DOKUZ EYLÜL ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

POPÜLERLİK SIRALAMASINA DAYALI DC^{3}MAB ÖNERİ SİSTEMİ VE UYGULAMASI

Najibeh MIRINEZHAD

Şubat, 2024

İZMİR

POPÜLERLİK SIRALAMASINA DAYALI DC^{3}MAB ÖNERİ SİSTEMİ VE UYGULAMASI

Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi
İstatistik Anabilim Dalı, Veri Bilimi Programı

Najibeh MIRINEZHAD

Şubat, 2024

İZMİR

YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

NAJIBEH MIRINEZHAD tarafından **DR. ÖĞR. ÜYESİ AYŞE ÖVGÜ KINAY** yönetiminde hazırlanan **"POPÜLERLİK SIRALAMASINA DAYALI DC^{3}MAB ÖNERİ SİSTEMİ VE UYGULAMASI"** başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

Dr.Öğr.Üyesi Ayşe Övgü KINAY

Danışman

Prof. Dr. Gözde ULUAGAY

Jüri Üyesi

Doç. Dr. Engin YILDIZTEPE

Jüri Üyesi

Prof. Dr. Okan FISTIKOĞLU

Müdür

Fen Bilimleri Enstitüsü

TEŞEKKÜR

Tez çalışmam boyunca değerli bilgi ve birikimlerini benimle paylaşan, çalışmam için beni motive eden ve tecrübeleriyle hem akademik hem de kişisel olarak bana her zaman destek olan saygıdeğer tez danışmanım Dr. Öğr. Üyesi Ayşe Övgü KINAY'a sonsuz teşekkür ve saygılarımı sunarım.

Yüksek lisans eğitimim boyunca bana değerli katkılar veren DEÜ Veri Bilimi ailesinin değerli hocalarına ayrı ayrı teşekkür ederim.

Tez süresince yanımda olan desteklerini her daim yanımda hissettiğim sevgili aileme ve sevgili eşime teşekkürlerimi sunarım.

Najibeh MIRINEZHAD

POPÜLERLİK SIRALAMASINA DAYALI DC^{3}MAB ÖNERİ SİSTEMİ VE UYGULAMASI

ÖZ

Gerçek dünya öneri sistemlerinde, kullanıcı ve öge sayısının artması, kullanıcı-öge etkileşimlerinin giderek seyrekleşmesine neden olmaktadır. Ayrıca yeni bir kullanıcı ve/veya yeni bir ürün sisteme eklendiğinde bunlar hakkında yeterli bilgi olmadığı için kullanıcılara ürün önerme sürecinde sorunlar oluşur. Öneri algoritmaları, kullanıcıların önceki tercihlerine dayanmaktadır. Aynı zamanda, bu algoritmaların kullanıcının ilgi alanlarını belirleme potansiyeli yüksek ürünleri tanımlaması gerekir.

Çok kollu haydut öneri sistemleri her ne kadar soğuk başlangıç problemlerine bir çözüm olsa da uygulanmasında bazı temel sorunlar ortaya çıkar. İlk olarak, büyük miktarda veri olması sebebiyle kullanıcı tercihlerinin belirlenmesi zorlaşır. Ayrıca, geleneksel haydut modelleri, ögeleri kollar olarak ele alır ve sürekli olarak artan öge sayısı ile başa çıkmakta zorluk çeker. Ek olarak, bu konuda yapılan çalışmalarda genellikle Bernoulli dağılımı temelli ikili geri bildirim göz önüne bulundurur.

Bahsedilen zorlukları aşmak için bu çalışmada DC^{3}MAB algoritmasına popüler ögelerin etkisi adımı eklenerek yöntem genişletilmiştir. DC^{3}MAB-π (DC^{3}MAB **P**opular **I**tems) olarak isimlendirdiğimiz bu yöntemde öncelikle dinamik bir kullanıcı kümeleme yaklaşımı uygulanır. Ögeler, İşbirlikçi Filtreleme ile dinamik olarak kümelere ayrılarak kol sayısı önemli ölçüde azaltılır. Daha sonra filtreleme adımında ögeler etkileşimine göre ağırlıklandırılır ve bu ögeler için toplam popülerlik dereceleri belirlenir. Böylece yüksek popülerliğe sahip ögelere öncelik verilmiş olur. Ayrıca, kullanıcı tercihlerinin daha doğru bir şekilde belirlenmesi için örtük geri bildirimlerden elde edilen çok sınıflı bir ödül sistemi kullanılır.

Son olarak bu çalışmada DC^{3}MAB-π algoritmasının etkinliği DC^{3}MAB algoritması ile karşılaştırılmış ve sonuçlara göre Duyarlılık ve F_1 puanı değerlerinde orta düzeyde, Ödül açısından ise ortalama yüzde 6 iyileşme olduğu gözlemlenmiştir.

Anahtar kelimeler: Çevrimiçi öneri sistemi, dinamik kümeleme, bağlamsal çok kollu haydut, örtük geri bildirim.

DC^{3}MAB RECOMMENDATION SYSTEM BASED ON POPULARITY RANKING AND ITS APPLICATION

ABSTRACT

In real-world recommendation systems, the increasing number of users and items leads to sparser user-item interactions. In addition, when a new user and/or a new item is added to the system, there is not enough information about them, which causes problems in the process of recommendation. Online recommendation algorithms are based on users' previous preferences. At the same time, these algorithms need to explore items with high potential to discover the users' interests.

Although multi armed bandit recommendation systems are a solution to the cold start problems, some key problems arise in their implementation. First, the large amount of data makes it difficult to determine user preferences. Second, the traditional bandit models treat the items as arms and find it difficult to deal with the ever-increasing number of items. In addition, previous research on this topic usually considers the Bernoulli distribution based on binary feedback.

To overcome these challenges, this study extends the DC^{3}MAB method by adding a popular items impact step to the DC3MAB algorithm. In this method, which we call DC^{3}MAB-π (DC^{3}MAB **P**opular **I**tems), a dynamic user clustering approach is applied. Items are dynamically clustered by Collaborative filtering and the number of arms is significantly reduced. Then, in the filtering step, the items are weighted according to their interaction and the total popularity levels for these items are determined. Thus, items with high popularity are prioritized. In addition, a multi-class reward system derived from implicit feedbacks is used to determine user preferences more accurately.

Finally, in this study, the efficiency of DC^{3}MAB-π algorithm was compared with DC^{3}MAB algorithm and the results showed a moderate improvement in Recall and F_1 score values and an average 6 percent improvement in Reward value.

Keywords: Online recommendation system, dynamic clustering, contextual armed bandit, implicit feedback.

SEMBOLLER LİSTESİ

r_A	: A kullanıcısının tercih vektörü
r_{A_i}	: A kullanıcısının i ürünü için derecelendirmesi
$\hat{r}_{A_i}$	: A kullanıcısının i ürünü için tahmin edilen derecelendirmesi
U	: Kullanıcılar kümesi
I	: Ögeler kümesi
R	: Derecelendirme matrisi
d	: Boyut
$R_{m\times n}$	: $m \times n$ boyutlu derecelendirme matrisi
S	: Tüm derecelendirmeler için (kullanıcı, öge) kümesi
A	: Tüm eylemler kümesi
$a_{t,k}$	: t anındaki k eylemi
$r_{t,k}$	: t anındaki k eyleminin aldığı ödül
A_t	: t anında seçilen eylem
$\mu(a)$	: t anında a eyleminin tahmini değeri
$N_t(a)$	: t anından önce a eyleminin seçilme sayısı
α	: Güven düzey parametresi
r_{t,a^*}	: Ajanın a^* seçtikten sonra aldığı ödül
R	: Ödül
$x_{t,a}$	: Özellik vektörü
$E[r_{t,a}\|x_{t,a}]$	: a eyleminin $x_{t,a}$ özellik vektörü bilindiğinde beklenen getirisi
θ_a^*	: a eyleminin beklenen getirisi için bilinmeyen bir katsayı vektörü
D_a	: Tasarım matrisi
C_a	: Yanıt vektörü
I_d	: $d \times d$ boyutlu birim matrisi
$\hat{\theta}_a$	: θ_a^* katsayısının tahmini
D_a^T	: Tasarım matrisinin transpozu
a_t	: Üst güven aralığını maksimize eden eylem
$x_{t,a}^T$	: Özellik vektörünün transpozu
β^*	: Tüm eylemler için ortak olan bilinmeyen katsayı vektörü

Reg : Pişmanlık

S : Etkileşim davranışı dizi kümesi

$\hat{S}_u^T$: Kullanıcı u'nun davranış dizisi

s_{ut} : Kullanıcı u'nun t anındaki etkileşim davranışı

c_t : t anındaki bağlam bilgisi

J : u tarafından t anında seçilecek mevcut kol(seçenek) sayısı

a : Süper kol

a^* : En iyi süper kol

A : Tüm süper kollar kümesi

$\mathbb{E}(R_a)$: a süper kolunun tahmin edilen ödülü

f_{UCB}^a : a süper kolunun üst sınır güven aralığı fonksiyonu

$\bar{I}_{ut}$: Kullanıcı u için t anındaki öneri listesi

R_t^* : Seçilen süper kol tarafından t anında beklenen ödül

μ_a : a süper kolunun beklenen ödülü

CB_t : Teorik üst güven sınırının basitleştirilmiş versiyonu

$\overline{w}_{c^*}$: Kullanıcının bulunduğu küme parametresi

c^* : Kullanıcının bulunduğu küme

N : Seçilen süper kolun uzunluğu

$\bar{I}_{\boldsymbol{ut}}$: t anında kullanıcı u için nihai öneri listesi

RW_{ui} : i ögesinin u kullanıcısına önerildiğindeki ödül ağırlığı

k : Kullanıcı küme sayısı

M_u : $d \times d$ boyutlu birim matrisi

b_u : d boyutlu sıfır matrisi

w_u : u kullanıcısının tercihini yansıtan bir katsayı

$\overline{w}_{c^*}$: u kullanıcısının bulunduğu küme için w_u'nun tahmini

$\overline{w}_{\bar{c}}$: u hariç kullanıcıların bulunduğu küme için w katsayısının tahmini

KISALTMALAR

ÖS	: Öneri Sistemleri
ÇKH	: Çok Kollu Haydutlar (Multi Armed Bandits, MAB)
MÖ	: Makine Öğrenimi
İTF	: İçerik Tabanlı Filtreleme
İF	: İşbirlikçi Filtreleme
HTİF	: Hafıza Tabanlı İşbirlikçi Filtreleme
MTİF	: Model Tabanlı İşbirlikçi Filtreleme
MF	: Matris Faktörizasyonu
ÜGA	: Üst Güven Aralığı (Upper Confidence Bound, UCB)
KNN	: K-En Yakın Komşuluk

İÇİNDEKİLER

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

TABLOLAR LİSTESİ

Sayfa

BÖLÜM BİR
GİRİŞ

Öneri sistemleri (ÖS), işbirlikçi filtreleme üzerine ilk makalelerin (Hill, Stead, Rosenstein, ve Furnas, 1995; Resnick, Iacovou, Suchak, Bergstrom, ve Riedl, 1994; Shardanand ve Maes, 1995) ortaya çıkmasıyla 1990'ların ortalarından itibaren önemli bir araştırma alanı haline gelmiştir (Adomavicius ve Tuzhilin, 2005).

Öneri (tavsiye) sistemleri, internet üzerindeki veri yoğunluğu ve kullanıcıların ilgi alanlarının çeşitliliği nedeniyle ortaya çıkan bir ihtiyaca cevap olarak geliştirilen bilgi işleme sistemleridir. İnternet günümüzde önemli bir bilgi kaynağıdır. Ancak, ihtiyaç duyduğumuz bilgiyi bulmak, mevcut kaynakların çokluğu nedeniyle zor olabilir. Öneri sistemleri, güvenilir bilgi sağlayarak bu sorunu çözmek için geliştirilmiştir.

İnternet kullanıcıları çoğunlukla bir şekilde öneri sistemleriyle karşılaşmışlardır. Örneğin, bir arkadaşınızın size yeni bir kitap önerdiğini ve ardından bir online kitapçıyı ziyaret ettiğinizi düşünün. Kitabın adını yazdıktan sonra, listelenen sonuçlardan sadece biri olarak görünür. Web sayfasının muhtemelen "Bu Ürünü alanlar bunları da aldı" olarak adlandırılan bir bölümünde, ilginizi çekebilecek başka kitaplar da gösterilir. Aynı çevrimiçi kitapçının düzenli bir kullanıcısıysanız, mağazaya girer girmez böyle kişiselleştirilmiş bir öneri listesi otomatik olarak görünecektir. Ziyaretçilere hangi kitapların gösterileceğini belirleyen yazılım sistemi, bir öneri sistemidir (Jannach, Zanker, Felfernig, ve Friedrich, 2010).

Öneri sistemleri, bir kullanıcının işine yarayacak ögeler için öneriler sunan yazılım araçları ve teknikleridir. Öneriler, hangi ürünlerin satın alınacağı, hangi müziğin dinleneceği veya hangi çevrimiçi haberlerin okunacağı gibi çeşitli karar verme süreçleriyle ilgilidir (Ricci, Rokach, ve Shapira, 2010). Öneri sistemleri, kullanıcılara çok fazla seçenek arasından yönlendirme yaparak karar verme sürecini kolaylaştırır ve kullanıcı deneyimini geliştirir.

Öneri sistemleri e-ticaret, e-sağlık, e-öğrenme, turizm ve bilgi yönetimi gibi çeşitli uygulama alanlarında hayata geçirilmiştir. İyi bilinen örnekler arasında, kullanıcıların geçmiş satın alma deneyimlerine dayalı olarak seçtikleri çeşitli ürünlerin önerildiği Amazon web mağazası yer almaktadır. Benzer şekilde, FourSquare yüksek puan alan

mekanları önermek için bütünleşmiş bir öneri sistemine sahiptir. Netflix film öneri sistemine sahiptir ve YouTube bir video öneri sistemine sahiptir (Khalid, Khan, ve Zomaya, 2019).

Ürün satışlarını artırmak bir öneri sisteminin birincil hedefidir. Sonuçta bu sistemler, satıcılar tarafından kârlarını artırmak için kullanılmaktadır. Öneri sistemleri, özenle seçilmiş ürünleri kullanıcılara önererek, ilgili ürünleri kullanıcıların dikkatine sunar. Bu da satıcının satış hacmini ve kârını artırır. Bir öneri sisteminin birincil hedefi satıcının gelirini artırmak olsa da bu genellikle ilk bakışta göründüğünden daha az açık olan yollarla elde edilir. Geliri artırmaya yönelik daha geniş işletme merkezli hedefe ulaşmak için, öneri sistemlerinin hedefleri aşağıdaki gibidir (Aggarwal, 2016):

1. Uygunluk: Bir öneri sisteminin en belirgin hedefi, kullanıcıya ilgili ögeleri önermektir. Kullanıcıların ilgili buldukları ögeleri tüketme olasılıkları daha yüksektir. Ancak ilgi düzeyi bir öneri sisteminin birincil hedefi olsa da tek başına yeterli değildir.

2. Yenilik: Öneri sistemleri, önerilen öge kullanıcının geçmişte görmediği bir şey olduğunda gerçekten yardımcı olur. Örneğin, tercih edilen bir türdeki popüler filmler kullanıcı için nadiren yeni olacaktır. Popüler ürünlerin tekrar tekrar önerilmesi de satış çeşitliliğinde azalmaya yol açabilir (Fleder ve Hosanagar, 2007).

3. Rastlantısallık: Bu şekilde önerilen ögeler beklenmedik oldukları için yüksek ihtimalle önerileceklerin aksine keşif unsurudurlar. Rastlantısallık yenilikten farklıdır çünkü öneriler kullanıcı için daha önce bilmedikleri bir şey olmaktan ziyade gerçekten şaşırtıcıdır. Belirli bir kullanıcının yalnızca belirli bir türdeki ürünleri tüketiyor olması sıklıkla söz konusu olabilir, ancak kullanıcının kendisinin de şaşırtıcı bulabileceği diğer türlerdeki ürünlere gizli bir ilgisi olabilir. Yeniliğin aksine, tesadüfi yöntemler bu tür önerileri keşfetmeye odaklanır (Good vd., 1999).

4. Öge çeşitliliğinin artırılması: Öneri sistemleri tipik olarak en iyi k ögelerin bir listesini önerir. Önerilen tüm bu ögeler birbirine çok benzediğinde, kullanıcının bu ögelerden hiçbirini beğenmeme riski artar. Öte yandan, önerilen liste farklı türlerde ögeler içerdiğinde, kullanıcının bu ögelerden en az birini beğenme ihtimali daha yüksektir. Çeşitlilik, kullanıcının benzer ögelerin tekrar tekrar önerilmesinden sıkılmamasını sağlama avantajına sahiptir.

1.1 Öneri Sistemlerinde Geri Bildirim Yaklaşımları

Web'in elektronik ve ticari işlemler için bir ortam olarak artan önemi, öneri sistemleri teknolojisinin gelişimi için itici bir güç olmuştur. Bu konuda önemli bir katalizör, Web'in kullanıcıların beğendikleri ya da beğenmedikleri içerikler hakkında geri bildirimde bulunmalarına olanak sağlamasıdır. Örneğin, Netflix gibi bir filim öneri sistemini düşünün. Bu gibi durumlarda kullanıcılar basit bir fare tıklamasıyla kolayca geri bildirimde bulunabilmektedir. Geri bildirim sağlamaya yönelik tipik bir metodoloji, kullanıcıların belirli bir değerlendirme sisteminden (örneğin, beş yıldızlı derecelendirme sistemi) çeşitli ögelere yönelik beğenilerini ve beğenmediklerini belirten sayısal değerleri seçtikleri derecelendirme şeklindedir (Aggarwal, 2016).

Öneri sistemlerinin temel fikri, müşterinin ilgi alanlarını çıkarmak için çeşitli kullanıcı geri bildirim verilerini kullanmaktır. Tavsiyenin sunulduğu varlık "kullanıcı" olarak adlandırılır ve önerilen ürün de "öge" olarak adlandırılır. Bu nedenle, tavsiye analizi genellikle kullanıcılar ve ögeler arasındaki önceki etkileşime dayanır, çünkü geçmiş ilgi alanları ve eğilimler genellikle gelecekteki seçimlerin iyi göstergeleridir.

Öneri sistemlerinde, kullanıcılarla ilgili veri toplama işlemi iki farklı yöntemle gerçekleştirilmektedir. Bu yöntemler, *açık* ve *örtük* geri bildirim olarak adlandırılır.

1.1.1 Açık Geri Bildirim

Açık geri bildirim, kullanıcıların doğrudan sistem üzerindeki etkileşimleri ve tepkileri yoluyla elde edilen bilgileri ifade eder. Kullanıcılar, ürünleri değerlendirme, puanlama, yorum yapma gibi açık şekilde ifade edilebilen geri bildirimlerde bulunarak tercihlerini belirtirler.

1.1.2 Örtük Geri Bildirim

Örtük geri bildirim kullanıcıların davranışları ve etkileşimleri aracılığıyla elde edilen bilgileri ifade eder. Kullanıcıların gezinme tercihleri, tıklama davranışları, satın alma alışkanlıkları gibi veriler analiz edilerek, kullanıcıların tercihleri ve ilgi alanları hakkında anlam çıkarılır. Bu bilgiler, öneri sistemlerinin kullanıcı deneyimini geliştirmek ve kullanıcılara daha uygun öneriler sunmak için kullanılır. Açık geri

bildirim, kullanıcıların bilinçli tercihlerini ifade etmelerine olanak sağlarken, örtük geri bildirim ise kullanıcı davranışlarından çıkarılan verilerle kullanıcıların ilgi alanlarını daha doğrudan tespit etmeye yardımcı olur.

1.2 Çalışma Amacı ve Özgünlüğü

Kişiselleştirilmiş çevrimiçi öneriler alanında, kullanıcıların değişen ihtiyaçlarını karşılamak için keşif ve sömürü arasında hassas bir denge kurmak önemli bir zorluktur. Bu klasik keşif/sömürü (Exploration/Exploitation, EE) ikilemi yaygın olarak kabul görmüştür ve geleneksel öneri yöntemleri, çevrimdışı eğitim paradigmaları nedeniyle sınırlamalarla karşı karşıya kalmıştır. Bu sorunu ele almak için çok kollu haydut (ÇKH) algoritmaları güçlü bir çözüm olarak ortaya çıkmıştır. Son yıllarda, ÇKH problemleri istatistik, makine öğrenimi, karar ve kontrol topluluklarında yoğun bir şekilde çalışılmıştır. Bunun nedeni, ÇKH'ın çevrimiçi öneri sistemleri gibi birçok gerçek dünya uygulamasının basit ama etkili bir modeli olmasıdır.

Geleneksel ÇKH öneri yaklaşımları, her bir ögeyi tek bir eylem gibi ele alarak hesaplama karmaşıklığı sorunlarına yol açmaktadır (Christakopoulou ve Banerjee, 2018). Bu tez çalışmasına temel oluşturan ve DC^{3}MAB olarak bilinen dinamik kümeleme tabanlı bağlamsal birleşimsel çok kollu haydutlar yaklaşımı, mevcut modellerin sınırlamalarının üstesinden gelmek için bir kullanıcı kümeleme politikası önermekte ve global haydutların basitliği ile bağımsız haydutların kişiselleştirilmesi arasında bir denge kurmaktadır. (Yan, Han, Zhang, Zhu ve Wan, 2022).

Ürün-kullanıcı eşleşmesinin azalması, ürün yelpazesinin genişlemesi (ürün kategorisinin artması), belirli tip ürünlere daha az talep olması gibi durumlar nedeniyle kullanıcı tercihlerini öğrenmek zorlaşır. Mevcut birçok yöntem ikili bir ödül mekanizması kullanır. Bu da potansiyelinden tam olarak yararlanamadıkları anlamına gelir. Bu sistemler, yalnızca satın alınan ögeleri başarılı öneriler olarak kabul eder ve örtük geri bildirim tercih keşfine katkıda bulunmaz. DC^{3}MAB algoritmasına dayanan yaklaşımı ise örtük geri bildirimi kategorize etmek veya doğrudan değerlendirmek yerine öge önerilerini optimize ederek tercihlerin dikkate alınmasını sağlar.

Bu çalışmada, DC^{3}MAB algoritmasına öge popülerliği ve kullanıcı etkileşimleri entegre edildi ve DC^{3}MAB-**P**opüler **I**tems (DC^{3}MAB-π) olarak isimlendirildi. Ağırlıklı etkileşim değerlerini göz önünde bulunduran bu yöntem sayesinde yüksek etkileşimli ögelerin popülerliği birleştirilerek öneri stratejisinde belirli ögelere öncelik verilmesi sağlanır.

1.3 Tezin Akışı

Birinci bölüm, metnin giriş aşamasını oluşturur. Giriş bölümünde çalışma konusu kısaca tanıtıldıktan sonra, geri bildirim yaklaşımları ele alınmış ve ayrıca çalışmanın amacı ile özgünlüğü vurgulanmıştır. İkinci bölümde, makine öğrenmesi çeşitleri ve terminolojisi detaylı bir şekilde sunulmuştur. Çalışmanın üçüncü bölümünde, öneri sistemlerinin işbirlikçi, içerik tabanlı, demografik tabanlı, bilgi tabanlı ve karma tabanlı algoritmaları, avantajları ve dezavantajları açıklanmıştır. Dördüncü bölümde, öneri sistemlerinin karşılaştığı sorunlar ele alınmıştır. Beşinci bölümde, gelişmiş modeller başlığı altında odak noktası olan ÇKH algoritması ve bu algoritmanın bir versiyonu olan DC^{3}MAB yöntemi ile bu tez çalışmasında önerilen DC^{3}MAB-π yöntemi detaylı bir şekilde açıklanmıştır. Altıncı bölümde, bu çalışmada kullanılan veri setinin özellikleri anlatılmış ve bu veri kümesinden elde edilen üç örneklem üzerinde de DC^{3}MAB ve DC^{3}MAB-π yöntemleri uygulanmıştır. Bu hesaplamalarda elde edilen performans ölçütleri ise istatistiksel yöntemlerle incelenmiştir. Son olarak yedinci bölümde de bulgular özet olarak sunulmuş ve genel değerlendirme yapılmıştır.

BÖLÜM İKİ
MAKİNE ÖĞRENMESİ

2.1 Makine Öğrenme Yaklaşımları

Makine öğrenimi (MÖ), bilgisayar sistemlerinin açıkça programlanmadan belirli bir görevi yerine getirmek için kullandıkları algoritmaların ve istatistiksel modellerin bilimsel çalışmasıdır (Mahesh, 2020). Fiziksel dünyada yapılan gözlemlerden örüntüler çıkarmaya ve fikir edinmeye çalışan bir dizi araç ve yöntem olarak düşünülebilir. Makine öğrenmesi algoritmaları denetimli, denetimsiz ve pekiştirmeli öğrenme olmak üzere üçe ayrılır (Gutierrez, 2015).

2.1.1 Denetimli Öğrenme (Supervised Learning)

Denetimli öğrenme, eğitim verilerindeki etiketli veriler kullanılarak bir modelin öğrenildiği bir makine öğrenme yöntemidir. Model, girdi verilerine karşılık gelen çıktıları (etiketleri) öğrenir. Bu süreçte, modelin eğitim verileri üzerindeki performansı değerlendirilir ve daha sonra yeni veriler için çıktı tahminleri yapılabilir.

2.1.2 Denetimsiz Öğrenme (Unsupervised Learning)

Denetimsiz makine öğrenimi, istatistiksel öğrenmenin daha esnek bir çeşididir. Etiketli veri kümelerini kullanmadan, yalnızca bir dizi öznitelik değişkeninin gözlemlendiği durumlar için uygun olan istatistiksel tekniklerdir. Bu durumda amaç, veri kümesi etiketsiz olduğundan ve analize rehberlik edecek ilişkili bir çıktı değişkeninin bulunmaması nedeniyle tahmin değildir. Amaç, öznitelik değişkenlerinin ölçümleri üzerinden yararlı bilgiler elde etmektir (Bonaccorso, 2017).

2.1.3 Pekiştirmeli Öğrenme (Reinforcement Learning)

Pekiştirmeli öğrenme, çevreden gelen geri bildirime dayanır, ancak bu geri bildirim genellikle bir ödül (veya bazen bir ceza) şeklinde gelir ve ajanın (örneğin bir öneri algoritması, robot veya bir oyun karakteri) belirli bir durumda hangi eylemlerin olumlu olduğunu anlamasına yardımcı olur. Ancak, bu geri bildirim genellikle kesin bir hata ölçüsü sağlamaz. Bu nedenle, ajanın hatasının kesin bir ölçüsünü belirlemesi zordur.

Bu, pekiştirmeli öğrenmenin bir denetleyicinin olmadığı durumlarda bile çalışabilmesini sağlar (Bonaccorso, 2017).

Pekiştirmeli öğrenme, bir ajanın çevresiyle etkileşime girerek ve deneyimlerinden öğrenerek belirli bir hedefi başarmasını sağlar. Bu öğrenme süreci, Şekil 2.1'de gösterildiği gibi bir şema ile açıklanabilir. Ajan, çevresiyle etkileşime girerken ve belirli bir hedefi başarmaya çalışırken ödüller veya cezalar alır. Bu ödüller, ajanın doğru eylemleri öğrenmesini teşvik eder ve yanlış eylemleri azaltır. Ajan, durumları algılar, eylemler gerçekleştirir ve elde ettiği ödüllerle öğrenme sürecini optimize eder.

Öneri sistemleri, kullanıcılara ilgi duyabilecekleri ürünleri, içerikleri veya hizmetleri sunmak için kullanılır ve kullanıcıdan tıklama veya satın alma şeklinde ödüller alır. Bir öge ne kadar çok tıklanır veya satın alınırsa, ödülü o kadar yüksek olur. Pekiştirmeli öğrenme, öneri sistemlerinde kullanıcı tercihlerini anlamak ve onlara en uygun önerileri sunmak için etkili bir yaklaşımdır.

Şekil 2.1 Öneri sistemlerinde pekiştirmeli öğrenmenin öğrenme süreci

2.1.3.1 Pekiştirmeli Öğrenmenin Unsurları

Ajan (Agent): Ajan, çevreyi algılayabilir ve çevre ile sürekli etkileşime girerek en büyük ödül değerini elde etmek için bir eylem seçebilir. Bu süreçte, ajan aynı zamanda kullanıcılara doğru ve kişiselleştirilmiş öneriler sunma görevini üstlenir (Qiang ve Zhongli, 2011). Öneri sistemlerinde ajanlar genellikle şu temel görevleri yerine getirir:

- **Öznitelik Toplama:** Kullanıcılar hakkında çeşitli öznitelikleri toplar, bu öznitelikler kullanıcıların geçmiş alışverişleri, izledikleri filmler, dinledikleri müzikler, okudukları kitaplar gibi tercihleri içerebilir.
- **Modelleme ve Öğrenme:** Toplanan öznitelikleri kullanarak bir model oluşturur.
- **Tahmin ve Öneri:** Oluşturulan modeli kullanarak, kullanıcılara özel önerilerde bulunur.
- **Geri bildirim Alımı:** Kullanıcıların tepkilerini ve geri bildirimlerini takip eder. Kullanıcıların önerilere nasıl tepki verdiklerini anlamak için geri bildirimleri kullanarak modeli günceller.

Çevre (Environment): Çevre, öneri sistemine dış dünyayı temsil eden kritik bir bölümü ifade eder. Çevre, ajanın etkileşimde bulunduğu kullanıcı verileri, ürün katalogları, puanlamalar, gezinme davranışları ve diğer ilgili bilgileri içerir. Ajan, çevreden veri alır, bu verileri işler, kararlarını verir ve çevreyle etkileşimde bulunarak önerileri sunar.

Politika (Policy): Öneri sistemlerinde politika, hangi önerilerin verileceğini ve nasıl sunulacağını belirleyen stratejidir. Politika, kullanıcının geçmiş tercihleri, popülerlik, benzerlik veya kişiselleştirme gibi faktörlere dayanabilir. Bu çok yönlü strateji, öneri sisteminin kullanıcıya en uygun içeriği sunma amacına hizmet eder.

Eylem (Action): Eylem, kullanıcılara öneri sistemleri tarafından sunulan çeşitli önerilerin genişliğini temsil eder. Örneğin, bir e-ticaret platformundaki ürün önerileri veya bir müzik akışı hizmetindeki şarkı önerileri. Öneri sistemleri, etkileşimlerine kucak açarak geniş bir eylem yelpazesi sunar. İncele, dinle, beğen veya al; her eylem, sistemlerin karmaşıklığını ve çeşitliliğini yansıtarak, kullanıcıları sıkıcı monotonluğun tuzaklarından uzak tutar. Her seferinde farklı bir heyecan vaat ederek, kullanıcıları keşif dolu bir serüvene davet eder.

Ödül (Reward): Pekiştirmeli öğrenmede ödüller kullanıcıların gerçekleştirdikleri eylemlerin sonuçlarına dayanır. Ödül sinyali hedefi belirler. Her bir zaman adımında, çevre öğrenene "ödül" adı verilen bir puan verir. Öğrenenin amacı, uzun vadede elde

ettiği birikimli ödülü maksimize etmektir. Ödül, öğrenen için olumlu ve olumsuz olayları belirtir. Mesela bir öneriye tıklama, satın alma ya da öneriyi beğenme gibi eylemler öğrenenin başarılı bir hamle yaptığını gösterir. Bu tür etkileşimler, öğrenenin elde ettiği başarıları çeşitlendirip zenginleştiren pozitif deneyimlerdir.

Durum (State): Durum, kullanıcının geçmiş tercihleri, demografik bilgileri, gezinme davranışları ve diğer ilgili faktörleri içerir. Bu faktörler, öneri sistemleri için kritik girdiler sağlar. Sistem, bu kapsamlı durum bilgilerini kullanarak kullanıcının mevcut durumunu anlamaya çalışır. Böylece, daha önceki tercih ve davranışlara dayanarak kişiselleştirilmiş ve çeşitli öneriler sunma kapasitesini artırır.

2.1.3.2 Pekiştirmeli Öğrenmenin Öneri Sistemlerinde Önemi

Pekiştirmeli öğrenme, kullanıcılara öneriler sunarken keşfedilmemiş alanları keşfetmelerine yardımcı olabilir. Kullanıcıların daha önce deneyimlemediği veya bilinmeyen içeriklere yönlendirilerek yeni deneyimler yaşaması sağlanabilir. Bu, kullanıcıların sınırlı bir alanda kalmaktan ziyade farklı ve çeşitli içerikleri keşfetmelerini sağlar. Örneğin, bir sistem belirli bir kullanıcıya aksiyon, polisiye ve romantik türlerde filmler önerdi. Bununla birlikte, kullanıcı sadece romantik filmleri izlemeyi tercih etti. Bu noktada, öneri sistemi basitçe kullanıcının sadece romantik filmleri sevdiğine ve bu tür filmleri tavsiye etmeye odaklanabileceğine karar verebilir. Ancak, kullanıcının romantik filmlerden başka ilgi çekici ve sistem tarafından daha önce tavsiye edilmemiş film kategorileri olabilir. Kullanıcının, sistem tarafından sunulmayan diğer kategorilerden de ilgi duyabileceği ve böylece daha fazla film izleyebileceği umut edilerek (artan ödül), sistemin daha önce keşfedilmemiş film kategorilerinden önerilerde bulunması ve keşif sürecini sürdürmesi faydalı olabilir (Glowacka, 2019).

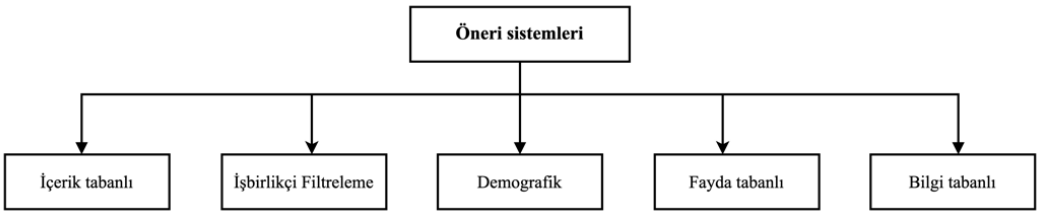
Pekiştirmeli öğrenme yöntemleri, kullanıcıların tercihlerinde zamanla meydana gelen değişikliklere uyum sağlar. Öneri sistemleri, kullanıcıların geri bildirimleri ve tepkileri üzerinden öğrenir ve bu bilgileri kullanarak öneri stratejisini günceller. Bu sayede sistem, kullanıcıların değişen ilgi alanlarına, trendlere veya kişisel tercihlere göre öneriler sunabilir. Aynı anda, gerçek zamanlı geri bildirimlere dayanır. Kullanıcıların verdiği geri bildirimler, önerilen içeriklerin kullanıcı tercihlerine

uygunluğunu doğrulamak veya düzeltmek için kullanılır. Bu sayede sistem, kullanıcı tercihlerine daha iyi uyum sağlamak için önerilerini güncelleyebilir. Pekiştirmeli öğrenme algoritmalarının önemli bir sınıfı çok kollu haydut algoritmalarıdır.

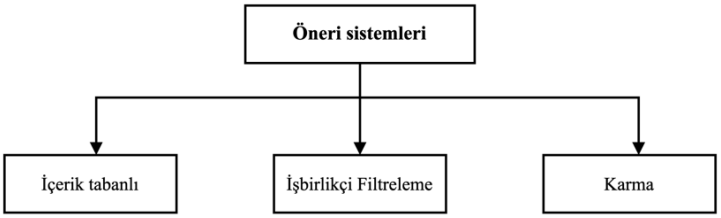
BÖLÜM ÜÇ
ÖNERİ SİSTEMLERİNİN TEMEL YÖNTEMLERİ

3.1 Yöntem

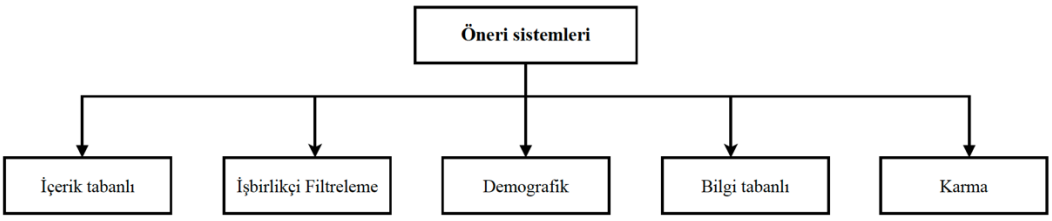
Öneri sistemleri alanında uzman kişiler tarafından literatürde çeşitli sınıflandırmalar yapılmıştır. Balabanovic ve Shoham (1997), öneri sistemlerini içerik bazlı ve işbirlikçi paradigmalar olmak üzere iki ana gruba ayırmıştır. Ayrıca, *Fab* mimarisi adlı bir karma yaklaşım önererek, içerik bazlı ve işbirlikçi metotları birleştirerek sistemlerin avantajlarını bir araya getirmiştir. Burke (2002), öneri sistemlerini girdi verilerini dikkate alarak beş kategoriye ayırmıştır. Bu sınıflandırmayı Şekil 3.1'deki gibidir. Adomavicious ve Tuzhilin (2005) tarafından önerilen ve daha geniş çapta kabul gören üç ana sınıflandırma Şekil 3.2'deki gibidir. Aggarwal (2016) öneri sistemlerinin temel yöntemlerini Şekil 3.3'te gösterildiği gibi beş alt kategoriye ayırmıştır.



Şekil 3.1 Öneri sistemi türlerinin Burke'e göre sınıflandırılması



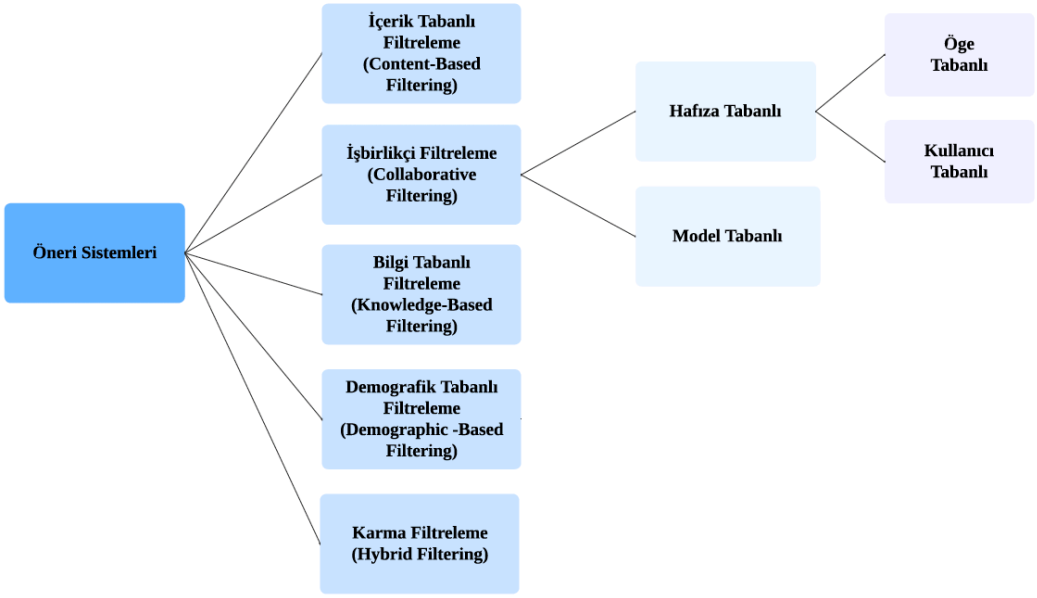
Şekil 3.2 Öneri sistemi türlerinin Adomavicious ve Tuzhilin'e göre sınıflandırılması



Şekil 3.3 Öneri sistemi türlerinin Aggarwal'e göre sınıflandırılması

Öneri sistemleri, kullanıcı-öge etkileşimlerini ve özellik bilgilerini kullanarak çeşitli modellerle öneriler oluşturmayı amaçlar. İşbirlikçi filtreleme yöntemleri, kullanıcıların benzer davranışları ve tercihleri üzerinden öneriler yapar. İçerik tabanlı öneri yöntemleri ise kullanıcı ve öge özelliklerini dikkate alarak öneriler oluşturur. Ancak, genellikle tüm kullanıcılar yerine tek bir kullanıcının derecelendirmelerine odaklanan modelin, içerik tabanlı sistemlerin çoğunlukla derecelendirme matrislerini kullandığını unutmamak gerekir. Bilgi tabanlı öneri sistemleri, kullanıcının belirttiği gereksinimlere dayalı öneriler sunar ve geçmiş derecelendirme veya satın alma verileri yerine dış bilgi kaynaklarını kullanır. Demografik öneri sistemleri, kullanıcıların demografik özelliklerine dayanarak öneriler sunar. Bu sistemler, kullanıcıların yaş, cinsiyet, coğrafi konum, gelir düzeyi gibi demografik verilerini kullanarak kişiselleştirilmiş öneriler sağlar. Bazı öneri sistemleri, bu farklı yaklaşımları birleştirerek karma sistemler oluşturur. Karma sistemler, farklı öneri yöntemlerinin güçlü yönlerini birleştirerek farklı ortamlarda daha iyi performans gösteren teknikler sunar. Bu modeller, öneri sistemlerinin temelini oluşturur ve çeşitli bilgileri kullanarak kullanıcılara kişiselleştirilmiş öneriler sunmayı hedefler. Şekil 3.4'te öneri sistemlerinin hiyerarşik yapısı verilmiştir:

kullanıcılara kişiselleştirilmiş öneriler sunmayı hedefler. Şekil 3.4'te öneri sistemlerinin hiyerarşik yapısı verilmiştir:

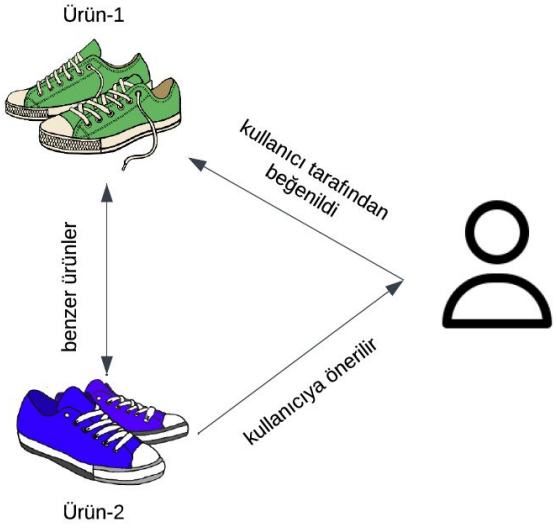


Şekil 3.4 Öneri sistemlerinin ağaç yapısı

3.2 İçerik Tabanlı Filtreleme (Content-Based)

Bu yaklaşımda, kişinin bir önceki seçimleri ve profili göz önünde bulundurularak kendisine benzer profillerin yaptığı seçimler dikkate alınır. Bu ögelerin içerik niteliklerini göz önünde bulundurarak, kişilerin önem verdikleri alanları belirler. Mesela kullanıcının bir önceki sevdiği, satın aldığı kitap türleri, yazarlar veya varsa yayınlarının benzer niteliklerine dayanarak yakın çeşitlilikte kitaplar tavsiye eder (Ricci vd., 2010). İçerik Tabanlı Filtreleme (İTF) çalışma prensibi Şekil 3.5'te gösterilmiştir. Burada, kullanıcının satın aldığı ürün-1'in nitelikleri, kişinin profilini oluşturmak için kullanılır. Bundan sonra, ürün-1 ile benzer niteliklere sahip olan ürün-2, kullanıcı profiliyle uyumlu olduğu için önerilir.

oluşturmak için kullanılır. Bundan sonra, ürün-1 ile benzer niteliklere sahip olan ürün-2, kullanıcı profiliyle uyumlu olduğu için önerilir.



Şekil 3.5 İçerik tabanlı filtreleme çalışma prensibi

Kişiye özel öneriler sunma yeteneği, İTF yönteminin avantajlarındandır. Bu yöntem, kullanıcının geçmişte beğendiği ya da ilgi duyduğu ögelerin özelliklerini inceleyerek, bağlantılı yeni içerikler önerir. Diğer insanların ya da harici verilerin etkisi olmadan kendi bağımsız filtreme yöntemini kullanır. Ayrıca İTF'nin hesaplama gücü ihtiyacının daha düşük olması sebebiyle öneri süreci daha hızlı ve verimli çalışır.

İTF yönteminin dezavantajları da bulunmaktadır. Bu yöntem kullanıcılara genellikle geçmişte beğendikleri veya aldıkları ögeleri benzer önerme eğilimindedir. Bu durum, kullanıcıların alışagelmedik ve çeşitli ögeleri keşfetme fırsatını sınırlar. Ayrıca, kullanıcıların ilgi alanları değiştiğinde veya yeni ilgi alanları geliştirdiklerinde, İTF sınırlı bir öneri çeşitliliği sunabilir. İTF'nin kullanıcılar arasındaki örtük ilişkileri keşfetme yeteneği kısıtlıdır ve diğer kullanıcıların benzer ilgi alanlarına sahip olabileceği potansiyelini göz ardı eder. Bu nedenle, İTF'nin tek başına kullanılması bazı durumlarda yetersiz kalır. Bu dezavantajlı duruma çözüm olarak daha etkili sonuçlar elde etmek için işbirlikçi filtreleme gibi diğer yöntemlerle birleştirilebilir (Adomavicius ve Tuzhilin, 2005).

İTF fonksiyonu Şekil 3.6'daki sözde kodda temsil edilir. Bu fonksiyon içerik ve kullanıcı profili parametrelerini alır ve içeriğin kullanıcı tarafından görüntülenip görüntülenmeyeceği bilgisini döndürür. Algoritma adımları *Benzerlik* fonksiyonunu kullanarak içeriğin kullanıcı profiline ne kadar benzediğini hesaplayan bir *benzerlik_puanı* ve *İçerik* fonksiyonunu kullanarak içeriğin ne kadar önemli veya ilgi çekici olduğunu belirleyen bir *içerik_puanı* içerir. Eğer *benzerlik_puanı* ve *içerik_puanı* belirlenen eşik değerlerinden büyük veya eşitse, içeriğin kullanıcı tarafından görüntülenmesine izin verilir ve True değeri döndürülür. Aksi takdirde, içeriğin kullanıcı tarafından filtrelenmesine karar verilir ve False değeri döndürülür.

İçerik Tabanlı Filtreleme Algoritması

```
Input: içerik (İçerik verisi), kullanıcı_profil (Kullanıcının profil verisi)
Output: Sonuç (İçeriğin görüntülenip görüntülenmeyeceği bilgisi)
1. def İTF (içerik, kullanıcı_profil)
2.
3.      benzerlik_puanı = Benzerlik (içerik, kullanıcı_profil)
4.      içerik_puanı = İçerik (içerik)
5.
6.      if benzerlik_puanı >= benzerlik_eşiği ve içerik_puanı >= içerik_eşiği ise:
7.          return True
8.      else:
9.          return False
```

Şekil 3.6 İçerik tabanlı filtreleme algoritmanın sözde kodu

3.3 İşbirlikçi Filtreleme (Collaborative Filtering)

İşbirlikçi Filtreleme (İF) 1990'ların ortalarından itibaren kullanılan bir yöntemdir (Adomavicius ve Tuzhilin, 2005). Bu yöntem en eski yöntem olmasına rağmen hala en yaygın kullanılan yöntemlerden biridir. İF, kullanıcıların veya ögelerin benzer davranışlarını analiz ederek önerilerde bulunur. Kullanıcıların geçmiş tercihlerini veya değerlendirmelerini kullanarak ilişkileri bulur ve aynı profil veya tercihlere sahip diğer kullanıcıların önerilerini sunar. Kullanıcı ve ürün benzerliklerini ölçmek ve

değerlendirmek için K – En Yakın Komşu Yaklaşımı ve Pearson Korelasyonu gibi çeşitli algoritmalar kullanılır. Bu yöntemde, kullanıcı-ürün matrisi adı verilen bir veri tabanı oluşturulur ve kullanıcı profilleri arasındaki benzerliklere göre ilgi ve tercihlere dayalı eşleştirmeler yapılır. Kullanıcılar, kendileriyle benzer özelliklere sahip diğer kullanıcıların beğendiği ancak henüz değerlendirmediği ürünler için öneriler alır. İF'de yaygın olarak kullanılan iki yöntem vardır: *hafıza tabanlı* (bellek tabanlı) yöntemler ve *model tabanlı* yöntemler.

3.3.1 Hafıza Tabanlı İşbirlikçi Filtreleme (Memory-Based)

Hafıza Tabanlı İşbirlikçi Filtreleme (HTİF) yöntemleri, aynı zamanda komşuluk tabanlı işbirlikçi filtreleme algoritmaları olarak bilinir (Aggarwal, 2016). Hafıza tabanlı yaklaşımlar, doğrudan geçmiş etkileşimleri kullanır ve önerileri en yakın komşu bilgileri kullanarak oluşturur. Bu bağlamda, en çok kullanılan yöntemlerden biri Kosinüs benzerliğidir. Denklem 3.1'de görüldüğü gibi, kosinüs benzerliği iki farklı kullanıcı vektörü arasındaki açının kosinüsünü kullanarak hesaplanır. A kullanıcısının i ögesi için derecelendirmesi, diğer tüm kullanıcıların i ögesi için derecelendirmelerinin toplam ağırlıkları kullanılarak hesaplanır. Denklem 3.2'ye göre, derecelendirilmemiş i ögesi için bir derecelendirme tahmini üretirken kullanılacak ağırlık, A kullanıcısı ile diğer kullanıcılar arasındaki benzerliği ifade etmektedir (Bozkurt ve Acı, 2021).

r_A ve r_B: Sırasıyla A ve B kullanıcılarının tercih vektörlerini temsil eder

$\|r_A\|$ ve $\|r_B\|$, sırasıyla A ve B vektörlerinin normlarını temsil eder

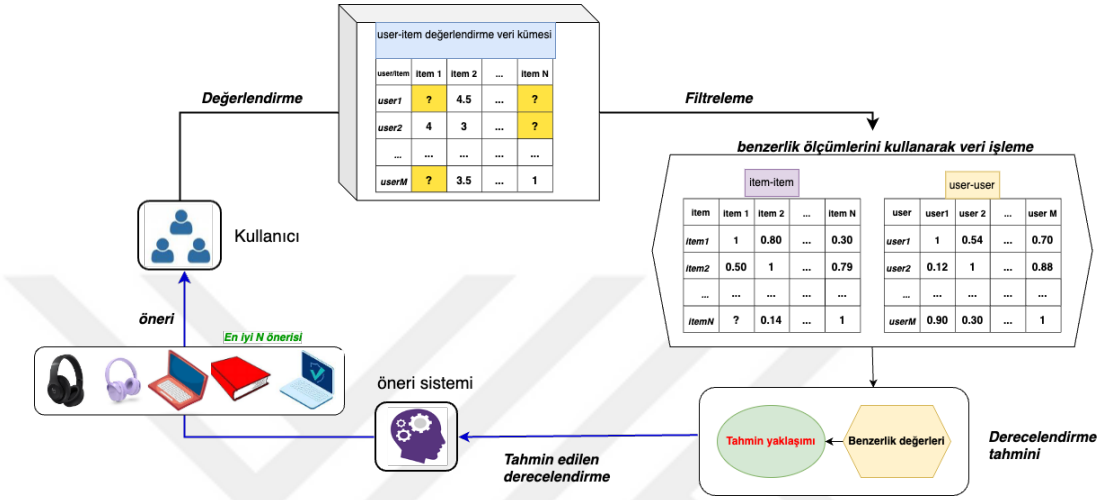
$$\text{benzerlik}(A,B) = \frac{r_A . r_B}{\|r_A\| . \|r_B\|} \tag{3.1}$$

r_{B_i}: B kullanıcısının i ürünü için derecelendirmesi

$\hat{r}_{A_i}$, A kullanıcısının i ürünü için tahmin edilen derecelendirme

$$\hat{r}_{A_i} = \frac{\sum_B benzerlik(A,B) . r_{B_i}}{\sum_B |benzerlik(A,B)|} \tag{3.2}$$

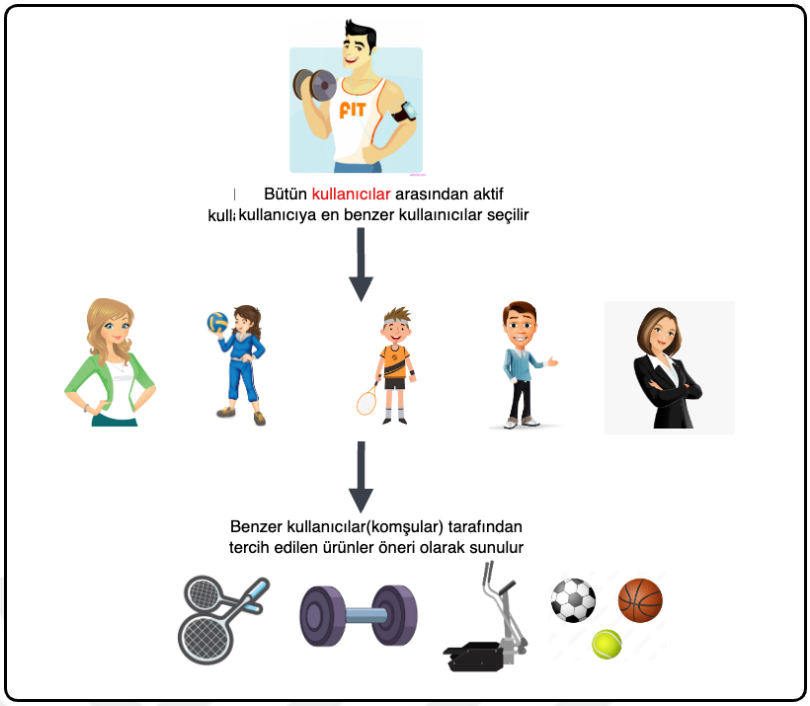
Hafıza tabanlı yaklaşımlar, kullanıcı tabanlı ve öge tabanlı olmak üzere iki kategoriye ayrılabilir. Kullanıcı tabanlı yöntem, benzer kullanıcıları bulmak ve bu kullanıcılara öneriler sunmak için kullanılır. Öte yandan öge tabanlı yöntemler, benzer özelliklere sahip ögeleri bulur ve bu ögeleri kullanıcılara önerir. Şekil 3.7 hafıza tabanlı yaklaşımının çalışma prensibini göstermiştir.



Şekil 3.7 Hafıza tabanlı işbirlikçi filtreleme yaklaşımının çalışma prensibi

3.3.1.1 Kullanıcı Tabanlı Filtreleme (User-Based)

Bu yöntemde, benzer özelliklere sahip kullanıcılar aynı kategoriye dahil edilir ve bu benzer kullanıcılar tarafından beğenilen ürünler temel alınarak ürün önerileri oluşturulur. Değerlendirme sırasında Pearson Korelasyonu veya kosinüs benzerliği gibi yöntemler kullanılmaktadır. Bu süreç Şekil 3.8'de görüldüğü üzere üç kademede tamamlanır. İlk olarak, aktif kullanıcıya benzer özelliklere sahip diğer kullanıcılar bulunur ve bu kullanıcıların beğendiği ürünler tespit edilir. Ardından bu ürünler belirlenen bir sıralama ile aktif kullanıcıya sunulur (Işık, 2022).



Şekil 3.8 Kullanıcı tabanlı işbirlikçi filtreleme yönteminin çalışma mekanizması

Şekil 3.9'daki sözde kod, *Hafiza_Tabanli_Kullanıcı* adlı bir fonksiyon tanımlar. Bu fonksiyon, bir kullanıcı ve bir öge parametresi alır ve kullanıcının ögeye verdiği puanı hesaplar ve bu puanı döndürür. Fonksiyon, *kullanıcı* ve *öge* isimli iki parametre alır (kullanıcının hedef ögeye verdiği puanı tahmin etmeye çalışılır). Fonksiyon, benzer kullanıcıları bulmak için *BenzerKullanicilariBul* fonksiyonunu kullanır ve kullanıcının öge ile ilgili değerlendirmelerini *İlgiliDegerlendirmeler* listesine ekler. Eğer ilgili değerlendirmeler boş değilse, bu değerlendirmelerin ortalamasını hesaplayarak puanı belirler. Eğer boşsa, varsayılan bir puan kullanılır. Bu şekilde, fonksiyon kullanıcının ve ögenin değerlendirmelerine dayanarak benzer kullanıcıları ve değerlendirmeleri kullanarak kullanıcıya öneri yapar.

fonksiyon kullanıcının ve ögenin değerlendirmelerine dayanarak benzer kullanıcıları ve değerlendirmeleri kullanarak kullanıcıya öneri yapar.

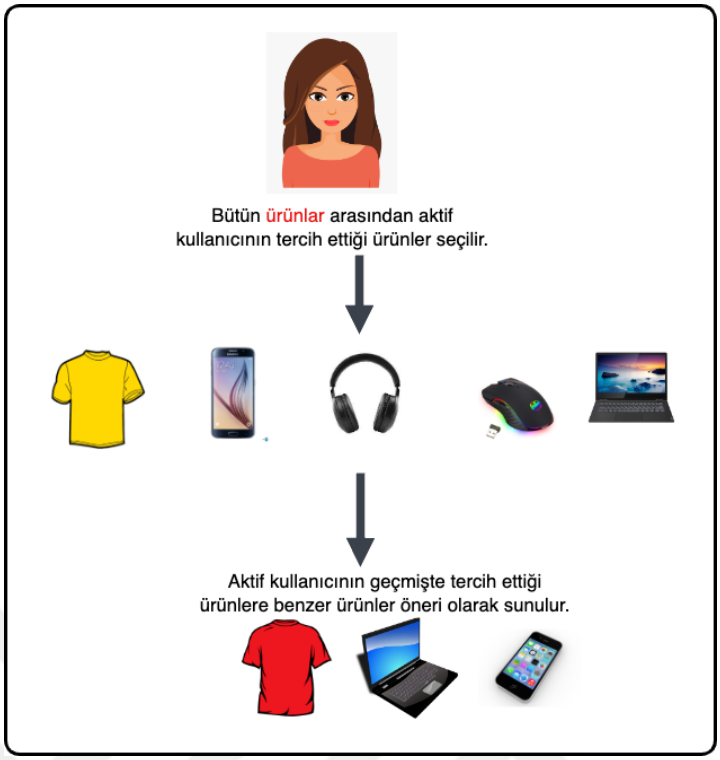
Kullanıcı Tabanlı İşbirlikçi Filtreleme Algoritması

```
Input: kullanıcı (Hedef kullanıcı), öge (Öneri yapılacak öge)
Output: puan (Kullanıcının ögeye verdiği puan)
1. Fonksiyon Hafıza_Tabanlı_Kullanıcı (kullanıcı, öge):
2.
3.   BenzerKullanıcılar = BenzerKullanıcılarıBul(kullanıcı)
4.   İlgiliDeğerlendirmeler = {}
5.
6.   for BenzerKullanıcı in BenzerKullanıcılar:
7.       BenzerKullanıcıDeğerlendirmeleri = DeğerlendirmeleriAl (BenzerKullanıcı)
8.
9.       for Değerlendirme in BenzerKullanıcıDeğerlendirmeleri:
10.          if Değerlendirme.öge == öge:
11.              İlgiliDeğerlendirmeler.append (Değerlendirme)
12.
13.  if İlgiliDeğerlendirmeler boş değilse:
14.      puan = DeğerlendirmelerinOrtalamasınıHesapla (İlgiliDeğerlendirmeler)
15.  else:
16.      puan = varsayılan_puan
17.
18.  Return puan
```

Şekil 3.9 Kullanıcı tabanlı işbirlikçi filtreleme yönteminin sözde kodu

3.3.1.2 Öge Tabanlı Filtreleme (Item-Based)

Bu yöntem, kullanıcılar arasındaki benzerlik yerine ürün benzerliğine odaklanır ve önceki algoritmayla çok benzer bir yapıya sahiptir. Bu algoritma sayesinde, herhangi bir ögeyi satın alan bir kullanıcıya kolaylıkla benzer ögeler önerilebilir. Kullanıcı tabanlı filtrelemeye kıyasla daha az zaman ve kaynak gerektirir. Bu süreç Şekil 3.10'da görüldüğü gibi üç kademede tamamlanır. İlk olarak, aktif kullanıcının geçmişte beğendiği ürünler belirlenir ve benzer özelliklere sahip diğer ürünler tespit edilir. Ardından, bu ürünler belirli bir sıralama ile aktif kullanıcıya sunulur (Işık, 2022).



Şekil 3.10 Öge tabanlı işbirlikçi filtreleme çalışma mekanizması

Şekil 3.11'deki sözde kod öge tabanlı işbirlikçi filtreleme yöntemini uygulayan *Hafiza_Tabanli_Öge* adında bir fonksiyon tanımlar. Bu fonksiyon, bir kullanıcı ve bir öge parametresi alır ve kullanıcının o ögeye verdiği puanı tahmin eder. Fonksiyon, öge tabanlı filtrelemenin çalışma mantığına göre, öneri yapılacak ögenin benzer ögelerini bulmak için *BenzerÖgeleriBul* adlı bir başka fonksiyonu çağırır. Daha sonra, kullanıcının o benzer ögelerle ilgili değerlendirmelerini *İlgiliDegerlendirmeler* adlı bir liste içinde toplar. Eğer bu liste boş değilse, bu değerlendirmelerin ortalamasını alarak puanı hesaplar. Ancak, eğer ilgili değerlendirmeler boş ise, varsayılan bir puanı kullanarak tahminde bulunur. Bu sayede fonksiyon, kullanıcının ve ögenin değerlendirmelerini kullanarak öge tabanlı işbirlikçi filtreleme algoritmasıyla kullanıcıya öneriler sunar.

değerlendirmelerini kullanarak öge tabanlı işbirlikçi filtreleme algoritmasıyla kullanıcıya öneriler sunar.

Öge Tabanlı İşbirlikçi Filtreleme Algoritması

```
Input: kullanıcı (Hedef kullanıcı), öge (Öneri yapılacak öge)
Output: puan (Kullanıcının öğeye verdiği puan)
1. def Hafıza_Tabanlı_Öge (kullanıcı, öge):
2.
3.    BenzerÖgeler = BenzerÖğeleriBul(öge)
4.    İlgiliDeğerlendirmeler = {}
5.
6.    for BenzerÖge in BenzerÖğeler:
7.        BenzerÖgeDeğerlendirmeleri = DeğerlendirmeleriAl (BenzerÖge)
8.
9.        for Değerlendirme in BenzerÖgeDeğerlendirmeleri:
10.           if Değerlendirme.kullanıcı == kullanıcı: # Kullanıcının değerlendirmesi var mı?
11.               İlgiliDeğerlendirmeler.append(Değerlendirme.puan)
12.
13.   if İlgiliDeğerlendirmeler :
14.       puan = DeğerlendirmelerinOrtalamasınıHesapla (İlgiliDeğerlendirmeler)
15.   else:
16.       puan = varsayılan_puan
17.
18.   Return puan
```

Şekil 3.11 Öge tabanlı işbirlikçi filtreleme algoritmasının sözde kodu

3.3.2 Model Tabanlı İşbirlikçi Filtreleme (Model-Based)

HTİF öneri sistemlerinin en önemli avantajlarından biri, kolay uygulanabilir olmalarıdır. Ancak bu yöntem, seyrek matrislerde yeterli öneri sonuçları üretemez. Bu tür durumlar için Model Tabanlı İşbirlikçi Filtreleme (MTİF) yöntemi daha avantajlı olabilir. Kullanıcının önceki etkileşimlerinin vektörünü kullanarak yeni temsiller oluşturmak için bir model hesaplar. Önceden hesaplanmış modele dayanarak anında öneriler yapmayı amaçlar. Bu sayede veri tabanındaki tüm verileri kullanmak zorunda kalmaz. Dolayısıyla sistem için daha verimli ve ölçeklenebilir bir yaklaşım sunar (Barakat, 2020). Hız açısından da MTİF, sadece modele dayalı veri tabanı sorgulaması yaparak daha hızlı sonuçlar elde edilir.

MTİF algoritmasının, tüm veri setini kullanarak öneri üreten HTİF algoritmasına göre performansı, doğru öneri üretme açısından biraz daha düşük olur. Bu nedenle, öneri kalitesi açısından veri tabanındaki tüm verileri kullanmak tercih edilir.

MTİF algoritması başlangıçta model oluşturmak açısından zorlu ve sofistike bir süreç gerektirebilir, ancak elde edilen öneri sonuçları ve sistem ölçeklenebilirliği açısından uzun vadede fayda sağlar. Model tabanlı öneri sistemleri, temelde benzer çalışma prensiplerine dayansa da model oluşturmak ve bu modelleri kullanmak için farklı yaklaşımlar kullanılır. Örneğin, Naïve Bayes, Sinir Ağları, Kümeleme ve Matris Faktörizasyon (Matrix Factorization-MF) gibi çeşitli yöntemler bulunmaktadır. En yaygın olarak kullanılan yöntem, MF temelli olanıdır (Raza ve Ding, 2019). Bu çalışmada MF yöntemi kullanıldığı için bu yöntemin detayları aşağıda açıklanmıştır.

U kullanıcılar, I ögeler ve $R^{|U|\times|I|}$ seyrek derecelendirme matrisi olsun. Matrisi R'deki eksik puanlamaları tahmin ederek tamamlamak istenmektedir. R matrisini Şekil 3.8'deki gibi daha küçük boyutlu P ve Q matrislerine ayrıştırır, hem kullanıcıları hem de ögeleri d boyutundaki gizli faktör uzayına taşır. Burada $d \ll \min(m,n)$, ve gizli faktörler ile hem kullanıcıları hem de ögeleri benzer şekilde tanımlayabilen bir dizi farklı özellik kastedilmektedir (Koren, Bell, ve Volinsky, 2009). Bu nedenle hafıza tabanlı yaklaşımda olduğu gibi kullanıcıları ve ögeleri temsil etmek için puanlama kullanmak yerine, R matrisindeki engelleyici vektörlere kıyasla sayısı daha az ve daha yoğun olan gizli faktörler kullanılır. Böylece MF'nin amacı, Denklem 3.3'te görüldüğü gibi R matrisine dayalı olarak kullanıcılar $P \in R^{|U|\times \mathrm{d}}$ ve ögeler $Q \in R^{|I|\times \mathrm{d}}$ için iki gizli faktör matrisini hesaplamaktır (Alnahhas, 2021):

Şekil 3.12 Matris faktörizasyonu yönteminin örneği

Matematiksel olarak, R 'deki her bir kullanıcı-öge derecelendirme puanı $\hat{r}_{u,i}$ Denklem 3.4 ile tahmin edilir:

$$R_{m\times n} \approx \tilde{R} = P_{m\times d} \times Q^T_{n\times d} \tag{3.3}$$

$$\hat{r}_{u,i} = p_u . q_i^T = \sum_{k=1}^{d} p_{uk} . q_{ki} \tag{3.4}$$

Kayıp fonksiyonu Denklem 3.5'teki gibi elde edilir:

$$\sum_{r_{u,i}\neq 0} \left(r_{u,i} - \hat{r}_{u,i}\right)^2 \tag{3.5}$$

Burada $r_{u,i}$ gerçek derecelendirme puanıdır.

Tahmin hatasını, gerçek ve tahmin edilen değerler arasındaki kök ortalama kare hatasını (RMSE) hesaplayarak aşağıdaki gibi Denklem 3.6 ve Denklem 3.7'de verilmiştir:

$$RMSE = \sqrt{\frac{1}{|S|} \sum_{(u,i)\in S} \left(r_{u,i} - \hat{r}_{u,i}\right)^2} \tag{3.6}$$

$$RMSE = \sqrt{\frac{1}{|S|} \sum_{(u,i)\in S} \left(r_{u,i} - p_u q_i^T\right)^2} \tag{3.7}$$

Burada S kullanıcı-öge kümelerinin sayısıdır. MF'yi uygulamak için öncelikle P ve Q matrislerinin elemanlarına başlangıç değerleri atanır. Bu değerlerin $\sqrt{b/d}$ değerine eşit ve denk olması, b'nin R'deki boş olmayan değerlerin ortalaması ve d'nin hem P hem de Q'daki en küçük boyut olması, çarpma işleminin hızlı yakınsamasına yol açacaktır (Leskovec vd., 2014). Başlangıç değerlerini atadıktan sonra, RMSE hatası hesaplanır ve iki dizinin değerleri değiştirilerek hata azaltılmaya çalışılır. Bu süreç hata minimum olana kadar tekrarlanır. Bu çözüm, farklı algoritmalarla ele

alabileceğimiz optimizasyon problemine benzer (Stokastik Gradyan İnişi, Bias Stokastik Gardiyan İnişi, Alternatif En Küçük Kareler veya ağırlıklı Alternatif En Küçük Kareler).

Model Tabanlı İşbirlikçi Filtreleme Algoritmsı

Input: kullanıcı (Hedef kullanıcı), öge (Öneri yapılacak öge)
Output: puan (Kullanıcının ögeye verdiği puan)

```
1. def ModelTabanlıİşbirlikçiFiltreleme (kullanıcı, öge):
2.
3.     KullanıcıModeli = KullanıcıModeliniOluştur (kullanıcı)
4.     ÖgeModeli = ÖgeModeliniOluştur (öge)
5.
6.     puan = BenzerlikHesapla (KullanıcıModeli, ÖgeModeli)
7.
8.   Return puan
```

Şekil 3.13 Model tabanlı işbirlikçi filtreleme algoritmasının sözde kodu

Model Tabanlı İşbirlikçi Filtreleme algoritması Şekil 3.13'teki sözde kod ile ifade edilebilir. Fonksiyon, kullanıcı ve öge parametrelerini alır ve kullanıcının ögeye verdiği puanı hesaplar. Bu hesaplama, kullanıcı ve öge modellerinin oluşturulması ve bu modeller arasındaki benzerliğin hesaplanmasıyla gerçekleştirilir.

İşbirlikçi Filtreleme yaklaşımının avantajlarından bazıları şunlardır: Kullanıcı tercihlerini analiz ederek benzer kullanıcıların beğendiği ögeleri önererek kişiselleştirilmiş öneriler sunması, yeni kullanıcılar ve daha önce görülmemiş ögelerle çalışabilme yeteneği sayesinde sistemdeki yeniliklere ve değişikliklere uyum sağlaması, benzer tercihlere sahip kullanıcılar arasındaki ilişkileri analiz ederek ilgi çekici ögeler sunabilmesidir. Ancak işbirlikçi filtreleme yaklaşımının bazı dezavantajları da vardır. Soğuk başlangıç problemi, yeni kullanıcıların veya yeni ögelerin yeterli veriye sahip olmaması durumunda doğru öneriler sunmada zorluk yaratabilir. Yeterli miktarda kullanıcı ve öge verisi gereklidir. Veri eksikliği özellikle yeni veya nadir ögeler için doğru önerilerin yapılamamasına neden olabilir. Büyük ölçekli sistemlerde ölçeklenebilirlik sorunları ortaya çıkabilir. Veri setinin büyüklüğü

ve kullanıcı sayısının artması hesaplama ve işlem süresini olumsuz etkiler. Ayrıca, kullanıcının geçmiş tercihlerine dayanan öneriler sınırlı bir çeşitlilik sunar, kullanıcıların farklı kategorilerdeki ögeleri keşfetme fırsatını kısıtlar. Bu avantajlar ve dezavantajlar, işbirlikçi filtreleme yaklaşımının kullanımını belirleyen önemli faktörlerdir. Her bir kullanım durumu farklı avantajlar ve dezavantajlarla karşılaşabilir ve bu nedenle işbirlikçi filtreleme yöntemi, spesifik uygulama senaryolarında dikkatlice değerlendirilmelidir.

3.4 Bilgi Tabanlı Filtreleme Yöntemi (Knowledge-Based)

Bilgiye dayalı öneri sistemleri, derecelendirmelerin öneriler için kullanılmadığı sistemlerdir. Bunun yerine, öneri süreci müşteri gereksinimleri ile ürün açıklamaları arasındaki benzerliklere veya kullanıcı gereksinimlerini belirleyen kısıtlamaların kullanımına dayanır. Bu süreç, geri alım süreci sırasında kullanılacak kurallar ve benzerlik fonksiyonları hakkında verileri içeren bilgi tabanları kullanılarak kolaylaştırılır. Aslında, bilgi tabanları, bu yöntemlerin etkili çalışmasında o kadar önemlidir ki, yaklaşım adını bu gerçekten almıştır (Aggarwal, 2016).

Bilgiye dayalı sistemler, nadiren satın alınan veya seyrek verilere sahip ürünler için özellikle kullanışlıdır. Örneğin, emlak, otomobil, turistik talepler, finansal hizmetler veya pahalı lüks ürünler gibi ürünler bu kategoriye girer. Bu tür durumlarda, öneri süreci için yeterli derecelendirmeler bulunmayabilir. Nadiren satın alınan ve farklı detaylı seçeneklere sahip olan ürünler, belirli bir örnekleme (yani seçeneklerin kombinasyonu) için yeterli sayıda derecelendirme elde etmek zor olabilir. Bu sorun, yeterli derecelendirmelerin öneri süreci için mevcut olmadığı soğuk başlangıç sorunu bağlamında da karşılaşılan bir sorundur.

yeterli derecelendirmelerin öneri süreci için mevcut olmadığı soğuk başlangıç sorunu bağlamında da karşılaşılan bir sorundur.

Bilgi Tabanlı Filtreleme Yöntemi Algoritması

```
Input: kullanıcı (Hedef kullanıcı), özellikler (Kullanıcının tercih ettiği özellikler)
Output: öneriler (Kullanıcıya yapılan öneriler)

1. def BilgiTabanliFiltreleme (kullanıcı, özellikler):
2.     öneriler = []

3.     for her_bir_öge in tum_ögeler:
4.         if özellikler_uyumlu (kullanıcı, her_bir_öge, özellikler):
5.             öneriler. append (her_bir_öge)
6.     return oneriler
8.
7 def özellikler_uyumlu (kullanıcı, öge, özellikler):
8.     for özellik in özellikler:
9.         if öge.özellikler [özellik] != kullanici.özellikler[özellik]:
10.            return False
11.    return True
```

Şekil 3.14 Bilgi tabanlı filtreleme algoritmasının sözde kodu

Bilgi Tabanlı Filtreleme algoritmasının sözde kodu Şekil 3.14'teki gibidir. Fonksiyon, hedef kullanıcı ve kullanıcının tercih ettiği özelliklerin girdi olarak alır ve kullanıcıya yapılacak önerileri hesaplar. Kod, tüm ögeleri dolaşarak her bir öge için kullanıcının tercih ettiği özelliklerle uyumunu kontrol eder. Eğer özellikler uyumlu ise, öge öneriler listesine eklenir. Son olarak, hesaplanan öneriler listesi döndürülür.

Kodun doğru çalışması için özelliklerin ve özellik uyumunun uygun şekilde tanımlanması gerekmektedir. Ayrıca *tüm_ögeler* adında bir veri yapısının (örneğin, liste veya veri tabanı) tanımlanması ve ögelerin uygun şekilde temsil edilmesi beklenir.

3.5 Demografik Öneri Sistemleri

Bu sistemler demografik verileri kullanarak kullanıcıları farklı segmentlere ayırır ve benzer demografik profillere sahip kullanıcıların benzer tercihleri olduğunu

varsayar. Bu tür sistemlerin ilk örneği, etkileşimli bir diyalog aracılığıyla toplanan kişisel bilgilere dayalı olarak kitap öneren Grundy'dir (Rich, 1979). Örneğin bu sistemde aynı yaş aralığındaki kullanıcıların benzer ilgi alanlarına sahip olabileceği düşünülerek öneriler yapar. Demografik öneri sistemleri, kullanıcıların tercihlerini daha iyi anlamak ve kişiselleştirilmiş öneriler sunmak için bir başlangıç noktası olarak kullanılabilir. Bu yaklaşım genellikle tek başına en iyi sonuçları vermese de karma veya bilgiye dayalı modellerin bir bileşeni olarak diğer öneri sistemlerinin gücüne önemli ölçüde katkıda bulunur.

Tablo 3.1'de çeşitli öneri sistemlerinin temel hedefleri yer almaktadır. I 'nın üzerinde önerilerde bulunulabilecek ögeler kümesi, U 'nun tercihleri bilinen kullanıcılar kümesi, u'nun kendisi için tavsiyeler üretilmesi gereken kullanıcı ve i'nin de u'nun tercihini tahmin etmek istediğimiz bir öge olduğunu varsayalım.

Tablo 3.1 Çeşitli öneri sistemlerinin temel hedefleri

Yaklaşım	**Arka plan**	**Girdi**	**Kavramsal hedef**
İşbirlikçi Filtreleme	I'daki ögelerin U'dan gelen derecelendirmeleri	Kullanıcı puanları + topluluk puanları	Kullanıcı ve topluluk derecelendirmelerini kullanarak, işbirlikçi bir yaklaşımla öneriler sunulur.
İçerik Tabanlı Filtreleme	I'daki ögelerin özellikleri	Kullanıcı puanları + öge özellikleri	Kullanıcının geçmiş derecelendirmeleri ve tercihleri doğrultusunda seçtiği özelliklere dayalı içerik tabanlı öneriler sunulur.
Bilgi Tabanlı Filtreleme	I'daki ögelerin özellikleri ve kullanıcının ihtiyaçlarına uygunluğu değerlendirmeleri	Kullanıcı özellikleri + öge özellikleri + alan bilgisi	Kullanıcının istediği özelliklere açık belirtim temel alınarak öneriler sunulur.
Demografik	U hakkındaki demografik bilgiler ve I'daki ögelere verilen derecelendirmeler	u hakkında demografik bilgiler	Kullanıcının demografik özelliklerine benzeyen diğer kullanıcıların derecelendirmeleri kullanılarak tahminler yapılır.

3.6 Karma Filtreleme (Hybrid)

Karma (Hibrit) öneri sistemleri, farklı yaklaşımların bir araya gelerek ortak güçlerini kullanan sistemlerdir (Ricci, Rokach, ve Shapira, 2015). Bu yaklaşım, içerik tabanlı filtreleme ve işbirlikçi filtreleme yöntemlerini birleştirerek daha kapsamlı ve doğru öneriler sunmayı hedefler. İçerik tabanlı filtreleme, içeriğin özelliklerini analiz ederek benzer içerikleri önerirken, işbirlikçi filtreleme kullanıcıların benzer tercihleri ve davranışları temel alarak öneriler sunar.

Karma öneri sistemleri, kullanıcılara daha kişiselleştirilmiş ve çeşitli öneriler sunma potansiyeline sahiptir. Bu sistemler, bir tekniğin dezavantajlarını diğerinin avantajlarıyla dengeleyerek daha iyi bir performans elde etmeyi amaçlar. Örneğin, içerik temelli filtreleme kullanarak ögelerin içeriksel benzerliklerini belirlerken, işbirlikçi filtreleme kullanarak kullanıcıların benzer davranışlarını değerlendirir.

BÖLÜM DÖRT
ÖNERİ SİSTEMLERİNDE SORUNLAR

Bir öneri sisteminde üç temel aşama bulunmaktadır: veri toplama, öğrenme aşaması ve tahmin veya önerilerin oluşturulması. Her bir aşamada kullanılan yönteme bağlı olarak bazı sorunlarla karşılaşılır. Bu sorunlar veri seyrekliği (sparsity), soğuk başlangıç (cold start), ölçeklenebilirlik (scalability), aşırı uzmanlaşma (over specialization), gecikme (latency), benzerlik (synonymy), kararsız kullanıcılar (gray sheep), yanlış yönlendirme (shilling attacks) ve gizlilik (privacy) olarak ifade edilebilir. Bu sorunlar içinden öneri sistemlerinin performansını etkileyen en önemlileri olan soğuk başlangıç, veri seyrekliği ve ölçeklenebilirlik problemleri üzerinde odaklanılacaktır (Su ve Khoshgoftaar, 2009).

4.1 Veri Seyrekliği (Sparsity)

Seyreklik sorunu, öneri sistemlerinde kullanıcıların ve ögelerin derecelendirmelerini belirtmek konusunda isteksiz olmaları veya yetersiz veri sağlamaları nedeniyle ortaya çıkan bir zorluktur. Bu durum, derecelendirme matrisindeki boşlukların ve eksik verilerin artmasına yol açar. Öneri sistemleri, kullanıcıların tercihlerini ve beğenilerini analiz ederek kişiselleştirilmiş öneriler sunmak için kullanıcı-öge derecelendirme matrisinden faydalanır. Ancak kullanıcıların çoğu, her öge hakkında derecelendirme yapmayı tercih etmez veya sadece bazı ögeler hakkında geri bildirimde bulunur. Bu da derecelendirme matrisinin seyrekleşmesine neden olur (Al-Bashiri, Abdulgabber, Romli, ve Hujainah, 2017).

Seyreklik sorunu, benzer tercihlere sahip kullanıcıları ve benzer özelliklere sahip ögeleri eşleştirmeyi zorlaştırır. Öneri sistemleri, bu eşleştirmeleri temel alarak benzer kullanıcıların beğendikleri ögeleri önerir. Ancak seyrekleşmiş bir matriste bu benzerlikleri bulmak ve doğru önerilerde bulunmak zorlaşır. Aynı zamanda tahminlerin kalitesini etkiler. Çünkü daha az veriye dayalı olarak yapılan tahminlerin doğruluğu azalır ve öneri sistemlerinin başarı oranı düşebilir. Ayrıca, derecelendirilmemiş ögeler ve aktif olmayan kullanıcılar gibi veriler de seyrekleşmeye katkıda bulunur. Bu veriler, öneri hesaplamalarında göz ardı edilebilir ve kullanıcı deneyimi olumsuz etkilenebilir. Seyreklik sorununu çözmek için farklı stratejiler

kullanılabilir. Örneğin, eksik derecelendirmelere varsayılan değerler atayarak veya içerik bilgilerini kullanarak boşlukları doldurabiliriz. Ayrıca, boyut azaltma veya grafik tabanlı yöntemler gibi teknikler de kullanılabilir (Kaya, 2019).

4.2 Soğuk Başlangıç Problemi (Cold Start)

Öneri sistemlerinde Soğuk Başlangıç Problemi, yeni kullanıcıların veya yeni ögelerin sisteme eklenmesiyle ortaya çıkan bir durumdur. Bu probleme göre, hakkında yeterli bilgi veya geri bildirim bulunmayan yeni kullanıcılar veya yeni ögeler, kişiselleştirilmiş önerilerin oluşturulmasında zorluklar yaşatabilir (Su ve Khoshgoftaar, 2009). Yeni kullanıcılar için, sistem kullanıcının tercihlerini ve ilgi alanlarını anlamak için yeterli veriye sahip olmadığından, kişiselleştirilmiş öneriler sunmak zorlaşır. Aynı şekilde, yeni eklenen ögeler henüz kullanıcılar tarafından keşfedilmemiş veya etkileşime girilmemiş olabilir, bu da sistemdeki ilişkileri ve benzerlikleri belirleme sürecini zorlaştırır (Adomavicius ve Tuzhilin, 2005). Soğuk Başlangıç Problemi, öneri sistemlerindeki başlangıç aşamasında karşılaşılan bir zorluktur ve kullanıcıların veya ögelerin geçmiş tercihleri veya geri bildirimleri olmadığı durumlarda doğru ve kişiselleştirilmiş öneriler sunmakta zorlanmayı ifade eder. Bu problemin çözümü için, yeni kullanıcıların veya ögelerin profillerini oluşturmak için alternatif yöntemler veya başlangıç verileri kullanma stratejileri geliştirilebilir (Ricci vd., 2015).

4.3 Ölçeklenebilirlik Problemi (Scalability)

Ölçeklenebilirlik, öneri sistemlerinde karşılaşılan önemli bir problemdir. Bilgisayar sistemlerinde olduğu gibi, ölçeklenebilirlik öneri sistemleri için de önemli bir faktördür. Sistem, kullanıcı veya öge sayısı beklenmedik bir şekilde artsa bile verimli ve etkili bir şekilde çalışacak şekilde tasarlanmış olmalıdır (Takács, Pilászy, Németh, ve Tikk, 2009). Bu sorun, İşbirlikçi Filtreleme algoritmasının büyük veri setleri üzerinde yavaş çalışması ve zamanla artan kullanıcı ve öge sayısıyla başa çıkamamasıdır. Bir öneri sistemi, milyonlarca kullanıcı ve ögeyle etkileşim halinde olduğunda gerçek zamanlı geri bildirimler sağlamalı ve hızlı öneriler sunabilmelidir. Ancak İşbirlikçi Filtreleme algoritmaları benzer komşuları bulmak için çok fazla

zaman gerektirebilir. Bu da ölçeklenebilirlik sorununu ortaya çıkarır (Su ve Khoshgoftaar, 2009).

Bu sorunu çözmek için boyut azaltma teknikleri kullanılabilir. Örneğin, Tekil Değer Ayrıştırma (Singular Value Decomposition – SVD) gibi hesaplamaları hızlandırmak için boyut azaltma kullanılabilir. Bu teknikler, mevcut kullanıcı verilerini kullanarak hesaplamalar yapar ve önerileri hızlı bir şekilde üretir.

BÖLÜM BEŞ
ÖNERİ SİSTEMLERİNDE İLERİ DÜZEY KONULAR

Öneri sistemleri, birçok öneri ortamında yeni kullanıcıların ve ürünlerin sürekli olarak sisteme dahil olması ve verilerdeki değişen desenlere uyum sağlaması gibi zorlu görevlerle karşı karşıyadır. Bu nedenle, çevrimiçi öneri algoritmaları, keşif ve sömürü arasında denge sağlayabilme yeteneğine sahip olmalıdır. Bir öneri işlemi, uygulama alanına bağlı olarak farklı stratejiler, nesneler veya algoritmalar arasından seçim yaparak gerçekleştirir. Bu problem, pekiştirmeli öğrenme alanında keşif ve sömürü arasındaki dengeyi sağlamak için çözülmesi gereken bir problemdir. Bu alanda, ÇKH algoritmaları önemli bir sınıf oluşturur ve arama uzayının keşfi ve kullanımını eş zamanlı olarak gerçekleştirirler (Wu, Wang, Gu, ve Wang, 2016).

Örneğin, haber makalelerinin önerildiği bir sistem için soğuk başlangıç sorunu yaygındır. Her zaman yeni makalelerin ortaya çıktığı ve çeşitli algoritmaların etkinliğinin de zamanla değişebildiği bir durumla karşılaşılabilir. Bu gibi durumlarda, verileri sürekli olarak keşfederek çeşitli seçeneklerin arama alanını genişletmek önemlidir. Aynı zamanda, kullanıcıların geri dönüş oranını optimize etmek için öğrenilen verilere dayanarak sömürüyü sağlamak önemlidir. Bu keşif ve sömürü arasındaki denge, çok kollu haydut algoritmaları ile yönetilebilir. Bu örnekte kullanıcının önceki ilgi geçmişine (tıklamalar gibi) bağlı olarak makale sunumunu yönlendiren bir haber portalı ele alınmış olsun. Kullanıcıların ilgi alanlarını karakterize eden özellik vektörleri ile kişiselleştirmenin mümkün olduğu durumda ÇKH algoritmaları kişiselleştirmeyi dahil etme imkânı sağlar. Örneğin, bir kullanıcının spora daha fazla ilgi duyduğu tespit edilirse, öneri sistemi bu bilgiyi kullanarak bu konulara ait önerileri sık sık gösterebilir. Bu bölümde, özellikle öneri sistemlerinde gelişmiş bir varyasyon olan ve bir makine öğrenimi yöntemi olan ÇKH algoritmaları incelenecektir.

5.1 Çok Kollu Haydut Problemleri (Multi-Armed Bandit Problems)

İlk çok kollu haydut problemi, 1933 yılında klinik araştırmaların uygulanmasında Thompson tarafından ortaya atılmıştır. İlk çalışmalarda, bu tür sıralı öğrenme problemlerine “sıralı deney tasarımı” adı verilmişti (Robbins, 1952; Bellman, 1956).

Çok kollu haydut (MAB) terimi ise 1950'lerin sonlarından 1960'ların başlarına kadar yayınlanan birkaç makalede ortaya çıkmıştır (Bradt, Johnson ve Karlin, 1956; Vogel, 1960a, 1960b ve Feldman, 1962). Bu terim, problemi birden fazla kolu olan bir slot makinesiyle oynamaya benzetilerek kullanılmıştır. Çok kollu haydut problemleri, otomatik bir ajanın hem bilgi edinmeye çalıştığı hem de mevcut bilgisini kullanarak en iyi seçimleri yapmaya çalıştığı bir problem sınıfını temsil eder.

Çok kollu haydut problemlerini açıklamak için kullanılan klasik bir örnek şu şekildedir: Şekil 5.1'de gösterildiği gibi farklı beklenen ödül miktarlarına sahip bir dizi slot makinesini (öneri algoritmaları veya stratejileri) düşünelim. Eğer slot makineleri belirli bir sayıda çekilebilirse, sınırlı sayıda çekme ile en fazla toplam ödülü nasıl elde edebiliriz? Çok kollu haydut problemleri, otomatik bir ajanın hem bilgi edinmeye çalıştığı hem de mevcut bilgisini kullanarak en iyi seçimleri yapmaya çalıştığı bir problem sınıfını temsil eder.

Şekil 5.1 Öneri sistemlerinde çok kollu haydutlar problemi

En basit haydut formasyonunda, her bir kolun ödülleri birbirinden bağımsızdır. Bu nedenle her turda ajan sadece seçtiği kol ile ilişkili ödülü görür ve diğer kollar hakkında bilgi sahibi olmaz. Kumarbaz, her bir makinenin kolunu çekerek, ortalama değeri bilinmeyen bir dağılıma sahip ödül alacaktır. Oyuncu, bu slot makinelerinden birinin diğerlerinden daha yüksek (beklenen) ödül ihtimali olduğunu düşünmektedir, ancak tüm makineleri oynamadan bu makineyi belirlemek imkansızdır. Bu makineleri

öğrenme amacıyla oynamak, stratejilerin arama alanının keşfi olarak görülebilir. Haydut algoritmaları, ajanın her turda hangi kolun seçileceğini belirlemek için bir strateji geliştirir. Elbette, bu öğrenme aşaması en iyi ödüle sahip olan makineye optimize edilmediği için denemeleri boşa harcayabilir. Ancak kumarbaz bir makinenin daha iyi bir ödül verdiğini öğrendikten sonra, o makineyi daha çok tercih eder. Bir haydut modeliyle en fazla birikimli ödülü elde etmek için sömürü eylemleri ve keşif eylemleri arasındaki doğru oranı bulmak önemlidir. ÇKH algoritmasının sözde kodu basit bir biçimde Şekil 5.2'de verilmektedir.

Çok Kollu Haydutlar Algoritması

Her turda $(t = 1,2, \dots, T)$:

1. Ajan, A kümesinden bir eylem seçer $(a_{t,k} \in A)$
2. Çevre, seçilen eylem için bir ödül açıklar $(r_{t,k})$

Şekil 5.2 Çok kollu haydutlar algoritmasının sözde kodu

Bir öneri sistemi, bir kullanıcıya bir Web sayfasını önermek konusunda karar vermek zorunda kaldığında, çeşitli stratejiler arasında bir dizi farklı seçimle karşı karşıyadır. Bu seçimler, çeşitli slot makinelerinin koluna karşılık gelir. Bir kullanıcı, önerilen bir sayfanın bağlantısına tıkladığında, öneri sistemi başarılı önerinin başarısı açısından bir ödül (Reward) alır. En basit durumda, tıklama oranı problemi, ikili ödüllerle modellenir, yani bir tıklama 1 birimlik bir ödül anlamına gelir. Bu ödül, bir kumarbazın slot makinesinden aldığı ödülüyle benzer bir şekilde görülebilir. Zamanla, çok kollu haydut algoritması öğrenmeye başlar. Daha fazla tıklama alan web sayfasının daha yüksek ödül kazanma ihtimaline sahip olduğunu anlar. Dolayısıyla, algoritma, daha fazla ödül beklenen web sayfalarına daha fazla ağırlık verir ve bu sayfaları daha sık önerir.

Bu stratejiler, ajanın (öneri algoritması) önceki deneyimlerine ve ödüllerine dayalı olarak hangi kolun daha yüksek bir beklenen değere sahip olduğunu tahmin etmeye çalışır. Farklı haydut algoritmaları farklı stratejiler kullanır ve öğrenme sürecindeki dengeyi farklı şekillerde yönetir.

Haydut algoritmalarının performansını değerlendirmek için birikimli ödül ve birikimli beklenen pişmanlık kullanılır. Bu, ajanın her zaman en yüksek beklenen değere sahip kolu seçtiği durumda elde edilebilecek birikimli ödül ile gerçek seçimlerinden elde ettiği birikimli ödül arasındaki farkı ifade eder. Amaç, bu pişmanlığı mümkün olduğunca düşük tutmak ve en iyi stratejiyi bulmaktır.

Kumarbazın arama alanını keşfetme ve kullanma arasındaki dengeyi düzenlemek için kullanabileceği Naive, ε-Açgözlü (ε-Greedy), Üst Güven Aralığı (Upper Bounding) algoritmaları gibi bir dizi yaygın strateji vardır ancak bu çalışmada Üst Güven Aralığı (ÜGA) yöntemi kullanılacaktır.

5.2 Üst Güven Aralığı (Upper Confidence Bound, UCB)

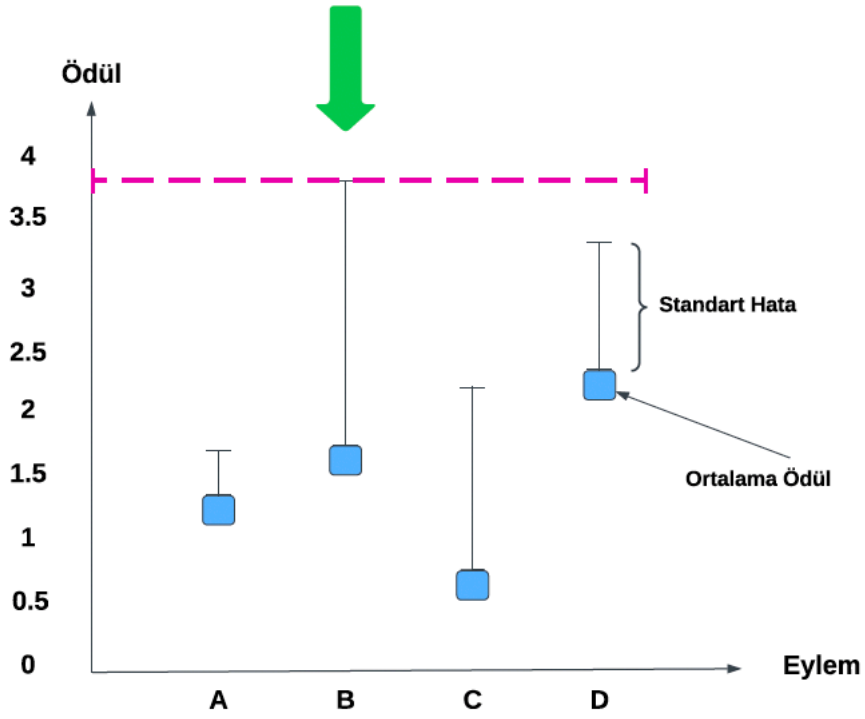
ÜGA algoritması, keşif ve sömürü arasında denge kurarak optimal çözümlere ulaşmayı hedefler. Bu algoritma, belirsizlikleri azaltarak ve istatistiksel olarak anlamlı eylemleri tercih ederek etkili bir şekilde keşif yapmayı sağlar. Başlangıçta, farklı eylemleri deneyerek keşif yapar ve elde ettiği ödüllere dayanarak değer tahminlerini günceller. Sonra, güncellenen üst sınırları kullanarak en yüksek üst sınıra sahip olan eylemi seçerek sömürü yapar. Her bir eylemin değer tahminlerindeki belirsizliği dikkate alarak en iyi eylemi seçme stratejisi sunar. Bu sayede, yüksek ödül olasılığına sahip eylemler zaman içinde daha sık seçilir ve performansı iyileştirilir (Auer, Cesa-Bianchi, ve Fischer, 2002).

ÜGA algoritmasında, kumarbaz slot makinesinin ortalama getirisini kullanmaz. Bunun yerine, yeterince denenmemiş eylemlere daha iyimser bir bakış açısıyla yaklaşır ve bu nedenle daha az keşfedilen veya belirsizlik içeren eylemlere öncelik verir. Nadiren test edilen eylemler daha büyük üst aralıklara sahip olma eğiliminde olacağı için (daha büyük güven aralıkları nedeniyle) daha sık denenirler. Deneme sayısını artırmak, güven aralığının genişliğini azaltır ve bu nedenle üst aralıklar zamanla azalma eğilimi gösterir. Sisteme yeni bir eylem (örneğin yeni bir ürün) eklendiğinde, üst aralığı mevcut eylemlerden birinin altına düşene kadar sıklıkla tekrar tekrar denenir.

ÜGA eylem seçimi Şekil 5.3'te gösterildiği gibi gerçekleşir ve algoritmanın sözde kodu, Şekil 5.4'te sunulmuştur. Denklem 5.1'de görüldüğü gibi ÜGA algoritmasında,

eylem değerinin belirsizliğini veya değişimi ölçen bir karekök terimi kullanılır. A_t, t anında seçilen eylem, $\mu(a)$, t anındaki a eyleminin ortalama değeri, $N_t(a)$ ise t anında a eyleminin seçilme sayısıdır. Maximum işlemi yapılan miktar, eylem a'nın olası gerçek değerinin bir tür üst aralığıdır ve α, güven düzeyini kontrol eden bir parametredir. a her seçildiğinde belirsizlik azalır, $N_t(a)$ artar ve dolayısıyla belirsizlik terimi azalır. Öte yandan, a hariç başka bir eylem seçildiğinde t artar, ancak $N_t(a)$ artmaz ve bu durumda belirsizlik terimi de artar. Doğal logaritma kullanılması, artışların zamanla daha küçük hale gelmesini sağlar, ancak sınırsızdır. Tüm eylemler sonunda seçilecek ancak değeri daha düşük olan veya daha önce sıkça seçilen eylemler zamanla daha az sıklıkla seçilecektir (Sutton ve Barto, 2018).

$$A_t = \underset{a \in (1,2,\ldots,k)}{\operatorname{argmax}} \left[\mu(a) + \alpha \sqrt{\frac{\ln(t)}{N_t(a)}} \right] \tag{5.1}$$



Şekil 5.3 Üst güven aralığı eylem seçimi

Üst Güven Aralığı Algoritması

Input: $A = \{1,2, \ldots, k\}$

1. $t \leftarrow 1$

2. $\forall a \in A$: $\hat{\mu}(a) \leftarrow 0$

3. While $t < T$ **do**

4. $A_t = \underset{a \in (1,2,\ldots,k)}{\operatorname{argmax}} \left[\mu(a) + \alpha \sqrt{\frac{\ln(t)}{N_t(a)}} \right]$

5. a kolu çek ve R ödülü al.

6. Ortalama ödül$\mu(a)$ ve seçim sayısını $N_t(a)$ güncelle.

Şekil 5.4 Üst güven aralığı sözde kodu

5.3 Bağlamsal Haydut Algoritmaları

Bağlamsal haydut (Contextual Bandit) algoritmaları, haydut modellerine bağlam bilgisi ekleyerek yeni bir algoritma sınıfı oluşturur. Genellikle, bir bağlam vektörü, belirli bir varlık hakkında bilgiler içerir. Örneğin bir kullanıcı için bağlam vektörü, kullanıcının önerilen makalelere tıklama sıklığı olabilir veya bir web sitesinde bir günde makale okuma süresi gibi istatistiksel bilgileri içerebilir. Bağlamsal haydut, istatistiksel olarak anlamlı hangi özelliklerin daha yüksek bir ödül elde etmek için önemli olduğunu bilmediğimiz durumlarda kullanışlıdır. Örneğin, kullanıcının cinsiyet kimliği, tercih ettikleri haber makalelerini etkileyebilir. Bağlamsal haydut algoritması, bu özellikleri dikkate alarak daha iyi öneriler üretir ve yeterli eğitimle performansını artırır (Healy, 2022).

Bağlamsal ÇKH problemi, bir dizi bağımsız denemede, çevrimiçi bir algoritmanın belirli bir bağlama (yan bilgi) dayalı olarak, seçilen eylemlerin toplam getirisini en üst düzeye çıkarmak için bir dizi olası eylemden bir eylem seçtiği bir durumu modeller. Ödül hem seçilen eyleme hem de bağlama bağlıdır (Lu, Pál, ve Pál, 2010). Her turda, öneri sistemi sadece bir haydut koluyla değil, aynı zamanda bu iterasyonla ilişkilendirilmiş bir bağlamla da etkileşime geçer.

Bu yaklaşım *soğuk başlangıç* problemini ele almak için kişiselleştirilmiş öneri hizmeti için yaygın olarak kullanılmaktadır (Chu ve Park, 2009). Ayrıca

kişiselleştirilmiş önerilerde, haber makaleleri, e-ticaret ürünleri, müzik veya film önerileri gibi çeşitli öneri sistemleri uygulamalarında kullanılabilir. Bağlam bilgisi, daha kesin ve kişiselleştirilmiş öneriler sağlar, böylece kullanıcılar ilgi alanlarına daha uygun içerikleri keşfeder ve tercihlerine daha iyi şekilde yanıt verir.

Bağlamsal haydutlar algoritmasında, A olarak adlandırılan bir eylem kümesini ve her eylem $a \in A$'nın d-boyutlu bir bağlam vektörüyle temsil edildiğini varsayalım. Bağlam, t anında a eyleminin ve hedef kullanıcının gözlemlenebilir bilgisini tanımlar. Ajan, hedef kullanıcı için bir a^* eylemini seçer ve ardından $x = \{0,1\}$ bir ikili ödülü temsil eden bir $r_{t,a^*} \in x$ ödülünü alır (Zhang, Xie, Li, ve CS Lui, 2020). Bağlamsal haydut problemine yönelik öncü algoritmalar arasında LinRel (Auer vd., 2002) yer almaktadır. LinRel, ödülün eylemin önceki ödüllerinin doğrusal bir kombinasyonu olduğunu varsayar. Aynı düşünceyi temel alan LinUCB bir diğer popüler doğrusal rastgele algoritmadır (Li, Chu, Langford, ve Schapire, 2010). LinUCB kullanıcılar ve ögeler hakkındaki bağlamsal bilgilere dayalı olarak kullanıcılara sunulacak ögeleri sırayla seçer ve toplam kullanıcı tıklamalarını en üst düzeye çıkarmak için öge seçim stratejisini kullanıcı tıklama geri bildirimlerine göre ayarlar. Bu çalışmada, LinUCB detaylı bir şekilde açıklanacaktır. Ayrıca, Agrawal ve Goyal (2013), stokastik bağlamsal haydutlar problemine doğrusal ödül fonksiyonlarıyla Thompson örnekleme yöntemini genişletmeye çalışmışlardır (Yan vd., 2022).

LinUCB veya doğrusal üst güven aralığı algoritması, Li vd. (2010) tarafından tanıtılmış ve en popüler doğrusal stokastik algoritmalardan biridir. Yazarlar, çalışmalarında Yahoo üzerinde kişiselleştirilmiş haber makalesi önerileri sorununu bağlamsal haydut perspektifinden çözmeye çalışmışlardır. Bu algoritma, kullanıcının ve makalelerin bağlam bilgisine dayanarak sıralı olarak makaleleri kullanıcılara sunar ve aynı zamanda kullanıcı tıklama geri bildirimine dayanarak makale seçim stratejisini uzun vadede toplam kullanıcı tıklamalarını maksimize etmek üzere uyarlar. Veri kümesi 33 milyondan fazla etkileşimden oluşmaktadır. Algoritma bağlamdan bağımsız olarak %12,5'lik bir tıklama artışı elde etmiştir.

LinUCB, ÜGA algoritmasının bir uzantısıdır. Her bir eylem için bağlam ve ödül arasında doğrusal bir ilişki olduğunu varsayan bir bağlamsal haydut algoritmasıdır. Bu

ilişkiyi her eylem için ayrı ayrı modeller. Algoritmanın Ayrık LinUCB ve Karma LinUCB olmak üzere iki versiyonu vardır.

5.3.1 Doğrusal Ayrık Üst Güven Aralık Modeli (Ayrık LinUCB)

A_t tüm eylemler kümesinden her bir a eyleminin beklenen getirisinin ($r_{t,a}$) Denklem 5.2 verildiği üzere, bilinmeyen θ_a^* katsayı vektörü ile d -boyutlu $x_{t,a}$ özelliğinde doğrusal olduğu varsayılır;

$$E\left[r_{t,a}\middle|x_{t,a}\right] = \left[x_{t,a}\right]^T \theta_a^* \tag{5.2}$$

Parametreler farklı eylemler arasında paylaşılmadığından bu model ayrık olarak adlandırılır. D_a , satırları m eğitim girişine karşılık gelen t denemesinde $m \times d$ boyutunda bir tasarım matrisi, $C_a \in \mathbb{R}^m$, karşılık gelen yanıt vektörü, I_d , $d \times d$ boyutlu birim matristir. Eğitim verileri (D_a, C_a) kullanılarak ridge regresyonu sonucunda $\hat{\theta}_a$ ile ifade edilecek katsayı tahmini Denklem 5.3 ile elde edilir.

$$\hat{\theta}_a=(D_a^T D_a + I_d)^{-1} D_a^T C_a \tag{5.3}$$

$A^{-1} \in \mathbb{R}^{d\times d}$, $\hat{\theta}_a$ katsayısının kovaryansı olarak yorumlanabilecek güncelleme ağırlıklarıdır. Seçilen a_t eylemi Denklem 5.4'te verildiği gibi, üst güven aralığını maksimize edendir:

$$a_t = \underset{a\in\mathcal{A}_t}{\operatorname{argmax}}\left(x_{t,a}^T \hat{\theta}_a + \alpha\sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}\right) \tag{5.4}$$

Burada $A_a = (D_a^T D_a + I_d)$, ve $b_a = D_a^T C_a$ dir.

Ayrık LinUCB sözde kodu Şekil 5.5'te verilmiştir.

Ayrık LinUCB Algoritması

Input: $\alpha \in R^+$

1. for $t = 1,2, \ldots, T$ **do**

2. Tüm $a \in A_t$ özellikleri gözlemlenir : $x_{t,a} \in R^d$

3. **for all** $a \in A_t$ **do**

4. **if** a is new **then**

5. $A_a \leftarrow I_d$($d \times d$ boyutlu birim matristir.)

6. $b_a \leftarrow 0_{d\times 1}$($d \times 1$ boyutlu sıfır vektörüdür.)

7. **end if**

8. $\hat{\theta}_a \leftarrow A_a^{-1}. b_a$

9. $\mu(a) \leftarrow x_{t,a}\hat{\theta}_a^T + \alpha\sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}$)

10. **end for**

11. $A_t = \underset{a\in(1,2,\ldots,k)}{\operatorname{argmax}} \left[\mu(a) + \alpha\sqrt{\frac{\ln(t)}{N_t(a)}}\right]$

12. a kolu çek ve r_t ödülü al

13. A_{a_t} ve b_{a_t} güncellenir: $\begin{cases} A_{a_t} \leftarrow A_{a_t} + x_{t,a_t}. x_{t,a_t}^T \\ b_{a_t} \leftarrow b_{a_t} + r_t. x_{t,a_t} \end{cases}$

14.end for

Şekil 5.5 Ayrık LinUCB algoritması

5.3.2 Doğrusal Karma Üst Güven Aralık Modeli (Karma LinUCB)

Birçok uygulamada, eyleme özgü olanların yanı sıra tüm eylemler tarafından paylaşılan özelliklerin kullanılması faydalıdır. Örneğin, haber makalesi tavsiyesinde, bir kullanıcı yalnızca sporla ilgili makaleleri tercih edebilir ve bu da bir mekanizma sağlar. Bu nedenle hem paylaşılan hem de paylaşılmayan bileşenleri olan özelliklere sahip olmak önemlidir. Denklem 5.2'nin sağ tarafına başka bir lineer terim ekleyerek karma modeli oluşur bu da aşağıdaki Denklem 5.5'te gösterilmektedir:

$$E\left[r_{t,a}\middle|x_{t,a}\right] = \left[x_{t,a}\right]^T \theta_a^* + \left[z_{t,a}\right]^T \beta^* \tag{5.5}$$

$z_{t,a} \in \mathbb{R}^k$ mevcut kullanıcı/öge kombinasyonunun özelliği ve β^* tüm eylemler için ortak olan bilinmeyen bir katsayı vektörüdür.

Modeller arasındaki temel fark, ayrık modelde yalnızca seçilen kolların özelliklerini güncellerken karma modelde ortak özelliğin etkileşimine karşılık gelen

ödülü güncellemektir. Karma model durumunda güven aralıkları biraz farklı hesaplanır. Algoritmanın sözde kodu Şekil 5.6'da verilmiştir.

Karma LinUCB Algoritması

Input: $\alpha \in R^+$

1. $A_0 \leftarrow I_k$ (k boyutlu birim matrisi)
2. $b_0 \leftarrow 0_k$ (k boyutlu sifir matrisi)
3. **for** $t = 1,2, \ldots, T$ **do**
4. Tüm $a \in A_t$ özellikleri gözlemlenir : $(z_{t,a}, x_{t,a}) \in R^{k+d}$
5. $\hat{\beta} \leftarrow A_0^{-1}. b_0$
6. **for all** $a \in A_t$ **do**
7. **if** a is new **then**
8. $A_a \leftarrow I_d$ ($d \times d$ boyutlu birim matrisi)
9. $B_a \leftarrow 0_{d \times k}$ ($d \times k$ boyutlu sıfır vektörü)
10. $b_a \leftarrow 0_{d \times 1}$ ($d \times 1$ boyutlu sıfır vektörü)
11. **end if**
12. $\hat{\theta}_a \leftarrow A_a^{-1}(b_a - B_a \hat{\beta})$
13. $s_{t,a} \leftarrow z_{t,a}^T A_0^{-1} z_{t,a} - 2\, z_{t,a}^T A_0^{-1} B_a^T A_a^{-1} x_{t,a} + x_{t,a}^T A_a^{-1} x_{t,a} + x_{t,a}^T A_a^{-1} B_a B_a^T A_0^{-1} B_a^T A_a^{-1} x_{t,a}$
14. $\hat{\mu}(a) \leftarrow z_{t,a}^T \hat{\beta} + x_{t,a}^T A_a^{-1} x_{t,a} \hat{\theta}_a + \alpha \sqrt{s_{t,a}}$
15. **end for**
16. $a_t = \underset{a \in \{1,2,\ldots,k\}}{\operatorname{argmax}} \left[\mu(a) + \alpha \sqrt{x_{t,a}^T A_a^{-1} x_{t,a}} \right]$
17. a kolu seçilir ve r_t ödülü alınır.
18. $A_0 \leftarrow A_0 + B_a^T A_{a_t}^{-1} B_a$
19. $b_0 \leftarrow b_0 + B_a^T A_{a_t}^{-1} b_{a_t}$
20. $A_{a_t} \leftarrow A_{a_t} + x_{t,a_t}. x_{t,a_t}^T$
21. $B_{a_t} \leftarrow B_{a_t} + x_{t,a_t} z_{t,a_t}^T$
22. $b_{a_t} \leftarrow b_{a_t} + r_t. x_{t,a_t}$
23. $A_0 \leftarrow A_0 + z_{t,a_t} z_{t,a_t}^T - B_{a_t}^T A_{a_t}^{-1} B_{a_t}$
24. $b_0 \leftarrow b_0 + r_t z_{t,a_t} - B_{a_t}^T A_{a_t}^{-1} b_{a_t}$
25. **end for**

Şekil 5.6 Karma LinUCB algoritması

Burada 5. ve 12. satırlarda katsayıların Ridge regresyonu çözümü ve 13. satırda güven aralığı hesaplanır. Algoritmadaki yapı taşlarının (A_0, b_0, A_a, B_a ve b_a) tümü sabit boyutlara sahip olduğundan ve aşamalı olarak güncellenebildiğinden, algoritma hesaplama açısından verimlidir.

5.4 Birleşimsel Haydutlar

Haydut algoritmalarında, her bir seçenek ya da "kol", tipik olarak bir öge olarak düşünülür ve kullanıcı bir kolu çektikçe, hedef kullanıcıya o kolun karşılık gelen bir ögeyi önerilir. Ancak gerçekte, öneri sistemlerinde her kullanıcıya bir seferde birden

fazla öge sunar (Qin, Chen, ve Zhu, 2014) ve tek bir öge önerme yaklaşımı yalnızca kullanıcıların farklı ihtiyaçlarını karşılamayı zorlaştırmakla kalmaz, aynı zamanda kullanıcının tercihini tam olarak eşleştirmeyi de zorlaştırır.

Kademeli haydut algoritmaları kullanıcıya sıralı bir öge listesi önerir ve kullanıcı, öge listesini kontrol ederken ilk memnun olduğu ögede durur (Craswell, Zoeter, Taylor, ve Ramsey, 2008), (Kveton, Szepesvari, Wen, ve Ashkan, 2015). Bu çalışma temel alınarak Zong vd. (2016), değişen bir öge kümesiyle başa çıkmak için bir dizi doğrusal kademeli haydut düşünmüştür.

Aynı zamanda, (Chen, Wang, ve Yuan, 2013), (Kveton, Wen, Ashkan, ve Szepesvari, 2015) gibi klasik bağlamsal olmayan haydut algoritmasını birleşimsel haydut algoritmasına genelleyen çalışmalar da mevcuttur. Birleşimsel ÇKH'da, öneri algoritması süper kol adı verilen ve belirli bir dağılıma uyan temel kollar kümesi olan bir dizi kolu seçebilir.

Dolayısıyla bu tezde, kol sayısını azaltmak ve çeşitliliği artırmak için bir öneri listesi oluşturmak üzere ögeleri dinamik olarak kümelemeye yönelik işbirlikçi filtrelemeye dayalı bir yaklaşım araştırılacaktır.

5.5 Kümelemeye Dayalı Haydutlar

Geleneksel ÇKH tabanlı öneri modelleri, her ögeyi bir kol olarak ele alır (Christakopoulou ve Banerjee, 2018). Büyük ölçekli kollar, algoritmaların her seferinde her bir kol için ödül dağılımını hesaplamaları gerektiği için daha yüksek hesaplama karmaşıklığına yol açacaktır. Bazı araştırmalar, öneri zorluğunu azaltmak ve umut verici bir pişmanlık kısıtlamasını sürdürmek için kümeleme fikrini haydut algoritmalarına dahil eder. Haydutlar, kollar ve kullanıcılar olmak üzere farklı açılardan kümeleme işlemleri gerçekleştirirler.

Gentile vd. (2014), çeşitli bağımlılıklara dayalı olarak kolları (ürünleri) kümelemeye odaklanan **CLUB** algoritmasını önermiştir. Nguyen ve Lauw (2014), daha iyi kollar elde etmek amacıyla önerdikleri **DynUCB** algoritması ile aynı kümedeki çeşitli kullanıcılar arasında haydut parametrelerin paylaşılması için kullanıcıları k kümeye ayırmıştır. Bu yöntem çevrimiçi kümelemeyi bağlamsal bir

haydut algoritmasıyla birleştirir, ancak kümeleme yalnızca kullanıcılara yapılır. Bu algoritma, zaman içinde değişen bağlamlara ve kullanıcı tercihlerine bağlı olarak kümeleri dinamik olarak yeniden atamak için "K-ortalamalar" benzeri bir kümeleme tekniği kullanır. Hem CLUB hem de DynUCB, dinamik kullanıcı kümelemesi uygular. Aradaki fark, CLUB'un kullanıcıları kümelerden sürekli olarak silmesi, DynUCB'nin ise hedef kullanıcının kümesini dinamik olarak ayarlamasıdır.

COFIBA algoritması birleşik etkiyi dikkate alır, yani bazı kullanıcılar önerilen ögelere benzer şekilde yanıt verir, böylece kullanıcılar etkileşimde bulunulan ögelere göre dinamik olarak gruplandırılır (Li, Karatzoglou, ve Gentile, 2016). **ClexB**, kullanıcılar arasında bilgi paylaşımı için uyarlanabilir bir kümeleme stratejisi tasarlar ve aynı zamanda kullanıcı kümelemesini keşfederek işbirlikçi filtreleme işini geliştirir (Yang vd., 2020). Sanz-Cruzado vd. (2019), en yakın komşu şemasının bir çeşidi olarak görülebilen basit bir kullanıcı kümeleme yöntemi geliştirmiştir. Yöntem, kullanıcılara benzer kullanıcıları rastgele keşfetmek için kontrollü bir yetenek sağlar. Ancak, her turda hedef kullanıcıyı diğer kullanıcılarla karşılaştırması ve ardından benzer kullanıcıları bulmak için kümeleme işlemini yeniden gerçekleştirmesi gerektiği için çok zaman alıcıdır.

5.6 Örtük Geribildirim Modelleme

Kullanıcı ve ögelerin sayısı arttıkça, derecelendirmeler ve satın almalar gibi açık etkileşimleri daha seyrek hale gelir ve kullanıcı tercihlerini yakalamayı daha zor hale getirir. Neyse ki, mevcut öneri sistemleri, kullanıcıların tercihleri hakkında büyük miktarda bilgi içeren zengin, örtük geri bildirim verileri toplamaktadır. Örneğin, kullanıcılar önerilen bir ürüne tıklar veya onu alışveriş sepetlerine ekler, bu da önerilen ürünlerle ilgilendikleri anlamına gelir. Ayrıca kullanıcılar, sadece bir kez tıklanan ürünler yerine iki kez tıklanan veya alışveriş sepetine eklenen ürünleri tercih edebilir. Bununla birlikte, mevcut ilgili çalışmaların çoğu, Bernoulli tabanlı bir ödül mekanizması kullanır. Yani, öneri başarılı olduğunda ödülü 1 olarak ayarlar, aksi durumda ise 0 olarak ayarlar. Bu durumda, önerilen ögelerden yalnızca satın alınanlar başarılı öneriler olarak kabul edilir ve örtük geri bildirim, tercihlerin belirlenmesinde hiçbir rol oynamaz.

Çoğu kullanıcı, önerilen ögelerden ne kadar memnun olduklarını sistemlere doğrudan söylemez, bu nedenle derecelendirmelerde veya yıldızlarda açık geri bildirim toplamak çok maliyetlidir ve toplanan geri bildirimler genellikle çok seyrektir. Aksine, öneri sistemleri, kullanıcı tercihlerinin daha kapsamlı bir resmini sağlayan örtük geri bildirim verilerini kolayca toplayabilir (Yan, Xian, Wan, ve Wang, 2021). Örtük verilerle başa çıkmanın en yaygın yolu, onları doğrudan 0 veya 1'e dönüştürmektir (Joachims, Granka, Pan, Hembrooke, ve Gay, 2017). Zoghi vd. (2017) kullanıcı davranışını karakterize etmenin basit ama etkili bir yolu olan basamaklı haydutu (Cascading Bandits) tanıtmıştır. Ancak, yalnızca bir davranış türünü dikkate alır ve davranışların çeşitliliğini göz ardı eder.

Daha yakın tarihli ilgili çalışma, örtük geri bildirimin farklı davranış türlerine daha fazla odaklanmıştır. Schoinas ve Tjortjis (2019) ilişkilendirme kurallarının davranış geçmişine dayalı olarak eksik değerleri ayıklamak ve doldurmak için kullanıldığı, ilişkilendirme kuralı madenciliğine (association rule mining) dayalı daha yoğun bir kullanıcı ögesi matrisi oluşturmuştur. Kullanıcı ögesi geri bildirimi etkileşimi olanlar için 1, etkileşimi olmayanlar için 0 alınır. Bu şekilde çoklu kolların ve güven aralıklarının ağırlıklı toplamı ile nihai tahmin puanı elde edilir. Gao, vd. (2019), kullanıcıların çoklu davranış verilerinden öğrenme tercihleri için sinirsel çoklu görev önerisi (NMTR) adlı bir çözüm sağladı. Kullanıcılar ve ögeler arasındaki karmaşık, çok türlü etkileşimleri yakalamak ve geri bildirim sinyallerinden yararlanmak için çok görevli bir öğrenme çerçevesine dayalı olarak bunları ortaklaşa optimize etmek için bir sinir ağı modeli kullanırlar.

Tablo 5.1 ÇKH tekniklerine dayalı birkaç öneri sistemi çalışmaları

Bağlamsal ÇKH	Kümelemeye dayalı ÇKH	Birleşimsel ÇKH	Örtük geribildirimin modellenmesi
LinRel (Auer vd., 2002)	**DyUCB** (Nguyen ve Lauw, 2014)	**C²UCB** (Qin vd., 2014)	**MIF-TS** (Yan vd., 2021)
LinUCB (Li vd., 2010)	**COFIBA** (Liv vd., 2016)	**An Experimental Comparison of Click Position-Bias Models** (Craswell, Zoeter, Taylor, ve Ramsey, 2008)	**Accurately Interpreting Click through Data as Implicit Feedback** (Joachims vd., 2017)
Thompson Sampling for Contextual Bandits (Agrawal, ve Goyal, 2013)	**CLUB** (Gentile vd., 2014)	**Cascading Bandits** (Kveton, Szepesvari, Wen, ve Ashkan, 2015)	**Batch Rank** (Zoghi, vd., 2017)
	ClexB (Yang vd., 2020)	**Cascading Bandits for Large-Scale Recommendation Problems** (Zong vd., 2016)	**MuSIF** (Schoinas ve Tjortjis, 2019)
		Combinatorial Multi-Armed Bandit (Chen vd., 2013)	**NMTR** (Gao vd., 2019)
		Combinatorial Cascading Bandits (Kveton vd., 2015)	

5.7 Problem Formülasyonu

Pekiştirmeli öğrenme paradigmasını izleyerek, ilk olarak tipik e-ticaret öneri görevini bağlamsal birleşimsel bir haydut problemi olarak modellenecektir. t zamanında öneri sistemi kullanıcı etkileşimini (bağlam) ve önceki turlardan topladığı kullanıcı geribildirimini (ödül) gözlemleyerek bir politika fonksiyonuna göre bir öneri stratejisi (eylem) belirler. Ardından, hedef kullanıcıya birden fazla ürün içeren bir öneri listesi sunar. Öneri sistemi, kullanıcının tepkisine karşılık gelen yeni kullanıcı geribildirimini (ödül) alır.

Kullanıcı tepkisi, önerilen ürünlerin kalitesine dair geri bildirim sağlar. Bu geri bildirim, öneri sisteminin parametrelerini güncellemek için kullanılır ve daha doğru eylemler yaparak bir sonraki turda daha iyi öneriler sunmaya yardımcı olur. Şekil 5.7 tipik bir kişiselleştirilmiş çevrimiçi öneri görevinin iş akışını ve haydut problemi mimarisini gösterir. Bu durum Şekil 2.1'de bağlamsal olmayan şema verilmiş olan şemaya bir ek ve bağlamsal detay eklenmiş halidir.

Bölüm 2'de haydut içindeki kritik bileşenlere kısa bir giriş yapılmıştı; ancak, bu bölümde bu bileşenlere daha fazla ayrıntı verilecek ve haydut içindeki bu unsurların etkileşimleri açıklanacaktır.

Şekil 5.7 Bağlamsal çok kollu haydutlar algoritmasının modellenmesi

Ajan: Pekiştirmeli öğrenmenin temel problemi, bir ajanın karmaşık ve belirsiz bir ortamda elde edebileceği ödülü nasıl maksimize edeceğidir. Benzer şekilde, öneri sistemleri kullanıcı memnuniyetini maksimize etmeyi amaçlar.

Çevre: Varsayalım ki, I öge kümesi, U kullanıcı kümesi ve aralarında etkileşim davranışı olan S etkileşim davranışı dizi kümesi olsun. Kullanıcı u'nun davranış dizisi $\hat{S}_u^T = (s_{u1}, s_{u2}, \dots, s_{uT})$, şeklinde ifade edilir. Burada T dizindeki son zamanı temsil eder ve s_{ut}, kullanıcı u'nun t zamandaki $(1 < t < T)$etkileşim davranışlarının bir

kaydını temsil eder. Kayıt, kullanıcı kimliği (user ID), öge kimliği (item ID), öznitelikler (attributes) ve etkileşim türünden (iteraction type) oluşur.

Bağlam: Birçok haydut problemde, ajanın eylemlerin kalitesini tahmin etmede yardımcı olabilecek ek bilgilere, yani bağlama erişimi vardır. Benzer şekilde, öneri alanında, kullanıcıların veya ögelerin yardımcı bilgileri kullanılarak kullanıcı tercihleri yakalanır ve bu şekilde kişiselleştirilmiş önerilerin kalitesi artırılır. Sistem için bağlam bilgisi, t zamanındaki c_t olarak temsil edilir.

Eylem: Bağlamsal birleşimsel haydutta, kullanıcı u tarafından t zamanında çekilebilecek J adet mevcut seçenek veya kol $A = \{a_1, a_2, \dots, a_J\}$ bulunmaktadır, burada $|A| << |I|$. Bir kol, tek bir ögeden ziyade öge alt kümesine karşılık gelir ve "süper kol" olarak adlandırılır. Her süper kol çekilmek için bir şansa sahiptir. Her bir süper kol $a \in A$ için haydut, c_t bağlamı gözlemler, burada d boyutlu bağlamsal özellik vektörü $x_{t,a} \in \mathbb{R}^d$ şeklinde ifade edilir. Önceki iterasyonlardaki ödül deneyimine dayanarak, Denklem 5.5'te görüldüğü gibi, ajan en iyi süper kol a^*çekmeyi seçer:

$$a^* = arg \max_{a \in A} \mathbb{E}(R_a) + f_{UCB}^a \tag{5.5}$$

Burada $\mathbb{E}(R_a)$, haydut parametreleri tarafından belirlenen a süper kolunun tahmin edilen ödülüdür ve f_{UCB}^a, a süper kolunun üst sınır güven aralığını hesaplamak için kullanılan bir fonksiyondur. Haydut her seferinde en yüksek üst sınır güven aralığına sahip olan süper kolu seçer. Ardından, çevrimiçi bağlamsal birleşimsel haydut tabanlı öneri sistemi, kullanıcı u için t zamanında a^*'e karşılık gelen bir öneri listesi $\bar{I}_{ut}(|\bar{I}_{ut}| \geq 1)$ oluşturur.

Ödül: Ajan a^* seçtikten sonra, r_{t,a^*} ödülünü gözlemler. Ödül, kullanıcıların öneri alanındaki önerilen ögelere verdiği yanıtları veya geri bildirimleri ifade eder. Ardından ajan, gelecekteki yinelemelerde en iyi süper kolu seçme stratejisini geliştirmek için devam eden yinelemede $(x_{t,a^*}, a^*, r_{t,a^*})$ gözlemlerinden öğrenir. T turundan sonra ajan, birikimli ödülünü $\mathbb{E}[\![\sum_{t=1}^{T} R_t^*]\!]$ gözlemler; burada R_t^*, seçilen süper kol tarafından t anında elde edilen beklenen ödül ve birikimli pişmanlık (Reg olarak kısaltılır) T turları için Denklem 5.6'daki şekilde tanımlanır:

$$Reg(T) = \mathbb{E}\left[\left[\sum_{t=1}^{T} \max_{a \in A} \mu_a\right]\right] - \left[\left[\sum_{t=1}^{T} R_t^*\right]\right] \quad (5.6)$$

Burada ilk terim, herhangi bir süper kol kullanılarak beklenen maksimum ödüldür, yani, $\mathbb{E}\left[\left[\sum_{t=1}^{T} \max_{a \in A} \mu_a\right]\right] \geq [[\sum_{t=1}^{T} R_t^*]]$ ve μ_a, a süper kolunun beklenen ödülünü temsil eder.

$Reg(T)$ genellikle haydut algoritmalarının performansını ölçmek için kullanılır ve küçük değerleri sonucun iyi olduğunu gösterir. Ancak her kolun getirisini gözlemlemek pek mümkün değildir. Bu yüzden $Reg(T)$'nin hesaplanması zordur. Birikimli pişmanlığı en aza indirme, birikimli ödül maksimizasyonuna eşdeğerdir. Dolayısıyla, bağlamsal birleşimsel haydut öğreniminin amacı resmi olarak T turlarından sonra birikimli ödülü maksimize etmektir.

5.8 DC^{3}MAB Çalışma Prensibi

5.8.1 Dinamik Kullanıcı Kümeleme

Kullanıcılar için dinamik kümelemede her C_j kullanıcı kümesi bir $\overline{w}_{C_j}$ küme parametresi tutar ve her u kullanıcı için özgün bir w_u parametresi bulunur. Başlangıçta, kullanıcı özellik vektörlerine dayanarak tüm kullanıcıları kümelemek için K-ortalama kümeleme (K-means) algoritması kullanılır. K-ortalama algoritmasının sadece başlangıçta kullanıcıları kümelemek için kullanıldığını belirtmek önemlidir. Daha sonra, k-ortalama algoritması çevrimiçi gerçek zamanlı öneriler için zaman alıcı ve uygun olmadığından, farklı bir strateji kullanılarak kullanıcıların kümeleri dinamik olarak ayarlanır. Hedef kullanıcının kümesini nasıl dinamik olarak ayarlayacağımız ayrıntılı olarak aşağıda açıklanacaktır. Öneri sisteminin t zamanında u kullanıcısına hizmet verdiğini ve girdi değerinin u'nun tercihini ortaya çıkarmak için u'nun etkileşim bilgisi olduğu varsayılsın.

Kullanıcı kümeleme yoluyla, öneri sistemi hedef kullanıcıya benzer kullanıcıları bulur ve benzer kullanıcıların işbirliğini kullanarak öneri listesini belirler. Bu, seyrek

veri nedeniyle mevcut bilginin yetersiz olduğu sorununu çözecektir. Strateji aynı zamanda benzer kullanıcıların bilgeliğini kullanarak kararlar alır.

5.8.2 Dinamik Öge Bölümlendirme

Çok sayıda öge nedeniyle, çevrimiçi önerilerin tur başına her öge için beklenen ödülü hesaplaması zaman alıcıdır ve ayrıca her kol için ödül dağılımını elde etmek amacıyla her kolu seçmek pratik değildir. Bu nedenle, pratik öneri sistemlerinde bir ögenin kol olarak modellenmesi uygun değildir.

Toplam ögeleri S_{ut}'ye göre bölmek için J ($J > 0$ ve $J \ll |I|$) farklı tahminci olduğunu varsayalım, her bir tahminci farklı bir işbirlikçi filtreleme tekniği kullanabilir. Yan vd. (2021) tarafından ortaya koydukları gibi tahmincileri iki türe ayırıyoruz: öznitelik tabanlı tahminciler ve davranış tabanlı tahminciler.

Öznitelik tabanlı tahminciler, etkileşimde bulunulan ögelerle benzer özniteliklere sahip ögeleri seçer. Örneğin, etkileşimde bulunulan ögelerle aynı renkte veya markada olan ögeleri veya aynı yaş grubundaki en popüler ögeleri getirir. Genel olarak, veri kümesinin özelliklerine göre, kullanıcıların veya ögelerin (filmler, makaleler, müzik, ürünler) öznitelikleri, toplam ögeleri filtrelemek için kullanılabilir, böylece etkileşimde bulunulan ögelerle aynı özelliklere sahip olan ögeler bir öge alt kümesine gruplanabilir ve bu alt küme bir süper kol ile eşleşir.

Davranış tabanlı tahminciler, olaylar veya etkileşim davranışları yoluyla ögeleri seçer. Örneğin, hedef kullanıcının benzer kullanıcılarla mevcut etkileşimi ile aynı etkileşim davranışına sahip kullanıcıları tanımlarız. Bu benzer kullanıcıların favori ögeleri, bir öge alt kümesini oluşturur.

Eylem setleri oluşturmak için öge niteliklerini çiftler halinde birleştirme gibi yöntemler kullanılır. Bu metodoloji, öge özelliklerinin eşleştirilmesini ve daha da önemlisi, her iki özelliği de aynı anda karşılayan ögelerin filtrelenmesini içerir. Amaç, aşırı küçük öge setlerinin oluşturulmasından kaçınırken, önerilerin son derece doğru olmasını ve kullanıcı tercihleriyle uyumlu olmasını sağlamaktır. Ek olarak, öge özellikleri ile kullanıcı davranışı arasındaki ilişki, eylemlerin oluşturulmasını kolaylaştırmak için kullanılır. Bu stratejik yaklaşım, önerilerin doğruluğunu sağlamayı

ve öge setinin hantallaşmasını önlemeyi amaçlamaktadır. Ayrıca, kullanıcı tercihleri değişebilir ve hatta aynı kullanıcı farklı zamanlarda farklı ögelere ilgi duymaya eğilimli olabilir. Her süper koldaki ögeler de kullanıcı davranışına göre dinamik olarak değişir.

5.8.3 Haydut

Algoritma her süper kolun bağlam bilgisini gözlemleyerek, üst güven aralığını en büyük yapan süper kol (a^*), Denklem 5.7 ile elde edilir.

$$a^* = \underset{a \in A}{\operatorname{argmax}}(\, \overline{w}_{c^*}^{T}\, x_{t,a} + CB_t) \tag{5.7}$$

$$CB_t = \sqrt{x_{t,a}{}^{T}\, \overline{M}_{c^*}^{-1}\, x_{t,a} \log(1+t)} \tag{5.8}$$

Burada Denklem 5.8'de verildiği üzere CB_t , teorik üst güven aralığın basitleştirilmiş bir versiyonudur, $\overline{w}_{c^*}$ kullanıcının bulunduğu kümenin parametresidir ve c^* kümesinin tercihini yansıtır. Burada α, keşfin önemi için bir parametredir. a^* Süper kolu seçildikten sonra, buna karşılık gelen öge alt kümesi önerilen aday ögeler olarak kullanılacaktır. Gerçek dünyada süper kolların hepsi her zaman mevcut değildir. Tahminciler ögeler arasında hiçbir korelasyon bulamadığında bu durum ortaya çıkar. Haydut politikası, böyle durumlarla etkin bir şekilde başa çıkabilir. Genellikle seçilen a^* süper kolun uzunluğu N'den ($|a^*| >= N$) büyük olacaktır, bu nedenle algoritma rastgele N öge seçecektir. N'den az olduğunda, süper kol içindeki tüm ögeler nihai önerilen ögeler olacaktır. Sıfır olduğunda, algoritma bu seçilen süper kolu atlayacak ve yeniden başka bir süper kol çekecektir.

5.8.4 Çok Sınıflı Ödül Mekanizması

$\bar{I}_{ut}$, t anında kullanıcı u için nihai öneri listesini temsil eder. Makine öğreniminde bir kayıp işlevini optimize etmek için kullandığımız yönteme benzer şekilde, haydut algoritması, bir sonraki eylemi sürekli olarak geliştirmek için önceki ödül geri bildirimine dayanır. T anında öneri performansını doğrularken, kullanıcının önerilen ögelere ilgi gösterip göstermediğini gözlemlemek gerekir. Eğer kullanıcı $t+1$ anında

önerilen ögeleri satın almazsa, bu önerinin başarısız olduğunu doğrudan sonuçlandıramayız. Burada, kullanıcıların bir ögeye ilgi gösterip göstermediğini belirtmek için "güçlü tercih" kavramını kullanılıyor. Örneğin, bir kullanıcı bir öge için favori, sepete ekle veya satın al düğmelerine tıklarsa, kullanıcının o ögeye çok ilgi gösterdiğini düşünülür ve bu üç davranışa "güçlü tercih davranışı" denir. Diğer taraftan, bir kullanıcı bir ögeye sadece bir kez tıklarsa, o ögeye çok ilgi göstermediğini düşünülür ve bu "zayıf tercih davranışı" olarak adlandırılır. Çoğu mevcut çalışma, Bernoulli tabanlı bir ödül mekanizması kullanır, başarılı bir öneri için ödül 1 ve aksi durumda 0'dır. Ancak, gerçek uygulamalarda, kullanıcı davranışı karmaşık ve çeşitlidir. Bernoulli tabanlı ödül ayarlaması, örtük geri bildirim bilgisini kullanmazken, örtük geri bildirim hedef kullanıcının tercihini ortaya koyabilir. Bu nedenle, burada "Çok Sınıflı Ödül Mekanizması" farklı davranış türleri için farklı geri bildirim değerlerini ödül olarak alır. Farklı etkileşim tipleri için farklı ödül ağırlıkları belirler ve bu şekilde tercih derecesini daha iyi açıklar. Diyelim ki beş davranış türü var, bunlar arasında etkileşim yok, tıklama, favori, sepete ekle ve satın alma bulunmaktadır. Yukarıdaki davranış türleri için ödül ağırlıkları, ortaya koyulan tercih derecesine göre kademeli olarak artar. Ayrıca, i ögesinin ödülü Denklem 5.9 gibi tanımlanır:

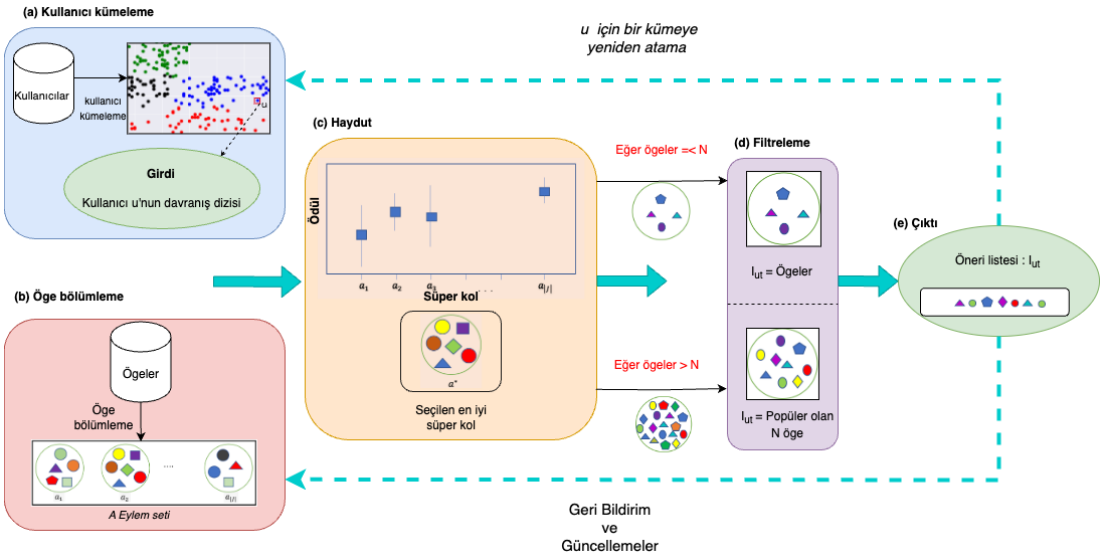
$$r_i = \begin{cases} RW_{ui}, & \text{Eğer başarılı bir öneriyse} \\ 0, & \text{Aksi takdirde} \end{cases} \tag{5.9}$$

Burada RW_{ui}, i ögesinin u kullanıcıya önerildiğinde ödül ağırlığıdır. Son olarak seçilen a^* süper kolu için ödül Denklem 5.10 şekilde tanımlanır:

$$r_{t,a^*} = \sum_{i \in \overline{I}_{ut}} r_i \tag{5.10}$$

Her turdan sonra, öneri sistemi u kullanıcının tercihlerinde değişiklikleri tespit eder ve u'ya ait ilgili parametreleri günceller. u ve her kullanıcı kümesi arasındaki tercih benzerliğine göre, öneri sistemi en uygun kümeyi bulacaktır.

Bu çalışmada DC^{3}MAB yönteminin gücü, gerçek dünyadaki çevrimiçi öneri sistemlerinde dinamik ihtiyaçlarını karşılamak için kullanılmış ve DC^{3}MAB-π algoritmasında, DC^{3}MAB algoritmasından dört temel bileşene (***Dinamik kullanıcı kümeleme***, ***Dinamik öge bölümleme***, ***Haydut*** politikası ve ***Çıktı)*** ek olarak beşinci bir modül "***Filtreleme***" eklenmiştir. Kullanıcı Kümeleme modülü, benzer etkileşim davranışına sahip kullanıcıları gruplandırırken, dinamik öge bölümleme modülü, ögeleri öznitelik ve davranışa dayalı olarak süper kollar adı verilen alt kümelere ayırır. Bu modül, ögelerin gruplandırmalarını sürekli olarak farklı alt kümelere dönüştürerek sistemde kritik bir rol oynamaktadır. Haydut modülü kullanıcılara politika tabanlı önerilerde bulunur. Algoritma, her süper kolun bağlamını göz önünde bulundurarak, üst güven aralığını en üst düzeye çıkararak a^* olarak gösterilen en uygun süper kolu seçer (Li vd., 2010). Yeni tanıtılan Filtreleme modülü, öge sayısının önceden tanımlanmış bir eşiği aştığı durumları ele almaktadır. Bu gibi durumlarda, bu modül en popüler ögelerin tavsiye için seçilmesini sağlar. Çıktı modülü daha sonra bu kişiselleştirilmiş önerileri sunarak tercih yakalamayı optimize eden ve gelişen kullanıcı tercihlerine uyum sağlayan gelişmiş bir algoritma ortaya çıkarır. DC^{3}MAB-π algoritması Şekil 5.8'de gösterilmektedir. Sonraki adımda, bu modüllerde yer alan temel işlevler açıklanacaktır.



Şekil 5.8 DC^{3}MAB-π algoritması

DC^{3}MAB-π'nin ayırt edici özelliği ise "Filtreleme" işlemidir. Bir süper kol içindeki sayı önceden tanımlanmış bir eşiği aştığında ögeleri rastgele seçen DC^{3}MAB'den farklı olarak, yöntemimiz ögeleri birikimli ağırlıklı eylem türüne göre seçer (tıklama=1, favori=2, sepete ekle=3, satın alma=4). Burada amaç rastgele herhangi bir ürünü seçmek yerine kullanıcı tarafından daha fazla ilgi duyabileceği ürünleri öncelik vermektir. Örneğin kullanıcılar bir ürünle daha çok etkileşimde bulunmuşlarsa o ürünlerin popular olmasınınım bir göstergesidir ve önerilecek listede bulunma şansı artmakta olmalıdır. Özellikle, daha yüksek birikimli etkileşim seviyelerine sahip ögelere öncelik verir ve en iyi süper kol içindeki en iyi N ögeyi seçer. Bir süper kol içindeki birden fazla öge aynı birikimli etkileşim türüne sahipse, sistem bunlardan birini rastgele seçer. Buna ek olarak, değerli öneriler sunmak için, seçilen a^* süper kolu üzerinde önceden tanımlanmış kriterlere dayalı bir filtreleme işlemi de uygular. Daha önce satın alınmış ürünleri ve tekrarlanan a^*'den silinir. Bu aşama, önerilerin çeşitliliğini korumasını ve aynı zamanda kullanıcı tercihlerine ve ürün popülerliğine uyum sağlamasını sağlar.

DC^{3}MAB-π algoritması, Şekil 5.9'da açıklanmıştır. Kullanıcı tercihleri kullanıcı kümeleri ile ilişkilendirilir. Özellikle, U kullanıcı seti $c_1, c_2, \dots, c_k$ şeklinde k kümeye bölünür. Aynı kümedeki kullanıcılar benzer tercihlere sahiptir, farklı kümedeki kullanıcılar ise farklı tercihlere sahiptir. Algoritma u kullanıcısına hizmet verdiğinde, birkaç tahminci toplam ögeleri farklı öge alt kümelerine bölerek bunları süper kollar $A = \{a_1, a_2, \dots, a_J\}$ olarak modellemektedir.

getActions işlevi, öge bölümlendirme işlevini uygulamak içindir. Süper kollar, t zamanında $\{x_{t,a_1}, x_{t,a_2}, \dots, x_{t,a_J}\} \in R^d$ bağlamsal özellik vektörleri kümesi olarak temsil edilir. Çünkü bağlamsal bilgi zamanla değişir, getActions işlevi her an çağrılmalıdır. Üst güven sınırını hesaplamak için u'nun bulunduğu C^* kümesinin parametresi kullanılır. Algoritma, 11. satırda gösterilen politika fonksiyonuna dayalı olarak bir a^* süper kolu seçer. Bu, süper kola karşılık gelen bir öge alt kümesinin kullanıcı u'ya filtreleme aşmasından geçerek önerilmesi anlamına gelir. 12 ve 13. satırlarında eğer ürünler listesi elemanların sayısı N değerinden küçük ise tüm a^*'deki elemanları u kullanıcısına önerilir. 14 ve 15. satırlarda eğer ürünler listesi elemanların

sayısı N değerinden büyük ise a^* süper kolu içindeki N yüksek birikimli etkileşim seviyelerine sahip ögeleri seçerek ödül r_{t,a^*} değeri verilir.

Bağlamsal özellik vektörü x_{t,a^*} ve seçilen süper kola karşılık gelen ödül r_{t,a^*}, kullanıcı u'nun parametresini güncellemek için kullanılır, bu da u'nun bulunduğu kullanıcı kümesini ayarlamaya yardımcı olur. Ayrıca, bunlar süper kolların ödül dağılımını güncellemek için kullanılır, bu da bir sonraki turda karar vermeye yardımcı olur. Özellikle, algoritma, u kullanıcısının tercihini yansıtan ve $w_u = M_u^{-1} b_u$ ile tanımlanan w_u'yu güncellemek için seçilen süper kolun bağlamsal özellik vektörünü x_{t,a^*} kullanır. Kullanıcı parametresini güncelleme işlemi yalnızca w_u için yapılır. Kullanıcı u dışındaki diğer kullanıcılar için parametreler güncelleme yapılmaz. w_u değiştiğinde, haydut algoritması, w_u'ya en yakın tercih katsayısına sahip bir küme oluşturarak kullanıcı u'yu yeniden bir kümelere atar. Ardından, $\overline{w}_{c^*}$ ve $\overline{w}_{\bar{c}}$ yeniden hesaplar. Böylece, dinamik kullanıcı kümeleme, hedef kullanıcıya en benzer tercih parametresini bulmak suretiyle gerçekleştirilir ve bu sayede algoritma kullanıcı tercihlerindeki değişikliklere uyum sağlayabilir.

DC³MAB-π algoritması

1. k kullanıcı kümesini başlatın ve $b_u = 0 \in R^d$,
   $M_u = I \in R^{d \times d}$ Tüm kullanıcılar için $u \in U$
2. *Her c_j kümesi için $\bar{w}_{c_j}$ katsayısını hesaplayın*
3. $\bar{M}_{c_j} = I + \sum_{u \in c_j} (M_u - I)$
4. $\bar{b}_{c_j} = I + \sum_{u \in c_j} b_u$
5. $\bar{w}_{c_j} = I + \bar{M}_{c^*}^{-1} . \bar{b}_{c_j}$
6. Hizmet vermek için bir kullanıcı u seçin
7. **for** $t \leftarrow 1,2, \ldots, \frac{2}{3}T$ **do**
8. u'nun kullanıcı kümesi C^*'yi elde edin
9. t zamanında süper kolları tanımlayın, $\mathcal{A} = \textbf{getActions}(S_{ut}, t)$
10. Süper kolların bağlamlarını gözlemleyin.
    $x_{t,a} = \frac{1}{|a|} \sum_{a \in a} x_a \ , \forall a \in \mathcal{A}$
11. Haydut politikasını çağırın ve bir eylem seçin,
    $a^* = \underset{a \in \mathcal{A}}{\text{argmax}} \left[\bar{w}_{c^*}^T . x_{t,a} + \alpha \sqrt{x_{t,a}{}^T . \bar{M}_{c^*}^{-1} . x_{t,a} . \log(1+t)} \right]$
12. **if** $|\bar{I}_{ut}| < N$:
13. $\bar{I}_{ut} = I_{u_t}$
14. **else:**
15. I_{u_t}dan en popüler N ögeyi seçin (eğer eşit değere sahip olan varsa onların arasından rastgele seç)
16. Seçilen a^* süper kola karşılık gelen çıkış sonucunu $\bar{I}_{ut}$ alın
17. Çok sınıflı ödül mekanizması aracılığıyla ödülü gözlemleyin, $r_{t,a^*} = \textbf{getReward}(\bar{I}_{ut}, t)$
18. $\bar{x}_t = x_{t,a^*}$ olarak ayarlayın
19. u' nun parametrelerini güncelleyin ve ayarlayın:
20. $M_u = M_u + \bar{x}_t . \bar{x}_t{}^T$
21. $b_u = b_u + r_{t,a^*} . \bar{x}_t$
22. $w_u = M_u^{-1} . b_u$
23. u'nun ait olduğu kullanıcı kümesini ayarla
24. $\bar{C} = \underset{i=1,2,\ldots,k}{\text{argmin}} \|w_u - \bar{w}_i\|$
25. **if** $\bar{C} \neq C^*$ **then**
26. u 'yu C^* kümesinden $\bar{C}$ kümesine taşıyın
27. $\bar{w}_{\bar{c}}$' yi yeniden hesapla
28. **end**
29. $\bar{w}_{c^*}$' yi yeniden hesapla
30. **end**

Şekil 5.9 DC³MAB-π algoritması

BÖLÜM ALTI
UYGULAMA

6.1 Veri Seti

IJCAI-15 veri seti, TIANCHI popular e-ticaret sitesinden anonim kullanıcıların 6 aya ait alışveriş kayıtlarını ve tekrarlı alıcı olup olmadıklarını belirten etiket bilgilerini içermektedir. (https://tianchi.aliyun.com/dataset/dataDetail?dataId=42)

DC^{3}MAB-π'nin performansı e-ticaret öneri sisteminde yaygın olarak kullanılan IJCAI-15 veri kümesi üzerinde değerlendirilmiştir. Bu veri setindeki her örnek, öge kimliği, kullanıcı kimliği, öge öznitelikleri, kullanıcı öznitelikleri, etkileşim süresi ve etkileşim türünü içerir. IJCAI-15 veri setinin özellikleri aşağıdaki şekilde özetlenmiş, ayrıca veri setine ait genel bilgiler de Tablo 6.1'de verilmiştir.

- Kullanıcı sayısı = 424170
- Ürün sayısı = 1090390
- Etkileşim sayısı = 54925330
- Kullanıcılar için demografik bilgiler (yaş, cinsiyet)
- Ögeler için yan bilgiler (ürün kategorisi, marka, satıcı)
- Etkileşim türü (tıklama 0, sepete ekleme 1, satın alma 2, favorilere ekleme 3)

Tablo 6.1 IJCAI-15 Veri Seti hakkında genel bilgiler

Veri alanları	Tanımlama
user_id	Müşteri için benzersiz bir kimlik
age_range	Kullanıcının yaş aralığı: (1: <18) ; (2: [18,24]) ; (3: [25,29]) ; (4: [30,34]); (5: [35,39]) ; (6: [40,49]) ; (7: >= 50) ; (8: NULL)
gender	(0: Kadın) ; (1: Erkek) ; (2: NULL)
item_id	Öge için benzersiz bir kimlik
cat_id	Ögenin ait olduğu kategorinin benzersiz kimliği
brand_id	Ögenin markası için benzersiz bir kimlik
Seller_id	Satıcı için benzersiz bir kimlik
time_tamp	Eylemin gerçekleştiği tarih
action_type	0: Tıklama 1: Sepete ekleme 2: Satın alma 3: Favorilere ekleme

IJCAI-15 veri seti oldukça büyük olduğundan yöntemin sonuçlarını değerlendirmek için yeterli büyüklükte üç adet alt küme oluşturulmuştur. Her bir alt kümede 1000 adet kullanıcıya ait veri yer almaktadır. Bu kümelerde yer alan ürün sayısı ve boyut bilgisi Tablo 6.2'de verilmiştir. Veri setlerinde yer alan kullanıcıların %3'ten azının ortak olduğu gözlenmiştir.

Tablo 6.2 Üç veri setine ait genel bilgiler

Veri seti	Kullanıcı sayısı	Ürün sayısı	Etkileşim sayısı
Veri Seti-1	1000	8453	15605
Veri Seti-2	1000	8348	15861
Veri Seti-3	1000	8652	15728

Bu çalışmada Python programlama dili kullanılmıştır. Önerilen DC^{3}MAB-π yönteminin temelini oluşturan ve Yan vd. (2022) tarafından önerilen DC^{3}MAB yönteminin kaynak kodu (https://github.com/HaixHan/DC3MAB) adresinden alınmış ve yeniden düzenlenmiştir. Tablo 6.2'deki veri setleri üzerinde her iki yöntem için de 20'şer deneme yapılmış ve her iki öneri sistemi için elde edilen sonuçlar Ekler bölümünde verilmiştir.

Bu tez çalışmasında önerilen DC^{3}MAB-π yönteminin etkinliğini incelemek için ise IBM SPSS Statistics 27 kullanılmıştır.

6.2 Değerlendirme Ölçütleri

Algoritmaları değerlendirmek için Kesinlik (Precision), Duyarlılık (Recall), F_1 puanı ve Birikimli ödül (Cumulative Reward) kullanılacaktır.

Kesinlilik: Öneri doğruluğunu ölçmek için tipik olarak kullanılan bir ölçüttür ve bu ölçüt, aşağıdaki Denklem 6.1'deki gibi şekilde hesaplanır.

$$\text{Kesinlilik}_u = \frac{1}{\left|\hat{S}_u^T\right|} \sum_{t=1}^{\left|\hat{S}_u^T\right|} \frac{\left|\bar{I}_{ut} \cap \bar{S}_u\right|}{\bar{I}_{ut}} \tag{6.1}$$

Burada $\bar{I}_{ut}$ önerilen öge listesidir, $\bar{S}_u$, u kullanıcısının davranış sıralamasında güçlü bir tercihe sahip öge listesini, $\hat{S}_u^T$ ise kullanıcı u'nun davranış dizisini ve T ise dizideki

bitiş zamanı temsil eder. Ayrıca $|U|$ toplam kullanıcı sayısını ifade eder. Ardından, test edilen tüm kullanıcıların bireysel Kesinlilik değerlerini toplanır ve toplamı kullanıcı sayısına bölünür, böylece genel bir ortalama Kesinlilik değeri elde edilir. Tüm test kullanıcılarının ortalama Kesinlilik değeri Denklem 6.2 ile hesaplanır.

$$\text{Kesinlilik} = \frac{1}{|U|}\sum_{u \in U} \text{Kesinlilik}_u \tag{6.2}$$

Duyarlılık: Öneri performansını değerlendirmek için kullanılan önemli bir ölçüdür ve aşağıdaki Denklem 6.3 ile elde edilir.

$$\text{Duyarlılık}_u = \frac{1}{|\hat{S}_u^T|}\sum_{t=1}^{|\hat{S}_u^T|} \frac{|\bar{I}_{ut} \cap \bar{S}_u|}{\bar{S}_u} \tag{6.3}$$

Ardından, tüm test kullanıcılarının ortalama Duyarlılık Denklem 6.4 ile elde edilir.

$$\text{Duyarlılık} = \frac{1}{|U|}\sum_{u \in U} \text{Duyarlılık}_u \tag{6.4}$$

***F_1* puanı:** Kesinlilik ve Duyarlılık arasındaki dengeyi sağlamak önemlidir. F_1, bu dengeyi harmonik bir ortalama ile ölçerek, bir sistem veya algoritmanın başarısını kapsamlı bir şekilde değerlendirir (Goutte ve Gaussier, 2005). F_1 puanı, yanlış pozitif ve negatifleri en aza indirme hedefini vurgular. Bu da modelin performansının daha hassas bir şekilde değerlendirmesini sağlar. Bu ölçüt Denklem 6.5 ile elde edilir.

$$F_1\ puanı = 2.\frac{\text{Kesinlilik}\,.\,\text{Duyarlılık}}{\text{Kesinlilik} + \ \text{Duyarlılık}} \tag{6.5}$$

F_1 puanı Kesinlilik ve geri Duyarlılık arasındaki değiş tokuşu temsil eder ve farklı algoritmaların performansını değerlendirmek için kullanılır.

Ödül: Algoritmanın *T* turlarında elde ettiği ödüllerin toplamı olarak tanımlanan, haydut algoritmasının performansını değerlendirmek için kullanılan tipik bir ölçüttür. Ödül ne kadar büyükse, algoritma en iyiye (optimal) o kadar yakındır.

6.3 Özellik Yapısı

Bağlamsal özellik vektörlerinin oluşturulması bağlamsal ÇKH'de önemli bir adımdır. Matris faktörizasyonu (MF), işbirlikçi filtreleme için basit ama etkili bir modeldir (daha önce bölüm 3.3.2'de bahsedilmiş). Kullanıcı ve öge özelliklerini çıkarmak için bu yöntem benimsenmiştir.

Bununla birlikte, yukarıda bahsedildiği gibi, çoğu pratik öneri sistemlerinde, kullanıcılardan çok az açık geri bildirim veya derecelendirme gelir ve ögeler üzerinde yalnızca tıklama ve sepete ekleme gibi örtük kullanıcı davranışları vardır. Bir derecelendirme matrisi oluşturmak için, RW_{ui} ödül ağırlıklarının atanmasıyla tutarlı olarak, farklı etkileşim türlerine farklı derecelendirme puanları atanmıştır. Deneyimizde tıklama, favori, sepete ekleme ve satın alma puanları sırasıyla 1, 2, 3 ve 4 olarak belirlenmiştir. Etkileşim yoksa derecelendirme puanı 0 olarak ayarlanır. Kullanıcının aynı ögeyle birden çok farklı türde etkileşimi varsa, maksimum puan olarak nihai puan kullanılmıştır.

Kullanıcıların ve ögelerin öznitelik vektörleri elde edildikten sonra her bir kolun öznitelik vektörü hesaplanır. Bir a süper kolu için, bağlamsal özellik vektörünü temsil etmek için a'daki tüm ögelerin özellik vektörlerinin ortalamasını Denklem 6.6'daki şekilde elde edilir.

$$X_a = \frac{1}{|a|}\sum_{\acute{a}\in a} X_{\acute{a}} \qquad ,\forall a \in A \tag{6.6}$$

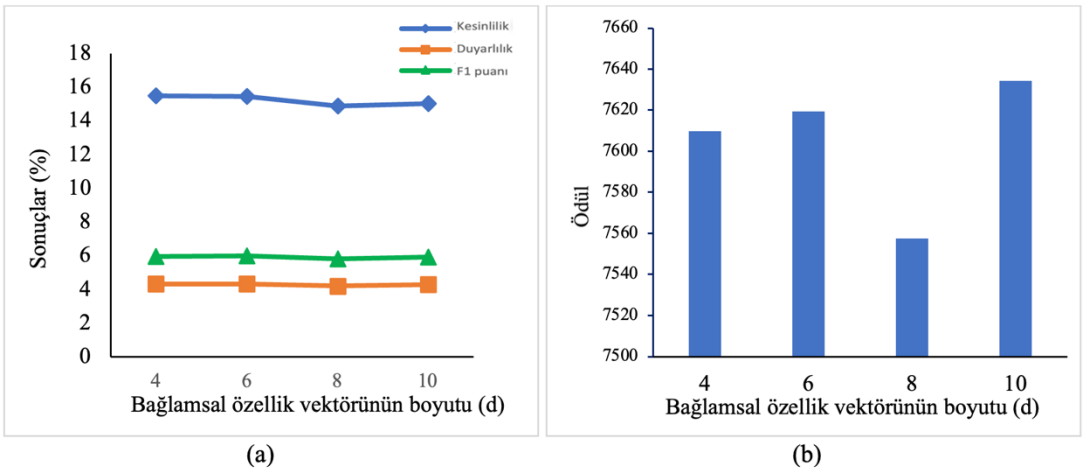
Burada $\acute{a}$, a'daki tek bir ögedir ve $X_{\acute{a}}$, $\acute{a}$ ögesinin özellik vektörünü temsil eder.

6.4 Önemli Parametrelerin Etkisi

Bu bölümde, temel parametrelerin öneri algoritmamızın performansı üzerindeki etkisi incelenmiştir. Bu parametreler arasında bağlamsal özellik vektörünün boyutu d ve kullanıcı kümelerinin sayısı k yer almaktadır.

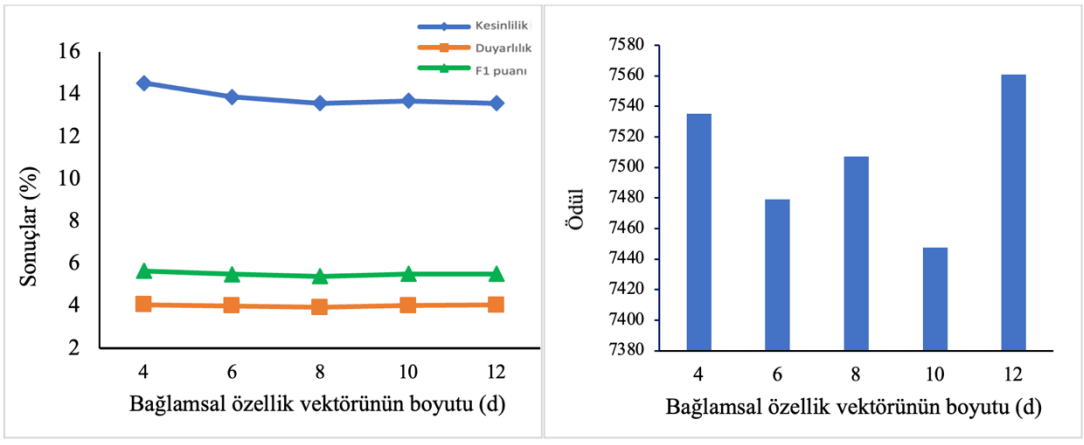
İlk olarak, Şekil 6.1'de bağlamsal özellik vektörünün boyutu olan d değeri ele alınmıştır. Veri Seti-1'de $d = 4$'ten $d = 1$'e artırıldığında, Kesinlilik performansının

düştüğü, Duyarlılık ve F_1 puanı performansının da sabit kaldığı görülmektedir. Ödül açısından en iyi performans ise $d = 10$ değeri için elde edilmiştir. Veri Seti-2 ve Veri Seti-3 için Şekil 6.2 ve Şekil 6.3'te görüldüğü üzere benzer bir durum gözlenmiştir. Öneri sistemlerinde daha az boyutla çalışmak hesaplama yükü açısından olumlu etki yaratacağından $d = 4$ boyutu seçilmiştir.



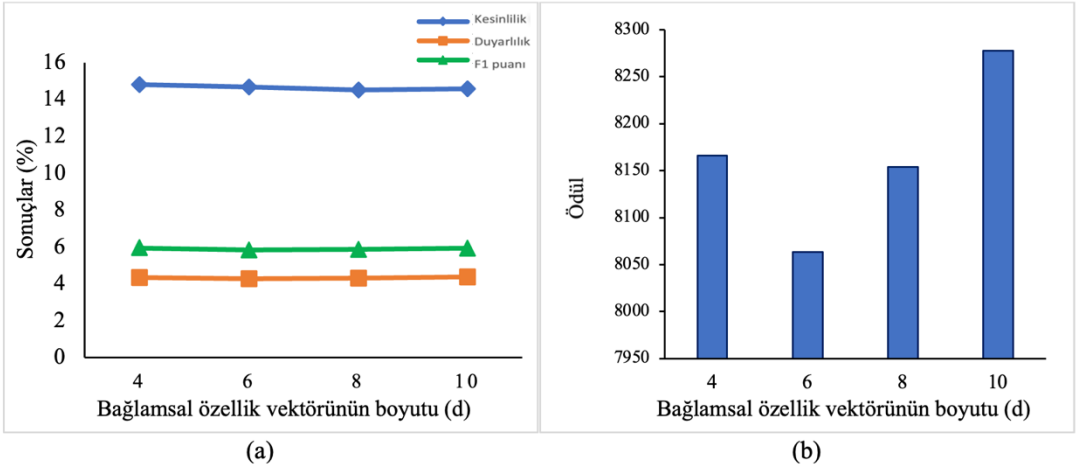
(a) (b)

Şekil 6.1 Veri Seti-1 için bağlamsal özellik boyutunun (a) Kesinlilik, Duyarlılık, F1 puanı, (b) Ödül üzerindeki etkisi



(a) (b)

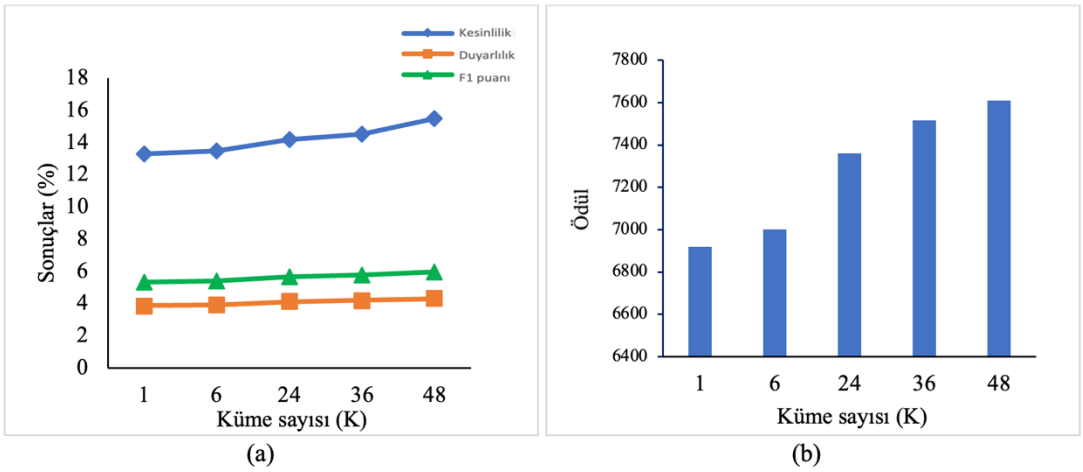
Şekil 6.2 Veri Seti-2 için bağlamsal özellik boyutunun (a) Kesinlilik, Duyarlılık, F1 puanı, (b) Ödül üzerindeki etkisi



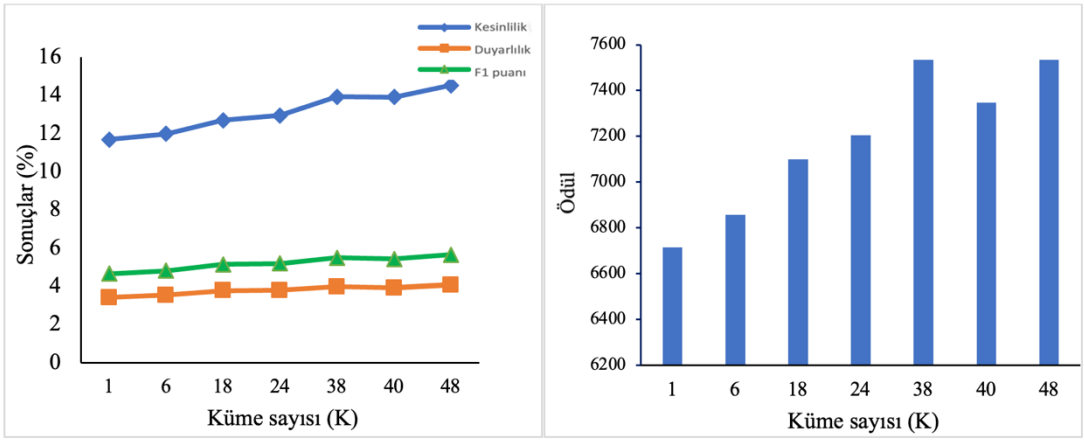
Şekil 6.3 Veri Seti-3 için bağlamsal özellik boyutunun (a) Kesinlilik, Duyarlılık, F_1 puanı, (b) Ödül üzerindeki etkisi

Kullanıcı kümelerinin sayısı olan k için farklı değerler incelendiğinde sonuçlar Şekil 6.4'teki gibidir. Veri Seti-1 için k'nın değeri arttıkça daha yüksek Kesinlilik, Duyarlılık, F_1 puanı ve Ödül değerlerinin elde edildiği gözlemlenmiştir.

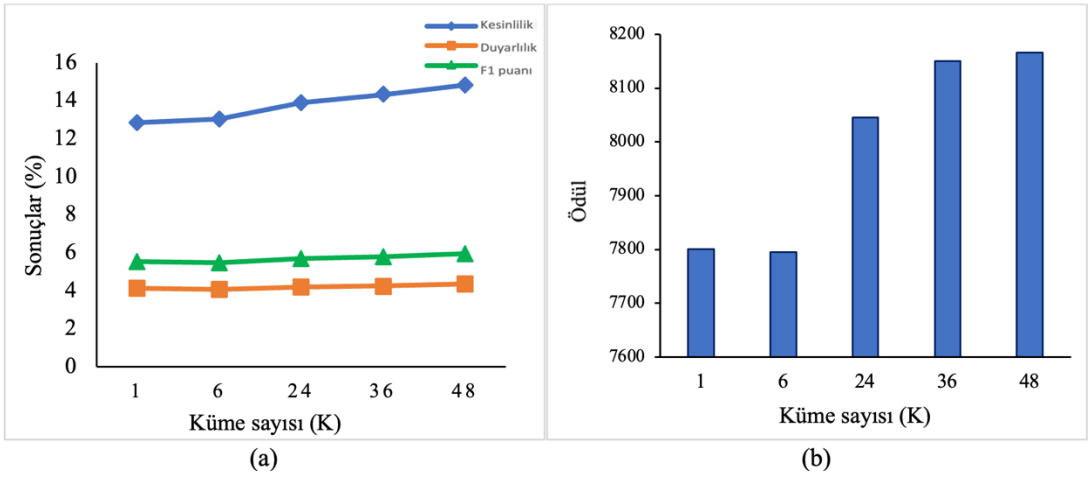
Veri Seti-2 ve Veri Seti-3 için de Şekil 6.5 ve Şekil 6.6'ten görüldüğü üzere benzer bir durum gözlenmektedir. Bununla birlikte, hesaplama karmaşıklığı ile öneri etkinliği arasında bir denge vardır. Optimum k değerini seçerken doğru dengeyi bulmak çok önemlidir. Yaptığımız denemelerde, Yan vd. (2022) çalışmasında elde edilen sonuçlarla tutarlı olarak $k = 48$ için performansın daha iyi olduğu görülmüştür.



Şekil 6.4 Veri Seti-1 için küme sayısının (a) Kesinlilik, Duyarlılık, F_1 puanı, (b) Ödül üzerindeki etkisi



Şekil 6.5 Veri Seti-2 için küme sayısının (a) Kesinlilik, Duyarlılık, F_1 puanı, (b) Ödül üzerindeki etkisi



Şekil 6.6 Veri Seti-3 için küme sayısının (a) Kesinlilik, Duyarlılık, F_1 puanı, (b) Ödül üzerindeki etkisi

6.5 DC^{3}MAB ve DC^{3}MAB-π Sonuç Karşılaştırmaları

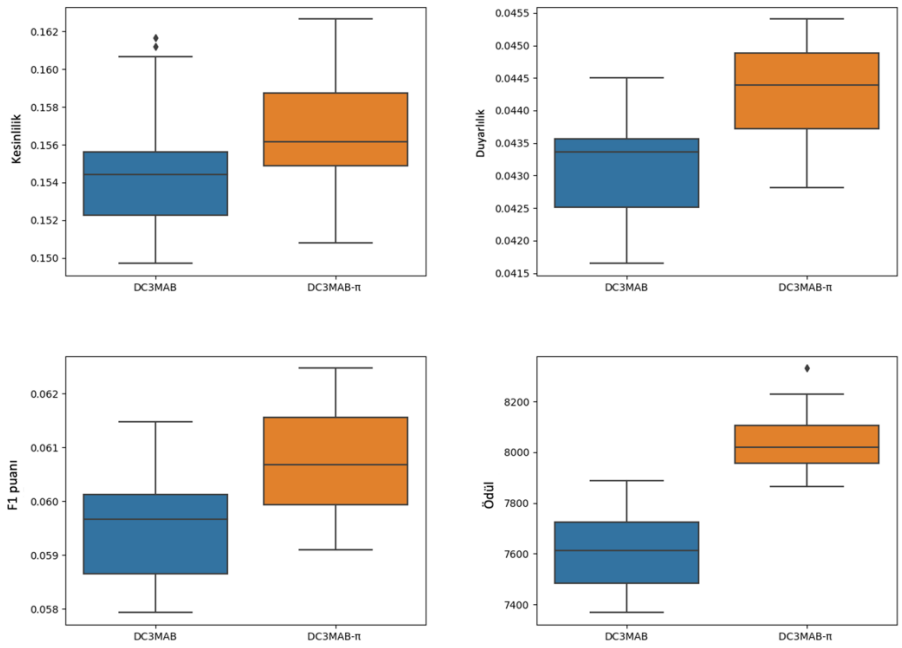
Yan vd. (2022) çalışmasında DC^{3}MAB algoritmasının **C^{2}UCB** (Qin, Chen, ve Zhu, 2014), **TPBandit** (Yan, Han, Wang, ve Zhang, 2021), **N-LinUCB** (Li, Chu vd., 2010), **TS** (Chapelle ve Li, 2011), **UCB** (Auer vd., 2002) ve **DynUCB** (Nguyen ve Lauw, 2014) dahil olmak üzere bir dizi iyi bilinen haydut algoritmasıyla karşılaştırmıştır. Sonuç olarak DC^{3}MAB algoritmasının Kesinlilik, Duyarlılık, F_1 puanı ve Ödül değerleri açısından diğer algoritmalardan daha iyi performans elde ettiği gösterilmiştir. Bu nedenle, bu çalışmada DC^{3}MAB-π yönteminin etkinliğini göstermek için yalnızca

DC^{3}MAB yöntemi sonuçları ile karşılaştırma yapılmıştır. Bunun için her bir yöntem 3 veri seti için de 20'şer kez çalıştırılmış ve bu yöntemler arasındaki performans farkları gösteren özet istatistikler Tablo 6.3'te gösterilmiştir.

Tablo 6.3 Veri Seti-1 için özet istatistikler

		Ortalama	Std. Sapma	Ortalamanın Std. Hatası
Kesinlilik	DC^{3}MAB	0,15482	0,00357	0,00080
	DC^{3}MAB-π	0,15684	0,00316	0,00071
Duyarlılık	DC^{3}MAB	0,04313	0,00081	0,00018
	DC^{3}MAB-π	0,04423	0,00075	0,00017
F_1 puanı	DC^{3}MAB	0,05960	0,00101	0,00023
	DC^{3}MAB-π	0,06074	0,00104	0,00023
Ödül	DC^{3}MAB	7609,85	153,95	34,42
	DC^{3}MAB-π	8041,85	126,0	28,17

Şekil 6.7'de gösterildiği gibi dört temel metriğin dağılımını görselleştirmek için kutu grafikleri kullanılarak DC^{3}MAB-π ve DC^{3}MAB arasında karşılaştırmalı bir analiz gerçekleştirildi. Kutu grafikleri, her iki yöntemdeki her bir ölçüt için merkezi eğilimin ve yayılımın net bir temsilini sağlar.



Şekil 6.7 DC^{3}MAB ve DC^{3}MAB-π yöntemleri için kutu grafikleri (Veri Seti-1)

Kutu grafikleri incelendiğinde DC^{3}MAB-π yöntemine ait sonuçların, daha yüksek minimum, ortalama ve maksimum değerlerle DC^{3}MAB'den daha iyi performans gösterdiği görülmektedir. Ayrıca Ödül açısından incelendiğinde, DC^{3}MAB-π sonuçlarında çeyrekler arası açıklığın DC^{3}MAB sonuçlarına göre daha küçük bir değere sahip olduğu görülmektedir.

İstatistiksel farklılıkların daha ayrıntılı bir şekilde anlaşılması için Shapiro-Wilk normallik testi uygulanmış ve veriler normal dağılıma uyduğu için t-testi ile karşılaştırmalar yapılmıştır. Bu testler, gözlemlenen farklılıkların istatistiksel olarak anlamlı olup olmadığının tespit edilmesine yardımcı olacak ve DC^{3}MAB-π ve DC^{3}MAB'in belirtilen ölçümlerle ilgili karşılaştırmalı etkinliği hakkında sonuçlara varmak için sağlam bir temel oluşturacaktır.

Tablo 6.4 Veri Seti-1 için Shapiro-Wilk normallik testleri

		Shapiro-Wilk	
	Gruplar	**İstatistik**	**Önem düzeyi**
Kesinlilik	DC^{3}MAB	0,907	0,055
	DC^{3}MAB-π	0,969	0,744
Duyarlılık	DC^{3}MAB	0,952	0,407
	DC^{3}MAB-π	0,960	0,534
F_1 puanı	DC^{3}MAB	0,954	0,425
	DC^{3}MAB-π	0,954	0,425
Ödül	DC^{3}MAB	0,948	0,343
	DC^{3}MAB-π	0,940	0,242

Örneklem genişliği 20 birim olduğu için Shapiro-Wilk normallik testi sonuçları Tablo 6.4'teki gibi elde edilmiştir. Kesinlilik, Duyarlılık, F_1 puanı ve Ödül değerlerinin her iki yöntem için de normal dağılım gösterdiği tablodan görülmektedir. Bu aşamadan sonra normal dağılım gösteren değişkenler için iki grubun ortalaması eşleştirilmiş örneklem testi ile karşılaştırılmış ve analiz sonuçları Tablo 6.5'teki gibi elde edilmiştir.

Tablo 6.5 Veri Seti-1 için eşleştirilmiş örneklem testi sonuçları

		Eşleştirilmiş Farklar					t	s.d.	*p*
		Ort	Std. Sapma	Ortalamanın Std Hatası	Farklar için %95 düzeyinde Güven Aralığı Alt	Üst			
Kesinlilik	DC^3MAB - DC^3MAB-π	-0,00203	0,00519	0,00116	-0,00446	0,00041	-1,744	19	0,097
Duyarlılık	DC^3MAB - DC^3MAB-π	-0,00109	0,00119	0,00027	-0,00166	-0,00054	-4,125	19	**0,001**
F_1 puanı	DC^3MAB - DC^3MAB-π	-0,00114	0,00162	0,00036	-0,00189	-0,00038	-3,144	19	**0,005**
Ödül	DC^3MAB - DC^3MAB-π	-432,0	180,41	40,34	-516,43	-347,56	-10,71	19	**0,001**

Duyarlılık, F_1 puanı ve Ödül değerleri için *p* değeri 0,05'ten küçük olduğundan DC^3MAB ve DC^3MAB-π yöntemleri arasında anlamlı farklılık vardır. Bu anlamlı farklılığın etkisinin ne kadar olduğunu belirlemek için ise etki büyüklüğü değerleri incelenmiştir. Bu analize dair sonuçlar Tablo 6.6'daki gibidir.

Tablo 6.6 Veri Seti-1 için etki büyüklüğü değerleri

			Standartlaştırıcı[a]	Nokta tahmin	%95 Güven Aralığı Alt	Üst
Duyarlılık	DC^3MAB - DC^3MAB-π	Cohen's d	0,001191	-0,922	-1,440	-0,388
		Hedges' düzeltmesi	0,001215	-0,904	-1,411	-0,380
F_1 puanı	DC^3MAB - DC^3MAB-π	Cohen's d	0,001620	-0,703	-1,187	-0,204
		Hedges' düzeltmesi	0,001653	-0,689	-1,163	-0,200
Ödül	DC^3MAB - DC^3MAB-π	Cohen's d	180,411	-2,395	-3,260	-1,513
		Hedges' düzeltmesi	184,072	-2,347	-3,195	-1,483

a. Etki büyüklüklerinin tahmin edilmesinde kullanılan payda. (The denominator used in estimating the effect sizes.)

Cohen's d, ortalama farkın örneklem standart sapmasını kullanır. (Cohen's d uses the sample standard deviation of the mean difference.)

Hedges'in düzeltmesi, ortalama farkın örneklem standart sapması artı bir düzeltme faktörü kullanır. (Hedges' correction uses the sample standard deviation of the mean difference, plus a correction factor.)

Tablo 6.6'daki sonuçlar Cohen'in *d* istatistiğinin çalışmalardaki örneklem sayısının 20'nin üzerinde olması durumunda daha doğru sonuç verdiği belirtilmektedir (Lipsey ve Wilson, 2001). Yöntemler her bir veri setinde 20'şer kez tekrarlandığı için Cohen'in *d* istatistik sonuçları da Hedges'in düzeltmesi istatistiği de dikkate alınabilir (Hedges, 1981). Etki büyüklüğü aralıkları ve anlamları Tablo 6.7'de verilmiştir (Cohen, Manion ve Morrison, 2007).

Tablo 6.7 Etki büyüklüğü değerleri ve yorumlanması

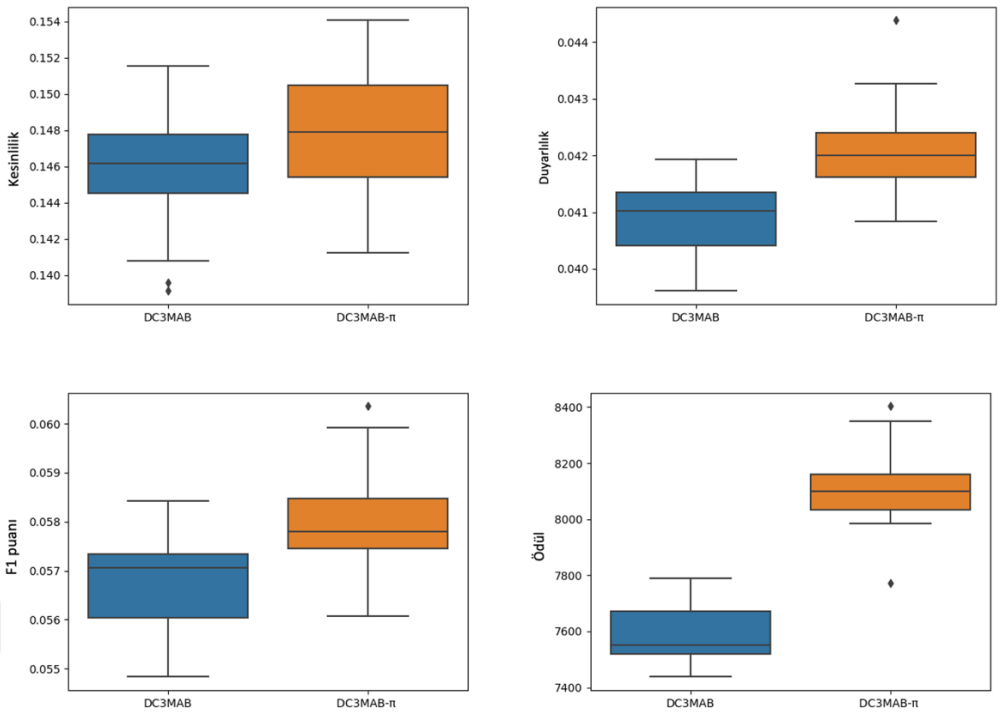
0-0,2	Zayıf derecede etki
0,21-0,50	Mütevazı derecede etki
0,51-1,0	Orta derecede etki
> 1,0	Güçlü derecede etki

Buna göre Veri Seti-1 için Cohen'in *d* istatistiği ve Hedges düzeltmesi sonuçları incelendiğinde Duyarlılık ve F_1 puanı için orta hatta güçlü dereceye yakın etki gözlenmişken Ödül için güçlü derecede etki olduğu söylenebilir.

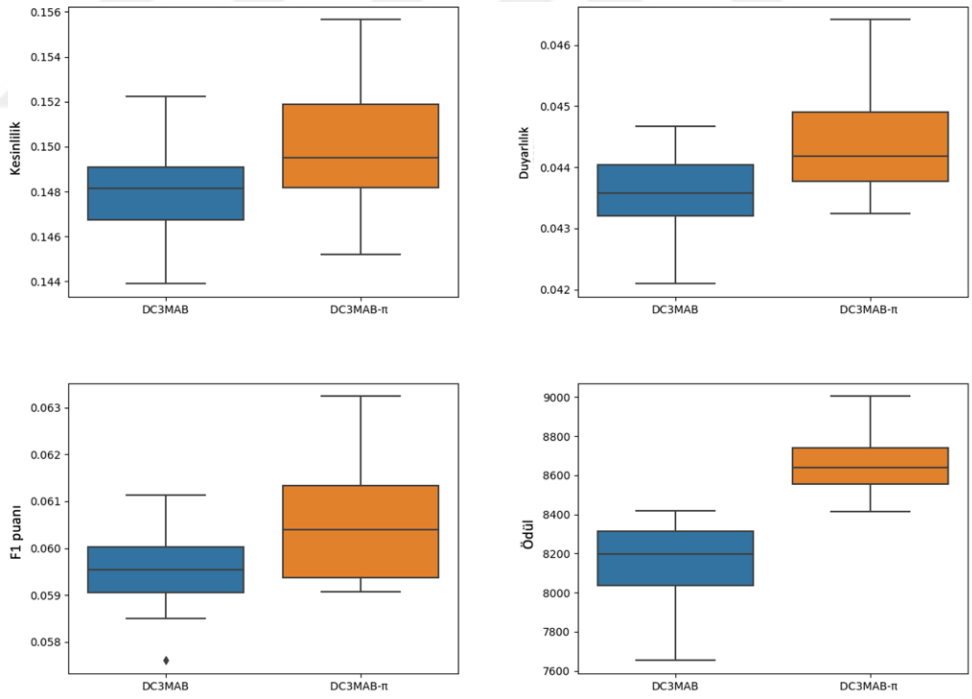
Veri Seti-2 ve Veri Seti-3 için özet istatistikler Tablo 6.8'de ve Kutu grafikleri sırasıyla Şekil 6.8 ve Şekil 6.9'da verilmektedir. Her iki yöntemde de DC^{3}MAB-π, daha yüksek minimum, ortalama ve maksimum değerlerle DC^{3}MAB'den daha iyi performans gösterdiği gözlemlenir. Ödül dağılımı açısından, DC^{3}MAB-π yönteminden elde edilen sonuçların, Veri Seti-1'deki gibi yine daha küçük çeyrekler arası açıklık değerine sahip olduğu görülmektedir.

Tablo 6.8 Veri Seti-2 ve Veri Seti-3 için özet istatistikler

			Ortalama	Std. Sapma	Ortalamanın Std. Hatası
Veri Seti-2	**Kesinlilik**	DC^{3}MAB	0,14594	0,00345	0,00077
		DC^{3}MAB-π	0,14783	0,00348	0,00077
	Duyarlılık	DC^{3}MAB	0,04090	0,00067	0,00014
		DC^{3}MAB-π	0,04215	0,00080	0,00017
	F_1 puanı	DC^{3}MAB	0,05680	0,00099	0,00022
		DC3MAB-π	0,05814	0,00138	0,00030
	Ödül	DC^{3}MAB	7578,2	98,44	22,01
		DC^{3}MAB-π	8114,5	138,77	31,03
Veri Seti-3	**Kesinlilik**	DC^{3}MAB	0,14814	0,00186	0,000416
		DC^{3}MAB-π	0,15006	0,00269	0,000602
	Duyarlılık	DC^{3}MAB	0,04357	0,00070	0,000157
		DC^{3}MAB-π	0,04436	0,00087	0,000195
	F_1 puanı	DC^{3}MAB	0,05955	0,00083	0,000185
		DC^{3}MAB-π	0,06044	0,00114	0,000255
	Ödül	DC3MAB	8165,8	218,75	48,91
		DC3MAB-π	8656,5	174,15	38,94



Şekil 6.8 DC^{3}MAB ve DC^{3}MAB-π yöntemleri için kutu grafikleri (Veri Seti-2)



Şekil 6.9 DC^{3}MAB ve DC^{3}MAB-π yöntemleri için kutu grafikleri (Veri Seti-3)

Veri Seti-2 ve Veri Seti-3 için Shapiro-Wilk normallik testi sonuçları Tablo 6.9'daki gibi elde edilmiştir. Veri Seti-2 için tüm ölçütlerin normallik varsayımını sağladığı görülmektedir. Veri Seti-3 için DC^{3}MAB yöntemiyle elde edilen Ödül değerleri hariç Kesinlilik, Duyarlılık ve F_1 puanı değerlerinin normal dağılım gösterdiği tablodan görülmektedir.

Tablo 6.9 Veri Seti-2 ve Veri Seti-3 için Shapiro-Wilk normallik testleri

			Shapiro-Wilk	
		Gruplar	**İstatistik**	**Önem düzeyi**
Veri Seti-2	**Kesinlilik**	DC^{3}MAB	0,936	0,205
		DC^{3}MAB-π	0,982	0,958
	Duyarlılık	DC^{3}MAB	0,949	0,356
		DC^{3}MAB-π	0,922	0,109
	F_1 puanı	DC^{3}MAB	0,932	0,166
		DC^{3}MAB-π	0,945	0,292
	Ödül	DC^{3}MAB	0,908	0,058
		DC^{3}MAB-π	0,948	0,339
Veri Seti-3	**Kesinlilik**	DC^{3}MAB	0,983	0,968
		DC^{3}MAB-π	0,973	0,823
	Duyarlılık	DC^{3}MAB	0,972	0,804
		DC^{3}MAB-π	0,939	0,233
	F_1 puanı	DC^{3}MAB	0,987	0,990
		DC^{3}MAB-π	0,924	0,120
	Ödül	DC^{3}MAB	0,877	**0,015**
		DC^{3}MAB-π	0,941	0,254

Bu aşamadan sonra normal dağılım gösteren değişkenler için iki grubun ortalaması ilişkili örneklem testi ile karşılaştırılmış ve analiz sonuçları Tablo 6.10'daki gibi elde edilmiştir.

Tablo 6.10 Veri Seti-2 ve Veri Seti-3 için eşleştirilmiş örneklem testi

			Eşleştirilmiş Farklar							
						Farklar için %95 düzeyinde Güven Aralığı				
			Ort	Std. Sapma	Ortalamanın Std Sapması	Alt	Üst	t	s.d.	*p*
Veri Seti-2	**Kesinlilik**	DC³MAB - DC³MAB-π	-0,00189	0,00509	0,00113	-0,00428	0,00048	-1,667	19	**0,112**
	Duyarlılık	DC³MAB - DC³MAB-π	-0,00125	0,00089	0,00019	-0,00169	-0,00084	-6,320	19	<0,001
	F_1 puanı	DC³MAB - DC³MAB-π	-0,00135	0,00181	0,00040	-0,00219	0,00050	-3,329	19	0,004
	Ödül	DC³MAB - DC³MAB-π	-536,3	181,1	40,51	-621,1	-451,5	-13,24	19	<0,001
Veri Seti-3	**Kesinlilik**	DC³MAB - DC³MAB-π	-0,00191	0,00346	0,00077	-0,00353	-0,00029	-2,476	19	0,023
	Duyarlılık	DC³MAB - DC³MAB-π	-0,00078	0,00129	0,00029	-0,00139	-0,00018	-2,738	19	0,013
	F_1 puanı	DC³MAB - DC³MAB-π	-0,00088	0,00158	0,00035	-0,00162	-0,00014	-2,504	19	0,022

Veri Seti-2 Kesinlilik hariç *p* değeri 0,05'ten küçük olduğundan DC³MAB ve DC³MAB-π yöntemleri arasında anlamlı farklılık olduğu görülmektedir. Veri Seti-3 için de *p* değeri 0,05'ten küçük olduğundan DC³MAB ve DC³MAB-π yöntemleri arasında anlamlı farklılık olduğu görülmektedir. Bu anlamlı farklılığın etkisinin ne kadar olduğunu belirlemek için Etki Büyüklüğü değerleri incelenmiştir. Bu analize dair sonuçlar Tablo 6.11'deki gibidir.

Tablo 6.11 Veri Seti-2 ve Veri Seti-3 için etki büyüklüğü değerleri

						%95 Güven Aralığı	
				Standartlaştırıcı[a]	Nokta Tahmini	Alt	Üst
Veri Seti -2	**Duyarlılık**	DC³MAB - DC³MAB-π	Cohen's d	0,00088	-1,413	-2,029	-0,779
			Hedges' düzeltmesi	0,00091	-1,385	-1,989	-0,763
	F_1 puanı	DC³MAB - DC³MAB-π	Cohen's d	0,00181	-0,744	-1,234	-0,239
			Hedges' düzeltmesi	0,00184	-0,730	-1,210	-0,235
	Ödül	DC³MAB - DC³MAB-π	Cohen's d	181,18	-2,960	-3,984	-1,92
			Hedges' düzeltmesi	184,85	-2,901	-3,905	-1,88
Veri Seti -3	**Kesinlilik**	DC³MAB - DC³MAB-π	Cohen's d	0,00346	-0,554	-1,019	-0,075
			Hedges' düzeltmesi	0,00353	-0,543	-0,999	-0,074
	Duyarlılık	DC³MAB - DC³MAB-π	Cohen's d	0,00129	-0,612	-1,084	-0,126
			Hedges' düzeltmesi	0,00131	-0,600	-1,063	-0,124
	F_1 puanı	DC³MAB - DC³MAB-π	Cohen's d	0,00158	-0,560	-1,026	-0,081
			Hedges' düzeltmesi	0,00161	-0,549	-1,006	-0,079

Buna göre Veri Seti-2 için Cohen'in d istatistiği ve Hedges düzeltmesi sonuçları incelendiğinde Duyarlılık ve Ödül ölçütleri için çok güçlü derecede etkili ve F_1 puanı için orta derecede etkili olduğu söylenebilir. Veri Seti-3 için bu sonuçlar incelendiğinde ise Kesinlilik, Duyarlılık ve F_1 puanı için orta derece etki olduğu görülmektedir.

Son olarak Veri Seti-3'te DC^{3}MAB'den elde edilen Ödül değerleri normal dağılım varsayımını sağlamadığı için bu değerlerin DC^{3}MAB- π yöntemi sonuçlarıyla karşılaştırılması parametrik olmayan testlerden Wilcoxon testi ile gerçekleştirilmiş ve sonuç Tablo 6.12'deki gibi elde edilmiştir. Buna göre iki grup arasındaki farkın istatistiksel olarak anlamlı olduğu görülmektedir.

Tablo 6.12 Wilcoxon testi sonuçları

Test İstatistikleri[a]

	DC^{3}MAB - DC^{3}MAB-π
z	-3,920[b]
Asimptotik Önem (2 kuyruklu)	,000

a. Wilcoxon İşaretli Rank Testi

b. Negatif sıralamalara göre.

BÖLÜM YEDİ
SONUÇLAR VE ÖNERİLER

Öneri algoritmalarının keşif ve sömürüyü dengeleyerek yönetmesi, kullanıcı deneyimini geliştirme ve içerik çeşitliliğini artırmada önemli bir role sahiptir. Bu denge, haydut problemi olarak modellenerek öneri sistemlerinin etkili bir şekilde kurulmasını sağlar. Ancak, genellikle geleneksel haydut düzeni tarafından kısıtlanır. Bu çalışmada, öneri sistemlerinde karşılaşılan temel sorunlara getirdiği bazı çözümler sebebiyle DC^{3}MAB algoritması üzerinde detaylı inceleme yapılmıştır. Bu algoritmaya ögelerin popülerliklerine göre önerilmesi adımı eklenmiş ve DC^{3}MAB-π (DC^{3}MAB **P**opular **I**tems) şeklinde isimlendirilmiştir.

IJCAI-15 veri kümesinden elde edilmiş üç farklı veri alt kümesi üzerinde gerçekleştirdiğimiz deneysel sonuçlar, DC^{3}MAB-π algoritmasının Precision değerini koruyarak Recall, F_1 puanı ve Reward açısından DC^{3}MAB'den daha iyi performans gösterdiğini ortaya koymaktadır. Gerçekleştirdiğimiz bağımlı t-testleri, Wilcoxon ve etki büyüklüğü testleri gibi istatistiksel analizlerle, değerlendirme sürecimiz ortalama performans ölçütlerini karşılaştırmanın ötesine geçmektedir. Özellikle etki büyüklüğü açısından, bu yöntemin Reward üzerinde güçlü bir değişim sağladığı gösterilmiştir. Bu kapsamlı analiz, sadece DC^{3}MAB-π'nin performansını doğrulamakla kalmaz aynı zamanda bu öneri sisteminin etkinliğini seyrek veri durumunda iyileştirmenin potansiyelini vurgular.

İleriki çalışmalar olarak aşağıdaki durumların öneri sistemlerinin sonuçlarına etkisinin incelenmesi olabilir.

- DC^{3}MAB-π yönteminde farklı dinamik kümeleme yöntemlerinin kullanımının büyük veri setlerinden elde edilecek sonuçlarda yaratacağı değişikliklerin incelenmesi,
- Kullanıcıların birbirleriyle olan ilişkilerini anlamak ve kullanıcılar arası etkileşimlerin modellenmesi için sosyal ağ analizi veya çizge teorisi temelli yöntemler kullanılması,
- Derin öğrenme modelleri ile ödül mekanizmalarının detaylandırılması veya kullanıcı davranışlarının daha ayrıntılı bir şekilde modellenmesi.

KAYNAKLAR

Adomavicius, G., ve Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering,* 17(6), 734-749. https://ieeexplore.ieee.org/document/1423975

Agrawal, S., ve Goyal, N. (2013). Thompson sampling for contextual bandits with linear payoffs. *In International conference on machine learning* (127-135). PMLR.

Aggarwal, C. C. (2016). *An introduction to recommender systems. Recommender systems: The textbook.* https://doi.org/10.1007/978-3-319-29659-3

Al-Bashiri, H., Abdulgabber, M. A., Romli, A., ve Hujainah, F. (2017). Collaborative filtering recommender system: overview and challenges. *Advanced Science Letters*, 23(9), 9045-9049.

Alnahhas, A. (2021*). Hybrid deep multi-criteria recommender system model* [Yüksek Lisans Tezi]. İstanbul Teknik Üniversitesi.

Auer, P., Cesa-Bianchi, N., ve Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning,* 47, 235-256. https://doi.org/10.1023/A:1013689704352

Balabanović, M., ve Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3), 66-72.

Barakat, M. O. S. (2020). *PubMed article recommendation system based on collaborative filtering* [Yüksek Lisans Tezi]. Dokuz Eylül Üniversitesi. https://doi.org/10.3390/su142417053

Bellman, R. (1956). A problem in the sequential design of experiments. Sankhyā: *The Indian Journal of Statistics* (1933-1960), 16(3/4), 221-229. https://www.jstor.org/stable/25048278

Bonaccorso, G. (2017). Machine learning algorithms. *Packt Publishing.* https://dl.acm.org/doi/10.5555/3165154

Bozkurt, M., ve Acı, Ç. İ. (2021). Öneri Algoritmalarının Film Önerme Problemi Üzerinde Karşılaştırılması: MovieLens Örneği. *Bilgisayar Bilimleri ve Teknolojileri Dergisi,* 2(2), 36-42.

Bradt, R. N., Johnson, S. M., ve Karlin, S. (1956). On sequential designs for maximizing the sum of n observations. *The Annals of Mathematical Statistics,* 27(4), 1060-1074. https://www.jstor.org/stable/2237195

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction,* 12, 331-370. https://doi.org/10.1023/A:1021240730564

Chapelle, O., ve Li, L. (2011). An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, 24.

Chen, W., Wang, Y., ve Yuan, Y. (2013). Combinatorial multi-armed bandit: General framework and applications. *In International conference on machine learning* (151-159). PMLR.

Christakopoulou, K., ve Banerjee, A. (2018). Learning to interact with users: A collaborative-bandit approach. *In Proceedings of the 2018 SIAM International Conference on Data Mining (612-620). Society for Industrial and Applied Mathematics.* https://doi.org/10.1137/1.9781611975321.

Chu, W., ve Park, S. T. (2009). Personalized recommendation on dynamic content using predictive bilinear models. *In Proceedings of the 18th international conference on World wide web* (691-700)

Cohen, L., Manion, L., ve Morrison, K. (2007). Research methods in education.

Craswell, N., Zoeter, O., Taylor, M., ve Ramsey, B. (2008). An experimental comparison of click position-bias models. *In Proceedings of the 2008 international conference on web search and data mining* (87-94).

Feldman, D. (1962). Contributions to the "two-armed bandit" problem. *The Annals of Mathematical Statistics*, 33(3), 847-856. https://www.jstor.org/stable/2237862

Fleder, D. M., ve Hosanagar, K. (2007). Recommender systems and their impact on sales diversity. *In Proceedings of the 8th ACM conference on Electronic commerce* (192-199). https://doi.org/10.1145/1250910.1250939

Gao, C., He, X., Gan, D., Chen, X., Feng, F., Li, Y., ... ve Jin, D. (2019). Neural multi-task recommendation from multi-behavior data. *In 2019 IEEE 35th international conference on data engineering (ICDE)* (1554-1557).

Gentile, C., Li, S., ve Zappella, G. (2014). Online clustering of bandits. *In International conference on machine learning* (757-765). PMLR.

Glowacka, D. (2019). Bandit algorithms in information retrieval. *Foundations and Trends® in Information Retrieval,* 13(4), 299-424. https://doi.org/10.1561/1500000067

Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B., Herlocker, J., ve Riedl, J. (1999). Combining collaborative filtering with personal agents for better recommendations. Aaai/iaai, 439(10.5555), 315149-315352.

Goutte, C., ve Gaussier, E. (2005). A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. *In European conference on information retrieval* (345-359). Berlin, Heidelberg: Springer Berlin Heidelberg.

Gutierrez, D. D. (2015). *Machine learning and data science: an introduction to statistical learning methods with R*. Technics Publications.

Hedges, L. V. (1981). Distribution theory for Glass's estimator of effect size and related estimators. *journal of Educational Statistics*, 6(2), 107-128.

Healy, A. (2022). Contextual Bandit Approaches to Personalized Tutoring (Doctoral dissertation, WORCESTER POLYTECHNIC INSTITUTE).

Hill, W., Stead, L., Rosenstein, M., ve Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. *In Proceedings of the SIGCHI conference on Human factors in computing systems* (194-201). https://doi.org/10.1145/223904.223929

Işık, A. (2022). *Derin özkodlayıcı ağlar ile öneri sistemlerinde tahmin doğruluğunun artırılması* [Yüksek Lisans Tezi]. Kto Karatay Üniversitesi.

Joachims, T., Granka, L., Pan, B., Hembrooke, H., ve Gay, G. (2017). Accurately interpreting clickthrough data as implicit feedback. *In Acm Sigir Forum* (Vol. 51, No. 1, 4-11). New York, NY, USA: Acm.

Jannach, D., Zanker, M., Felfernig, A., ve Friedrich, G. (2010). *Recommender systems: an introduction*. Cambridge University Press.

Kaya, T. S. (2019). *Veri madenciliği algoritmaları ile kredi kartı kullanım alışkanlıklarının incelenmesi ve kişiye özgü kampanya teklifi* [Yüksek Lisans Tezi]. İstanbul Üniversitesi. https://acikbilim.yok.gov.tr/handle/20.500.12812/141404

Khalid, O., Khan, S. U., ve Zomaya, A. Y. (Ed.). (2019). Big Data Recommender Systems: Algorithms, architectures, big data, security and trust (Vol. 1). *Institution of Engineering and Technology.*

Koren, Y., Bell, R., ve Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer,* 42(8), 30-37. https://ieeexplore.ieee.org/document/5197422

Kveton, B., Szepesvari, C., Wen, Z., ve Ashkan, A. (2015). Cascading bandits: Learning to rank in the cascade model. *In International conference on machine learning* (767-776). PMLR.

Kveton, B., Wen, Z., Ashkan, A., ve Szepesvari, C. (2015). Combinatorial cascading bandits. *Advances in Neural Information Processing Systems*, 28.

Leskovec, J., Rajaraman, A., ve Ullman, J. D. (2014). Dimensionality reduction. *Mining of Massive Datasets*, 415-447. https://doi.org/10.1017/CBO9781139924801

Lu, T., Pál, D., ve Pál, M. (2010). Contextual multi-armed bandits. *In Proceedings of the Thirteenth international conference on Artificial Intelligence and Statistics* (485-492). Workshop and Conference Proceedings.

Li, L., Chu, W., Langford, J., ve Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. *In Proceedings of the 19th international conference on World wide web* (661-670). https://doi.org/10.48550/arXiv.1003.0146

Lipsey, M. W., ve Wilson, D. B. (2001). *Practical meta-analysis*. SAGE publications, Inc.

Li, S., Karatzoglou, A., ve Gentile, C. (2016). Collaborative filtering bandits. *In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval* (539-548). https://doi.org/10.1145/2911451.2911548

Mahesh, B. (2020). Machine learning algorithms- a review. *International Journal of Science and Research,* 9(1), 381-386. https://doi.org/ 10.21275/ART20203995

Nguyen, T. T., ve Lauw, H. W. (2014). Dynamic clustering of contextual multi-armed bandits. *In Proceedings of the 23rd ACM international conference on conference on information and knowledge management* (1959-1962).

Qiang, W., ve Zhongli, Z. (2011). Reinforcement learning model, algorithms and its application. *In 2011 International Conference on Mechatronic Science, Electric Engineering and Computer* (1143-1146). https://doi.org/10.1109/MEC.2011.6025669

Qin, L., Chen, S., ve Zhu, X. (2014). Contextual combinatorial bandit and its application on diversified online recommendation. *In Proceedings of the 2014 SIAM International Conference on Data Mining* (461-469). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611973440.53

Raza, S., ve Ding, C. (2019). Progress in context-aware recommender systems—An overview. *Computer Science Review*, 31, 84-97. https://doi.org/10.1016/j.cosrev.2019.01.001

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., ve Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. *In Proceedings of the 1994 ACM conference on Computer supported cooperative work* (175-186).

Rich, E. (1979). User modeling via stereotypes. *Cognitive science*, 3(4), 329-354. https://doi.org/10.1016/S0364-0213(79)80012-9

Ricci, F., Rokach, L., ve Shapira, B. (2010). Introduction to recommender systems handbook. *In Recommender systems handbook* (1-35). Boston, MA: springer US. https://doi.org/10.1007/978-0-387-85820-3_1

Ricci, F., Rokach, L., ve Shapira, B. (2015). Recommender systems: introduction and challenges. *Recommender systems handbook.*

Robbins, H. (1952). Some aspects of the sequential design of experiments.

Sanz-Cruzado, J., Castells, P., ve López, E. (2019). A simple multi-armed nearest-neighbor bandit for interactive recommendation. *In Proceedings of the 13th ACM conference on recommender systems* (358-362).

Schoinas, I., ve Tjortjis, C. (2019). MuSIF: a product recommendation system based on multi-source implicit feedback. *In Artificial Intelligence Applications and Innovations: 15th IFIP WG 12.5 International Conference, AIAI 2019, Hersonissos, Crete, Greece, May 24–26, 2019, Proceedings 15 (660-672)*. Springer International Publishing.

Shardanand, U., ve Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". *In Proceedings of the SIGCHI conference on Human factors in computing systems* (210-217).

Sutton, R. S., ve Barto, A. G. (2018). *Reinforcement learning: An introduction.* MIT press. https://doi.org/10.1109/TNN.1998.712192

Su, X., ve Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence, 2009.*

Takács, G., Pilászy, I., Németh, B., ve Tikk, D. (2009). Scalable collaborative filtering approaches for large recommender systems. *The Journal of Machine Learning Research*, 10, 623-656.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika,* 25(3-4), 285-294. https://doi.org/10.2307/2332286

Vogel, W. (1960). A sequential design for the two armed bandit. *The Annals of Mathematical Statistics*, 31(2), 430-443. https://www.jstor.org/stable/2237959

Vogel, W. (1960). An asymptotic minimax theorem for the two armed bandit problem. *The Annals of Mathematical Statistics,* 31(2), 444-451. https://doi.org/ 10.1214/aoms/1177705907

Wu, Q., Wang, H., Gu, Q., ve Wang, H. (2016,). Contextual bandits in a collaborative environment. *In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval.*

Yan, C., Han, H., Wang, Z., ve Zhang, Y. (2021). Two-phase multi-armed bandit for online recommendation. *In 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)* (1-8). IEEE. https://doi.org/10.1109/DSAA53316.2021.9564225

Yan, C., Han, H., Zhang, Y., Zhu, D., ve Wan, Y. (2022). Dynamic clustering based contextual combinatorial multi-armed bandit for online recommendation. *Knowledge-Based Systems,* 257, 109927.

Yan, C., Xian, J., Wan, Y., ve Wang, P. (2021). Modeling implicit feedback based on bandit learning for recommendation. *Neurocomputing,* 447, 244-256.

Yan, Han, Zhang, Zhu ve Wan, (2022). Dynamic clustering based contextual combinatorial multi-armed bandit for online recommendation. *Knowledge-Based Systems*, *257*, 109927. https://doi.org/10.1016/j.knosys.2022.109927

Yang, L., Liu, B., Lin, L., Xia, F., Chen, K., ve Yang, Q. (2020). Exploring clustering of bandits for online recommendation system. *In Proceedings of the 14th ACM Conference on Recommender Systems* (120-129).

Zhang, X., Xie, H., Li, H., ve CS Lui, J. (2020). Conversational contextual bandit: Algorithm and application. *In Proceedings of the web conference 2020* (662-672). https://doi.org/10.48550/arXiv.1906.01219

Zoghi, M., Tunys, T., Ghavamzadeh, M., Kveton, B., Szepesvari, C., ve Wen, Z. (2017). Online learning to rank in stochastic click models. *In International conference on machine learning* (4199-4208). PMLR.

Zong, S., Ni, H., Sung, K., Ke, N. R., Wen, Z., ve Kveton, B. (2016). Cascading bandits for large-scale recommendation problems. *arXiv preprint arXiv:1603.05359*.

EKLER

Ek-1: Veri seti-1 Deneme Sonuçları

Veri seti-1							
DC^{3}MAB				DC^{3}MAB-π			
Kesinlilik	**Duyarlılık**	**F_1 puanı**	**Ödül**	**Kesinlilik**	**Duyarlılık**	**F_1 puanı**	**Ödül**
0,15554	0,04334	0,05995	7728	0,16119	0,04540	0,06239	8331
0,16005	0,04340	0,06062	7817	0,15655	0,04475	0,06152	8175
0,15443	0,04342	0,05962	7621	0,15425	0,04446	0,06060	8035
0,15587	0,04285	0,05963	7614	0,15266	0,04340	0,05956	8021
0,15482	0,04228	0,05865	7574	0,15868	0,04454	0,06112	8013
0,16121	0,04451	0,06147	7887	0,15307	0,04379	0,06000	7967
0,14970	0,04259	0,05828	7393	0,16119	0,04490	0,06180	7923
0,16168	0,04442	0,06133	7765	0,15838	0,04418	0,06069	8020
0,15282	0,04338	0,05959	7613	0,15634	0,04381	0,06010	7901
0,15517	0,04297	0,05946	7604	0,15871	0,04495	0,06145	8228
0,15162	0,04228	0,05866	7368	0,15501	0,04281	0,05910	7865
0,15418	0,04353	0,05999	7785	0,15081	0,04311	0,05918	7897
0,14998	0,04224	0,05844	7417	0,15574	0,04448	0,06068	8158
0,15376	0,04350	0,06004	7596	0,15453	0,04377	0,05993	8088
0,15437	0,04369	0,05989	7708	0,16265	0,04536	0,06247	8215
0,15195	0,04165	0,05798	7405	0,15563	0,04432	0,06075	8024
0,16066	0,04328	0,06041	7660	0,15594	0,04358	0,05992	7896
0,15236	0,04168	0,05793	7505	0,16098	0,04496	0,06218	7986
0,15486	0,04412	0,06042	7725	0,15580	0,04330	0,05972	8018
0,15138	0,04364	0,05972	7412	0,15886	0,04488	0,06165	8076

Ek-2: Veri seti-2 Deneme Sonuçları

Veri seti-2							
DC³MAB				**DC³MAB-π**			
Kesinlilik	**Duyarlılık**	**F₁ puanı**	**Ödül**	**Kesinlilik**	**Duyarlılık**	**F₁ puanı**	**Ödül**
0,14457	0,04150	0,05720	7438	0,15351	0,04318	0,05992	8403
0,14764	0,04099	0,05721	7685	0,15113	0,04438	0,06036	8351
0,13912	0,04018	0,05539	7543	0,14417	0,04084	0,05608	7773
0,14871	0,04094	0,05714	7459	0,14483	0,04156	0,05715	8004
0,14753	0,03961	0,05564	7505	0,14697	0,04155	0,05746	8093
0,15156	0,04120	0,05777	7566	0,14463	0,04200	0,05746	8132
0,14775	0,04193	0,05801	7547	0,14668	0,04225	0,05778	8240
0,14177	0,04010	0,05556	7506	0,14908	0,04201	0,05787	8092
0,14656	0,04106	0,05699	7561	0,14745	0,04327	0,05914	8158
0,14609	0,04106	0,05697	7556	0,14123	0,04141	0,05619	8031
0,14618	0,04141	0,05757	7790	0,14848	0,04164	0,05745	8068
0,14078	0,03998	0,05533	7472	0,15012	0,04169	0,05783	8152
0,14609	0,04050	0,05619	7525	0,15407	0,04289	0,05949	8051
0,14942	0,04097	0,05725	7529	0,15031	0,04255	0,05844	8170
0,14619	0,04076	0,05676	7539	0,14569	0,04211	0,05762	8291
0,15113	0,04177	0,05842	7756	0,15089	0,04229	0,05855	8126
0,14779	0,04132	0,05720	7677	0,14260	0,04146	0,05649	7986
0,14592	0,04180	0,05771	7554	0,14559	0,04234	0,05795	8105
0,14440	0,04114	0,05689	7670	0,15101	0,04189	0,05821	8034
0,13959	0,03988	0,05483	7687	0,14831	0,04185	0,05774	8031

Ek-3: Veri seti-3 Deneme Sonuçları

Veri seti-3							
DC³MAB				**DC³MAB-π**			
Kesinlilik	**Duyarlılık**	**F₁ puanı**	**Ödül**	**Kesinlilik**	**Duyarlılık**	**F₁ puanı**	**Ödül**
0,14648	0,04456	0,06012	8314	0,14854	0,04337	0,05906	8596
0,14994	0,04387	0,05999	8254	0,15174	0,04521	0,06167	8852
0,15009	0,04420	0,06038	8314	0,14995	0,04413	0,06015	8656
0,14731	0,04402	0,05990	7658	0,15189	0,04556	0,06178	8817
0,14887	0,04325	0,05939	8278	0,15268	0,04388	0,06041	8716
0,14663	0,04305	0,05879	8012	0,14611	0,04390	0,05930	8449
0,14877	0,04365	0,05944	8142	0,14904	0,04328	0,05914	8445
0,14935	0,04349	0,05970	8290	0,14800	0,04480	0,06074	8668
0,14679	0,04255	0,05849	8041	0,15565	0,04641	0,06324	9004
0,14626	0,04269	0,05892	8180	0,14904	0,04385	0,05988	8429
0,15222	0,04444	0,06113	8364	0,14790	0,04344	0,05918	8415
0,14390	0,04209	0,05760	7653	0,15394	0,04453	0,06127	8620
0,15038	0,04399	0,06027	8412	0,15093	0,04454	0,06081	8826
0,14616	0,04385	0,05941	8412	0,14825	0,04378	0,05967	8592
0,14825	0,04270	0,05876	8204	0,15314	0,04521	0,06161	8991
0,14805	0,04413	0,05991	8012	0,14834	0,04463	0,06037	8608
0,14767	0,04349	0,05910	8137	0,14782	0,04324	0,05918	8579
0,14790	0,04346	0,05929	8030	0,14519	0,04371	0,05940	8485
0,14893	0,04467	0,06086	8417	0,15188	0,04423	0,06041	8687
0,14901	0,04329	0,05963	8192	0,15123	0,04553	0,06154	8695