YZM 4008
Veri Madenciliği Dersi Proje Raporu

Algoritma Adı: FDB-RUN-kNN

| Runge Kutta Optimizer with Fitness Distance Balance

Öğrenci Adı ve Soyadı: Fatih Üstün

Öğrenci No: 397204

Öğrenci İletişim Bilgileri

Cep tel: 0539 250 3537

***e-mail adresi: fatihustunx@gmail.com**

| 397204@ogr.ktu.edu.tr

Öz

Sınıflandırma problemleri günümüzde önemli bir yer kaplamaktadır. Fakat sınıflandırma problemlerinde bir sorunla karşılaşmaktayız. Bu sorun problem niteliklerini ölçeklendirme dolayısıyla niteliklerin doğadaki haliyle probleme aktarılamamasıdır. Bu problemin çözümü için nitelik ağırlıklandırma yapılmaktadır. Ayrıca problem boyutu fazla olan problemler için nitelik ağırlıklandırılıp boyut indirgeme yapılarak performans artışı sağlanabilir. Yapay zeka algoritmalarının da her geçen gün geliştirilmesi bu problemlere yeni çözümler sunmamızı sağlar. Bu çalışmamızda fitness distance balance tabanlı runge kutta optimizasyon algoritması kullanılarak sınıflandırma problemlerinde nitelik ağırlıklandırılması yapılarak sınıflandırma problemlerindeki başarım ölçülmüştür. Bu çalışmamızda sınıflandırma algoritması olarak k-nn kullanılmaktadır. Deneysel çalışmalarda ise FDB-RUN tabanlı nitelik ağırlıklandırmasıyla beraber k-nn sınıflandırma algoritmalarında daha etkili sonuçlar elde edildiği gösterilmektedir.

1. Giriş

Yapay zeka uygulamalarında veriler farklı ölçeklerde olduğu için tutarlı, doğru bir sonuç alabilmek için ölçeklendirme yapılır. Fakat yapay zeka uygulamalarında problem verilerimizin homojen olması ve problemi doğal hayatta olduğu gibi temsil etmesi gerekmektedir. Verilerimizde ölçeklendirme yaptığımız için verilerimiz problem uzayını doğal süreçte olduğu gibi temsil edemeyebilir. Bu sebeple problemdeki bağımlı değişkenleri temsil eden bağımsız değişkenlerin ağırlıklandırılması gerekmektedir. Niteliklerin ağırlıklandırılması çok boyutlu problemlerde bağımlı değişkeni en iyi şekilde temsil eden bağımsız değişkenleri de ortaya çıkarmaktadır. Bu sayede çok boyutlu problemlerde bir eşik değeri belirlenerek ağırlıklandırılan nitelikler arasında boyut indirgeme yapılır. Çok boyutlu problemlerde ise boyut indirgeme yapılarak performans artışı sağlanır. Bu süreçte ağırlıkların en optimum şekilde seçilmesi için meta-sezgisel arama algoritmalarına ihtiyaç vardır.

Bu çalışmada melez bir yapay zeka algoritması geliştirilmiş, sınıflandırma problemlerinde nitelikler ağırlıklandırılarak daha iyi sınıflandırma sonuçları elde edilmeye çalışılmıştır. Sınıflandırma algoritması olarak k-nn kullanılmıştır. Meta sezgisel arama algoritması fitness distance balance tabanlı runge kutta optimizasyon algoritması kullanılmıştır. Veri seti olarak ise UCI Machine Learning veri havuzundan Data User Modeling veri seti kullanılmıştır. Bir sonraki bölümde bu çalışmada geliştirilen melez algoritmanın temel öğeleri olan k-nn ve fdb-run algoritması tanıtılmaktadır. Ardından adım adım önerilen yöntem açıklanmıştır.

2. Materyal Ve Yöntem

Bu kısımda geliştirilen melez algoirtmanın temel öğeleri ve kendisi anlatılmaktadır.

2.1 k-En Yakın Komşu Algoritması

k-nn algoritması, basit bir temele dayanır. Temel fikir, bir veri noktasının etiketini o noktaya en yakın k komşusunun etiketleri temelinde tahminlemektir. k-nn algoritmasının ilk adımı, etiketlenmiş eğitim verilerini saklamaktır. Her veri noktası, özellik değerlerinden oluşan bir vektör ve bir etiket içerir.

Sınıflandırmak veya tahminlemek istediğimiz veri noktası ile eğitim verileri arasındaki mesafeler hesaplanır. Öklid bağıntısı genellikle kullanılan bir mesafe metriğidir.

Bu adım, veri noktasının hangi komşulara daha yakın olduğunu belirlemek için önemlidir. Hesaplanan mesafelere göre, veri noktasına en yakın k komşu seçilir. Bu komşular, veri noktasını sınıflandırmak veya tahminlemek için kullanılacak "en yakın komşular" olarak adlandırılır. Sınıflandırma problemleri için, en yakın komşuların etiketleri dikkate alınarak veri noktasının sınıfı belirlenir. Genellikle, çoğunluk oylaması yöntemi kullanılır, yani en çok oylanan etiket veri noktasının tahmini sınıfı olur. k-nn algoritmasının sözde kodu Algoritma 1'de verilmiştir.

Algoritma 1. *k*-nn Algoritmasının Sözde Kodu

1. *Veri setinin tanımlanması:* probleme ait *n*-adet örnek gözlemleri içeren ve problem uzayını temsil etme kabiliyeti yüksek (gözlem uzayını homojen olarak örnekleyen) **X** veri setini oluştur.

2. *Uzaklık bağıntısının belirlenmesi:* gözlemler arasındaki uzaklıkların hesaplanmasında kullanılacak yöntemi belirle.

3. *k-değerinin belirlenmesi:* gözlem sayısına ve veri setinin karakteristiğine bağlı olarak k-komşu sayısı için arama uzayının sınırlarını tanımla.

4.
```
for each kj
    kj için sınıflandırma performansını SPkj = fk-nn(kj)
        if (SPkj > SPkj-1)
            k= kj
        end if
end
```

5. En iyi sınıflandırma performansı sağlayan *k*-değerini kaydet

6. Sınıf etiketi belirlenecek olan *q* sorgu gözlemini tanımla

7.
```
for i=1:n
D[i]=q ile Xi arasındaki uzaklığı hesapla
end
```

8. Xq[k]=D[i] uzaklık dizisinden *q* sorgu gözlemine en yakın *k*-adet gözlemi belirle

9. Xq[k] gözlemlerinin sınıflarını dikkate

2.2 Fitness Distance Balance Bazlı Runge Kutta Optimizasyon Algoritması

Runge-Kutta optimizasyon algoritması, Runge-Kutta metodunun optimizasyon problemlerine uyarlanmış bir versiyonudur.

Runge-Kutta optimizasyon algoritması, bir hedef fonksiyonun minimum veya maksimum değerini bulmak için kullanılan bir optimizasyon yöntemidir. Bu algoritma, sayısal diferansiyel denklemleri çözmek için yaygın olarak kullanılan Runge-Kutta metoduyla birlikte gradyan tabanlı optimizasyon yaklaşımını birleştirir.

Optimizasyon probleminin başlangıç noktası belirlenir. Bu nokta genellikle rastgele seçilir veya başka bir optimizasyon yöntemiyle elde edilir. Başlangıç noktasında hedef fonksiyonun değeri hesaplanır. Hedef fonksiyonun gradientı veya gradyan yaklaşımı hesaplanır. Gradient, fonksiyonun her bir parametreye göre türevidir ve fonksiyonun artış veya azalış yönünü gösterir. Gradientı veya gradyan yaklaşımı kullanılarak, adım büyüklüğü ve yönü belirlenir. Adım büyüklüğü, her iterasyonda alınacak adım miktarını belirler. Belirlenen adım büyüklüğü ve yönü kullanılarak, yeni bir nokta hesaplanır. Bu nokta, optimizasyon probleminin bir sonraki adımındaki parametre değerlerini temsil eder. Yeni noktadaki parametre değerleri kullanılarak, hedef fonksiyonun değeri yeniden hesaplanır. Hedef fonksiyonun değeri bir önceki adımdakinden daha iyi ise tekrardan adım büyüklüğü ve miktarı hesaplanılarak işlemler tekrar edilir fakat daha iyi değilse optimizasyon tamamlanır.

Runge-Kutta optimizasyon algoritması, gradient hesaplaması için Runge-Kutta metodu kullanır. Bu, gradient hesaplamasını daha doğru ve istikrarlı hale getirebilir. Runge-Kutta optimizasyon algoritması, genellikle karmaşık ve çok değişkenli optimizasyon problemlerinde kullanılır.

Runge-Kutta optimizasyon algoritmasında arama sürecine yön veren çözüm adayları Fitness-Distance Balance yöntemi kullanılarak geliştirilmiş ve ortaya FDB-RUN algoritması ortaya çıkmıştır.

Fitness-Distance Balance yöntemi ise rehberlik seçimi sırasında adayların sadece fitness değerine bakılmaksızın uzaklık değerleri de hesaba katılarak çeşitlilik ve dinamiklik sağlayan bir yöntemdir.

FDB-RUN algoritmasının sözde kodu Algoritma 2'de verilmiştir.

Algoritma 2: *FDB-RUN* Algoritmasının Sözde Kodu

Başla
Problemin yaratılması (uygunluk fonksiyonunun, ceza fonksiyonunun tanımlanması)
Çözüm adayının tasarımı ve çözüm adayları topluluğunun yaratılması
Çözüm adaylarının uygunluk değerlerinin hesaplanması
İteratif süreç (sonlandırma kriteri sağlanıncaya kadar devam et: amaç fonksiyonu azami değerlendirme sayısı)
    Çözümlerin ortalaması ve uyarlanabilir faktör belirlenir.
    For i=1: nP
        Runge-Kutta yöntemine göre yeni çözüm belirlenir. (İt++)
        Çözüm kalitesi geliştirilir. (it+2)
        En iyi çözüm hesaplanır.
    End
Her iterasyonda en iyi çözüm kaydedilir.

2.3 Önerilen Yöntem: FDB-RUN-kNN ile Nitelik Ağırlıklandırma

Bu yöntem veri setindeki niteliklere ağırlık katsayısı vererek hangi niteliklerin diğerlerine göre daha etkin nitelikler olduğunu bulmayı amaçlar. FDB-RUN özelliklerinden olan gradient hesaplama sayesinde kullanılan diğer boyut azaltma veya nitelik ağırlıklandırma yöntemlerinden daha iyi sonuçlar üretebilir.
Popülasyonun oluşturulmasıyla başlayan yöntem sınır değerler arasında rastgele çözümler üretir. Bu problemde ağırlık sınırları [0,1] olacaktır. FDB-RUN işlemleri doğrultusunda çözüm adayları yani nitelik ağırlıkları sırasıyla amaç fonksiyona gönderilir. Amaç fonksiyonu olarak kNN kullanılır. kNN'de tahmin edilen sınıf değerleri asıl sınıf değerleri ile karşılaştırılır. Yapılan yanlış tahmin sayısının toplam tahmin edilen sınıf sayısına bölümü ile oluşturulan hata değeri gönderilen çözüm adayının uygunluk değeri olarak kaydedilir. Algoritma sonucunda ise en optimum ağırlık değerleri ve en iyi fitness değeri çıktı olarak verilir. Önerilen yöntem olan FDB-RUN-kNN algoritmasının sözde kodu Algoritma 3' te verilmiştir.

Algoritma 3: FDB-RUN-kNN Algoritmasının Sözde Kodu

1. **Başla**
2. P: Problemin *n* nitelikli ağırlık dizisini temsil eden çözüm adaylarından rastgele bir popülasyon oluşturulur
3. **for** i = 1:*n*
4. F: Her bir çözüm adayının ağırlık dizisini *k*-nn'e göndererek uygunluk değeri (hata değeri) hesaplanır
5. end for
6. //Arama süreci yaşam döngüsünün başlangıcı
7. **while** (G1'den Gmax'a (maksimum uygunluk değerine) kadar git)
8. **for** i = 1:*n*
9. Runge-Kutta yöntemine göre yeni çözüm belirlenir.
   Yeni çözümün fitness değeri kNN ile hesaplanır. (it++)
10. Çözüm kalitesi geliştirilir.
    Maximum 2 defa kNN çağrılabilir. (it+2)
11. En iyi çözüm hesaplanır
12. end for
13. Her iterasyonda en iyi çözüm kaydedilir.
18. **Bitir**

3. Deneysel Sonuçlar

Önerilen yöntemin sonuçlarını görebilmek adına 1 veri seti ile karşılaştırma yapılmıştır. Bu karşılaştırma FDB-RUN-kNN algoritmasının farklı iterasyon sayılarında çalıştırılıp kendi aralarında karşılaştırılması ve daha sonra FDB-RUN uygulanmamış yanlızca KNN ile tahmin performansı ile FDB-RUN-kNN sezgisel nitelik ağırlıklandırma tahmin performansı karşılaştırılmıştır.

Sonuçları elde etmede kullanılan algoritma ayarları verilmiştir

FDB-RUN-kNN	K komşu sayısı=3 Uzaklık Hesaplama Bağıntısı= öklid
Popülasyon Büyüklüğü= 30	
K-NN	K komşu sayısı=3, Uzaklık Hesaplama Bağıntısı= ÖKLİD

Kullanılan veri setinin özellikleri Tablo 1'de verilmiştir.

Tablo 1	Boyut	Eğitim Örnek Sayısı	Test Örnek Sayısı
Örnek Veri Seti	5	258	145

Algoritmaların karşılaştırmaları sınıflandırma hata değerlerine göre yapılmıştır. Hata değeri her yanlış sınıf tahmininin toplam sınıf tahminine oranına göre hesaplanmıştır.Hata değerleri yüzde cinsinde çevrilmiştir.

Algoritma	FDB-RUN-kNN			kNN
İterasyon	100	500	1000	
En İyi	%19	%19	%17	
En Kötü	%21	%20	%20	
Ortalama	%20	%20	%18	
Sonuç				%25.52

4. Sonuç Ve Tartışma

Bu çalışmada FDB-RUN-kNN melez algoritması kullanılarak sınıflandırma problemi parametreleri için ağırlıklandırılma yapılmıştır. Sonuçlarda görüldüğü üzere k-nn algoritması ortalama %25.52'lik bir hata ile sınıflandırma yaparken FDB-RUN-kNN algoritması ise en iyi %17, ortalama %20'lik bir hata yüzdesi ile sınıflandırma gerçekleştirmiştir. Görüldüğü üzere %20 oranında bir iyileşme söz konusudur. Bu durum problem niteliklerinin ağırlıklandırılarak bağımlı değişkeni temsil eden bağımsız değişkenlerin öne çıkmasına, niteliklerin doğal yaşamda olduğu gibi problemi temsil etme ve problem verilerinin homojen bir şekilde ele alınmasına katkı sağlamıştır.