Yapay Zekâ Dersi Projesi

Mastermind

Rana Demir

I. GİRİŞ

*A. Problem Tanımı*

Temelde, problem şundan ibaret: Rastgele bir dizilim mevcut. Bu dizilimi kod oluşturucu belirliyor. Bunu bulabilmek adına adım adım dizilim tahminleri yapılıyor. Tahminleri yapana ise kod kıran adı veriliyor. Bu tahminlerin her biri orijinal dizilimle karşılaştırılarak da özelliklerine göre göstergeler elde ediliyor. Bu göstergeler kullanılarak orijinal dizilim bulunmaya çalışılıyor.

Elde toplamda 8 renk bulunuyor. Her renkten yeterli sayıda kutu olduğu farz ediliyor. Bu varsayım altında anlaşılıyor ki bir dizilim; tek bir renkten de oluşabilir, tamamı farklı renklerden de oluşabilir. Bu durum hem tahmin dizilimleri hem de oluşturulacak orijinal dizilim için geçerli.

Simülasyonu gerçekleştirilecek durum, iki temel kısımdan oluşuyor:

1) Dizilim Oluşumu: Bu kısım, orijinal dizilimin oluşturulduğu bölüm. Eldeki tüm kutulardan istenilen dört tanesi kullanılarak tahmin edilecek bir dizilim oluşturuluyor.

2) Dizilim Tahmini: Bu kısımda, ilk kısımda oluşturulan orijinal dizilim saptanana kadar art arda tahminler yapılıyor. Bu tahmin dizilimlerinin her birinin sonucunda, o dizilimdeki her bir kutu için en fazla üç durum söz konusu olabiliyor. Bunlar, o kutu için kırmızı gösterge çıkması, beyaz gösterge çıkması veya hiçbir gösterge çıkmaması durumlarından oluşuyor. Yeni oluşturulan tahmin diziliminde, orijinal dizilimde bulunan her bir renk için iki göstergeden biri beliriyor. Orijinal ve tahmin dizilimlerindeki aynı olan bu renk iki dizilimde de aynı konumda bulunuyorsa bu gösterge kırmızı oluyor. Orijinal ve tahmin dizilimlerindeki aynı olan bu renk iki dizilimde farklı konumlarda bulunuyorsa bu gösterge beyaz oluyor. Tahmin diziliminde bulunan herhangi bir renk orijinal dizilimde mevcut değilse hiçbir gösterge belirmiyor.

Örneğin, herhangi bir oyundaki orijinal dizilim Figür 1’de, bu dizilim için yapılmış olan bir tahmin ise Figür 2’de bulunuyor. Buna göre önce aynı renkte ve aynı yerde kutu var mı, diye bakılıyor. İki dizilimde de ikinci sırada mavi renkte kutular olduğu için bir kırmızı gösterge elde ediliyor. Daha sonra aynı renk ve yerde olan başka kutu kalmadığı için sıradan bakılıyor. Tahmin diziliminde olan pembe kutu, orijinalde olmadığı için gösterge çıkmıyor. Maviye zaten bakıldığı için mor kutuya geçildikten sonra bunun da orijinal dizilimde yer almadığı saptanıyor ve gösterge çıkmıyor. Son olarak yeşil kutuya bakıldığında orijinal dizilimde yer aldığı fakat farklı yerde bulunduğu için beyaz gösterge elde ediliyor. Sonuç olarak bir kırmızı, bir beyaz gösterge elde edilmiş olunuyor.

Figür 1. Orijinal dizilim Figür 2. Tahmin dizilimi

*B. PEAS Analizi*

Başarım Ölçütü: Orijinal dizilime ulaşmanın kaç tahmin sürdüğü ele alınıyor.

Çevre: Önceden yapılmış tahmin dizilimleri, orijinal dizilim ve kalan olası dizilimler

Eyleyiciler: Tahmin yapan fonksiyon

Alıcılar: Orijinal dizilimle tahmin diziliminin karşılaştırılması sonrası oluşan göstergeleri söyleyen fonksiyon

*C. Çevre Özellikleri*

Orijinal dizilimin ne olduğu, tasarlanan algoritmanın bilgisi dahilinde değil. Bu yüzden kısmi gözlemlenebilir bir çevrenin söz konusu olduğu söylenebilir.

Çevre, gösterge sayısını ve varlığını da kapsadığından dolayı yapılacak tahminler ilerideki tahminleri de etkiliyor. Bu yüzden rastgele olmayan bir yapı mevcut.

Aracının hareketi yani tahmini gerçekleştirme hareketi, sadece o tahminin yapıldığı bölüme değil, ondan önceki bölümlere de bağlıdır. Bu yüzden takip eden bir problem söz konusu.

Aracı tarafından tahminin yapıldığı süreç içerisinde çevrenin özelliklerinde herhangi bir değişim meydana gelmiyor. Yani değişmez bir problem.

Aracı, yaptığı tahminler sonucu elde ettiği göstergeleri kullanarak yeni tahminler yaptığı için sürekli bir çevre var.

Temelde yalnızca tahminin yapıldığı algoritma mevcut. Yani tek aracı bulunuyor.

*Ç. Problemin Çeşidi*

Problemin çevre özellikleri de göz önüne alındığında bu problemin gerektirdiği, adım adım tahminler yapmak ve bu tahminleri önceki tahminlere ve elde edilen göstergelere göre kısıtlayarak oluşturmak. Bu noktada, problemin bir kısıt sağlama problemi olduğu belirleniyor.

II. LİTERATÜR TARAMASI VE ÇÖZÜM YOLLARI

*A. Kısıt Sağlama Problemi Nedir?*

Bir kısıt sağlama probleminde, tanım kümesini bildiğimiz değişkenler ve bu değişkenlerin değerlerini sınırlayan kısıtlar bulunuyor. Buradaki görev ise etkili bir biçimde her değişken için kısıtları sağlayan birer değer atayabilmek [1].

Kısıt sağlama problemini tanımlamak için gerekli olanlar:

Değişken seti X = {,…,}

Değişkenlerin tanım kümeleri D = {,…,}

Kısıtlar seti K = {,…,}, her biri herhangi bir sayıda değişken barındırabilir:

: kuvvet kümesi(D) → {doğru, yanlış}, k = {1, 2, …, m}

*B. Mastermind Problemi Nasıl Çözülür?*

Her tahminde daha da iyi bir yaklaşımda bulunmak için kısıt sağlama optimizasyonu kullanılması gerekiyor. Bunun gerçekleştirilmesi aşamasından önce benzer yapıdaki problemlere ve onların çözümlerinde kullanılan yaklaşımlara göz atmak faydalı olacaktır.

*C. Benzer Kısıt Sağlama Problemleri ve Yaklaşımlar*

1) Harita Renklendirme: Belirli bir sayıda renk ile bir harita boyanması amaçlanıyor. Bu esnada haritadaki komşu her bir bölgenin rengi birbirinden farklı olmalı.

Her bölgenin alabileceği renk setleri mevcut. Bu setler, komşu bölgeler için her boyamadan sonra güncelleniyor. Boyama sırası ise şu mantık izlenerek belirleniyor: Her adımda en fazla kısıda sahip bölge seçiliyor. Bu yaklaşıma “minimum kalan değerler sezgisel yöntemi” adı veriliyor. Bahsi geçen bölgeden birden fazla olduğu durumda ise bunlardan komşu bölgelerini en az sınırlayacak bölge seçiliyor. Bu yaklaşıma “rütbe sezgisel yöntemi” adı veriliyor. Daha sonra o bölge, en az yeri sınırlandıracak şekilde boyanıyor. Böylece en kötü durum, olabilecek en iyi şekilde engellenmiş oluyor.

2) Ders Programı Hazırlama: Genel bir ders programı hazırlama probleminde; ders ve sınavlar belli zaman aralıklarına yerleştiriliyor. Bu yerleştirme işlemi yapılırken birtakım kısıtların göz önünde bulundurulması gerekiyor. Bu da bir gerçek hayat problemi olarak ele alındığında işi zorlaştıran bir durum hâline gelebiliyor [2].

Her durum için her dersin ve sınavın konulabileceği saatler ve günler seti mevcut. Bu setler, herhangi bir ders, uygun bir saate yerleştirildikten sonra boşta kalan tüm dersler için güncelleniyor. Bu güncellemeleri yapabilmek içinse genelde sezgisel yöntemler tercih ediliyor.

Bahsi geçen iki problemde de belirli bir mantığa göre değer atanacak değişkenler belirleniyor. Bu belirleme gerek kısıt sağlama optimizasyonu gerekse sezgisel yöntemlerle yapılabiliyor. Elimizdeki probleme daha yakın olan problem harita renklendirme olduğu için bunda kullanılan yönteme çok benzer bir yol izleme kararı alınıyor.

III. ÇÖZÜM YOLU

*A. Kurulan Mantık*

Mevcut herhangi bir tahmin diziliminden, orijinale bakılarak bir gösterge elde ediliyor. Bu gösterge, şu şekilde kullanılıyor: Eğer tahmin, orijinal dizilim olsaydı; orijinal ise mevcut tahmin olsaydı aynı gösterge elde edilirdi. O hâlde şu sonuç çıkarılıyor: Orijinal olabilecek tüm tahmin dizilimleri de mevcut tahmin dizilimi ile aynı göstergeyi vermeli. Bu yüzden tüm olası dizilimleri tutacak bir veri yapısı tutulmasına karar veriliyor. Yapının her tahmin sonucu giderek küçülmesi planlanıyor.

*B. Genel Algoritma Yapısı*

D yapısı oluştur, tüm olası tahminleri burada tut.

Birbirinden farklı iki A ve B renklerini rastgele seçerek {A, A, B, B} şeklinde ilk tahmini yap.

Her tahmin için amaç testi beklenen sonucu vermediği sürece

D yapısındaki tüm dizilimleri gez

Her dizilim için mevcut tahmini orijinal gibi alarak gösterge elde et.

Bu gösterge, eldeki göstergeden farklı ise bu dizilimi D yapısından çıkar.

Güncellenmiş D yapısından yeni bir dizilimi rastgele seçerek yeni tahmin olarak ata.

*C. Algoritmada Kullanılan Setler*

Mevcut renkler seti, R = {Mavi, Pembe, Yeşil, Mor, Sarı, Kırmızı, Beyaz, Siyah}. Bu setin adı programda “colors” olarak geçiyor. Bu, temel bir yapı olarak hem orijinal dizilimi hem de tüm tahmin dizilimlerini belirlerken kullanılıyor. Neticede rastgele veya bilinçli seçilen tüm kutular, bu renk setine ait oluyor.

Tüm olası dizilimler, D = {[Mavi, Mavi, Mavi, Mavi], …, [Siyah, Siyah, Siyah, Siyah]}. Bu yapının adı programda “all\_possibilities” olarak geçiyor. Başta boş bir şekilde oluşturulduktan sonra henüz orijinal dizilim bile belirlenmeden tüm renkler dört kez gezilerek olası tüm dizilimler bu sete yerleştiriliyor. Böylece dört tane aynı renkten oluşan dizilimlerden dört adet farklı renkten oluşan dizilimlere kadar her olası tahmin ele alınıyor.

*Ç. Algoritmada Kullanılan Fonksiyonlar*

Tüm Olasılıkları Oluşturma: Programdaki adı “create\_possibilities”. Boş bir set ve kullanılacak renk sayısını parametre olarak alıp bu renklerle oluşabilecek tüm olası dizilimleri sete aktarıyor. Böylece bu renkleri kapsayan tüm olası dizilimler elde edilmiş oluyor.

Orijinal Dizilimi Oluşturma: Programda adı “create\_sequence” olarak geçiyor. R setinden dört kere renk seçerek rastgele bir dizilim elde ediyor. Bu dizilim, dört aynı renkten oluşabildiği gibi dört farklı renkten de oluşabiliyor.

İlk Tahmin: Programdaki adı “first\_prediction”. R setinden, birbirinden farklı iki A ve B renklerini rastgele olarak seçiyor, {A, A, B, B} olacak şekilde ilk tahmini gerçekleştiyor.

Tahmin: Programda “predict” olarak geçiyor. İlk tahminden sonraki tüm tahmin durumları için bu fonksiyon kullanılıyor. O adıma kadarki tüm geçerli tahminleri kapsayan D seti içerisinden rastgele bir dizilim seçip bunu yeni tahmin olarak belirlemiş oluyor.

Gösterge Bulma: Programdaki adı “compute\_signs”. Parametre olarak orijinal dizilimi, tahmin olarak tutulan dizilimi alıyor. Eldeki tahminin orijinal dizilime göre olan göstergelerini veriyor. Bunu yaparken de önce orijinal dizilimde ve tahmin diziliminde aynı yerde aynı renk var mı, diye bakıyor. Varsa bunu kaydedip devam ediyor. Bu kaydedilen yerlere tekrar bakmıyor. Böylece tahmin dizilimindeki bir kutu için hem beyaz hem de kırmızı gösterge koyma olasılığını engellemiş oluyor. Kaydedilmeyen yerlere de bakarak o renk orijinal dizilimde varsa beyaz gösterge sayısını bir arttırıyor, yoksa değiştirmiyor. Fonksiyon sonucunda elde edilen değer şu şekilde oluyor: [Kırmızı gösterge sayısı, Beyaz gösterge sayısı, Toplam gösterge sayısı].

Amaç Testi: Programda “prediction\_accuracy” şeklinde geçmekte olan bu fonksiyon, parametre olarak aldığı mevcut tahminin göstergelerine bakıyor ve bunların dört kırmızı göstergeden oluşması durumunda oyunun bittiğini söylüyor.

Olasılıkları Güncelleme: Programda “update\_possibilities” adıyla bulunabiliyor. Parametre olarak tahmin dizilimini ve bu dizilimin orijinal dizilime bakılarak elde edilmiş göstergelerini alıyor. Burada kullanılan mantık daha ayrıntılı irdelendiğinde şöyle söyleniyor: Biliniyor ki eldeki gösterge, tahmin diziliminin orijinal dizilime bakılarak çıkarılmış göstergesi. Bu gözlem, tersinden değerlendirildiğinde; yani orijinal dizilimin tahmin dizilimine bakılarak göstergeleri çıkarılmış olsaydı aynı göstergelerin elde edileceği gözlemleniyor. Öyleyse bu mantık, eldeki tüm olasılıklara uygulandığında bunlardan birinin orijinal dizilim olduğu bilindiğinden her tahminde belirli sayıda olasılık elenmiş oluyor ve geriye orijinal dizilimin de içinde bulunduğu daha dar bir küme kalıyor. Başka bir deyişle tüm olasılıkların eldeki tahmine göre göstergeleri saptandığında bu göstergelerin orijinal göstergeyle aynı olması bekleniyor. Aynı ise orijinal dizilim olma ihtimali olduğundan yapıda tutuluyor. Farklı ise orijinal dizilim olamayacağından tüm olasılıklar setinden çıkarılıyor. Böylece her tahmin sonucunda olasılıklar daralacak şekilde güncellenmiş oluyor.

Oyun Simülasyonu: Bundan önce bahsedilmiş olan tüm fonksiyonları kullanarak oyunun gerçekleştirimini sağlayan fonksiyon, programda “game” olarak geçiyor. Öncelikle renk sayısına göre tüm olasılıkları oluşturup D yapısını elde ediyor. Daha sonra orijinal dizilimi oluşturup ilk tahmini yapıyor. Buradan sonra ise ilki de dahil her tahminde amaç testinin gerçekleşip gerçekleşmediğini kontrol ediyor. Oyun bitmediği sürece tüm olasılıkları yani D yapısını, yaptığı tahmine göre güncelleyip yeni tahminler geliştiriyor. Bir yandan da adım sayısını arttırarak bunun kontrolünü sağlıyor. Bu fonksiyonun bir diğer türevi de programda “game\_only\_rounds” şeklinde geçmekte. Bu fonksiyon ise çoklu denemelerde kullanılıyor. İkisinin farkı, ilkinde her buluntu ekrana bastırılırken ikincisinde yalnızca adım sayısı elde ediliyor ve ekrana hiçbir şey bastırılmıyor.

Denemeler: Programda “execute” adında olan bu fonksiyon, oyun simülasyonunun ortalama kaç adımda başarıya ulaştığını ortalamalarını hesaplayarak elde ediyor. Parametre olarak kaç renk kullanılacağını ve kaç deneme yapılacağını alıyor. Bu deneme adedince denemelerde kullanılan oyun simülasyonu fonksiyonunu çağırıyor. Bunların da ortalamasını alıp ortalama bir başarı ölçütü sağlıyor.

IV. SONUÇ, ÇIKARIMLAR VE ANALİZ

*A. Genel Başarı*

Temel amacımız 10’un altında tahminle doğru dizilime ulaşmak olduğundan bu başarının yakalandığı Tablo I üzerinde gösteriliyor. Burada, art arda 20 kez çalıştırılmış oyunun sonucu veriliyor. Tüm oyunlarda da oluşturulan rastgele dizilimler; A, B, C ve D harflerinin her biri farklı birer rengi temsil edecek şekilde gösteriliyor. Buradan da anlaşılacağı üzere genelde 4 ile 8 aralığında bir adım başarısı sağlıyor. Ayrıca, en büyük adımların genel olarak dört farklı renk bulunduğunda meydana çıktığı söylenebilir.

TABLO I

RASTGELE 10 DENEMENİN BAŞARIMI

|  |  |  |
| --- | --- | --- |
| Deneme sırası | Başarım (adım sayısı) | Orijinal Dizilim |
| 1 | 5 | {A, A, B, B} |
| 2 | 5 | {A, B, C, D} |
| 3 | 5 | {A, B, C, A} |
| 4 | 6 | {A, B, C, D} |
| 5 | 5 | {A, B, C, D} |
| 6 | 6 | {A, A, B, C} |
| 7 | 6 | {A, B, B, C} |
| 8 | 7 | {A, B, C, D} |
| 9 | 3 | {A, A, B, C} |
| 10 | 4 | {A, B, C, A} |

*B. Farklı Renk Setleriyle Denemeler*

Mevcut problemde 8 adet renk mevcut. Fakat algoritmanın daha kapsamlı bir gözlemini yapabilmek adına 8 ile birlikte 6 ve 10 renk ile de denemeler yapılıyor. Bu durumlar için gerekli kodlar, programın sonunda yorum şeklinde bulunuyor. Bu denemelerin ardından elde edilen sonuçlar, Tablo II üzerinde gözlemlenebilir. Doğal olarak kullanılan renk sayısı, yani problemin mevcut uzayı, arttıkça daha fazla adımda doğru tahmine ulaşılıyor. Sekiz renk içeren ana versiyondaki amacın (on veya daha az adımda dizilimi başarıyla tahmin etme) on renk için de sağlandığı açıkça görülebiliyor.

TABLO II

RENK SETLERİNDE PERFORMANSLAR

|  |  |  |  |
| --- | --- | --- | --- |
| Deneme sayısı | Kullanılan renk sayısı | | |
| 6 | 8 | 10 |
| 20 | 4.75 | 5.85 | 6.0 |
| 50 | 4.66 | 5.26 | 6.4 |
| 100 | 4.55 | 5.34 | 6.23 |
| 500 | 4.578 | 5.498 | 6.324 |
| 1000 | 4.563 | 5.424 | 6.347 |

*C. Geliştirilebilir Yönler*

Mevcut versiyonda tüm olasılıkların bulunduğu ve adım adım güncellendiği bir veri tutuluyor. Bu güncelleme işlemi yalnızca o ana kadar yapılan tahminler göz önünde bulundurularak yapılıyor. Fakat bunun yerine bir sonraki adımda yapılacak tahmin olarak tüm olası dizilimler için olasılık hesaplaması yapılarak gelecek adımda seçilecek tahmin dizilimi, daha net bir şekilde kararlaştırılabilir.

KAYNAKÇA

1. Sally C. Brailsford, Chris N. Potts ve Barbara M. Smith, (16.12.1999). “Constraint satisfaction problems: Algorithms and applications”. *European Journal of Operational Research,* 119 (3), s. 557-581
2. Edmund K Burke, Barry McCollum, Amnon Meisels, Sanja Petrovic, Rong Qu, (2007). “A Graph-Based Hyper-Heuristic for Educational Timetabling Problems”. *European Journal Operational Research*, 176, s. 177-192.