SINIFLANDIRMA KURALLARININ KARINCA KOLONİ ALGORİTMASIYLA KEŞFİ

Bilal ALATAŞ, Erhan AKIN

Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Fırat Üniversitesi, 23119, Elazığ

e-posta: {balatas, eakin}@firat.edu.tr

Anahtar sözcükler: Veri Madenciliği, Sınıflandırma Kuralları, Karınca Koloni Algoritması

ABSTRACT

Methods for mining of classification rules employ a set of classified examples to form a model that can be used as a classifier against new examples. Classifiers can be formed by decision trees, decision lists, evolutionary algorithms, instance based learning, artificial neural networks, Bayesian networks, logistic regressions, and most recently ant colony optimization algorithm (Ant-Miner). In this work, a new ant colony algorithm different from 'separate-and-conquer' approach that is used in Ant-Miner is proposed for mining comprehensible classification rules. Different heuristic and fitness function have been used. The experimental results show that, this method finds more accurate rules than Ant-Miner due to its more global control of direction choosing when constructing rules.

1. GİRİŞ

Veri madenciliği, büyük miktarda veri yığınlarından istatistik, makine öğrenmesi, yapay zeka ve örüntü tanıma yöntemleriyle anlamlı veri ilişkilerinin süzülerek çıkartılması süreci olarak tanımlanmaktadır [1]. Sınıflandırma kurallarının madenciliği de, en çok kullanılan veri madenciliği tekniklerindedir. Sınıflandırmada amaç, yeni bir veri elemanını daha önceden belirlenmiş sınıflara atamak için bir model oluşturmaktır. Veritabanında yer alan örnekler, bir sınıflama fonksiyonu yardımıyla kullanıcı tarafından belirlenmiş ya da karar niteliğinin bazı değerlerine göre anlamlı ayrık alt sınıflara ayrılır. Sınıflama algoritması, bir sınıfı diğerinden ayıran örüntüleri keşfeder. Bu bağlamda kullanılan belki de en önemli değerlendirme kriterleri, tahmini doğruluk ve anlaşılabilirliktir. Tahmini doğruluk genelleme olarak ta bilinir ve oluşturulan modelin daha önce görülmemiş örnekleri sınıflandırmada ne kadar performanslı olduğunun bir ölçüsüdür. Anlaşılabilirlik ise, oluşturulan modelin kullanıcılar tarafından anlaşılabilir olmasını sağlar.

Sınıflandırma için çeşitli yöntemler ve algoritmalar bulunmaktadır. *Karar ağaçları*, sınıflandırma için güçlü bir modeldir. Bunlar, *C4.5* [2] ve *CART* [3] gibi tekniklerle oluşturulur ve 'böl-ve-yönet' stratejisini uygular. Veri ayrı alt kümelere ayrılır ve algoritma her kümeye tekrarlı olarak uygulanır. Karar ağaçlarının en önemli avantajı türetilen bir model olarak karar verme işlemine açık bir şekilde hakimdir. Ancak bunlar, oluşturulmaları sırasında eğitim verisinde örneklerin saf alt kümelerini belirleme eğilimindedir. Bu da yanlış ya da tutarsız olan örneklere aşırı uymaya neden olabilir ve böylece son-modelin genelleme gücünü azaltır. Bu problemin üstesinden gelmek için kural budama ve bunun gibi yardımcı prosedürler kullanılmaktadır.

Karar listeleri de eğitim verisinden çıkarılan bilginin açık bir temsilini belirli şekilde göstermesiyle karar ağaçlarına benzer. Ancak bunlar 'ayır-ve-yönet' yaklaşımını kullanır ve bir kural, eğitim verisinin bir alt kümesini kapsamak için oluşturulur ve sonra daha fazla kural, kalan örnekleri tekrarlı olarak kapsamak için üretilir. Bu strateji, ilk olarak algoritmanın *AQ* ailesinde [4] uygulanmıştır ve daha sonra *CN2* [5] gibi algoritmalara temel teşkil etmiştir. Ayrıca, *Ant-Miner* [6] algoritmasının da temeli bu yönteme dayanmaktadır. Algoritmanın sonunda sıralı *EĞER-O ZAMAN* kurallarının listesi elde edilir ve yeni bir örneğin sınıflandırılmasında sırayla uygulanır. Eğer listedeki ilk kural örneği kapsamıyorsa, yani hem kural hem örnekteki nitelikler için eşleşen değerler yoksa, o zaman bir sonraki denenir. İkincisi de çalışmazsa listedeki üçüncü kural denenir ve böylece devam eder. Bir örnek bir kural tarafından sınıflandırılırsa daha fazla kural denenmez. Eğer kuralların hiçbiri örneği kapsamıyorsa o zaman karar listesinin en altındaki varsayılan bir kural işletilir, yani varsayılan kurala ulaşan tüm sınıflandırılmamış örnekler bu kuralın sınıf etiketiyle işaretlenir.

Sıralı listelerin bir dezavantajı, bireysel kuralların kendilerinin anlaşılma bakımından zor olabilmesinden kaynaklanır. Bir listedeki bir kural, önceki tüm kuralların bağlamında ele alınmalıdır. Karar ağaçları gibi, karar listeleri de gürültülü eğitim verisine aşırı uyma problemiyle karşı karşıyadır ve bu yüzden genellikle kural budama işlemi uygulanır.

Evrimsel hesaplama, özellikle *genetik algoritma* ve *genetik programlama*, da etkili şekilde sınıflandırma

kural madenciliğinde kullanılmıştır [7-8]. Bu yaklaşımla arama uzayı üzerinde global bir arama yapılır ve greedy algoritmalara göre nitelik etkileşimiyle daha iyi baş edilebilir.

Sınıflandırmaya ayrıca *örnek-tabanlı öğrenme*, *yapay sinir ağları*, *lojistik gerileme* ve *Bayesian ağları* yaklaşımları da vardır. Bu metotların çoğunun temel dezavantajı, tahmini doğrulukları bazı durumlarda iyi olmasına rağmen, açıklayıcı güçlerinin eksikliğidir. Literatürdeki bu yöntemlere bazen *bulanık mantık* ta eklenerek bulanık kurallar üretilmiştir.

Kural budama gürültülü eğitim verisine aşırı uymadan kaçınmak için gerekli bir işlemdir. Karar listelerinde kural budama için iki temel strateji vardır. Birincisi komple bir kural kümesi oluşturulur ve sonra nitelikleri kurallardan elimine edilerek ya da bireysel kurallar silinerek kural kümesi basitleştirilir. Bu global olarak kural kümesinin önceden tanımlı bazı budama kriterlerine bağlı olarak optimizesiyle yapılır. İkinci strateji ise artımsal budama olarak adlandırılır, çünkü her kural algoritmayla oluşturulduktan hemen sonra basitleştirilir. *Ant-Miner* ve bu çalışmada ikinci yöntem, artımsal yaklaşım, uygulanmıştır.

Aşağıdaki bölümlerde, karınca koloni algoritması, önerilen yöntem ve deneysel sonuçlar ayrıntılı olarak açıklanmıştır.

2. KARINCA KOLONİ ALGORİTMASI

Karıncalar, yuvaları ile yiyecek kaynağı arasındaki en kısa yolu bulma kabiliyetine sahiptirler ve ayrıca çevredeki değişimlere adapte olabilmektedirler. Örneğin, yuva ile yiyecek arasındaki en kısa yol belirli bir zamanda keşfedilir ve bir yiyeceğe giden yolda herhangi bir problem meydana gelmesi (bir engelin ortaya çıkması gibi) ve yolun kullanılamaz olması durumunda, yeniden en kısa yolu bulurlar. Karıncaların en kısa yolu bulma kabiliyetleri, birbirleri arasındaki kimyasal haberleşmenin bir sonucudur. Karıncalar birbirleriyle haberleşmede feromon olarak adlandırılan kimyasal bir madde kullanmaktadır. Karıncalar ilerlerken yolları üzerine bir miktar feromon maddesi bırakır ve her bir karınca yuva ya da yiyecek bulmak için bir doğrultuyu seçer. Bir yönün seçilme ihtimali, bu yön üzerindeki feromonun miktarına bağlıdır. Her karıncanın, ortalama aynı hızda ve aynı miktarda feromon bıraktığı göz önüne alınırsa, daha kısa yollarda birim zamanda daha çok feromon bulunacaktır. Dolayısıyla, karıncaların büyük çoğunluğu hızla en kısa yolları seçecektir. Bu geri besleme işlemi otokatalitik işlem olarak bilinir. Otokatalitik işlem araştırmanın çok hızlı bir şekilde optimal yola yakınsamasına neden olur ve sonuçta bir alt optimal yola takılmadan yuva ve yiyecek arasındaki en kısa yol bulunur.

Karınca kolonilerinin yuvaları ile yiyecekleri arasındaki bu davranışları Dorigo tarafından problem çözme tekniği olarak kullanılmıştır [9]. Daha sonra birçok arama ve optimizasyon probleminde de uygulanmıştır. Bunun için geliştirilen algoritmaların genel adımları Şekil 1'de gösterilmiştir.

```
Procedure Ant_Colony_Algorithm
Begin
   İlk feromon miktarının hesaplanması
   while(not durdurma_kriteri)
        Aday çözümler oluştur
        Lokal arama gerçekleştir
        Feromonları güncelle
   end while
   return bulunan en iyi çözüm
End
```

Şekil 1. Karınca Koloni Algoritmasının Yapısı

Bütün karıncalar başlangıç safhasında bir ilk çözüm oluşturur. Oluşan bu çözümler, feromon miktarları göz önünde bulundurularak geliştirilir. Her iterasyonun sonunda feromon miktarı güncellenir. Çünkü doğal ortamda karıncaların yolları üzerine bıraktıkları feromon miktarı zamanla azalmakta ya da artmaktadır. İterasyonlar önceden belirlenen durdurma şartı sağlanana kadar devam edecektir.

3. SINIFLANDIRMA KURALLARININ MADENCİLİĞİNDE KARINCA KOLONİ ALGORİTMASI

Karınca algoritmalarının sınıflandırma kural madenciliğinde kullanımı çok yenidir ve yakın zamanda Parpineli ve arkadaşları tarafından *Ant-Miner* [6] önerilmiştir. Daha sonra da Liu ve arkadaşları [10] tarafından aynı mantıkla, ancak farklı sezgisel ve feromon güncelleme stratejisiyle *Ant-Miner* geliştirilmiş. Karınca tabanlı aramanın geleneksel metotlardan daha esnek ve sağlam olduğunu savunmuşlar ve daha global arama yaptıklarını belirtmişlerdir. Ayrıca greedy algoritmalara göre nitelik etkileşimleriyle genetik algoritmalar gibi iyi baş edebilir. Başka bir özellik de karınca algoritması uygulamalarının problem alanıyla minimum bilgi gerektirmesidir.

Bu algoritmada kurallar *EĞER* <şartlar> *O ZAMAN* <sınıf> yapısındadır. <şartlar> ya da ata kısmı, bir nitelik ve bunun alanında özel bir değer mesela $nitelikA=deger_1$ olan terimlerin mantıksal kombinasyonudur. <sınıf> ya da sonuç kısmı ise, özel terimlerin kısmi kombinasyonlarına atanan sınıf etiketini içerir.

Ant-Miner algoritması 'ayır-ve-yönet' mantığıyla çalışır. Önce tüm eğitim verisiyle başlar. Eğitim verisinin bir alt kümesini kapsayan bir 'en iyi' kural oluşturur ve bu en iyi kuralı `Kesfedilen_kurallar_listesi`' ne ekler. Sonra bu kural tarafından kapsanan örnekleri eğitim verisinden kaldırıp azaltılmış bir eğitim kümesiyle tekrar başlar. Bu iş eğitim verisinde sadece birkaç örnek kalana (`KapsananMaksOrn`'ten az) kadar devam eder. Bu aşamada, bu kalan örnekleri

kapsayacak varsayılan bir kural oluşturulur. Şekil 2, algoritmanın aşamalarını göstermektedir.

```
Kesfedilen_kurallar_listesi=[]
                              //Başlangıçta boş
Eğitim kümesi=tüm eğitim örnekleri
while (Eğitim kümesindeki kapsanmayan örnek
      sayısı>KapsananMaksOrn)//KarıncaKoşması
   i=0
   repeat                     //iterasyon
      i=i+1
      Karinca(i) artımsal olarak
       sınıflandırma kuralını oluşturur
      Kuralın sonucunu ata
      Yeni oluşturulan kuralı buda
      Karinca(i) tarafından takip edilen
       yolun feromonunu güncelle
   until (i>=KarincaSayisi) or (Karinca(i)
           bir önceki MaksKuralYakinsa-1
           karınca ile aynı kuralı
           oluştursa)
   Oluşturulan tüm kuralların en iyisini seç
   Kuralı Kesfedilen_kurallar_listesi'ne ekle
   Seçilen kural tarafından doğru olarak
    kapsanan eğitim örneklerini Eğitim
    Kümesi'nden kaldır
end while
Varsayılan kuralı oluştur
Kesfedilen_kurallar_listesi'ni ver
```

Şekil 2. *Ant-Miner*'ın yapısı

Aslında önerilen bu algoritma, 'ayır-ve-yönet' mantığı ve her iterasyonda tek bir karınca koşturmakla diğer karınca algoritmalarından biraz farklıdır.

Bir KarıncaKoşması boyunca her karınca boş bir kuralla (yani şartlar kısmında terim olmayan kuralla) başlar, her seferinde bir terim ekler. Mevcut kısmi kuralın şart kısmına eklenecek bir terimin seçimi, hem sezgisel değere (entropi tabanlı) hem de her terimle ilişkili feromon miktarına bağlıdır. Seçim olasılıksal olarak yapılır ve bağıl olarak yüksek sezgisel ve feromon değerlerine sahip terimlerin seçilmesi daha olasıdır. Bir terim daha önceden seçilmişse tekrar seçilememektedir.

Bir karınca, mevcut niteliklerin her birinden bir terim seçmişse ya da bir sonra eklenebilecek herhangi bir terim önceden tanımlı eşiğin altında, `KuralBasinaMinOrn`, mevcut kuralın ata kısmıyla kapsanan eğitim örneklerinin sayısını azaltacaksa kuralın ata kısmını oluşturmayı durdurur. Bu kriter, eğitim verisine izin verilen aşırı uymanın miktarı üzerinde bir kontrol olarak düşünülebilir. Bu eşik değeri ne kadar büyükse, karıncalar tarafından oluşturulan kuralın ata kısmının o kadar genel olması zorlanır.

Bir karınca, bir kuralın ata kısmını oluşturmayı bitirdiğinde kuralın bir sonuç kısmı seçilir. Bu, sonuç kısmının oluşturulan ata kısım tarafından kapsanan örnekler arasında çoğunluk sınıfına atanmasıyla halledilir. Sonra kural kısaltılarak kalite ve anlaşılabilirliğin artması için budanır. Temel fikir, bir uygunluk fonksiyonu tarafından tanımlanan kural kalitesini geliştirinceye kadar her seferinde bir terimi iteratif olarak kaldırmaktır. Budama işlemi, herhangi bir noktada bir terimin kaldırılması kural kalitesini arttırmıyorsa durdurulur ve bu noktada terimlerin feromon miktarı güncellenir. Ata kısımda yer alan terimler feromon miktarlarını arttırırken diğer terimler azaltır.

Bir karıncanın bir kural oluşturması en fazla, önceden belirlenen karınca sayısı (`KarincaSayisi`) kadar tekrarlanır. Ancak, bu işlem mevcut karınca bir önceki (`MaksKuralYakinsa-1`) kural ile tamamen aynı kuralı oluşturuyorsa da durabilir.

4. YÖNTEM

Bu çalışmada önerilen yöntem 'ayır-ve-yönet' stratejini kullanmaz. Onun yerine, veritabanını azaltmadan her seferinde farklı kaliteli kuralların keşfi için farklı bir yol izler. İlk en iyi kural bulunduktan sonra, bu kuralın kapsadığı örneklerdeki terimlere bir işaret verilir ve bir nevi eğitim örneklerinin dağılımını değiştirir. Bu şekilde *Ant-Miner*'da ortaya çıkabilecek kurallar arasında beklenmedik etkileşimler ortadan kalkacaktır. Bu etkileşimler, eğer bir örnek farklı sınıfların birkaç kuralı tarafından kapsandığı zaman ortaya çıkabilir.

Ayrıca farklı sezgisel ve uygunluk fonksiyonu kullanır. Bu şekilde, daha doğru ve anlaşılabilir kural keşfi amaçlanmıştır. Ayrıca *Ant-Miner* algoritmasındaki keşif-kullanma dengesi eksikliği de kullanılan geçiş kuralıyla giderilmeye çalışılmıştır.

Bu çalışmadaki başka bir farklılık da her iterasyonda tek bir karınca yerine, belli sayıdaki populasyonla aramaya başlanmasıdır. Bir iterasyonda her karınca tek bir kural oluşturur. Budamadan sonra her kural eğitim kümesindeki örnekleri ne kadar iyi kapsadığına bağlı olarak bir kalite değeri alır. Her iterasyonun en iyi kuralı geçici olarak kaydedilir ve bir sonraki iterasyon başlamadan terimlerin feromon miktarları güncellemede kullanılır. Bir KarıncaKoşması ile çalışacak iterasyon sayınını kontrol etmek için iki parametre kullanılır. `MaksKuralYakinsa` ve `MaksIterasyonSayisi`. İlkinde, ardışık *t* ardışık iterasyon boyunca aynı en iyi *t* kural elde ediliyorsa, yani bir gelişme sağlanmıyorsa, KarıncaKoşması durur ve kaydedilmiş her iterasyonun en iyi kuralının en iyisi seçilir ve son kural kümesine eklenir. İkinci terim `MaksIterasyonSayisi` ise eğer bu KarıncaKoşması boyunca `MaksKuralYakinsa` değerine ulaşılmamışsa kullanılır. Ayrıca, ilk feromon değeri belirleme de bu çalışmada farklıdır. Her kural oluştuktan sonra, kural tarafından kapsanan niteliklerin feromon değeri azaltılarak bir sonraki kural arama işlemine başlanır. Algoritmada kullanılan diğer parametre ve terimler aşağıda sırayla açıklanmıştır.

4.1. Sezgisel Fonksiyon

$terim_{ij}$'nin $A_i=V_{ij}$ şeklinde bir kural şartı olduğunu ve A_i'nin *i.* nitelik, V_{ij}'nin de A_i'nin alanının *j.* değeri

olduğunu varsayalım. Mevcut kuralın şart kısmına eklenebilecek her $terim_{ij}$'nin η_{ij} sezgisel değeri vardır. Bu değer, oluşturan kuralın tahmini doğruluğunu geliştirme kabiliyetine bağlı olarak bu terimin kalitesinin bir tahmini hesabını verir. *Ant-Miner*'da entropi tabanlı bir sezgisel fonksiyon kullanılmıştı, ancak bu çalışmada bu kadar kompleks bir sezgisel yerine daha basit bir sezgisel kullanılmıştır. Çünkü, zaten feromon sezgisel değerlerdeki küçük potansiyel hataları düzeltecektir. Kullanılan fonksiyon şu şekilde seçilmiştir:

$$\eta_{ij} = \frac{Çoğunluk\_SınıfıT_{ij} - k \times işaret}{\left|T_{ij}\right|} \qquad (1)$$

Burada T_{ij}, A_i niteliğinin V_{ij} değerine sahip olduğu bölümdür. $Çoğunluk\_SınıfıT_{ij}$ ise T_{ij} bölmesindeki çoğunluk sınıfını sağlayan örnek sayısıdır. *işaret*, algoritma ilk çalıştırıldığında sıfırdır. En iyi kural bulunduktan sonra bulunan bu en iyi kural tarafından kapsanan örnekler işaretlenir ve daha sonra ikinci ve daha sonraki kuralların bulunmasında aynı en iyi kuralın bulunmaması için cezalandırıcı bir faktördür. Yani bu terim, daha önce bulunan kurallar tarafından işaretlenen bir örnekte bulunuyorsa bunun tekrar seçilme şansı biraz azalacaktır. *k* da bu değerin etkisini kontrol için kullanılır. Ayrıca bu sezgisel, sadece algoritma başında hesaplanır, *Ant-miner*'daki gibi her iterasyon başında hesaplanmaz. Zaten kapsanan örnekler işaretlendiğinden sezgisel kolayca hesaplanabilmektedir.

Bu sezgisel değer lokal bir sezgiseldir çünkü bireysel terimlere uygulanır ve nitelik etkileşimlerine hassastır. Bir terimin feromon miktarı, algoritmada diğer bir sezgiseldir fakat daha global etkisi vardır. Feromon miktarı kuralın tamamının uygunluk fonksiyonuna bağlı olarak değişir ve kuralda yer alan nitelikler arası etkileşimi de dikkate alır.

4.2. Uygunluk Fonksiyonu

Bir kuralın kalitesi, *Ant-Miner*'da kullanılana benzer, ancak biraz daha geliştirilmiş bir şekilde belirlenir:

$$Q = \omega_1 \times hassasiyet \times belirlilik - \omega_2 \times uzunluk - \omega_3 \times işaret$$

$$Q = \omega_1 \times \frac{DP}{DP+YN} \times \frac{DN}{DN+YP} - \omega_2 \times uzunluk - \omega_3 \times işaret \qquad (2)$$

- *DP*, doğru pozitiflerdir ve kuralla aynı sınıf etiketine sahip olan, kural tarafından kapsanan örneklerin sayısıdır.
- *YP*, yanlış pozitiflerdir ve kuraldan farklı sınıf etiketine sahip olan, kural tarafından kapsanan örneklerin sayısıdır.
- *YN*, yanlış negatiflerdir ve kural tarafından kapsanmayan fakat kuralla aynı sınıf etiketine sahip örneklerin sayısıdır.
- *DN* ise doğru negatiflerdir ve kural tarafından kapsanmayan ve kuralla da aynı etikete sahip olmayan örneklerin sayısıdır.

Aslında hassasiyet pozitif örnekler arasındaki, belirlilik de negatif örnekler arasındaki doğruluktur. *uzunluk* da kuraldaki terim sayısıdır ve daha anlaşılabilir, basit kuralların keşfi için kullanılır. *işaret* terimi de sezgisel fonksiyonda açıklanan şekilde hesaplanır ve uygunluğa azaltıcı yönde etki eder. ω_1, ω_2 ve ω_3 ise kullanıcı tanımlı ağırlıklardır ve deneysel olarak en etkili değer belirlenebilir.

4.3. Geçiş Kuralı

Geçiş kuralı *Ant-Miner*'dakinden biraz farklı seçilmiştir. *Ant-Miner*'da kullanılan geçiş kuralı şu şekildedir:

$$P_{ij} = \frac{\left[\eta_{ij}\right]^{\alpha} \cdot \left[\tau_{ij}(t)\right]^{\beta}}{\sum_{i=1}^{a}\sum_{j=1}^{b_i}\left(\eta_{ij} \cdot \tau_{ij}(t)\right)}, \quad \forall i \in I \qquad (3)$$

Burada P_{ij}, $terim_{ij}$'nin mevcut kısmi kuralın ata kısmına eklenme olasılığıdır. η_{ij}, $terim_{ij}$ ile ilişkili sezgisel değer; $\tau_{ij}(t)$, *t*. iterasyonda bir $terim_{ij}$ ile ilişkili feromon miktarı; *a*, toplam nitelik sayısı; b_i, *i*. niteliğin alan değerlerinin sayısı; *I* karınca tarafından henüz kullanılmamış niteliklerdir. α ve β ise sırasıyla sezgisel ve feromon değerlerinin bağıl ağırlıklarını kontrol eden parametrelerdir ve *Ant-Miner*'da ikisi de 1 olarak belirlenmiştir. Bu şekilde geçiş olasılığı önceki bilginin kullanımına fayda sağlar ancak (3)'e göre daha önce keşfedilen kurallara ait olan terimlerin seçilme olasılığını arttırır. Böylece karıncaları keşfe doğru bir eğilim sergilemesini yavaşlatır. Keşfin rolünü arttırmak için [10]'da kullanılan geçiş kuralı kullanılmıştır (Şekil 3). Burada q_1 ve q_2 rassal sayı, φ [0, 1] aralığında bir parametre, J_i, *i*. niteliğin değerlerinin sayısı P_{ij} de denklem (3) ile hesaplanan olasılıktır. $q_1 \geq \varphi$ problem hakkında mevcut bilginin kullanımına tekabül ederken, $q_1 \leq \varphi$ daha fazla keşfi tercih eder. φ, keşfi kontrol için ayarlanabilir.

```
if q1 ≤ φ
   repeat
      Pij'leri topla
   until (q2 ≤ Σ_{j∈Ji} Pij)
   terimij'yi seç
else
   Maksimum Pij'li terimij'yi seç
```

Şekil 3. Geçiş kuralı

4.4. Feromon Güncelleme

Bir karınca bir kuralı oluşturmayı tamamladığında, tüm terimlerin feromon miktarı güncellenir. Oluşturulan *R* kuralında yer alan terimlerin feromonu kalitesiyle, *Q*, orantılı olarak artar. Feromon güncelleme kuralı aşağıdadır:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t) \times Q, \quad \forall i,j \in R \qquad (4)$$

Tüm terimlerin feromon miktarları normalize edilir ve bu şekilde *R* kuralında yer alan terimler, feromon miktarlarını arttırırken diğer terimler azaltır. *Ant-Miner*'dan farklı olarak bir iterasyonda birden fazla karınca bulunduğundan, feromon miktarını güncelleme işini populasyondaki en iyi karınca yapar.

5. DENEYSEL SONUÇLAR

Deneylerde UCI veri kümesi ambarından iki veri kümesi kullanılmıştır: Tic-Tac-Toe veritabanı ve Ljubljana Breast Cancer veritabanı. Bu veritabanlarının temel özellikleri Tablo 1'de verilmiştir. Bu veritabanlarının ikisinde de nümerik değerli veriler yoktur. Nümerik değerler içeren veritabanları öncellikle, herhangi bir teknikle ayrıklaştırılmalıdır.

Tablo 1. Veritabanı özellikleri

Veritabanı	Örnek sayısı	Nitelik sayısı	Sınıf sayısı
Ljubljana BC	286	9	2
Tic-Tac-Toe	958	9	2

Önerilen yöntemle *Ant-Miner*'ın performans karşılaştırması için on kere çapraz-geçerlilik testi uygulanmıştır. Her veritabanı on parçaya ayrılmış ver her metot, her seferinde farklı bir bölümü test verisi ve diğer dokuzunu eğitim verisi olarak kullanarak on kere çalıştırılmıştır. Her on on-kez çapraz-geçerlilik testi için algoritmalar aynı verilerle çalıştırılmıştır.

Parametre değerleri olarak populasyon sayısı=40, iterasyon sayısı=75, geçiş kuralındaki φ =0.4, `KapsananMaksOrn`=10, `KuralBasinaMinOr`= 10 ve `MaksKuralYakinsa`=10 olarak seçilmiştir.. Yapılan karşılaştırma sonucu Tablo 2'de gösterilmiştir.

Tablo 2. Tahmini doğrulukların karşılaştırılması

	Ant-Miner		Bu uygulama	
Veritabanı	Tahmini doğruluk	Standart sapma	Ort. tahmini doğruluk	Ort. standart sapma
Ljubljana BC	75.42	10.99	76.21	6.69
Tic-Tac-Toe	73.04	7.60	76.84	4.81

Çalışmada kullanılan yöntemle, ortalama doğruluğu daha fazla kurallar bulunmuştur. Yani, genelleme yeteneği fazla kural kümesi keşfedilmiştir. Bu daha çok önerilen işaretleme metodundan kaynaklanmaktadır. Ayrıca geçiş kuralı da burada etkili olmuştur. Çünkü keşif-kullanma dengesi sağlanmaya çalışılmıştır.

Keşfedilen kural kümesindeki kural sayısı ve terim sayısı ile ilgili karşılaştırmalar da Tablo 3'tedir.

Tablo 3. Kural kümelerinin karşılaştırılması

	Ant-Miner		Bu uygulama	
Veritabanı	Kural sayısı	Terim sayısı	Kural sayısı	Terim sayısı
Ljubljana BC	7.2	9.8	8.4	10.2
Tic-Tac-Toe	8.5	10.0	9.2	10.1

Bu tablodan, önerilen yöntemin daha fazla kural bulduğu görülmektedir. Bu da yine işaretleme yönteminin getirdiği bir sonuçtur. Kural sayısı fazladır, ancak tahmini doğruluk artmıştır. Kuraldaki terim sayısı ise hemen hemen aynıdır. Bu da, kullanılan uygunluk fonksiyonundan ileri gelmektedir.

6. SONUÇ

Yeni veri tiplerinin madenciliği, geniş hacimli ve çok boyutlu veri madenciliği için yeni algoritma, teknik ve sistemler sürekli geliştirilmektedir. Bu çalışmada; esnek, sağlam ve global aramada etkili olan, ayrıca greedy algoritmalara göre nitelik etkileşimleriyle iyi baş edebilen karınca koloni algoritmasıyla sınıflandırma kurallarının daha etkili keşfi yapılmıştır. Önerilen yöntemle, tahmini doğrulukları yüksek anlaşılabilir kurallar keşfedilmiştir.

Bu çalışmada birçok sistem parametresi vardır ve bunların en etkili şekilde belirlenmesi ve bunların sonucu nasıl etkilediğinin testi için ayrıntılı deneyler yapılmalıdır. İleriki çalışmalarda, sürekli değerli veriler üzerinden direkt olarak kural keşfinin yapılması hedeflenmektedir. Ayrıca algoritmanın paralel ve dağıtık versiyonu hazırlanarak diğer veri madenciliği tekniklerinde, daha etkili ve hızlı şekilde kullanılması düşünülmektedir.

KAYNAKLAR

[1] Han J., Kamber M., Data Mining: Concepts and Techniques, Morgan Kaufmann, 2001.

[2] Quinlan J.R., C4.5: Programs for Machine Learning, Morgan Kaufmann, 1992.

[3] Breiman L., Friedman J.H., Olshen R.A., Stone C.J., Classification and Regression Trees, Wadsworth, 1984.

[4] Michalski R.S., On the Quasi-Minimal Solution of the Covering Problem, Proc of the Fifth Int. Symp on Information Processing, pp. 125-128, Bled, Yugoslavia, 1969.

[5] Clark P., Niblett T., The CN2 Induction Algorithm, MACHINE LEARNING JOURNAL, pp. 261-283, The Netherlands, Kluwer, 1989.

[6] Parepinelli R. S., Lopes H. S., Freitas A., An Ant Colony Algorithm for Classification Rule Discovery, In Abbass HA, Sarker RA Newton CS (Ed.), DATA MINING: HEURISTIC APPROACH, Idea Group Publishing, 2002.

[7] Alataş B., Arslan A., Mining of Interesting Prediction Rules with Uniform Two-Level Genetic Algorithm, IJCI Proceedings-Vol 1, Iss 1, pp. 65-70, 2003.

[8] Gündoğan K.K., Alataş B., Karcı A., Mining Classification Rules by Using Genetic Algorithms with Non-random Initial Population and Uniform Operator, TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES, Vol 12, Iss 1, pp. 43-52, 2004.

[9] Dorigo M., Maniezzo V. Colorni A., Positive Feedback As A Search Strategy, Technical Report N.91-016, Politecnico di Milano, 1991.

[10] Liu B., Abbass H.A., McKay B., Classification Rule Discovery with Ant Colony Optimization, IEEE COMPUTATIONAL INTELIGENCE BULLETIN, Vol 3, Iss 1, 2004.