Bulanık Mantık

Yapay Zeka Ders Serisi: Bulanık Mantık ve Nöro-Bulanık Sistemler

Bölüm 7: Yapay Sinir Ağlarının Ötesi: Belirsizlik ve Dilsel Mantıkla Başa Çıkmak

Önceki derslerde yapay sinir ağlarının (YSA) temel yapı taşlarını, öğrenme mekanizmalarını ve pratik uygulamalarını inceledik. Bu bölümde, YSA'ların güçlü yönlerini bir temel olarak kabul ederken, bu modellerin doğasında bulunan bazı temel sınırlılıklara odaklanacak ve bu sınırlılıkların üstesinden gelmek için geliştirilmiş tamamlayıcı bir paradigma olan bulanık mantığı keşfedeceğiz.

7.1. Giriş: Yapay Sinir Ağlarının "Siyah Kutu" Problemi ve Yorumlanabilirlik İhtiyacı

Derin öğrenme modelleri ve özellikle de çok katmanlı yapay sinir ağları, görüntü tanıma, doğal dil işleme ve karmaşık tahminleme görevlerinde insanüstü performanslar sergileyebilmektedir. Ancak bu yüksek doğruluk, genellikle bir bedelle gelir: şeffaflık eksikliği. Bir YSA, milyonlarca (bazen milyarlarca) ağırlık ve bias parametresi arasındaki karmaşık ve doğrusal olmayan etkileşimler sonucunda bir karara varır. Bu sürecin karmaşıklığı, modelin belirli bir sonuca _nasıl_ulaştığını adım adım takip etmeyi veya mantıksal olarak açıklamasını neredeyse imkansız hale getirir. Bu durum, yapay zeka literatüründe "siyah kutu" (black box) problemi olarak bilinir.1

Tıp alanında bir hastalığı teşhis eden, finans sektöründe bir kredi başvurusunu reddeden veya otonom bir aracın direksiyonunu çeviren bir yapay zeka modelinin sadece *ne* karar verdiği değil, bu kararı *neden* verdiği de hayati önem taşır. Şeffaflığın olmaması, bu tür kritik sistemlerde güven, hesap verebilirlik ve hata ayıklama süreçlerini ciddi şekilde zorlaştırır.2 Örneğin, modelin eğitim verisindeki istenmeyen bir önyargıyı (bias) öğrenip öğrenmediğini anlamak, iç işleyişi anlaşılamayan bir "siyah kutu" modelinde oldukça zordur.1

Bu noktada, "Açıklanabilir Yapay Zeka" (Explainable AI - XAI) alanı devreye girer. XAI, yapay zeka modellerinin karar mekanizmalarını insanlar için anlaşılır ve yorumlanabilir hale getirmeyi amaçlayan bir araştırma ve uygulama disiplinidir.1 İşte bulanık mantık, bu noktada YSA'lara güçlü bir alternatif ve tamamlayıcı olarak ortaya çıkar. YSA'ların bu temel sınırlılıkları, bulanık mantık gibi yaklaşımlara olan ihtiyacı doğrudan tetiklemiştir. YSA'ların iki temel eksikliği bulunmaktadır:

1. **Yorumlanabilirlik Eksikliği:** Daha önce belirtildiği gibi, YSA'lar kararlarını açıklayamazlar. Bu "siyah kutu" doğası, güvenilirlik gerektiren uygulamalarda büyük bir engel teşkil eder.
2. **Dilsel Belirsizliği Yönetememe:** İnsanlar akıl yürütürken ve karar verirken sıklıkla "hava *oldukça* sıcak", "proje riski *yüksek*" veya "hedefe *neredeyse* ulaştık" gibi kesin olmayan, nitel ve dilsel ifadeler kullanır.6 Standart YSA'lar ise doğaları gereği sayısal ve kesin girdilerle çalışmak üzere tasarlanmıştır. Bu tür dilsel belirsizlikleri doğal bir şekilde işleyemezler ve deterministik yapıları nedeniyle tahminlerindeki belirsizlik seviyesini doğal olarak ifade edemezler.8

Bulanık mantık, bu iki temel soruna doğrudan çözümler sunmak üzere tasarlanmıştır. İnsan tarafından kolayca okunabilen EĞER-İSE (IF-THEN) kurallarına dayalı yapısı sayesinde, modelin karar mantığını şeffaf bir şekilde ortaya koyar ve doğal bir XAI aracı görevi görür.10 Aynı zamanda, "yüksek", "düşük", "ılık" gibi dilsel kavramları matematiksel olarak modelleyerek, gerçek dünyadaki belirsizlikle başa çıkmak için güçlü bir çerçeve sunar. Bu nedenle bulanık mantığın geliştirilmesi, yalnızca akademik bir arayış değil, aynı zamanda YSA'ların pratik uygulamalardaki temel eksikliklerine verilmiş anlamlı ve nedensel bir yanıttır.

7.2. Kesinlikten Esnekliğe: Boolean Mantığından Bulanık Mantığa Geçiş

Bilgisayar programcılığının temelini oluşturan klasik mantık, Aristo veya Boolean mantığı olarak bilinir. Bu mantık sisteminde her şey ikilidir: bir önerme ya tamamen "doğru" (1) ya da tamamen "yanlış" (0) olabilir. Arada bir değer yoktur.6 Örneğin, bir `if` koşulu (`if sicaklik > 25`) ya karşılanır ya da karşılanmaz.

Bulanık mantık ise bu keskin ikili yapıyı reddederek, gerçek dünyadaki belirsizliği ve derecelendirmeyi kucaklar. 1965 yılında Lotfi Zadeh tarafından ortaya atılan bu yaklaşım, "doğruluk dereceleri" veya "üyelik dereceleri" kavramını temel alır. Bulanık mantıkta bir önerme, 0 (tamamen yanlış) ile 1 (tamamen doğru) arasında herhangi bir reel sayı değeri alabilir.6

Bu farkı somut bir analoji ile anlayalım: Bir odanın sıcaklığını ele alalım.

- **Boolean Mantığı:** Bir kural tanımlayabiliriz: "Eğer sıcaklık 25°C'den büyükse, oda 'Sıcak'tır." Bu durumda 24.9°C "Sıcak Değil" (0) iken, 25.1°C "Sıcak" (1) olur. Bu 0.2°C'lik küçük fark, algıda mutlak bir değişime neden olur ki bu, insan deneyimiyle örtüşmez.
- **Bulanık Mantık:** "Sıcak" kavramını keskin bir sınır yerine kademeli bir geçiş olarak tanımlarız. Örneğin, 20°C "Sıcak" kümesine 0.0 üyelik derecesiyle ait olabilir (yani hiç sıcak değil). 26°C, bu kümeye 0.6 üyelik derecesiyle ait olabilir (yani *oldukça* sıcak). 35°C ise 1.0 üyelik derecesiyle tam üye olabilir (yani *kesinlikle* sıcak).6 Bu yaklaşım, "ılık", "biraz sıcak", "çok sıcak" gibi dilsel ifadelerin temsil ettiği kademeli geçişleri matematiksel olarak modellememizi sağlar.7

Öğrencilerin sıkça karıştırdığı bir diğer nokta, bulanık mantık ile olasılık arasındaki farktır. Bu iki kavram da aralığında değerler alsa da, felsefi olarak tamamen farklıdırlar:

- **Olasılık**, bir olayın gerçekleşme *ihtimalini* veya belirsiz bir sonuç hakkındaki *bilgi eksikliğimizi* (cehaletin ölçüsü) ifade eder. Bir madeni parayı havaya attığımızda tura gelme olasılığı 0.5'tir, ancak deneyin sonunda sonuç kesindir: ya tura ya da yazı.
- **Bulanık Mantık**, bir elemanın belirsiz veya muğlak bir kavrama ne kadar *uyduğunu* veya o kümeye ne derecede *ait olduğunu* (belirsizliğin ölçüsü) ifade eder. Bir durumun "orta sıcaklıkta" olma derecesi kalıcı olarak 0.5 olabilir; bu bir ihtimal değil, durumun kendisinin bir özelliğidir.6

Aşağıdaki tablo, bu iki mantık sistemi arasındaki temel farkları özetlemektedir.

Tablo 3: Boolean Mantığı ve Bulanık Mantık Karşılaştırması

Özellik	Boolean Mantığı	Bulanık Mantık
Doğruluk Değerleri	Sadece 0 (Yanlış) ve 1 (Doğru) 6	$$ aralığındaki tüm reel sayılar 6
Temel Felsefe	Bir şey ya bir kümenin üyesidir ya da değildir (kesin sınırlar).	Bir şey bir kümeye kısmen üye olabilir (kademeli üyelik).16
Temsil Ettiği Kavram	Kesinlik, ikilik.	Belirsizlik, muğlaklık, dilsel kavramlar.6
Gerçek Dünya Analojisi (Sıcaklık)	Sıcaklık > 25°C ise "Sıcak" = 1, değilse = 0.	Sıcaklık 26°C ise "Sıcak" üyeliği = 0.6; 35°C ise = 1.0.6
Uygulama Alanı	Dijital devreler, klasik programlama koşulları.	Kontrol sistemleri, karar destek sistemleri, insan benzeri akıl yürütme.18

Bölüm 8: Bulanık Çıkarım Sisteminin (FIS) Anatomisi

Bulanık mantığı kullanarak karar veren veya bir sistemi kontrol eden yapıya **Bulanık Çıkarım Sistemi (Fuzzy Inference System - FIS)** denir. Bir FIS, insan benzeri akıl yürütme sürecini taklit eden üç temel aşamadan oluşur: Bulanıklaştırma (Fuzzification), Kural Değerlendirme (Inference) ve Durulaştırma (Defuzzification). Bu bölümde, bir FIS'in anatomisini bu üç aşama üzerinden detaylı bir şekilde inceleyeceğiz.

8.1. Kavramları Sayısallaştırmak: Üyelik Fonksiyonları ve Bulanıklaştırma (Fuzzification)

Bir FIS'in ilk adımı **Bulanıklaştırma**'dır. Bu süreç, bir sensörden gelen sıcaklık değeri (örneğin 28°C) veya bir proje yönetim aracından gelen bütçe aşımı oranı (örneğin %75) gibi kesin (crisp)

sayısal girdileri alır ve bunları "sıcak", "orta riskli" gibi dilsel (linguistic) kavramlara dönüştürür.7

Bu dönüşümün matematiksel temelini **Üyelik Fonksiyonları (Membership Functions - MF)** oluşturur. Bir üyelik fonksiyonu, $\mu(x)$ ile gösterilir ve bir x girdi değerinin, belirli bir bulanık kümeye (örneğin, "ılık" kümesi) hangi derecede ait olduğunu aralığında bir değerle tanımlar.17 Örneğin, 28°C'lik bir sıcaklık değeri, "Ilık" bulanık kümesine 0.6 üyelik derecesiyle ait olurken, aynı anda "Sıcak" bulanık kümesine 0.4 üyelik derecesiyle ait olabilir.17 Bu, bir değerin birden fazla kavrama kısmen ait olabilme esnekliğini sağlar.

Bir bulanık kümeyi ve üyelik fonksiyonunu tanımlarken kullanılan bazı temel terimler şunlardır 20:

- **Evren (Universe of Discourse):** Bir değişkenin alabileceği tüm olası kesin değerler kümesi (örneğin, sıcaklık için aralığı).
- **Destek (Support):** Üyelik derecesi sıfırdan büyük olan tüm elemanların kümesi.
- **Çekirdek (Core):** Üyelik derecesi tam olarak 1 olan tüm elemanların kümesi.
- **Geçiş Noktası (Crossover Point):** Üyelik derecesi 0.5 olan elemanlar.

Sistem tasarımında en yaygın kullanılan üyelik fonksiyonu tipleri şunlardır:

- **Üçgen (Triangular MF):** Üç noktayla (alt sınır, üyelik derecesinin 1 olduğu tepe noktası ve üst sınır) tanımlanan basit ve hesaplama açısından verimli bir fonksiyondur.17
- **Yamuk (Trapezoidal MF):** Dört noktayla tanımlanır ve belirli bir değer aralığının tamamının kümeye tam üye (üyelik derecesi 1.0) olduğu durumları modellemek için idealdir.17
- **Gauss (Gaussian) ve Sigmoid MF:** Bu fonksiyonlar, çan eğrisi veya S-şekilli eğriler gibi pürüzsüz ve sürekli formlara sahiptir. Keskin köşeler yerine yumuşak geçişler sağlarlar, bu da onları biyolojik veya doğal süreçler gibi kademeli değişimlerin olduğu sistemleri modellemek için daha uygun hale getirir. Bu pürüzsüz yapı, sistemin daha sağlam ve hassas olmasını sağlar.17

Aşağıdaki tablo, bu yaygın üyelik fonksiyonlarını ve temel özelliklerini karşılaştırmaktadır.

Tablo 4: Yaygın Üyelik Fonksiyonları ve Özellikleri

Fonksiyon Tipi	Grafiksel Gösterim	Tanımlayıcı Parametreler	Avantajları	Dezavantajları/Kullanım Yeri
Üçgen (Triangular)	Basit üçgen çizimi	3 nokta (a, b, c)	Hesaplaması çok hızlı, sezgisel ve basit.20	Geçişler keskin köşelere sahiptir, pürüzsüz değildir.
Yamuk (Trapezoidal)	Basit yamuk	4 nokta (a, b, c, d)	Bir değer aralığının tam	Geçişler keskin köşelere sahiptir.

Fonksiyon Tipi	Grafiksel Gösterim	Tanımlayıcı Parametreler	Avantajları	Dezavantajları/Kullanım Yeri
	çizimi		üye olmasını sağlar.20	
Gauss (Gaussian)	Çan eğrisi çizimi	2 parametre (merkez, standart sapma)	Pürüzsüz ve sürekli geçişler, doğal olaylara daha uygun.17	Üçgen/Yamuk'a göre hesaplaması biraz daha maliyetlidir.
Sigmoid	S-şekilli eğri çizimi	2 parametre (eğim, merkez)	"Çok yüksek" gibi açık uçlu kavramlar ve asimetrik geçişler için idealdir.17	Hesaplaması diğerlerine göre daha maliyetlidir.

8.2. İnsan Bilgisini Kodlamak: Kural Tabanı ve Çıkarım Motoru (Inference Engine)

Bulanıklaştırma adımı tamamlandıktan sonra, sistemin "beyni" olarak kabul edilen **Kural Tabanı (Rule Base)** ve **Çıkarım Motoru (Inference Engine)** devreye girer.7

Kural Tabanı, bir uzmanın bilgi birikimini veya sistemden beklenen davranışı yansıtan, insan tarafından okunabilir bir dizi EĞER-İSE (IF-THEN) kuralından oluşur.10 Bu kurallar, bulanıklaştırılmış girdi değişkenleri ile çıktı değişkenleri arasındaki mantıksal ilişkiyi kurar. Örneğin, bir klima kontrol sistemi için bir kural şöyle olabilir:

EĞER sıcaklık YÜKSEK ise VEYA nem YÜKSEK ise, O HALDE kompresör_hızı YÜKSEK olmalıdır. 6

Çıkarım Motoru, bulanıklaştırılmış girdi değerlerini (üyelik derecelerini) alır ve bunları kural tabanındaki her bir kurala uygular.10 Bu süreç birkaç adımdan oluşur:

1. **Bulanık Operatörlerin Uygulanması:** Bir kuralın "EĞER" (antecedent) kısmı birden fazla koşul içeriyorsa, bu koşullar bulanık mantık operatörleri kullanılarak tek bir değere indirgenir.
   - **Fuzzy AND (Kesişim):** Genellikle `min` operatörü ile gerçekleştirilir. Örneğin, `μ_A(x) AND μ_B(y)` ifadesinin doğruluk derecesi `min(μ_A(x), μ_B(y))` olarak hesaplanır.
   - **Fuzzy OR (Birleşim):** Genellikle `max` operatörü ile gerçekleştirilir. Örneğin, `μ_A(x) OR μ_B(y)` ifadesinin doğruluk derecesi `max(μ_A(x), μ_B(y))` olarak hesaplanır.6
2. **Çıkarım (Implication):** Her bir kural için, antecedent kısmından elde edilen tek doğruluk derecesi (kuralın "ateşleme gücü" veya "firing strength" olarak da bilinir), kuralın "İSE" (consequent) kısmındaki çıktı üyelik fonksiyonunu şekillendirmek için kullanılır. Bu

genellikle, çıktı üyelik fonksiyonunun ateşleme gücü seviyesinde "kesilmesi" (clipping) veya ateşleme gücü ile "ölçeklendirilmesi" (scaling) şeklinde yapılır.

3. **Birleştirme (Aggregation):** Tüm kurallardan elde edilen bu şekillendirilmiş veya kesilmiş çıktı bulanık kümeleri, tek bir birleşik bulanık küme oluşturmak üzere bir araya getirilir. Bu işlem genellikle `max` operatörü kullanılarak, tüm çıktı fonksiyonlarının zarfının (envelope) alınmasıyla gerçekleştirilir.25

8.3. Karara Varmak: Durulaştırma (Defuzzification) Yöntemleri

Bulanık çıkarım sürecinin son adımı **Durulaştırma**'dır. Bu aşama, bir önceki adımda elde edilen birleşik bulanık çıktı kümesini, bir motorun hızını ayarlamak, bir vanayı açmak veya bir riski puanlamak gibi eyleme dönüştürülebilir, tek ve kesin (crisp) bir sayısal değere dönüştürür.21

Pek çok durulaştırma yöntemi bulunmakla birlikte, en yaygın kullanılan ve en sezgisel olanı **Ağırlık Merkezi (Centroid of Area - COA veya Center of Gravity - COG)** yöntemidir. Bu yöntem, birleşik çıktı bulanık kümesinin oluşturduğu geometrik şeklin "ağırlık merkezini" veya "denge noktasını" hesaplamaya dayanır.29 Bu yaklaşımın temel avantajı, nihai sonuca katkıda bulunan tüm kuralların etkisini (hem ateşleme güçlerini hem de çıktı üyelik fonksiyonlarının şekillerini) hesaba katması ve böylece dengeli ve sağlam bir sonuç üretmesidir.

Sürekli bir evren için Centroid yönteminin matematiksel formülü şu şekildedir 28:

$$x_{centroid} = \frac{\int \mu(x) \cdot x \, dx}{\int \mu(x) \, dx}$$

Bu formül, çıktı kümesinin şekli altındaki alanın birinci momentinin (her bir x noktasının kendi üyelik derecesiyle ağırlıklandırılmış toplamı) toplam alana bölünmesi anlamına gelir.

8.4. İki Yaklaşımın Karşılaştırılması: Mamdani ve Takagi-Sugeno Sistemleri

Bulanık çıkarım sistemleri, kurallarının sonuç (consequent) kısımlarını nasıl tanımladıklarına göre temel olarak iki ana kategoriye ayrılır: **Mamdani** ve **Takagi-Sugeno (TSK)**. Bu iki model arasındaki farkı anlamak, bir sonraki bölümde ele alacağımız Nöro-Bulanık sistemlerin (ANFIS) neden Sugeno modelini temel aldığını kavramak için kritik öneme sahiptir.

İki model arasındaki en temel ve belirleyici fark, EĞER-İSE kurallarının "İSE" (THEN) kısmının yapısıdır 32:

- **Mamdani Tipi FIS:** Bu modelde, kuralın sonuç kısmı da bir bulanık kümedir. Örneğin: `İSE fan_hızı YÜKSEK'tir`. Burada "YÜKSEK", bir üyelik fonksiyonu ile tanımlanmış dilsel bir kavramdır. Bu yapı, insan uzmanlığının dilsel doğasını doğrudan yansıttığı için son derece sezgisel ve yorumlanabilirdir.33 Bir uzmanın "Eğer durum kritikse, acil müdahale gerekir"

şeklindeki ifadesi, Mamdani kurallarıyla doğrudan modellenebilir. Ancak bu yüksek yorumlanabilirlik, hesaplama maliyetiyle birlikte gelir. Çünkü sistemin nihai kesin çıktısını üretmek için, birleştirilmiş çıktı bulanık kümesi üzerinde karmaşık bir durulaştırma işlemi (örneğin, Centroid integralinin hesaplanması) yapılması gerekir.35

- **Takagi-Sugeno (TSK) Tipi FIS:** Bu modelde ise kuralın sonuç kısmı bir bulanık küme değil, girdilerin (genellikle doğrusal) bir fonksiyonudur. Örneğin: `İSE çıktı = p \cdot x + q \cdot y + r`. Burada x ve y sistemin girdileri; p, q ve r ise o kurala özgü sayısal katsayılardır.6 Bu yapı, "çıktı YÜKSEK'tir" demekten daha az sezgiseldir ve dilsel yorumlanabilirliği bir miktar azaltır. Ancak devrim niteliğinde bir avantaj sunar: Her kuralın çıktısı zaten kesin bir sayıdır. Bu nedenle, karmaşık ve zaman alıcı bir durulaştırma sürecine ihtiyaç duyulmaz. Sistemin nihai çıktısı, her bir kuralın çıktısının, o kuralın ateşleme gücüyle ağırlıklandırılmış ortalaması alınarak doğrudan ve verimli bir şekilde hesaplanır.32

Mamdani ve Sugeno arasındaki bu tercih, sadece teknik bir detay değil, aynı zamanda bir modelleme felsefesi seçimidir. Mamdani, insan benzeri dilsel akıl yürütmeyi ve şeffaflığı en üst düzeye çıkararak uzman sistemler ve karar destek araçları için ideal bir yapı sunar. Sugeno ise, yorumlanabilirlikten bir miktar ödün vererek hesaplama verimliliği, matematiksel kesinlik ve optimizasyona uygunluk kazanır. Sonuç kısmındaki p, q, r gibi sayısal katsayılar, tıpkı bir sinir ağındaki ağırlıklar gibi, veriden öğrenilebilecek parametrelerdir. Bu özellik, Sugeno modelini, sinir ağlarının öğrenme algoritmalarıyla birleştirilmeye (hibritleşmeye) son derece uygun hale getirir. Bu stratejik takas, bir sonraki bölümde göreceğimiz ANFIS modelinin temelinde Sugeno mimarisinin yatmasının ana nedenidir.34

Aşağıdaki tablo, iki FIS tipini temel özellikleri açısından karşılaştırmaktadır.

Tablo 5: Mamdani ve Takagi-Sugeno FIS Karşılaştırması

Özellik	Mamdani Tipi FIS	Takagi-Sugeno Tipi FIS
Kural Sonucu (Consequent)	Bir bulanık küme (örn., `çıktı YÜKSEK'tir`) 32	Bir matematiksel fonksiyon (örn., $z = px + qy + r$) 32
Çıktı Üyelik Fonksiyonları	Gereklidir.	Gerekli değildir (çıktı fonksiyoneldir).32
Durulaştırma (Defuzzification)	Gerekli ve hesaplama maliyetli (örn., Centroid).35	Gerekli değil; ağırlıklı ortalama ile doğrudan hesaplanır.35
Hesaplama Verimliliği	Daha yavaş.35	Daha hızlı ve verimli.34
Yorumlanabilirlik	Çok yüksek, insan diline yakın.33	Daha düşük, sonuçlar matematiksel fonksiyonlardır.32

Özellik	Mamdani Tipi FIS	Takagi-Sugeno Tipi FIS
Uygun Olduğu Alanlar	Uzman sistemler, karar destek, dilsel modelleme.33	Kontrol sistemleri, optimizasyon, uyarlanabilir sistemler (ANFIS).33

Bölüm 9: Sinir Ağları ve Bulanık Mantığın Sentezi: Nöro-Bulanık Sistemler

Şimdiye kadar yapay sinir ağlarının veri odaklı öğrenme gücünü ve bulanık mantık sistemlerinin kural tabanlı yorumlanabilirliğini ayrı ayrı inceledik. Bu bölümde, bu iki güçlü paradigmayı bir araya getirerek her iki dünyanın en iyi özelliklerini birleştiren hibrit sistemleri, özellikle de **ANFIS (Uyarlanabilir Ağ Tabanlı Bulanık Çıkarım Sistemi)**modelini ele alacağız.

9.1. İki Dünyanın En İyisi: ANFIS'e (Uyarlanabilir Ağ Tabanlı Bulanık Çıkarım Sistemi) Giriş

ANFIS (Adaptive Neuro-Fuzzy Inference System), adından da anlaşılacağı gibi, yapay sinir ağları ile bulanık çıkarım sistemlerini bütünleştiren bir hibrit zeki sistemdir.37 ANFIS'in temel felsefesi, bulanık mantığın EĞER-İSE kuralları ve dilsel değişkenler gibi insan tarafından anlaşılabilir, şeffaf yapısını bir iskelet olarak kullanmak ve bu iskeletin parametrelerini (üyelik fonksiyonlarının şekli, kural sonuçlarının katsayıları vb.) veriden otomatik olarak öğrenmek için yapay sinir ağlarının öğrenme ve adaptasyon yeteneğinden faydalanmaktır.36

Bu hibrit yaklaşım, her iki dünyanın da temel avantajlarını bir araya getirir:

- **YSA'lardan Gelen Avantajlar:** Veriden öğrenme, karmaşık ve doğrusal olmayan ilişkileri modelleme, genelleme yapabilme ve adaptasyon yeteneği.12
- **Bulanık Mantıktan Gelen Avantajlar:** İnsan uzman bilgisini sisteme dahil edebilme, EĞER-İSE kuralları sayesinde şeffaf ve yorumlanabilir bir yapı, dilsel belirsizlikle başa çıkabilme.36

Sonuç olarak ANFIS, hem YSA'lar gibi yüksek tahmin doğruluğuna sahip olabilen hem de bulanık sistemler gibi "siyah kutu" olmayan, kararlarının arkasındaki mantığın anlaşılabilir olduğu modeller oluşturma potansiyeli sunar.12

9.2. ANFIS Mimarisi: 5 Katmanlı Yapının İncelenmesi

ANFIS, fonksiyonel olarak bir Takagi-Sugeno bulanık çıkarım sistemine eşdeğerdir ve bu sistemi 5 katmanlı bir ileri beslemeli sinir ağı mimarisi olarak uygular.37 Bu mimarideki her katman, hem bir sinir ağı katmanının hem de bulanık çıkarım sürecinin belirli bir adımının görevini yerine getirir. Şimdi bu katmanları adım adım inceleyelim 37:

- **Katman 1: Bulanıklaştırma Katmanı (Fuzzification Layer)**

    - **Amaç:** Sisteme giren kesin (crisp) sayısal değerleri alır ve her bir girdi değişkeni için tanımlanmış olan dilsel terimlere (bulanık kümelere) ait üyelik derecelerini hesaplar.
    - **İşlem:** Bu katmandaki her bir düğüm (nöron), bir üyelik fonksiyonunu temsil eder. Düğümün çıktısı, girdinin o üyelik fonksiyonuna göre aldığı üyelik derecesidir. Matematiksel olarak, i. düğümün çıktısı $O_{1,i} = \mu_{A_i}(x)$'dir.
    - **Bağlantılar:** Bu katman, bir FIS'in **Bulanıklaştırma** adımına karşılık gelir. Bu katmandaki üyelik fonksiyonlarının şeklini belirleyen parametreler (örneğin, bir Gauss fonksiyonunun merkezi ve genişliği) "öncül parametreler" (premise parameters) olarak adlandırılır ve öğrenme sırasında ağ tarafından ayarlanabilen (adaptive) parametrelerdir.37
- **Katman 2: Kural Katmanı (Rule Layer / Product Layer)**
    - **Amaç:** Her bir bulanık kuralın ateşleme gücünü (firing strength) hesaplar. Ateşleme gücü, o kuralın antecedent (EĞER) kısmının mevcut girdiler için ne kadar doğru olduğunu gösterir.
    - **İşlem:** Bu katmandaki her düğüm, bir EĞER-İSE kuralını temsil eder. Girdi olarak bir önceki katmandan gelen üyelik derecelerini alır ve bunları bir T-norm operatörü (genellikle basit bir çarpma işlemi) ile birleştirir. Örneğin, $w_i = \mu_{A_i}(x) \times \mu_{B_i}(y)$.
    - **Bağlantılar:** Bu adım, bir FIS'in **Çıkarım Motoru**'ndaki antecedent birleştirme işlemine karşılık gelir. Bu katmandaki düğümler genellikle sabittir ve öğrenilebilir parametreleri yoktur.37
- **Katman 3: Normalizasyon Katmanı (Normalization Layer)**
    - **Amaç:** Her bir kuralın ateşleme gücünü, tüm kuralların toplam ateşleme gücüne bölerek normalize eder. Bu, her bir kuralın nihai çıktıya yapacağı göreceli katkıyı belirler.
    - **İşlem:** Her düğüm, $\bar{w}_i = w_i / \sum_j w_j$ hesaplamasını yapar. Çıktılar "normalize edilmiş ateşleme güçleri" olarak adlandırılır.
    - **Bağlantılar:** Bu katman da genellikle sabittir ve öğrenilebilir parametre içermez.37
- **Katman 4: Durulaştırma Katmanı (Defuzzification Layer)**
    - **Amaç:** Her bir kuralın normalize edilmiş ateşleme gücünü, o kuralın sonuç (consequent) fonksiyonuyla çarparak o kuralın çıktıya olan katkısını hesaplar.
    - **İşlem:** Bu katmandaki her düğüm uyarlanabilirdir. Her düğüm, $O_{4,i} = \bar{w}_i \cdot f_i$ işlemini gerçekleştirir. Burada $\bar{w}_i$ bir önceki katmanın çıktısıdır ve $f_i = p_i x + q_i y + r_i$ şeklindeki Takagi-Sugeno tipi doğrusal fonksiyondur.
    - **Bağlantılar:** Bu katmandaki p_i, q_i, r_i katsayıları, "sonuç parametreleri" (consequent parameters) olarak bilinir ve öğrenme sırasında ağ tarafından ayarlanırlar.37
- **Katman 5: Toplam Katmanı (Output Layer)**
    - **Amaç:** Bir önceki katmandan gelen tüm kuralların katkılarını toplayarak sistemin nihai, tek ve kesin (crisp) çıktısını üretir.

- **İşlem:** Bu katmanda genellikle tek bir sabit düğüm bulunur ve basitçe tüm girdilerini toplar: $Genel\_Çıktı = \sum_i O_{4,i} = \sum_i \bar{w}_i f_i$.
- **Bağlantılar:** Bu işlem, Takagi-Sugeno modelindeki ağırlıklı ortalama ile durulaştırma işlemini tamamlar ve bir YSA'daki çıktı düğümüne benzer bir işlev görür.37

Aşağıdaki tablo, ANFIS'in katmanlı yapısını, her katmanın görevini ve hem bulanık mantık hem de sinir ağları dünyasındaki karşılıklarını özetlemektedir.

Tablo 6: ANFIS Katmanlarının Görevleri ve Bağlantıları

Katman	Katman Adı	Amaç/İşlem	FIS Karşılığı	YSA Karşılığı	Öğrenilebili Parametrele
1	Bulanıklaştırma	Üyelik derecelerini hesaplar ($\mu(x)$).	Bulanıklaştırma	Girdi/Özellik Katmanı	Öncül Parametreler (MF parametreler örn., c, σ)
2	Kural (Çarpım)	Kuralların ateşleme gücünü hesaplar ($w_i = \mu_A \times \mu_B$).	Kural Antecedent Birleştirme	Gizli Katman (AND operatörü)	Yok
3	Normalizasyon	Ateşleme güçlerini normalize eder ($\bar{w}_i$).	Kural Ağırlıklandırma	Gizli Katman (Normalizasyon)	Yok
4	Durulaştırma	Normalize edilmiş kural çıktılarını hesaplar ($\bar{w}_i \cdot f_i$).	Kural Consequent Uygulama	Gizli Katman (Doğrusal Kombinasyon)	Sonuç Parametreler (p, q, r)
5	Toplam (Çıktı)	Tüm kural çıktılarını toplar ($\sum \bar{w}_i f_i$).	Ağırlıklı Ortalama ile Durulaştırma	Çıktı Katmanı (Toplama)	Yok

9.3. Veriden Öğrenme: ANFIS Hibrit Öğrenme Algoritması

ANFIS'in en yenilikçi yönü, parametrelerini veriden öğrenmek için kullandığı **hibrit öğrenme algoritmasıdır**. Standart bir YSA'da tüm ağırlıklar genellikle sadece Geri Yayılım

(Backpropagation) ve Gradyan İnişi (Gradient Descent) ile güncellenir. ANFIS ise daha verimli bir "böl ve yönet" stratejisi izler.39

ANFIS'in öğrenme problemi, iki farklı parametre setine ayrılabilir: doğrusal olmayan öncül parametreler (Katman 1'deki üyelik fonksiyonu parametreleri) ve doğrusal olan sonuç parametreleri (Katman 4'teki fonksiyon katsayıları). Hibrit algoritma, bu iki parametre setini farklı ve en verimli yöntemlerle optimize eder. Her bir eğitim iterasyonu (epoch), iki geçişten (two-pass) oluşur 43:

1. **İleri Geçiş (Forward Pass):** Bu aşamada, öncül parametreler (üyelik fonksiyonları) sabit tutulur. Girdi verileri ağdan ileri doğru beslenir. Katman 4'e ulaşıldığında, sistemin genel çıktısı, sonuç parametrelerinin (p, q, r) doğrusal bir fonksiyonu haline gelir. Bu durum, $y = Ax$ formunda bir lineer denklem sistemine karşılık gelir. Bu sistem, doğrusal problemler için en hızlı ve en optimum çözümü sunan **En Küçük Kareler Yöntemi (Least Squares Estimation - LSE)** kullanılarak tek bir adımda analitik olarak çözülür. Bu sayede, mevcut öncül parametreler için en iyi sonuç parametreleri anında bulunur.39
2. **Geri Geçiş (Backward Pass):** Bu aşamada ise, ileri geçişte LSE ile bulunan optimum sonuç parametreleri sabit tutulur. Ağın çıktısı ile hedeflenen gerçek değer arasındaki hata hesaplanır. Bu hata, standart YSA'larda olduğu gibi **Geri Yayılım (Backpropagation)** algoritması kullanılarak ağdan geriye doğru yayılır. **Gradyan İnişi (Gradient Descent)** optimizasyon yöntemi, bu hata gradyanını kullanarak sadece öncül parametreleri (üyelik fonksiyonlarının merkez, genişlik gibi parametrelerini) günceller. Bu güncelleme, hatayı azaltacak yönde küçük bir adım atarak yapılır.39 Parametre güncellemesi, zincir kuralı (chain rule) kullanılarak hatanın (E) herhangi bir öncül parametreye (α) göre kısmi türevinin hesaplanmasıyla yönlendirilir: $\frac{\partial E}{\partial \alpha} = \frac{\partial E}{\partial f} \cdot \frac{\partial f}{\partial \alpha}$.47

Bu hibrit yaklaşımın stratejik verimliliği, öğrenme problemini zekice ikiye bölmesinden kaynaklanır. Eğer tüm parametreler aynı anda sadece gradyan inişi ile optimize edilmeye çalışılsaydı, arama uzayı çok büyük ve karmaşık olur, bu da yavaş yakınsamaya ve yerel minimumlara takılma riskine yol açardı.46 ANFIS, problemin doğrusal kısmını son derece verimli olan LSE ile anında çözerek, yavaş ve iteratif olan gradyan inişi yöntemini sadece daha karmaşık olan doğrusal olmayan kısım için kullanır. Bu akıllıca iş bölümü, ANFIS'in standart geri yayılıma göre çok daha hızlı ve güvenilir bir şekilde yakınsamasını sağlayan temel algoritmik üstünlüğüdür.

Bölüm 10: İnteraktif Simülasyon ve Uygulama Atölyesi

Teorik bilgileri kalıcı becerilere dönüştürmenin en etkili yolu, bu kavramları uygulamalı olarak deneyimlemektir. Bu atölye bölümünde, Python'un güçlü kütüphanelerini kullanarak bir temel bulanık çıkarım sistemi ve ardından veriden öğrenen bir ANFIS modeli oluşturacağız.

10.1. Pratik 1: Temel Bulanık Kontrolcü - `scikit-fuzzy` ile Proje Risk Analizi

Bu pratikte, Bölüm 8'de öğrendiğimiz FIS anatomisinin tüm adımlarını (Bulanıklaştırma, Kural Tabanı, Durulaştırma) kod üzerinde uygulayarak basit bir karar destek sistemi oluşturacağız.

Senaryo: Bir yazılım projesinin genel "risk seviyesini", 'bütçe aşımı' ve 'zaman aşımı' yüzdelerine göre değerlendiren bir sistem tasarlayacağız.

Adımlar ve Kod Uygulaması:

1. Kütüphanelerin Kurulumu ve İçe Aktarılması:

   İlk olarak, gerekli olan numpy ve scikit-fuzzy kütüphanelerini kurup projemize dahil etmemiz gerekiyor.7

   Python

   ```
   # Gerekli kütüphaneleri yükleyin: pip install numpy scikit-fuzzy
   import numpy as np
   import skfuzzy as fuzz
   from skfuzzy import control as ctrl
   ```

2. Girdi ve Çıktı Değişkenlerinin Tanımlanması:

   Sistemimiz için iki girdi (Antecedent) ve bir çıktı (Consequent) değişkeni tanımlıyoruz. Her değişkenin değer alabileceği evreni (universe) belirliyoruz.7

   Python

   ```
   # Girdi değişkenleri
   butce_asimi = ctrl.Antecedent(np.arange(0, 101, 1), 'bütçe_aşımı')
   zaman_asimi = ctrl.Antecedent(np.arange(0, 101, 1), 'zaman_aşımı')

   # Çıktı değişkeni
   proje_riski = ctrl.Consequent(np.arange(0, 101, 1), 'proje_riski')
   ```

3. Üyelik Fonksiyonlarının Oluşturulması:

   Her değişken için 'düşük', 'orta', ve 'yüksek' gibi dilsel kavramları, üçgen üyelik fonksiyonları (trimf) kullanarak tanımlıyoruz.7

   Python

   ```
   # Otomatik üyelik fonksiyonu oluşturma (3 seviyeli: düşük, orta, yüksek)
   butce_asimi.automf(3)
   zaman_asimi.automf(3)
   ```

```
# Çıktı için özel üyelik fonksiyonları tanımlayalım
proje_riski['düşük'] = fuzz.trimf(proje_riski.universe, )
proje_riski['orta'] = fuzz.trimf(proje_riski.universe, )
proje_riski['yüksek'] = fuzz.trimf(proje_riski.universe, )
```

4. Kural Tabanının Yazılması:

   Proje riskini belirlemek için uzman bilgisine dayalı basit EĞER-İSE kuralları oluşturuyoruz.

   Python

```
kural1 = ctrl.Rule(butce_asimi['poor'] | zaman_asimi['poor'],
proje_riski['düşük'])
kural2 = ctrl.Rule(butce_asimi['average'] & zaman_asimi['average'],
proje_riski['orta'])
kural3 = ctrl.Rule(butce_asimi['good'] | zaman_asimi['good'],
proje_riski['yüksek'])
# Not: automf 'poor', 'average', 'good' isimlerini kullanır. Bunları
'düşük', 'orta', 'yüksek' olarak düşünebiliriz.
```

5. Kontrol Sisteminin Oluşturulması ve Simülasyon:

   Tanımlanan kuralları bir kontrol sisteminde birleştirip simülasyon için hazır hale getiriyoruz.7

   Python

```
risk_kontrol_sistemi = ctrl.ControlSystem([kural1, kural2, kural3])
risk_simulasyonu = ctrl.ControlSystemSimulation(risk_kontrol_sistemi)
```

6. Simülasyon ve Sonuçların Yorumlanması:

   Belirli girdi değerleri için sistemi çalıştırıp durulaştırılmış (defuzzified) sonucu elde ediyoruz.

   Python

```
# Örnek bir senaryo: Bütçe %65 aşılmış, zamanlama ise %20 aşılmış.
risk_simulasyonu.input['bütçe_aşımı'] = 65
risk_simulasyonu.input['zaman_aşımı'] = 20

# Sistemi çalıştır
risk_simulasyonu.compute()

# Sonucu yazdır
print("Hesaplanan Proje Riski:
{:.2f}%".format(risk_simulasyonu.output['proje_riski']))

# Sonucun nasıl hesaplandığını görselleştir
proje_riski.view(sim=risk_simulasyonu)
```

Bu kod, bütçe aşımının 'orta' ile 'yüksek' arasında, zaman aşımının ise 'düşük' ile 'orta' arasında olduğunu değerlendirecek, ilgili kuralları ateşleyecek ve Centroid yöntemiyle nihai risk skorunu hesaplayacaktır.

10.2. Pratik 2: Öğrenen Bulanık Sistem - `scikit-anfis` ile Fonksiyon Yaklaşımı

Bu pratikte, bir bulanık sistemin parametrelerinin bir veri setinden nasıl otomatik olarak öğrenilebileceğini göreceğiz. Bu, Bölüm 9'da anlatılan ANFIS teorisinin pratik uygulamasıdır.

Senaryo: Girdi-çıktı veri noktalarını kullanarak karmaşık, doğrusal olmayan bir matematiksel fonksiyonu ($y = \frac{(x)}{e^{x/}}$) modellemeye çalışacağız. Bu, ANFIS'in bir regresyon ve sistem tanımlama problemi için nasıl kullanılabileceğini gösteren klasik bir örnektir.

Adımlar ve Kod Uygulaması:

1. Kütüphanelerin Kurulumu ve İçe Aktarılması:

   Bu pratik için scikit-anfis gibi modern bir ANFIS kütüphanesine ihtiyacımız olacak. Bu kütüphane, scikit-learn ile uyumlu bir arayüz sunar.50

   Python

   ```
   # Gerekli kütüphaneleri yükleyin: pip install scikit-anfis
   import numpy as np
   import matplotlib.pyplot as plt
   from skanfis import scikit_anfis
   from sklearn.model_selection import train_test_split
   from sklearn.metrics import mean_squared_error
   ```

2. Veri Setinin Oluşturulması:

   Hedef fonksiyondan eğitim ve test için veri noktaları üretiyoruz.45

   Python

   ```
   # Veri setini oluştur
   x = np.arange(0, 10.1, 0.1).reshape(-1, 1)
   y = np.sin(2 * x) / np.exp(x / 5)

   # Veriyi eğitim ve test setlerine ayır
   X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
   random_state=42)
   ```

3. ANFIS Modelinin Oluşturulması:

scikit_anfis nesnesini, veriyi ve eğitim parametrelerini belirterek oluşturuyoruz. hybrid=True parametresi, hibrit öğrenme algoritmasını aktive eder.51

Python

```
# ANFIS modelini oluştur
# 'data' parametresi, modelin girdi uzayını ve kural sayısını otomatik
olarak belirlemesine yardımcı olur.
# 'label='r'' regresyon görevini belirtir.
model = scikit_anfis(data=X_train, description="Function Approx",
epoch=150, hybrid=True, label='r')
```

4. Modelin Eğitilmesi:

fit metodu, ANFIS'in hibrit öğrenme algoritmasını kullanarak üyelik fonksiyonu (öncül) ve kural sonucu (sonuç) parametrelerini veri setine göre optimize etmesini sağlar.

Python

```
# Modeli eğit
model.fit(X_train, y_train)

# Eğitim sürecindeki hatanın değişimini görselleştir
model.plot_errors()
plt.title("Eğitim Süreci Hata Grafiği")
plt.xlabel("Epoch")
plt.ylabel("RMSE")
plt.show()
```

5. Sonuçların Değerlendirilmesi ve Görselleştirilmesi:

Eğitilmiş modelin performansını test verisi üzerinde değerlendiriyor ve tahminlerini gerçek fonksiyonla karşılaştırarak görselleştiriyoruz.

Python

```
# Test verisi üzerinde tahmin yap
y_pred = model.predict(X_test)

# Modelin performansını değerlendir (RMSE)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Test Seti Üzerindeki RMSE Değeri: {rmse:.4f}")

# Gerçek fonksiyon ile ANFIS tahminlerini karşılaştır
plt.figure(figsize=(10, 6))
plt.plot(x, y, 'b-', label='Gerçek Fonksiyon')
plt.plot(X_test, y_pred, 'r.', label='ANFIS Tahminleri')
```

```
plt.title("ANFIS ile Fonksiyon Yaklaşımı")
plt.xlabel("Girdi (x)")
plt.ylabel("Çıktı (y)")
plt.legend()
plt.grid(True)
plt.show()

# Modelin öğrendiği üyelik fonksiyonlarını görselleştir
model.plot_mfs()
plt.title("Öğrenilmiş Üyelik Fonksiyonları")
plt.show()
```

Bu pratik, öğrencilere sadece bir kütüphaneyi nasıl kullanacaklarını öğretmekle kalmaz, aynı zamanda bir "siyah kutu" olmayan, iç yapısı (öğrenilmiş üyelik fonksiyonları) incelenebilen ve yorumlanabilen güçlü bir modelin nasıl oluşturulabileceğini somut bir şekilde gösterir.