6	**98.0**	**97.1**	**97.1**	**98.0**	*91.92*
Splice	**92.7**	90.4	89.0	67.9	73.4	91.2	84.5	83.7	83.2	89.0	*84.50*
Promoter	97.5	95.0	**100**	**100**	97.5	87.5	**100**	**100**	**100**	97.5	*97.50*
Ortalama	*83.77*	*80.03*	*81.7*	*81.99*	*81.24*	*80.57*	*84.07*	*86.92*	*84.03*	*85.42*	

KAYNAKLAR

[1]. FEIGENBAUM, E. A., "Expert System in the 1980s", in Infotech State of the Art Report on Machine Intelligence, Ed:A. Bond, Maidenhead, Pergamon, Infotch, 1981.

[2]. BOSE, I., MAHAPATRA K.R., "Business Data Mining – A Machine learning Perspective", Information & Management 39, 211-225, 2001.

[3]. VAPNIK V.N., CHERVONENKIS A.Y., "On the uniform convergence of relative frequencies of events to their probabilities", Theory of Probability and Its Applications, 16(2), 264-280, 1971.

[4]. VALIANT, L.G., "A theory of the learnable", Communications of the ACM, Vol.27, 1134-1142; or, Readings in Machine Learning, (1990) Shavlik, J.W. and Dietterich, T.G. (Eds), Morgan Kaufmann, San Mateo, CA, 192-200, 1984.

[5]. OBLOW, E.M., "Implementing Valiant's learnability theory using random sets", Machine Learning, 8(1), Kluwer Academic Publishers, Boston, 45-73, 1992.

[6]. KRZYSZTOF J. C., "An Algorithm Which Learns Multiple Covers Via Integer Linear Programming Part I: the CLILP2 algorithm", Kybernetes, Vol. 24 No. 2, pp. 29-50, MCB University Press, 0368-492X, 1995.

[7]. MICHALSKI, R.S., "A theory and methodology of inductive learning", Machine Learning - An Artificial Intelligence Approach, Michalski; R.S., Carbonell, J.G. and Mitchell, T.M. (Eds), Morgan Kaufmann, Los Altos, CA, 83-134, 1983.

[8]. MITCHELL, T.M., "The need for biases in learning generalizations", Readings in Machine Learning, Shavlik, J.W. and Dietterich, T.G. (Eds), Morgan Kaufmann, San Mateo, CA, 184-191, 1990.

[9]. QUINLAN, J.R., "Learning efficient classification procedures and their application to chess end games", Machine Learning - An Artificial Intelligence Approach, Eds: Michalski; R.S., Carbonell, J.G. and Mitchell, T.M. (Eds), Tioga Publishing Co, Palo Alto. CA, 463-482, 1983.

[10]. DIETTERICH, T.G., "Limitations on inductive learning", Proceedings of the 6th International Workshop on Machine Learning (89 ML), Ithaca, NY. and Segre, A.M. (Ed), Morgan Kaufmann, San Mateo, CA, 124-128, 1989.

[11]. BOLC, L., CYTOWSKI, J., "Search Methods for Artificial Intelligence", Academic Press, London, 1992.

[12]. HAUSSLER, D. "Quantifying inductive bias: AI learning algorithms and Valiant's learning framework", Artificial Intelligence, 36, 177-221; reproduced in:, Readings in Machine learning, 1990.

[13]. WANG, X., "Inductive Learning Algorithms", Ph.D. Thesis, University of Wales Cardiff, 1997.

[14]. MICHALSKI, R.S., KODRATOFF, Y., "Research in machine learning: recent progress, classification of methods, and future directions", Machine Learning Vol.3, Morgan Kaufmann, San Mateo, CA, 3-30, 1990.

[15]. CENDROWSKA, J., "Knowledge Acquisition for Expert Systems: Inducing Modular Rules from Examples", PhD Thesis, The Open University, 1990.

[16]. BRAMER, M.A., "Automatic Induction of Classification Rules from Examples Using N-Prism",. Research and Development in Intelligent Systems XVI. Springer-Verlag, pp. 99-121, 2000.

[17]. HUNT, E. B., MARIN, J., STONE, P. J., "Experiments in induction", Academic Pess, New York, 1966.

[18]. QUINLAN, J.R., "Induction of decision trees", Machine Learning Vol.1, Kluwer Academic Publishers, Boston, 81-106; reproduced in:, Readings in Machine learning, 1990.

[19]. CHENG, J., et al., "Improved decision trees: A generalized version of ID3", Proceedings of the Fifth international conference on Machine Learning, Ann Arbor, Michigan, 100-106, 1988.

[20]. SCHLIMMER, J.C., FISHER, D.H., "A case study of incremental concept induction", AAAI 86-Proceedings of the 5th National Conference on Artificial Intelligence, Philadelphia, PA, 496-501, 1986.

[21]. UTGOFF, P.E., "ID5:An incremental ID3", Proceedings of the Fifth International Conference on Machine Learning, The University of Michigan, 107-120, 1988.

[22]. UTGOFF, P.E., "Incremental induction of decision trees", Machine Learning Vol.4, 161-186, 1989.

[23]. QUINLAN, J.R., "C4.5: Programs for Machine Learning", Morgan Kaufmann, San Mateo, CA, 1993.

[24]. BREIMAN, L.,et al. "Classification and Regression Trees", Wadsworth International Group, Belmont, California, 1984.

[25]. CRAWFORD, S.L., "Extensions to the CART algorithm", Machine Learning and Uncertain Reasoning - Knowledge-Based Systems, Vol. 3, Gaines, B and Boose, J. (Eds), Academic Press, London, 15-35, 1990.

[26]. ZHONG, N., DONG, J., OHSUGA, S., "Rule discovery by soft induction techniques", Neurocomputing 36, p: 171-204, 2001.

[27]. MICHALSKI, R.S., "Synthesis of optimal and quasi-optimal variable-valued logic formulas", Proceeding of the 1975 Int. Symposium on Multiple-Valued Logic, Bloomington, Indiana, 76-87, 1975.

[28]. AKSOY, M.S., "New Algorithms for Machine Learning", Thesis of PhD, University of Walles, Cardiff, United Kingdom, 1993.

[29]. TOLUN, M. R., ABU-SOUD S.M., "ILA:An Inductive Learning Algorithm For Rule Extraction", Expert Systems With Applications, Vol: 14, p:361-370, 1998.

[30]. TOLUN, M. R., et al., "Improved Rule Discovery Performance on Uncertainty", The Second Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-98), Melbourne, Australia, 15-17 April 1998.

[31]. AKGÖBEK, Ö., "Endüktif Öğrenmede Bilgi Kazanımı için Yeni Algoritmalar", Doktora Tezi, Sakarya Üniversitesi, Adapazarı, 2003.

[32]. BLAKE, C.L., MERZ, C.J., "UCI Repository of Machine Learning Databases", [http://ftp.ics.uci.edu /pub/ml-repos/machine-learning-databases/], 1998.

[33]. SGI Standard Template Library Programmer's Guide, Silicon Graphics Inc., http://www.sgi. com/tech/mcl/db., 1996.

[34]. BRAMER, M.A., "Using J-pruned to reduce overfitting in classification trees", Knowledge-Based Systems, Vol:15, 301-308, 2002.

[35]. WU, X., "Rule Induction with Extension Matrices", Journal of the American Society for Information Science, Volume 49, Vol 5, 435-454, 1998.

[36]. HAMILTON, H. J., et al., "RIAC:A Rule Induction Algorithm Based on Approximate Classification Technical Report", S4S 0A2, CS-96-06, ISSN 0828-3494 ISBN 0-7731-0321-X, 35-37, 1996.

[37]. THRUN, S.B., BALA, J.,"The MONK's Problems A Performance Comparison of Different Learning Algorithms", Carnegie Mellon University, CMU-CS-91-197, 1991.

[38]. AN, A., "Learning Classification Rules From Data", Computers & Mathematics with Applications, Vol 45, Issues 4-5, 737-748, 2003.

[39]. FOURNIER, D., CREMILLEUX, B., "A Quality Index For decision Tree Pruning", Knowledge-Based System 15, 37-43, 2002.