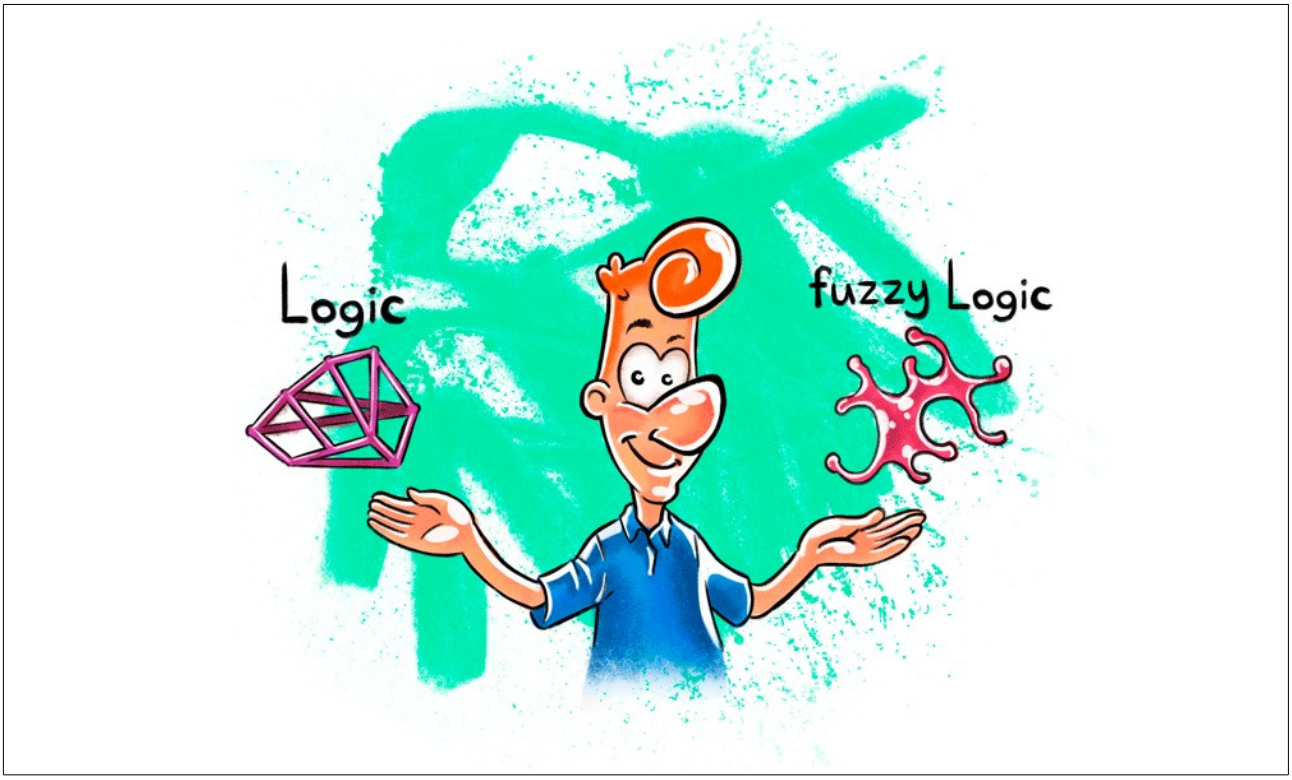
Bulanık Mantık Notları

Versiyon: 0.3



Hazırlayan: Ahmet Ataşoğlu

ahmetatasoglu98@gmail.com

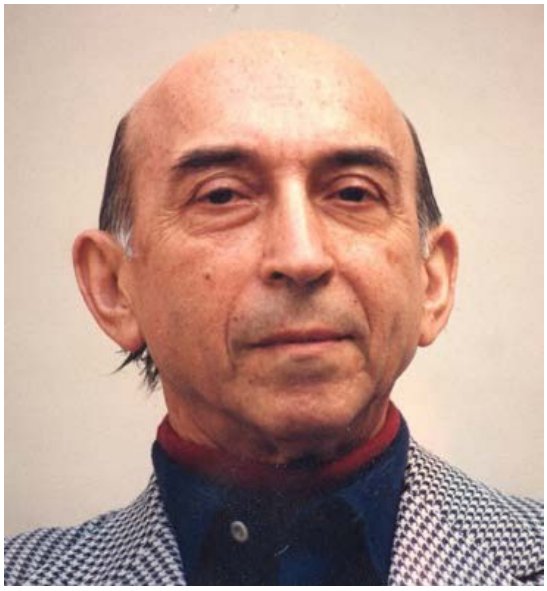
İçindekiler

1. Bulanık Mantık Nedir?

Bulanık Mantık (ing. Fuzzy Logic) ilk kez 1961 yılında Azeri kökenli matematikçi Lütfi Alasker Zade tarafından temelleri atılmış matematiksel bir kuramdır. Adında da geçtiği gibi bir 'mantık' çeşididir.

Aslında lisedeki matematik derslerinden mantık konusuna pek de yabancı sayılmayız. Özellikle de matematik derslerini sevmeyenler için yerinin daha ayrı olduğunu düşünüyorum. Matematiğin ender sözel ağırlıklı konularından biri olduğu içindir belki de. Gelgelelim "ve, veya, ise, ancak ve ancak" gibi hafızamızda yer edinen kavramları hepimiz bu derslerden anımsarız. Önerme dediğimiz cümlelerde geçen olguların, gerçeklerle örtüşüp örtüşmediğini anlamaya çalışırdık bu derslerde. Gerçeklerle örtüşen, diğer bir deyişle mantığa mugayır olmayan önermelere 'doğru' (1) derken; gerçeğin aksini söyleyenlere 'yanlış' (0) diyorduk. Az evvel bahsettiğimiz kavramlar da bu önermelerin, yani 1 ve 0'ların birbirleriyle olan ilişkilerini anlamlı hale getirmeye yarıyordu. O zamanlar bizim için garip gözükse de ve tam olarak bize ne anlatmaya çalıştığını yeterince anlamamış olsak da; bugünün bilgisayar ve elektronik teknolojilerinin altyapısı aslında tam da lisede gördüğümüz mantık konusundan ortaya çıkmıştır.

Çünkü mantık en basit anlamda; gerçek hayattaki herhangi bir problemi çözebilmek için onu belirli düşünce yöntemleriyle sayılar dünyasına nasıl aktarabileceğimizi kulağımıza fısıldar. Sonrasında çözüm için matematiğin imkanlarını kullanır ve yine mantığın bize sunduğu yöntemlerden faydalanırız. İşte Bulanık Mantık da tam olarak aynı amaç doğrultusunda çalışıyor. Fakat bir kaç farkla: biraz daha kapsamlı bir perspektifden meseleleri ele alıyor ve bu bildiğimiz klasik mantığın aksine, daha 'insansı' şekilde kurgulanmış sonuçlar üretecek şekilde yapıyor bunu. Bu yüzdendir ki günümüzde 'insansı' deyince akla ilk gelen yapay zeka başta olmak üzere; kontrol sistemleri, otomasyon, robotik ve sibernetik gibi mühendislik bilimlerinin pek çok alt dalında bulanık mantığın izlerine rastlıyoruz.



Resim 1. *Bulanık mantığın babası Lütfi A. Zade. (1921–2017)*

1.1. Bulanık Kavramı

Klasik mantığın çizdiği sınırlar içindeyken, bir önermenin yalnızca iki koşuldan birini sağlayabileceğini ön görürüz (doğru ya da yanlış). Çünkü varlığın mutlak gerçeklikle olan ilişkisi bir çeşit ‘tekillik’ durumundadır. Örneğin yaşınızın hem 18’den küçük, hem de büyük olamaması; ya da bugünün hem pazartesi hem de salı olmaması gibi. Ya pazartesidir, ya salıdır ya da ikisi de değildir. Fakat hem pazartesi hem de salı kesinlikle değildir! Objektif olarak ele alınan bütün olguların bu tekillik durumuna derinden sadakati, klasik mantığı ortaya atan Aristoteles’den bu yana matematikçilerin ve felsefecilerin epey ilgisini çekmiştir. Fakat zamanca fark edilmiş ki: bazı durumlarda önermelerin doğru ya da yanlış zeminine oturtulması pek de mümkün olamıyor. Çünkü gerçeklik dediğimiz şeyin bir kısmının insan zihninin bir ürünü olması ve bunların tam olarak ölçülebilir şeyler olmaması, işleri bu noktada biraz ‘bulandırıyor’.

Yine aynı örnek üzerinden gidelim: yaşınızın kaç olduğunu ölçebiliriz. Bunu yalnızca kimliğinize bakarak değil, çeşitli tıbbi testlerle yapmanın ve ‘kesin’ sonuçlara varmanın yolları da var üstelik. Bu sonuçlara bakarak, yaşınız hakkında yapılan ’bu kişi 18 yaşından büyüktür’ gibi bir önermeye yüzde yüz yanlış ya da yüzde yüz doğru gibi bir yorum getirmek mümkündür. Fakat bu önerme ‘bu kişi gençtir’ şeklinde ortaya atılsaydı, aynı şekilde yüzde yüz cevaplar verebilmek mümkün olabilir miydi? ‘Gençlik’ kavramının pek çok yorumu olduğu açık. Burada kastedilen fiziksel bir gençlik mi yoksa duygusal bir gençlik mi? Bunlara cevap verebilecek kadar yeterince doneye sahip olsak dahi, gençliğin insanlardaki algısının birbirinden farklı olması yine bir başka çıkmazı ortaya çıkarıyor. Çünkü objektif olması gereken önermemiz, önermeyi ortaya atan kişinin öznelinden kendini koparamamış oluyor.

Bu çıkmaz, klasik mantığın sınırlarını zorlayan bir durum. Fakat hal böyleyken, insan mantığı aynen bu şekilde çalışıyor. Gerçekliği yorumlamamız, mutlak ifadelerden ziyade belirsizlik çizgisinde ilerliyor. Örneğin birisini betimlerken; genç, yaşlı, uzun boylu, kısa boylu gibi belirsiz ama yine de bizim için bir anlam ifade eden olgulara başvuruyoruz. Aslında bizler de bir makine gibi, dış dünyayı sensörlerimizle algılıyoruz. Bu sensörler duyu organlarımız ve bunlardan gelen bilgileri sezgilerimize ve tecrübelerimize göre yorumluyoruz. Gerçek bir makinenin sensörleri, insan algılarından çok daha gelişmiş olabildiği için bunu büyük bir hassasiyetle yapabiliyor. Fakat biz bu hassasiyette olmadığımızdan, en azından doğru sonuçlara yaklaşabilmek adına, belirsiz yani ‘bulanık’ yorumlar yapıyoruz. Çünkü böylesi işimize geliyor.

İnsan mantığının doğası:

- Belirsiz çıkarımlar yoluyla dış dünyayı anlamlandırması
- Tecrübelerinden ve gözlemlerinden yola çıkarak fiziksel ve zihinsel işlemler yapabilmesi

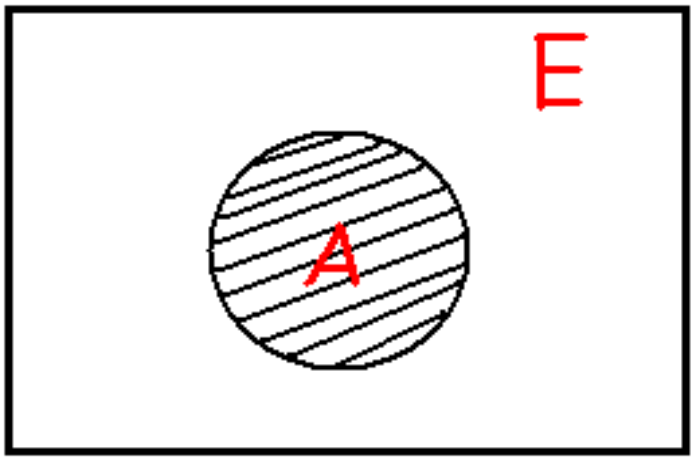
üzerine kuruludur. Bulanık mantıksa insan mantığına yaklaşmaya ve onu taklit etmeye çalışır. Dolayısıyla insan mantığının bu özellikleri, bulanık mantıkta da kendine birer karşılık bulur. Yani bulanık mantığın ‘bulanık’ kısmı, aslında insan zihninin bir yansımasıdır.

1.2. Bulanık Özellikler

Yukarıda da bahsettiğim gibi, bulanık mantığı klasik mantıktan ayıran önemli farklardan biri ‘belirsizliktir’. Geleneksel mantık diyebileceğimiz Aristo mantığında belirsizliğin yeri yoktur. Belirsizlikler genel olarak paradokslara yol açtığı ve göreceli dayanaklara sahip olduklarından, doğru ve yanlışın ötesi burada çalışmaz.

Aristo mantığı önermelerin doğruluk değerlerini yalnızca {0, 1} değerleriyle ifade eder. Önermelerin yada elemanların alabileceği bu iki değer, o önermenin/elemanın üyelik derecesidir. Eğer bir eleman, herhangi bir A kümesine dahil olmak (diğer deyişle kümenin üyesi olmak) için gereken özellikleri taşıyorsa, onun A kümesinin bir elemanı olduğu söyleriz. Bu durumda söz konusu eleman için:

- A kümesine olan üyelik derecesi ‘1’ iken,
- A kümesi dışında kalan bütün kümeleri kapsayan evrensel E kümesi için üyelik derecesi ‘0’dır. (Şekil 1)

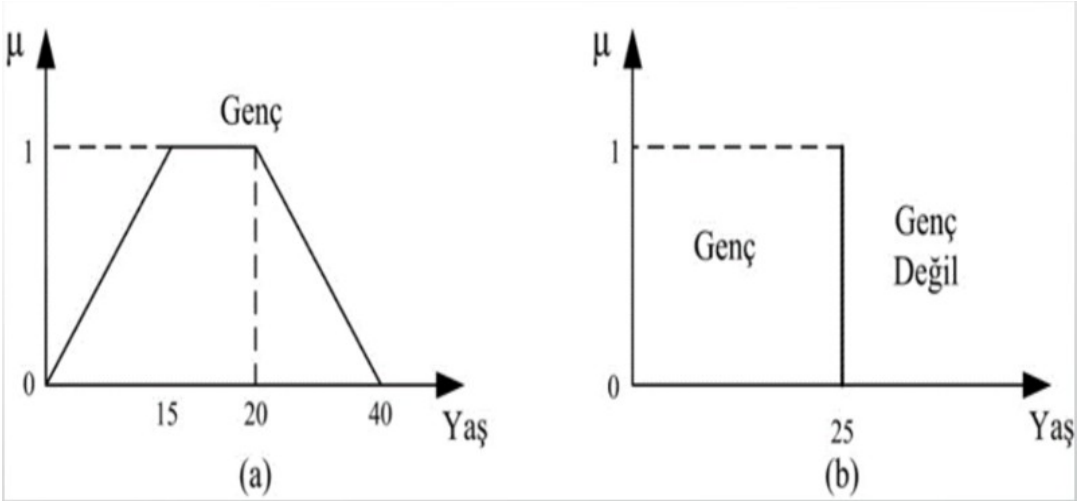


Şekil 1. *A kümesi ve onun dışında kalan E-evrensel kümesi. Bu tür kümeler kesin sonuçlara işaret ettiğinden 'Kesin kümeler' (ing. crisp sets) olarak adlandırılır.*

Bulanık mantığa geldiğimizdeyse, üyelik derecelerini belirlemek için, deyim yerindeyse, daha cömert bir yola başvuruyoruz.

Bulanık mantıkta geçen her bir küme bulanık küme olarak adlandırılır. Ve her bir elemanın bulanık kümelere olan üyelik dereceleri yalnızca 0 ve 1 sayılarını değil, bu iki sayı arasındaki bütün reel sayı değerlerini alabilir. (0 ve 1 de dahil) Yani, bulanık mantıkta üyelik derecelerinin alabileceği değerler sonsuzdur.

Şekil 2.de, bahsettiğimiz örnekte olduğu gibi ‘genç’ dilsel değerinin bulanık ve kesin kümeler açısından karşılaştırılmasına göz atalım.



Şekil 2. *Bulanık (solda) ve Kesin (sağda) kümelerin karşılaştırılması.*

- Grafiklerin x-ekseni yaş değerini gösterirken; y-ekseni yaş değerlerinin ‘genç kümesine’ olan üyelik derecesini gösterir.
- Sağdaki kesin küme grafiğine göre, bir kişinin genç sayılabilmesi için onun ancak 25 yaşından küçük olması gerekir. 26 yaşındaki bir insan için, gençliği ellerinden çoktan kayıp gitmiştir.
- Soldaki bulanık küme grafiği ise; 0-15 yaşları arasında gençliğin orantılı şekilde arttığını, 15-20 yaşlarında insanın artık tamamen genç sayıldığını ve 20-40 yaşları arasında da yine orantılı olarak gençliğin azaldığını söylemektedir.

Gerçekte 25 ile 26 yaşlarındaki insanların arasında neredeyse hiçbir fiziksel fark olmamasına rağmen, klasik mantık 25 yaşının üstündeki insanların hiçbirini genç olarak görmüyor. Bulanık mantık içinse; 20 yaşındaki bir birey genç iken (üyelik derecesi 1), 21 yaşındaki başka bir birey daha az genç oluyor. (üyelik derecesi 1'e çok yakın bir değer) Görüldüğü gibi bulanık mantıktaki üyeliklerin derecelendirilmesi, klasik mantıktaki yöntemlere nazaran akla ve gerçek dünyaya daha yatkın gelmektedir.

1.3. Neden Bulanık Mantık?

Bulanık mantığı karşılaştığımız problemleri çözebilmek için kullanıyoruz. Tıpkı diğer tüm matematiksel araçlar gibi. Fakat onu diğerlerinden daha cazip kılan bazı özellikleri mevcut Bunlardan bazıları:

- İnsan düşünce sistemi sayısal değil sözel bilgileri işlemeye odaklıdır. Başka bir deyişle ‘insan zihni, kelimelerle işlem yapar’**.** İfade etmeye çalıştığı şeyler için dilsel değerleri kullanır. Bulanık mantık da benzer şekilde çalıştığı için, insan mantığına yaklaşık sonuçlar üretebilir.
- Gerçek dünyadaki problemleri çözebilmek için gerekli sayısal verilere ulaşmak, genellikle zahmetli ve maliyetlidir. Elimizde yeterli bilgilerin bulunmadığı bu durumlarda bulanık mantık; insanın değer yargılarını, düşünme ve karar verme gücünü kullanarak, kesin olmayan durumlarda bile makul sonuçlar üretebilir.
- Bulanık sistemi tasarlayan uzman kişilerin, sistem hakkında sayısal olmayan veya matematiksel olarak modellenemeyen tecrübeleri olabilir. Örneğin: öğrencilerini yeteri kadar iyi tanıyan bir öğretmenin, sınavın zorluk seviyesine göre öğrencilerin alabileceği notları tahmin edebilmesi gibi. Her bir öğrenci için gerçeğe çok yaklaşık tahminler yapılabilir fakat bu tahminleri formülize etmek epey zordur. Sayısal bilgilere ulaşılamadığı bu gibi durumlarda; tecrübelerin bir makine yada bilgisayara aktarılabilmesi, bulanık mantıkla mümkündür.
- İnsan zihnine yakın çalışabildiği için, pek çok kontrol yöntemine göre anlaşılır seviyededir. Arka planda çalışan teorileri ve matematiği oluşturmak zor değildir.
- Esnek yapısı sayesinde kesin olmayan olasılık durumlarında; karmaşık ve doğrusal olmayan fonksiyonlarla ve sürekli değerlerle -yani, sadece belirli bir aralıkta olduğunu bildiğimiz değerlerle- bile uyumlu şekilde çalışabilir.
- Günlük hayattaki konuşma diliyle bulanık bir sistem modellenebilir. Bu açıdan bulanık sistemleri geliştirirken, programlama ve sistem tasarımı konularında teknik bilgiye sahip olmayan insanlarla bile etkili şekilde çalışmaya imkan tanır.
- Bulanık mantık klasik mantığı da kapsadığı için, klasik mantıkla oluşturulan her sistem bulanık mantıkla da oluşturulabilir.

Sonuç olarak bulanık mantık; klasik mantığın kapsamından daha öteye gidebilmiş, insanın gerçekte sahip olduğu ve kullandığı mantık modelini bilgisayarlar ve makineler üzerinde inşa etmeye çalışan bir mantık yapısıdır.

2. Bulanık Kümeler

Bulanık kümelere geçmeden, konuyu biraz daha detaylandırmak ve aralarındaki farkları daha iyi görebilmek adına, önceki yazıda da bahsettiğim klasik mantığı oluşturan küme modelinden biraz daha bahsetmemiz faydalı olacaktır.

2.1. Kesin Kümeler

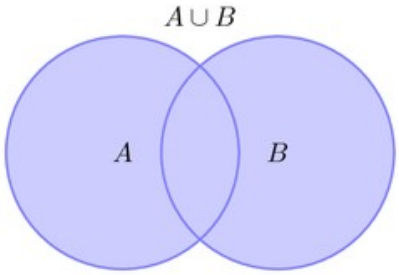
Bir keskin küme (ing. crisp set), ya tam üye ya da hiç üye olmayan elemanlardan oluşan bir küme olarak tanımlanabilir. Bir kümeyi oluşturan her bir elemanın, o kümeyle olan ilişkisinin üyelik derecesi denilen bir değer ile belirtildiğinden bahsetmiştik. Bu durumda, herhangi bir A keskin

kümesi ve bu kümenin de dahil olduğu evrensel küme elemanlarının üyelik dereceleri hakkında yorum yapmak gerekirse:

- A kümesine dahil olan elemanların üyelik derecesi 1,
- A kümesine dahil olmayan elemanların üyelik derecesi 0'dır.

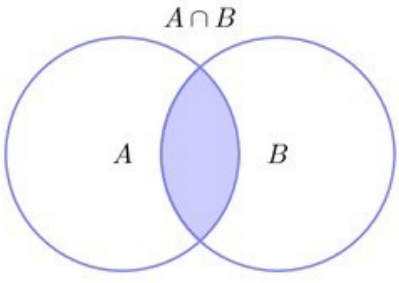
Herhangi iki veya daha fazla keskin kümenin birbirleriyle olan ilişkilerinden bahsederken (örneğin A ve B kümeleri), aşağıdaki kavramlar üzerinde dururuz:

- A veya B kümesine üyelik derecesi 1 olan bütün elemanların bulunduğu küme, bu iki kümenin birleşim kümesidir. A∪B şeklinde gösterilir. (Şekil 2.A)



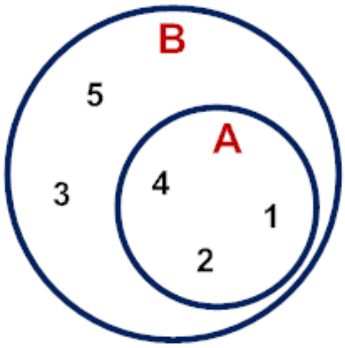
Şekil 2.A. *Birleşim kümesi.*

- Hem A hem de B kümesine olan üyelik derecesi 1 olan elemanların -yani ortak elemanların- bulunduğu küme, bu iki kümenin kesişim kümesidir. A∩B şeklinde gösterilir. (Şekil 2.B)



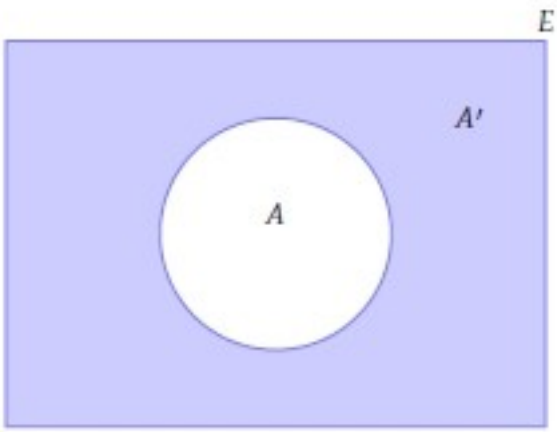
Şekil 2.B. *Kesişim kümesi.*

- Bu kümelerden herhangi birinin bütün elemanları, aynı zamanda diğer kümenin de elemanlarının tamamını veya bir kısmını oluşturuyorsa, bu kümeye alt küme denir. A ⊂ B şeklinde gösterilir. (Şekil 2.C)



Şekil 2.C. *Alt küme*
kaynak

- A kümesine olan üyelik derecesi 0 olan tüm elemanların (A kümenin dışında kalan elemanların) bulunduğu bir B kümesi varsa, B kümesi için A'nın tümleyenidir denir. A'nın tümleyeni A' şeklinde gösterilir. (Şekil 2.D)



Şekil 2.D. *A'nın tümleyeni.*

Bu kavramlara değinmemin sebebi, birazdan bahsedeceğim bulanık kümelerde de, buradaki sisteme oldukça benzer ifade şekillerinin olmasıdır.

Son olarak keskin kümelerin bize hangi noktalarda açık kapı bırakmadığına değinmemiz gerekiyor. Önceki yazıda klasik mantık kurallarının, bir olguyu ifade etmede son derece kesin ve objektif yargılara başvurduğundan bahsetmiştik. Bir önermenin doğruluğu ya da yanlışlığı tartışma götürmez temellere dayanmak zorundaydı. Fakat gerçek hayattaki düşünce sistemimiz, kesin ölçütler yerine bulanık sezgilere ve kesin olmayan gerçekliğe göre şekillenmekte idi. Örneğin, dünyadaki bütün araba markalarını listeleyerek bir keskin küme modeli oluşturmaya çalıştığımızı düşünelim. Buradaki önermemiz:

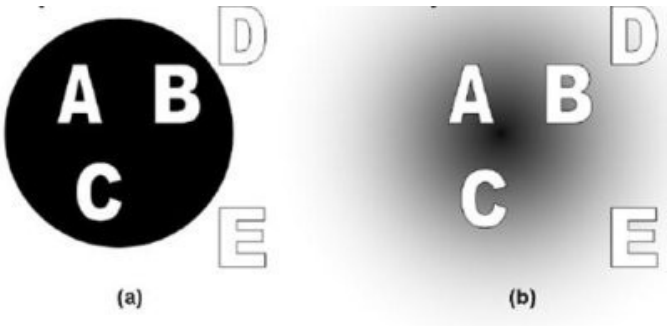
"Eğer X araba markası, Amerikan malı ise A kümesinin bir elemanıdır."

şeklinde olsun. Bu durumda, X elemanına karşılık gelen her bir araba markasının menşesini sorguladığımız zaman, karşımıza iki durumdan birinin çıkmasını bekleriz; bu marka ya Amerikan malıdır (1), ya da Amerikan malı değildir (0). Bir arabanın Amerikan malı sayılıp sayılmaması konusunda da tartışmaya mahal vermemek adına; o markanın Amerika'da kurulmuş bir marka olması, onu Amerikan markası yapmak için yeterli bir sebep olarak görülmüş olsun. Buraya kadar bir problem yok gibi. Fakat, kurguladığımız modelden kafamızı kaldırarak günümüzün gerçekliğinde bu önermeyi yorumlamak istersek, karşımıza başka bir tablo çıkacaktır. Burada biraz durup düşündüğümüzde şunu söyleyebiliriz: bir araba markasının Amerika'da kurulmuş olması, onun tamamen Amerikan malı sayılması için yeterli olmayabilir. Çünkü marka, üretimini Amerika dışında başka bir ülkede yapıyor olabilir. Veya Amerika'da üretim yapıyor olsa bile, ürettiği arabaların önemli parçalarının bir kısmı diğer ülkelerden ithal ediliyor olabilir. Bu detaycı perspektif, markaların tamamen Amerikan malı sayılıp sayılamayacağı konusunda mantığımızı şüpheye düşürmektedir. Klasik mantığın bizi zorladığı şekilde bu markaları sınıflandırmak için, elemanların (markaların) A kümesine aidiyeti hakkında yüzde yüz kesinlikte hükümler verebilmemiz gerekirdi. Fakat yaptığımız bu düşünce deneyinden çıkan sonuç; bahsettiğimiz 'Amerikan malı' kavramının aslında kesin bir sınırın olmadığıdır. Bir araba pek çok açıdan Amerikan malı sayılabilir; yine pek çok açıdan sayılmayabilir.

Geleneksel küme kuramına göre, bir keskin kümenin çok iyi tanımlanmış özelliklere sahip olması gereklidir. Bahsettiğimiz örnekteki küme tanımlaması, bu açıdan hala doldurulması gereken boşluklara sahip, iyi tanımlanmamış veya tanımlanması mümkün olmayan bir kümedir. Geleneksel küme kuramının çıkmaza düştüğü bu noktada durumun üstesinden gelebilmek için, imdadımıza şimdi bahsedeceğimiz bulanık küme kuramı yetişiyor.

2.2. Bulanık Kümeler

Bulanık kümeler (ing. fuzzy sets), keskin kümelerden farklı şekilde küme elemanlarının kısmi üyeliğine de izin verir. Bulanık kümeleri oluşturan elemanların alabileceği üyelik dereceleri [0, 1] kapalı aralığındaki bütün reel sayılardır. Bir elemanın A bulanık kümesine olan üyelik derecesini belirlerken, keskin kümelerde olduğu gibi 2 seçeneğe değil, çok daha fazlasına sahip oluruz. Öyle ki, 0 ve 1 aralığında sonsuz reel sayı bulunduğu için, teorik olarak bir elemanın alabileceği sonsuz farklı üyelik derecesi vardır diyebiliriz.



Şekil 3. *Kesin sınırlara sahip bir keskin küme(a), kesin olmayan sınırlara sahip bir bulanık küme(b)*

Bulanık kümelerde, her bir eleman aynı anda birden fazla kümeye ait olabilir. Fakat buradaki ‘aidiyet’ kavramını, keskin kümelerdeki haliyle karıştırmamak gerekir. Aynı önermenin farklı sonuçlarını temsil eden kümeler için (yukarıdaki örnekte, Amerikan malı olan araçları tutan küme ve Amerikan malı olmayan araçları tutan küme) her bir eleman kesinlikle bu kümelerden yalnız birine ait olmak zorundaydı.

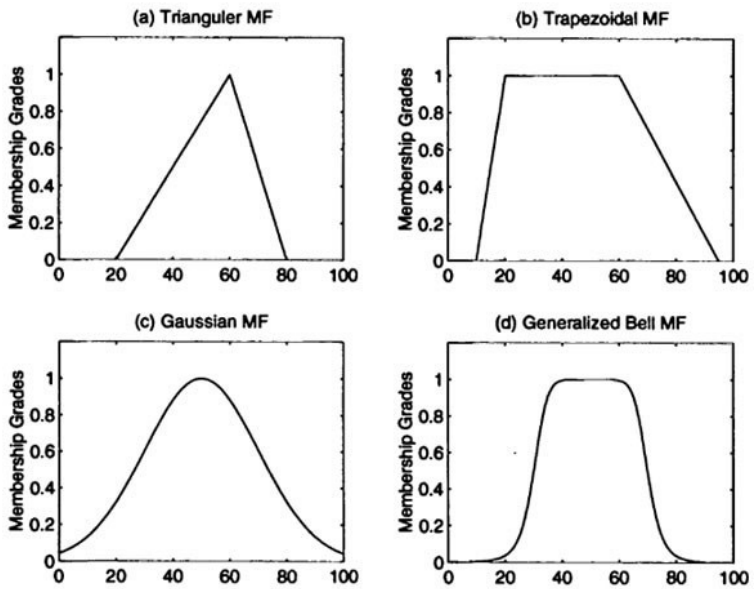
Fakat bulanık kümelerde, önermeyi oluşturan kavramların göreceli ve sezgisel çıkarımlardan oluşmuş olmasından dolayı, her bir eleman birbirine zıt gözüken her iki kümeye de aynı anda ait olabilir. İşte tam burada aitlik kavramının keskin kümelerdeki haliyle, bulanık kümelerdeki hali yollarını ayırır.

Bulanık kümelerde, her bir elemanın kümelere olan aidiyetleri belirli kurallar baz alınarak derecelendirilir. Keskin kümelerde kümenin bütün elemanları tamamen aynı üyelik derecesine sahipken (1) ; bulanık kümelerde ise aynı kümenin elemanları, birbirlerinden bambaşka üyelik derecelerine sahip olabilirler. Çünkü her bir elemanın o kümeye olan ‘aidiyeti’ kendine özeldir. 1’e çok yakın değerler, elemanın o kümeye yüksek dereceden bir aidiyeti olduğunu belirtir. Aynı şekilde 0’a yakın değerler, düşük dereceli bir aidiyetin göstergesidir.

2.3. Üyelik Derecelerinin Belirlenmesi

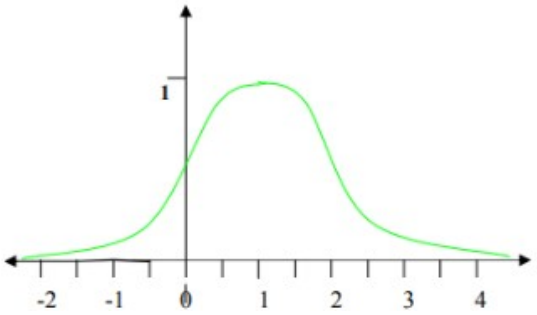
Her bir elemanın kendine özel bir üyelik derecesi ile ifade edildiğinden bahsettik. Peki bu üyelik derecelerini nasıl belirleyebiliriz, bundan bahsedelim.

Herhangi bir değer aralığındaki elemanlar için, bir kümeye hangi dereceden ait olduklarını gösteren fonksiyonlara üyelik fonksiyonları (ing. membership function) denir. µF şeklinde belirtilir. Bu fonksiyonları ifade etmek içinse genelde üçgen, yamuk, çan eğrisi, üstel, gausyen vb. foknsiyonlar kullanılır. (Şekil 4)



Şekil 4. *Üyelik fonksiyonları. Üyelik derecelerinin her koşulda [0, 1] aralığında değerler aldığına dikkat edin.*

Diyelim ki ‘1 sayısına yaklaşan tam sayılar’a ait bir bulanık küme tanımlamak istiyoruz (F kümesi). Bunun için, aşağıdaki gibi (Şekil 5) çan eğrisi şeklinde üyelik fonksiyonu belirlenmiş olsun.



Şekil 5. *1'e yaklaşan reel sayılara ait üyelik fonksiyonu, μF(ai)*

Tanım kümemizin ise aşağıdaki değerlerden oluştuğunu varsayalım:

A tanım kümesi için, A = {-2, -1, 0, 1, 2, 3, 4}

Her bir A elemanı *(ai)* için, üyelik fonksiyonundan alınan değerler şu şekilde olsun:

- *μF(a1) = μF(-2) = 0*
- *μF(a2) = μF(-1) = 0.3*
- *μF(a3) = μF(0) = 0.6*
- *μF(a4) = μF(1) = 1*
- *μF(a5) = μF(2) = 0.6*
- *μF(a6) = μF(3) = 0.3*
- *μF(a7) = μF(4) = 0*

Görüldüğü gibi, 1 sayısına pozitif veya negatif taraftan yaklaşan tam sayıların üyelik dereceleri artmaktadır. Uzaklaşıldığında ise üyelik derecelerinin azaldığı gözlemlenebilir. Son olarak, F bulanık kümensini oluşturan elemanların, üyelik dereceleriyle beraber ifade edilmesi şu şekilde olur:

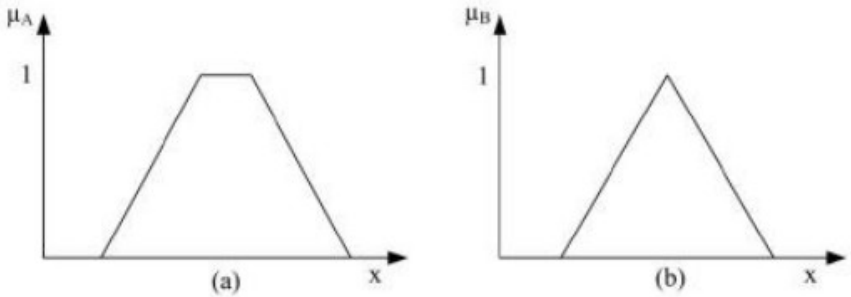
F = {(-2, 0), (-1, 0.3), (0, 0.6), (1, 1), (2, 0.6), (3, 0.3), (4, 0)}

Bulanık kümelerin tanımlanması ve üyelik fonksiyonlarıın oluşturulması basitçe bu şekildedir. Kümelerin ayrık ya da sürekli olması durumuna göre ifadeler ve gösterimler değişebilir.

2.4. Bulanık Küme İlişkileri

Bulanık kümelerin birbiriyle olan bağlantıları, keskin kümelere oldukça benzer özellikler gösterir. Keskin kümelerde bahsettiğimiz birleşim, kesişim, alt küme ve tümleyen özellikleri kavramsal olarak bulanık mantıkta da kendine yer bulur.

Yukarıdaki 1'e yaklaşık tam sayılar örneğinden devam ederek ilerleyelim.



Şekil 6. *(a) Abulanık kümesi, (b) B bulanık kümesi*

2.4.1 Birleşim

Diyelim ki, üyelik fonksiyonları birbirlerinden farklı iki bulanık kümeye sahibiz (A ve B kümesi, Şekil 6). Bir küme 1 sayısına olan yakınlığı irdeliyorken, diğeri başka bir tam sayıya olan yakınlığı irdelemekte. Böylece aynı tanım kümesi elemanları için, her bir bulanık kümede alacakları üyelik dereceleri farklı olacaktır. Bu durumda A ve B kümelerinin birleşimi:

$$A \cup B(t) = max[A(t), B(t)] = A(t)\ OR\ B(t)$$

şeklinde gösterilir. Her bir tanım kümesi elemanı için, A ve B kümelerindeki üyelik derecelerine bakılır. Hangi değer daha büyükse, o değer seçilir ve birleşim kümesine yazılır. (Şekil 7.d) Daha büyüğünü seçme işlemi OR (VEYA) mantıksal ifadesi ile gösterilir.

2.4.2. Kesişim

A ve B bulanık kümelerinin kesişimi ise, bu iki kümeye ait değerler arasından en düşük üyelik dereceli elemanların seçilmesi ve kesişim kümesine yazılması ile gösterilir. (Şekil 7.c)

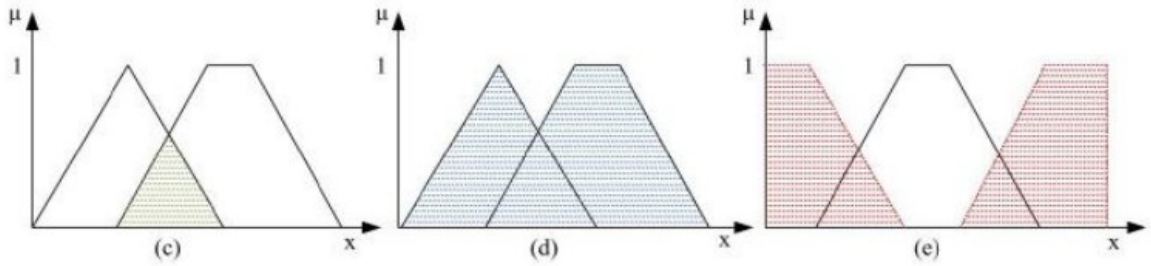
$$A \cap B(t) = min[A(t), B(t)] = A(t)\ AND\ B(t)$$

Bu işlem, diğer bir mantıksal ifade olan AND (VE) ile gösterilir.

2.4.3. Tümleyen

A kümesinin dışında kalan tüm elemanları belirten tümleyen kümesi için, her bir A elemanının üyelik derecesi 1'den çıkartılarak yazılır. (Şekil 7.e)

$$A'(t) = 1 - A(t)$$



Şekil 7. c. bulanık kesişim kümesi, d) bulanık birleşim kümesi, e) bulanık tümleyen kümes

3. Üyelik Fonksiyonları

3.1. Sözel Değişkenler

Yukarıdaki kısımlarda klasik mantık ile bulanık mantık arasındaki farklardan epey bahsettik. Herhangi bir önermenin içerdiği objektif ya da subjektif çıkarımlara bakarak, bu iki mantık modelinden hangisine uyum sağladığına karar verebiliyoruz. Örneğin:

"Bugün hava 27 santigrat derece."

önermesinde geçen havanın sıcaklığı, önermeyi ortaya kişiden bağımsız olarak ölçülüp test edilebilir. Bunun sonucunda sıcaklık ya 27 santigrat derece olarak görülür ya da görülmez. Önermemiz, iki gerçeklikten ancak birine işaret ettiği için klasik mantık sınırları içinde anlaşılabilir bir önerme olarak kendine yer bulur.

Ancak önermemiz şu şekilde, somut verilerden yoksun olsaydı:

"Bugün hava biraz sıcak."

klasik mantığın keskin gerçeklik ilkesine uymayan bir önerme olurdu. Zira havanın 'biraz' sıcak olmasının, tam olarak hangi sıcaklık değerini ifade ettiği belirsizdir (bulanıktır). Bu gibi bulanık değerler de, her ne kadar gözlemci bir için bir anlam ifade etse de, klasik mantık perspektifiyle yorumlanamıyor. Fakat bulanık mantığın kapsamına burada dahil oluyoruz.

Önermede de geçen 'biraz' ifadesinde olduğu gibi; görünürde anlamsız fakat önermeyi ortaya atanlar için yine de bir anlam ifade eden "az, fazla, biraz, çok, daha" gibi değerlere sözel değişkenler veya dilsel değerler denir. Yine de bir anlamı vardır diyoruz fakat bu anlamın neye göre; ne kadar ve nasıl bir ilişki içinde olduğunu belirlemek gerekiyor. Sonuç olarak bu bilgilerin bilgisayar ve makinelerce işlenebilmesi için sayısal bir zemine oturtulması gerekli.

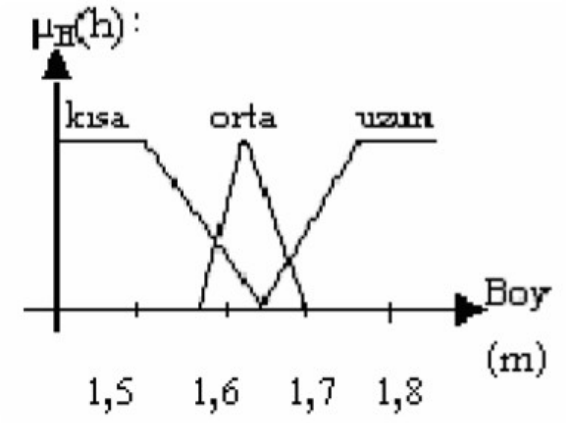
Sözel değişkenleri ifade ederken, önceki konuda bahsettiğimiz bulanık kümeleri kullanıyoruz. Herhangi bir değer aralığı için; sözel değişkenin karşılığı olan sayısal ifade, o değişken için oluşturulmuş üyelik fonksiyonu ile kolayca hesaplanabiliyor. Fakat burada dikkat çekmek istediğim nokta; sözel değişkenlerin üyelik fonksiyonlarının ne şekilde olacağı, tamamen bulanık sistemi tasarlayan uzmanın bilgilerine ve sistem hakkındaki tespitlerine bağlıdır. Nihayetinde sözel değişkenlerin göreceli olması dolayısıyla, 'biraz' gibi bir ifadenin de kişiden kişiye değişiklik gösterdiği; herkesteki 'biraz' algısının farklı olabildiği bir gerçek.

Örnek olarak herhangi bir gruptaki insanları boylarına göre sınıflandıran basit bir bulanık sistem tasarlayalım. Boyları ifade etmek için kullanılacak dilsel değerlerimiz şu şekilde olsun:

kısa, orta, uzun

Bu durumda tanım kümemiz; ölçülen boy değerlerinin metre biriminden karşılıklarını tutacaktır. Bunun için [0, 2.5] aralığını belirleyebiliriz.

Dilsel değerlerin tanım kümesiyle olan ilişkisinin üyelik fonksiyonlarıyla kurulacağından bahsetmiştik. Şimdilik, buradaki detayların üzerinde durmadan, aşağıdaki şekilde (Şekil 8) olduğu gibi dilsel değerleri basitçe üyelik fonksiyonlarıyla eşleştirelim:



Şekil 8. *Buradaki yatay eksen {0, 2.5] aralığındaki tanım kümesini, dikey eksen ise [0, 1] aralığındaki üyelik değerlerini temsil etmektedir.*

- [0 m, 1.65 m] aralığında ölçülen bir boya sahip insanlar ‘kısa’ dilsel değeriyle,
- [1.6 m, 1.7 m] aralığında ölçülen boya sahip insanlar ‘orta’ dilsel değeriyle,
- [1.65 m, 2.5 m] aralığında ölçülen boya sahip insanlar da ‘uzun’ dilsel değeriyle, şekildeki üçgen üyelik fonksiyonları kullanılarak eşleştirilmiş olsun.

Artık buradaki grafik üzerinden, bu sistemi tasarlayan kişi için kısa-orta-uzun boylu insanların aslında tam olarak hangi aralıkta boylara sahip olabileceği yorumlanabilir. Aynı şekilde, örneğin boyu 1.62 m olan biri için, ne kadar kısa ve ne kadar orta boylu olduğu yine bu grafik üzerinden ölçülebilir. Fakat buradan sonrası, önümüzdeki konuların detayları olduğu için şimdilik değinmiyorum.

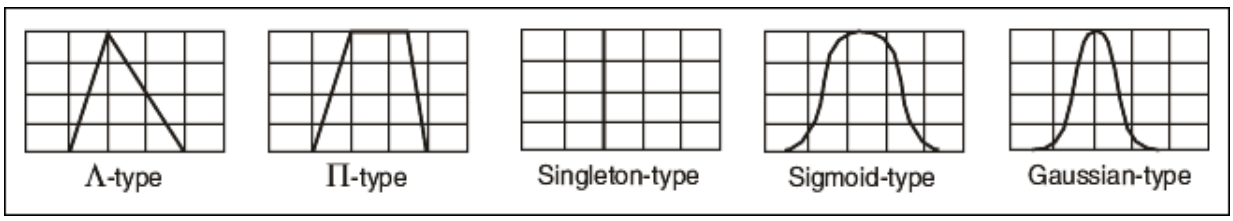
Yine belirtmekte fayda var: buradaki dilsel değerlerin söz konusu aralıklara hizalanması tamamen öznel bir yorumdur. Herkes için buradaki değerlerin aynı şekilde belirlenmesi tabi ki mümkün olmayacaktır. Zaten bulanık mantığın özü de burada saklıdır: öznel yorumlardan bile somut veriler çıkarabilir. Dolayısıyla, sizin bu noktada getirdiğiniz yorum bambaşka olacaktır. Basit bir deney olarak siz de; buradaki örnek üzerinden giderek, kendi algılarınız ve tecrübelerinize göre kısa-orta-uzun boylu insanların (sizin açınızdan) hangi aralıklardaki uzunlukları ifade ettiğinin grafiğini çizmeyi deneyebilirsiniz.

3.2. Üyelik Fonksiyonlarının Belirlenmesi

Bulanık bir sistemin tasarlanma aşamasında geçen küme ve değişken kavramına değindik. Şimdi de değişkenlerin, bulanık kümelerle olan ilişkilerini matematiksel olarak ifade etmek için kullandığımız üyelik fonksiyonu kavramına göz atma vakti.

Önceki yazıda da bu kavrama kısaca değinmiştik. Herhangi bir tanım aralığında bulunan bir değerin, sözel değişkenlerle oluşturulan bulanık kümelere olan aidiyetini bu fonksiyonlarla belirliyorduk. Örneğin yukarıdaki örnekte; boyu 1.62 m olan biri için, kısa ve orta bulanık kümelerine aynı anda dahil olduğunu söyledik. Her bir boy değeri için, dahil olduğu bulanık kümeye ‘ne kadar ait’ olduğunu ifade eden başka bir değerin olması gerekir. Bu değere üyelik derecesi deniyor ve bu değer, [0, 1] kapalı aralığındaki bir reel sayıya eşit oluyor. İşte üyelik fonksiyonları tam olarak: üyelik derecesinin, hangi reel sayıya eşit olduğunu söyleyen fonksiyonlardır.

Üyelik fonksiyonunun tanım kümesi, bütün reel sayılar kümesidir. Değer kümesi ise yukarı da da bahsettiğimiz gibi [0, 1] aralığındaki reel sayılar kümesidir. Üyelik fonksiyonlarını ifade etmek için pek farklı fonksiyon türlerinden biri kullanılabilir. Genellikle karşımıza çıkan üçgen, yamuk, gausyen, polinomal vb. fonksiyonlar, üyelik fonksiyonlarını ifade etmede çok defa kullanılmış fonksiyonlardır. (Şekil 9)



Şekil 9. *Çeşitlik üyelik fonksiyonları*

Tabi yazımız boyunca sık sık tekrar ettiğimiz gibi; bir problemin çözümünde kullanılacak fonksiyonun seçilmesi, sorunun özüne ve elimizdeki bilgilere/tecrübelere bağlıdır. Örneğin; piyasaya yeni sürülecek bir ürünün potansiyel müşterilerini belirlemek amacıyla bir araştırma yaptığınızı düşünün. Bunun için elinizde önceden yapılan satışlar ve alıcıların belirli kriterlerini tutan (kalite ve fiyat beklentisi, gelir durumu gibi) bir veri tabanınız olsun. Aslında bu noktada da bulanık mantığın dışında kullanabileceğimiz pek çok makine öğrenmesi ya da istatistiksel analiz yöntemi bulunuyor. Fakat bu örneği bulanık mantığa daha yatkın hale getirebilmek adına elimizdeki verilerin: az kaliteli, çok kaliteli, orta fiyatlı, yüksek fiyatlı, az kazanan, çok kazanan gibi sayısal verilerden ziyade sözel değişkenlerden oluştuğunu düşünelim. Verileri piyasa konusunda uzman pazarlamacılar veya yöneticilerle değerlendirdiğinizde; bu verilerin nasıl bir ilişki içerisinde olabileceğinin genel hatları ortaya çıkacaktır. Bunlar; ürün fiyatının artması durumunda kalite beklentisinin de aynı şekilde artacağı (1), ürün fiyatının artması durumunda gelir düzeyi görece azalan kitle tarafından daha az erişilebilir bir ürün olacağı (2) yönünde tespitler olabilir.

Burada, ürünün fiyatıyla sözel değişkenler arasındaki ilişki sezgisel anlamda ortaya çıkmış durumda. Yapılması gereken üyelik fonksiyonlarını oluşturarak, sistemi artık sayısal bir hale

getirmek. 1 no'lu öngörüde belirtilen durumda; kalite beklentisiyle fiyat arasında doğrusal bir ilişkiden bahsedilebilir. Fakat 2 no'lu öngörüde, doğrusal yerine polinomal bir ilişki söz konusu olabilir. Zira gelir durumu nispeten daha az olan insanlar bile, ürün için kayda değer bir talep içerisinde olabilirler.

Örnekte de bahsetmek istediğim gibi: üyelik fonksiyonlarının oluşturulması, sistemle ilgili verilere ve sistemi tasarlayan uzmanların öngörülerine göre değişiklik gösterir.

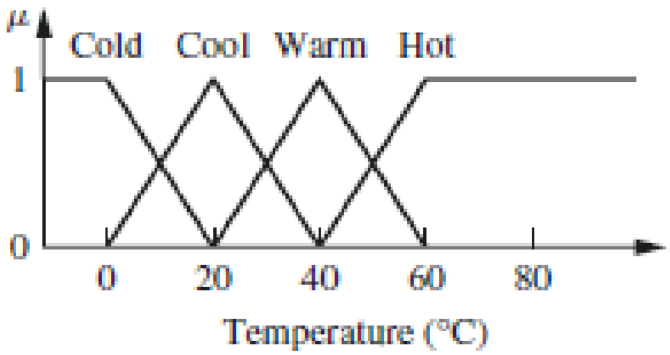
Günümüzde, gelişen yapay zeka araçları sayesinde, üyelik fonksiyonlarını belirlemede çeşitli optimizasyon algoritmalarını kullanabiliyoruz. Bu şekilde, sistemle ilgili öngörülerin dışarıdan hazır olarak dahil edilmesi yerine; verilerin bu algoritmalarla işlenmesi sonucu üyelik fonksiyonları en iyi değerlere ulaşabileceğimiz şekilde optimize edilebiliyor. Bu şekilde, sistem hakkında uzman kişilerle çalışma yükümlülüğü de bilfiil kalkmış oluyor. Bu konu de epey ilgi çekici. Şimdilik burada virgül koyalım. İlerleyen yazılarda değineceğiz.

Üyelik fonksiyonlarını oluşturmada genel olarak kullanılan yöntemler şu şekildedir:

1. *Sezgisel*
2. *Çıkarımsal*
3. *Sıralama*
4. *Optimizasyon Yöntemleri*

3.2.1. Sezgisel

İnsan sezgilerini kullanarak üyelik fonksiyonlarının belirlenmesi; aslında en başından beri verdiğimiz örneklerde geçen yöntemin ta kendisidir. Burada, sistem hakkında bilgi ve tecrübe sahibi bir kişiye yada gruba danışarak üyelik fonksiyonları belirlenir. Genellikle sayısal verilerin yetersiz olduğu veya üzerinde çalışılan sistemi modellemeye yetmediği durumlarda, sezgisel yaklaşımlara güvenerek üyelik fonksiyonlarını oluşturabiliriz. Aşağıdaki şekilde (Şekil 10) olduğu gibi, dilsel değerlerle ifade edilen hava sıcaklıklarının hangi değerlere karşılık geldiğini gösteren bir örnek; soğuk ve sıcak ortamları yeteri kadar tecrübe etmiş bir kimse tarafından, sezgilerini kullanmak suretiyle, kolaylıkla oluşturulabilir.



Şekil 10. *Soğuk, serin, sıcak, hafif sıcak, sıcak gibi dilsel değerlerin üçgen fonksiyonlarla ifade edilmesi.*

3.2.2. Çıkarımsal

Çıkarım yönteminde ise: tümdengelim gibi sistematik akıl yürütme pratiklerini kullanarak, sezgilerimize nazaran daha rasyonel biçimde oluşturulmuş fonksiyonlara ulaşmaya çalışırız.

Örneğin, iç açıları (A, B ve C) arasındaki bağıntı şu şekilde olan bir üçgen tanımlanmış olsun:

$$\boldsymbol{A \geq B \geq C}$$

Bu üçgenin ait olabileceği üçgen sınıfları şöyledir:

- Dik üçgen
- İkizkenar üçgen
- Eşkenar üçgen
- Diğer üçgenler

Bu sınıfları bulanık kümeler olarak düşünürsek; yukarıda tanımladığımız üçgenin, her bir üçgen kümesine üyeliğini belirten üyelik fonksiyonunu kolaylıkla oluşturabiliriz. Burada bir uzman görüşü yerine, geometri hakkında yeteri kadar temel bilgilere sahip olmamız dolayısıyla; ortaya atılan önermelerin sistemi tanımlamada herhangi hata ya da eksiklik taşımadığını biliyoruz. (İç açılarının 180 derece olması gibi olmazsa olmaz kıstaslara riayet ediyoruz zira) Eğer:

$$\boldsymbol{A = B \text{ veya } B = C}$$

gibi bir durum söz konusuysa, bu üçgenin ikizkenar üçgen kümesine olan üyelik fonksiyonunu şu şekilde (Şekil 11) oluşturabiliriz:

$$1 - \frac{1}{60^\circ} \min(A - B, B - C)$$

Şekil 11. *Üçgenimizin ikizkenar üçgen kümesine ait üyelik fonksiyonu. min fonksiyonu: aldığı parametreler arasından en küçük olanı geri döndürür.*

A = 75°, B = 75° ve C = 30° olması durumunda, değerlerini yerine yazdığımızda çıkan sonucun 1 olduğu görülür. Hatırlayacağınız gibi 1 değeri de en yüksek üyelik derecesiydi. Yani bu durumda üçgenin, tam olarak bir ikizkenar üçgen olduğu söylenebilir. Bu şekilde farklı değerleri yerine yazarak, o üçgenin ne kadar ikizkenar üçgen olduğuna dair çıkarımlar yapılabilir.

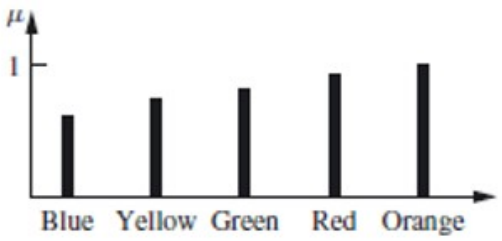
3.2.3. Sıralama

Bazen üzerinde çalıştığımız örneklem veri kümesinin daha gerçekçi bir numune olabilmesi için; yani belirli bir kitleyi yada sistemi daha iyi ifade edebilen bir özet olabilmesi için anketler yapılabilir. Bu anketler sonucunda çıkan sonuçları; en çok tercih edilenlerden, en az tercih edilenler şeklinde sıralayarak üyelik fonksiyonları oluşturulabilir. Örneğin 10 bin kişiden oluşan bir gruptaki

insanlara en çok sevdiği renk sorulmuş olsun. Amaç, en çok sevilen renk için bir bulanık küme oluşturmak. Alınan cevaplara göre sonuçlar şekildeki (Şekil 12) gibi olsun:

	Number who preferred						
	Red	**Orange**	**Yellow**	**Green**	**Blue**	**Total**	**Percenta**
Red	–	517	525	545	661	2 248	22.5
Orange	483	–	841	477	576	2 377	23.8
Yellow	475	159	–	534	614	1 782	17.8
Green	455	523	466	–	643	2 087	20.9
Blue	339	424	386	357	–	1 506	15
Total						10 000	

Şekil 12. *anket sonuçları*



Şekil 13. *Üyelik fonksiyonu*

Bu şekilde en çok belirtilen değerleri sıralayarak üyelik fonksiyonları oluşturulabilir.

3.2.4. Optimizasyon Yöntemleri

Şimdiye kadar gördüğümüz yöntemlerin tümünde, fonksiyonların belirlenmesi için bir uzman görüşüne, diğer bir deyişle: sistem hakkında bilgili ve öngörülü kişilere danışma ihtiyacının olduğunu gördük. Fakat bulanık sistemi oluştururken bu gibi kişilere ulaşma şansımız olmayabilir. Dahası, sistemi tasarlayan kişi olarak bizim de konu hakkında yeterli bilgi ve tecrübemiz olmayabilir.

Bu gibi durumlarda üyelik fonksiyonlarını olabildiğince en iyi şekilde belirlemek için optimizasyon yöntemleri kullanılır. Genel olarak kullanılan yöntemler:

- Popülasyon tabanlı stratejiler
- Ajan tabanlı sistemler
- Yapay Sinir Ağları

şeklindedir. Özellikle popülasyon tabanlı stratejilerden Genetik algoritmalar, üyelik fonksiyonlarının optimizasyonunda sıkça tercih edilen algoritmalardan birisidir.

Bu konu bulanık mantığın yapay zeka ve otonom sistem projelerine dahil edilmesiyle de ilişkilidir. Çünkü bulanık sistemi tasarlayabilmek için elimizde yalnızca salt verilerin olduğu durumlarda, daha iyi sonuçlar elde edebilmek için bu yöntemler oldukça işe yaramaktadır.

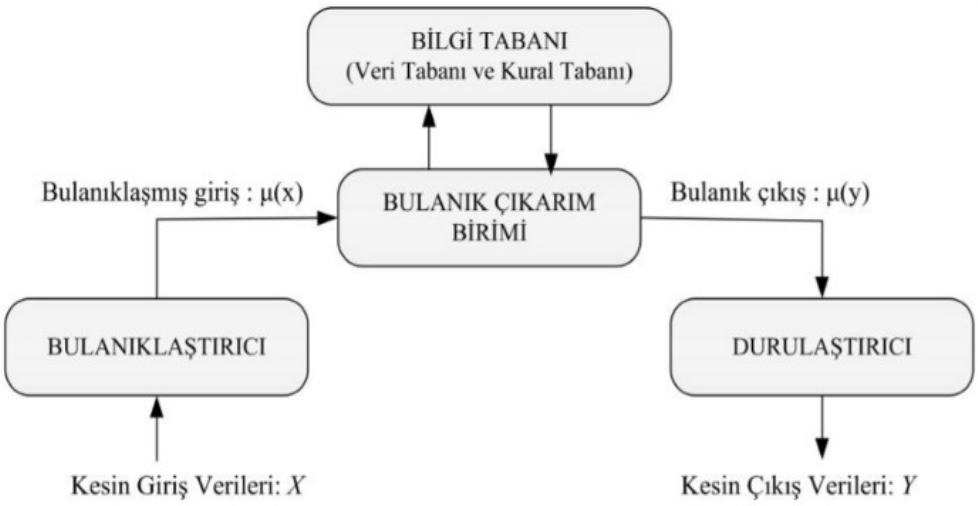
4. Bulanık Kurallar ve Çıkarım

Bu başlıkta basit bir bulanık sistem tasarımından ve buradaki aşamalarda geçen kural ve çıkarım kavramlarından bahsedeceğiz.

4.1. Bulanık Sistem

Önceki kısımlarda bulanık mantığı oluşturan temel birimlerden söz ettik. Önemli kısımlardan yer yer bahsettiysek de, bulanık mantıkla oluşturulmuş bir sistemin iskeletinden tam olarak bu yazıda bahsedeceğiz.

Şimdiye kadar değindiğimiz bulanık kümeler ve dereceli üyelik prensibiyle oluşturulan kontrol sistemlerine bulanık sistemler deniyor. Burada kontrol kelimesinin altını çizmek istiyorum. Çünkü bulanık mantığı şimdiye kadar bir yapay zeka uğraşı olarak gördüysek de; bulanık mantık, yapay zekanın önemli bir bölümü olan makine öğrenmesinden kendini epey ayrı tutar. Makine öğrenmesi yöntemleriyle oluşturulmuş bir sistem, kontrol ve karar mekanizmalarını devreye sokmadan önce (yani, gelecek veriler üzerinde tahminler yapmadan önce), elde tutulan verilerden anlamlı sonuçlar üretebilir ve bunu kendi karar mekanizmasına yansıtabilir. Fakat bulanık mantık, özü itibariyle makinelerin karar verebilme süreciyle ilgilenir. Veri kümesinden anlamlı bir model oluşturmak gibi işlemler, bulanık mantığın öncelikli alanlarından biri değildir. Yani, bulanık mantık bir öğrenme gerçekleştirmez. Yalnızca, öğrenilmiş veriler üzerinden bir kontrol mekanizması inşa edebilir. Tabi günümüzde çeşitli algoritmalarla desteklenerek oluşturulmuş bulanık sistemler; başından sonuna bir makine öğrenmesi sistemi gibi çalışabilir. Fakat bulanık mantığın temelde bu konuyla ilgili olmadığını not edelim.



Şekil 14. *Bulanık sistem.*

Aşağıdaki şekilde (Şekil 14) bir bulanık sisteminin temel yapısı gösterilmektedir. Verilerin; en yalın haliyle sisteme alındığı giriş kısmından, anlamlı sonuçlara dönüşmüş olarak geldiği çıkış kısımına kadar bütün süreçleri adım adım takip edelim:

- Verilerin bulanık sistem üzerinden geçerek işlenmesi süreci, okla gösterildiği yönde ilerlemektedir. İlk olarak giriş verilerini (X) dahil ederek, bulanık sistemin çarklarını döndürmeye başlarız. Giriş verilerinin üzerinde herhangi bir yapısal işlem gerçekleştirmek (normalizasyon vb.) zorunlu değildir. Buradaki işlemlere geçmeden önce, önceki konularda bahsettiğimiz kavramlardan dilsel değişkenleri ve buna bağlı olarak; bulanık kümeleri ve üyelik fonksiyonlarını oluşturmuş olmak gerekir. Şekilde gösterilen adımlar, sistemimizi bulanık mantık kuramıyla tanımladıktan sonrasını ifade etmektedir.
- Bulanıklaştırıcı: Sisteme yalın haliyle alınmış değerleri, üyelik fonksiyonunu kullanarak, bulanık değerlere (hatırlayın, 0 ile 1 arasındaydı) dönüştüren birimdir. Yani her bir giriş değerinin, bulanık kümeye/kümelere olan üyelik derecesini hesaplar diyebiliriz.
- Bulanık Çıkarım Birimi: Bu kısım, bilgi tabanıyla ortak çalışarak, kendisine gelen bulanık değerlerden sonuçlar çıkarmaya çalışır. Bu sonuçların neye göre ve nasıl çıkarılacağının bilgisi (ismiyle müsemma) bilgi tabanında tutulmaktadır.
- Bilgi Tabanı: Bulanık kümeler arasındaki ilişkiler burada tutulur. Gelecek verilere göre hangi çıkarımların yapılacağıyla ilgili kurallar yine buradadır. (Bu kısmı, bulanık sistemin anayasası gibi düşünebiliriz.)
- Durulaştırıcı: Çıkarım yapılmış veriler buraya kadar bulanık değer aralığında gelmektedir. Oysa bizim ihtiyacımız olan çıkış verilerin bambaşka bir aralıkta olması gerekebilir. Durulaştırıcı; gelen bulanık değerleri, istediğimiz bir aralığa göre ölçeklendirmeyi sağlar.

Basit bir bulanık sistemin çalışması bu şekilde özetlenebilir. Şimdi biraz daha detaylarına göz atalım.

4.2. Çıkarım

Üyelik fonksiyonları konusunda çıkarım yöntemine kısaca değinmiştik. Bu yöntemi üyelik fonksiyonlarının belirlenmesinde kullanılan bir yöntem olarak gördük fakat; aslında bundan çok daha fazlasıdır. Çıkarım yöntemi daha geniş bir açıdan; var olan bilgileri kullanarak, yeni bilgilerin elde edilmesini ifade eder.

Bulanık sistemleri oluştururken birden fazla dilsel değer ve bulanık küme kullanabiliriz. İncelediğimiz ilk örneklerden olan ‘bu kişi genç mi, değil mi?’ örneğini hatırlayın. Belirli bir yaş aralığındaki kişiler için yaptığımız gençlik kabulünü, hem kesin hem de bulanık kümeler açısından irdelemiştik. Şimdi bu örneğe, kişinin yaş değerinin yanı sıra, cinsiyet bilgisini de ikame edelim. Gençlik algısının yalnızca yaşla değil, cinsiyetle de ilgili olduğuna dair farazi bir öngörümüz olduğunu düşünelim. Bu durumda; kişinin gençliğine karar verebilmemiz için yaş bilgisine ek olarak, cinsiyet bilgisini de sorgulamamız ve bu iki farklı türden bilgiyi, anlamlı şekilde bir araya getirmemiz gerecektir.

İşte burada, bu iki farklı değeri mantıksal ifadelerle tanımlayan **kurallara** ihtiyacımız vardır. Bu kurallar, geleneksel mantıkta da olduğu gibi, basit bir bileşik önerme formunda tanımlanabilir. Örneğin:

"Kişi, 14 yaşındaysa ve kadınsa gençtir."

Burada iki farklı değeri (önermeyi) birbirine bağlayan 've' bağlacına dikkat ediniz. Bu iki önermeyi, aslında iki bulanık kümenin (yaş ve cinsiyet kümeleri) bir temsili olarak düşünelim. Bulanık kümelerin ilişkilerinden bahsettiğimiz yazıda, birleşim ve kesişim durumlarının mantıksal ifadelerle de gösterilebildiğine atıf yapmıştık. Nitekim, buradaki bağlaç da, aslında bu iki bulanık kümenin ilişkisini tanımlamaktadır. Bu şekilde önceden yaptığımız tanımlarla, bulanık sistemin bilgi tabanında tutulacak kurallarını oluşturmuş oluruz. Daha sonra da (birazdan bahsedeceğimiz) çıkarım yöntemlerini kullanarak, oluşturduğumuz kuralların direktifinde yeni bilgiler elde ederiz. Bu bilgiler de sistemden alınan sonuçlara karşılık gelecektir.

Not: Örnek olarak verdiğimiz ifadenin kesin hükümler içermesi nedeniyle kesin kümeler açısından da değerlendirilmesi olasıdır. Fakat aynı örneğe sadık kalmak için bu yanılgıyı göz ardı etmiş bulundum. Bulanık sistem içerisinde daha doğru bir önerme şu şekilde olabilir: "Kişi, küçük yaştaysa ve kadınsa gençtir".

4.2.1. Mantıksal Bağlaçlar

Birden fazla önermeyi birbirine bağlamak için kullanılan ifadelere **mantıksal bağlaçlar** denmektedir. Bu bağlaçları kullanarak, kendilerine özgü doğruluk değerleri olan önermelerden, üçüncü bir değer elde edilebilir. **(Şekil 15)** Yani, bir çıkarım sağlanabilir.

P	Q	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
0	0	0	0	1	1
1	0	0	1	0	0
0	1	0	1	1	0
1	1	1	1	1	1

***Şekil 15**. Mantıksal bağlaçlar.*

- P ve Q ifadeleri birer önermeyi temsil eder. Her bir satır, P ve Q önermelerinin doğruluk değerini ve mantıksal bağlaçlarla ifadelerinden çıkan yeni değeri gösterir.
- Bağlaçlar sırasıyla: **'ve', 'veya', 'ise', 'ancak ve ancak'** şeklinde ifade edilir.
- Bağlaçlı önermelerin (bileşik önermelerin) okunması şu şekildedir; her bir önermenin gerçek değerine bakılır ve mantıksal bağlacın, bu değerlere göre hangi sonucu döndürdüğü tablodan bulunur. Örneğin: **P** önermesinin değeri 1, **Q** önermesinin ise 0 olsun. Bu durumda **P ⇒ Q (P ise Q)** bileşik önermesinin değeri **0** olacaktır.

Bileşik önermelerin her zaman 1 sonucunu ürettiği durumlara **totoloji**, her zaman 0 sonucunu ürettiği durumlara ise **çelişki** denir. Örneğin:

$$(P \Lambda (P \Rightarrow Q)) \Rightarrow Q$$

bileşik önermesi, P ve Q değerleri ne olursa olsun, her zaman 1 sonucunu üretir. Yani totolojidir (önermedeki ifadeler ne olursa olsun, önermeden çıkarılan yorumlar doğrudur) Verilen örnek, özel bir bileşik önermeyi ifade etmektedir. Bu ifadeye modus ponens (doğrulama) denir. Burada; P bir olguyu, (P ⇒ Q) bir kuralı, (en sağdaki) Q ise sonucu temsil etmektedir. Örneğin:

- P = Ali'nin para kazanması (olgu)
- Q = Ali'nın yeni bir araba alması (sonuç)
- (P ⇒ Q) = Ali para kazanıyorsa, araba alabilir (kural)

Bu durumda; Ali para kazanıyorsa ve kurala göre Ali para kazanıyorken araba alabilirse, bu durumda 'Ali araba alacaktır' çıkarımı yapılabilir.

Diğer bir önerme ise:

$$(\sim Q \Lambda (P \Rightarrow Q)) \Rightarrow \sim P$$

şeklindedir. Bu bileşik önerme de bir totolojidir ve modus tollens (yanlışlama) olarak bilinir. Benzer şekilde ~Q olgu, (P ⇒ Q) kural ve ~P ise sonuçtur. Aynı örnek üzerinden incelersek (~ işareti, önermenin olumsuzunu belirtmektedir):

- ~P = Ali'nin para kazanmaması (olgu)
- ~Q = Ali'nin yeni bir araba almaması (sonuç)
- (P ⇒ Q) = Ali para kazanıyorsa, araba alabilir (kural)

Ali para kazanmıyorsa ve Ali, para kazanıyorken araba alabilirse; bu durumda 'Ali araba almamıştır' çıkarımı yapılır.

Modus ponens ve modus tollens, temel çıkarım yöntemlerinden ikisidir. Örnekleri incelerken; sonuç (çıkış) önermesinin, olgu (giriş) ve kural önermelerine bağlı olarak değiştiğini göz önünde bulundurunuz.

4.2.2. Kural Sunumu

Bir giriş değerine ve bir kurala bağlı olarak yeni bir bilginin nasıl çıkarılacağı konusunu inceledik. Şimdi ise bir kuralın, bulanık sistemleri kurgularken nasıl ifade edilebileceğinden bahsedelim.

Şekil 1.e geri dönersek; bilgi tabanı biriminin, bulanık sistemin anayasası olduğu yönünde bir benzetim yapmıştık. Bu benzetme, akılda kalıcı olmasının yanı sıra, aynı zamanda kuralların bulanık sistemler için ne kadar önemli olduğuna da atıf yapmaktadır. Çünkü birden fazla olguyu anlamlı bir şekilde bir araya getirmek gerektiğinde, bu şekilde kuralların oluşturulması ve ileride yapılacak çıkarımların, bu kurallara riayet etmesi gerekmektedir.

Kuralların sunumu basitçe; EĞER — O HALDE (IF — THEN) denilen bir yöntem kullanılarak gerçekleştirilir. Aşağıdaki örnekleri inceleyelim:

- EĞER (IF) yol kaygan ise, O HALDE (THEN) araba kullanmak tehlikelidir.
- EĞER (IF) elma kırmızı ise, O HALDE (THEN) olgundur.
- EĞER (IF) hava kapalı VE/VEYA saat geç ise, O HALDE (THEN) sıcaklık düşüktür.

Bu yöntem kısaca: ‘Eğer X A ise, o halde Y B’dir’ şeklinde ifade edilebilir. Son örnekte olduğu gibi; birden fazla olguyu da, mantıksal bağlaçları kullanarak, aynı kural içerisinde bir araya getirebiliriz.

4.3. Bulanık Çıkarım Yöntemleri

Bulanık mantıkla çalışırken yukarıda bahsettiğimiz temel yöntemlerin yanı sıra, kullanabileceğimiz pek çok çıkarım yöntemi bulunmaktadır. Bu yöntemlerden bazıları (bizim de değineceğimiz) doğrudan bulanık mantık üzerinde çalışmak üzere geliştirilmiştir. Her bir yöntem temelde aynı işlevi (belirlenen kurallara göre yeni bilgiler çıkarma görevini) yerine getirse de; verimlilik, karmaşıklık gibi yönlerden bazıları diğerlerine göre tercih sebebi olabilir. Bu tercihler, üzerinde çalışılan problemin parametrelerine bağlıdır.

Bulanık sistemlerde en çok kullanılan çıkarım yöntemleri şu şekildedir:

- Mamdani Çıkarımı
- Sugeno Çıkarımı

Bu yöntemlere ek olarak; Tsukamoto çıkarımı ve Larsen çıkarımı yöntemleri de bulunmaktadır. Fakat literatürde en çok geçen ve en bilindik çıkarım yöntemleri oldukları için Mamdani ve Sugeno yöntemini inceleyeceğiz.

İsimlerinden de anlaşılabileceği gibi, her bir çıkarım yöntemi onu geliştiren bilim insanının soyadını taşımaktadır.

4.3.1. Mamdani Çıkarımı

En çok kullanılan bulanık çıkarım yöntemidir. Bunun başlıca sebepleri; Mamdani çıkarımının insan algısına daha çok hitap etmesi, tasarımının nispeten kolay olması ve yorumlanabilirliliği daha fazla olmasından dolayıdır.

İlk kez 1975 yılında, Londra Üniversitesi’nde çalışan matematikçi ve bilgisayar bilimci İbrahim Mamdani tarafından geliştirilmiştir.

- Mamdani çıkarımında giriş ve çıkışlar bulanık değerlerdir.
- Giriş değerlerinin tetiklediği kurallara göre, üyelik değerleri hesaplanır. Daha sonra hesaplanan değerler, kuralların içerisinde geçen ve/veya mantıksal bağlaçlarına göre max ya da min operatörüne verilirler. Eğer, kural içerisinde geçen olgular birbirine ‘ve’ ile bağlıysa, hesaplanan üyelik değerleri min operatörüne; ‘veya’ ile bağlıysa max operatörüne verilir. Bu operatörler, adlarından da anlaşılabildiği gibi, aldıkları birden çok değer arasından en küçüğü ya da en büyüğü döndürürler.

Resim 2. *Mamdani çıkarım yönteminin babası İbrahim Mamdani (1942–2010).*

Örnek: 100 tane ikinci el arabanın model ve kilometre bilgisini tutan bir veritabanı olduğunu düşünelim. Amacımız; bu bilgilere bakarak, araçların yaklaşık fiyatlarını tahmin eden bir bulanık sistem tasarlamak. Bu durumda:

- Giriş değerleri: model ve km
- Çıkış değeri: fiyat

olduğunu söyleyebiliriz. Her bir değişken için belirlediğimiz dilsel değerler şu şekilde olsun:

- *Model: Düşük [2002, 2007 yıl], Orta [2002, 2012 yıl], Yüksek [2007, 2012 yıl]*
- *Kilometre: Düşük [0, 50K km], Orta [0, 100K km], Yüksek [50K, 100K km]*
- *Fiyat: Düşük [0, 20K TL], Orta [0, 40K TL], Yüksek [20K, 40K TL]*

Her bir dilsel değerin tanım aralığı ve birimi yanında belirtilmiştir. Yine her bir değer için üçgen üyelik fonksiyonu kullanılacaktır. Üyelik fonksiyonlarının grafiği Şekil 3.te gösterilmektedir.

Buraya kadar oluşturduğumuz kısımlar, bulanık sistemde geçen bilgi tabanı kısmının bir bölümünü oluşturmaktadır (Veri tabanı). Diğer önemli bölümü ise kuralların oluşturulmasıdır (Kural tabanı). Şimdi de kurallarımızı oluşturalım:

- *KURAL 1: EĞER model Düşük VE kilometre Yüksek ise, O HALDE fiyat Düşüktür.*
- *KURAL 2: EĞER model Orta VE kilometre Orta ise, O HALDE fiyat Ortadır.*
- *KURAL 3: EĞER model Yüksek VE kilometre Düşük ise, O HALDE fiyat Yüksektir.*

Bu kuralların oluşturulması, diğer yazılarda da bahsettiğimiz gibi, sistemi oluşturan olgular hakkında yeteri kadar bilgi sahibi olmayı (yani ‘uzman’ olmayı) gerektirmektedir. Buradaki

kurallar, sistem hakkında daha isabetli ve doğru bilgilere göre yorumlanıp yeniden düzenlenebilir. (Yani, optimize edilebilir.)

Kural tabanımız da artık hazır. Çıkarım birimimizi de, konu başlığı olduğu üzere, Mamdani çıkarımı olarak belirledik. Artık sistemimiz giriş değerlerini almaya hazır hale geldi. Giriş değerlerimiz şu şekilde olsun:

- Model = 2011
- Kilometre = 25K

Giriş değerlerini aldıktan sonra yapmamız gereken ilk iş bulanıklaştırma işlemidir (Bkz. Şekil 1). Öncelikle Model için, 2011 değerinin her bir bulanık kümeye (düşük, orta, yüksek) olan üyelik değerini hesaplayalım (M: Model):

- Mdüşük(2011) = model_düşük = 0
- Morta(2011) = model_orta = 0.24
- Myüksek(2011) = model_yüksek = 0.8

Ardından Kilometre için 25K değerinin, yine her bir dilsel değerin bulanık kümesine olan üyelik değerini hesaplayalım (K: Kilometre):

- Kdüşük(25000) = kilometre_düşük = 0.48
- Korta(25000) = kilometre_orta = 0.52
- Kyüksek(25000) = kilometre_yüksek = 0

Bulanıklaştırma işlemini tamamladık. Şimdi ise çıkarım işleminin ilk adımı olarak, giriş değerlerinin hangi kuralları tetiklediğini değerlendirmemiz gerekiyor. Burada ‘kuralların tetiklenmesi’nden kasıt; kurallarda geçen dilsel değerlerin üyelik değerlerini yerine yazdığımız zaman, en az birini 0'dan farklı olmasıdır. Bu durumda, ‘kural tetiklenmiştir’ yorumunu yapabiliriz. Aksi durumda (bütün değerlerin üyelik derecesi sıfırsa), ‘kural tetiklenmemiştir’ diyebiliriz.

Kuralları ve elimizdeki değerleri incelersek, 2. ve 3. kuralların tetiklendiğini söyleyebiliriz.

Mamdani çıkarım yöntemine göre; hesaplanan uygunluk değerleri, tetiklenen kurallar için max/min operatörüne verilir ve alınan değerler, yine kurallarda verilen sonuç değerinin dilsel değerleriyle eşleştirilir.

Tetiklenen 2. kural için,

- *min(model_orta, kilometre_orta) = fiyat_orta*

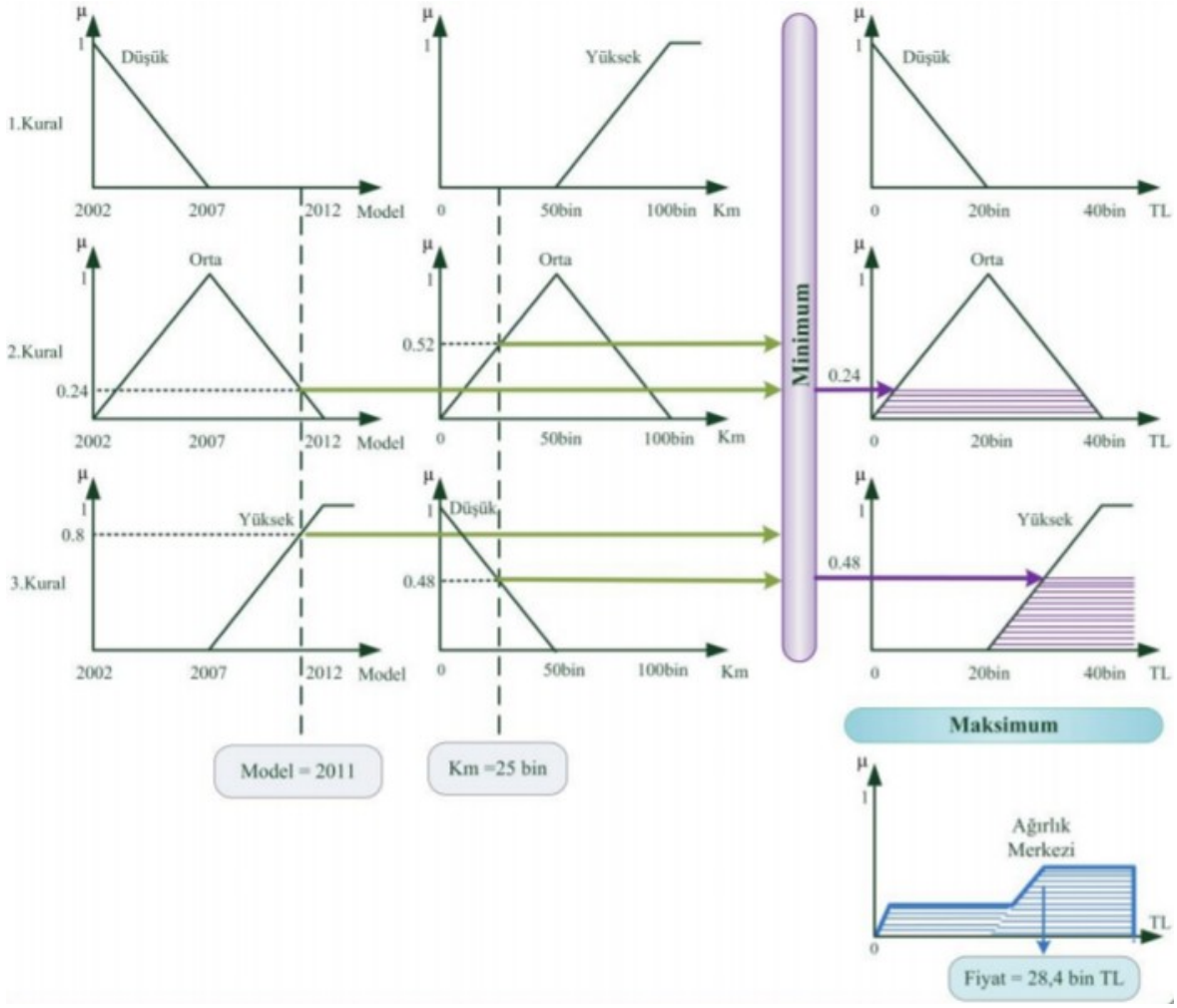
değerlerini yerine koyarsak:

- *min(0.24, 0.52) = fiyat_orta = 0.24*

Aynı şekilde 3. kural için aynı işlemleri yaparsak:

- *min(model_yüksek, kilometre_yüksek) = fiyat_yüksek*
- *min(0.8, 0.48) = fiyat_yüksek = 0.48*

Şimdiye kadar yaptığımız işlemleri aşağıdaki şekil üzerinden (Şekil 16) inceleyebiliriz.



Şekil 16. *Araç problemi için uygulanan Mamdani çıkarımı.*

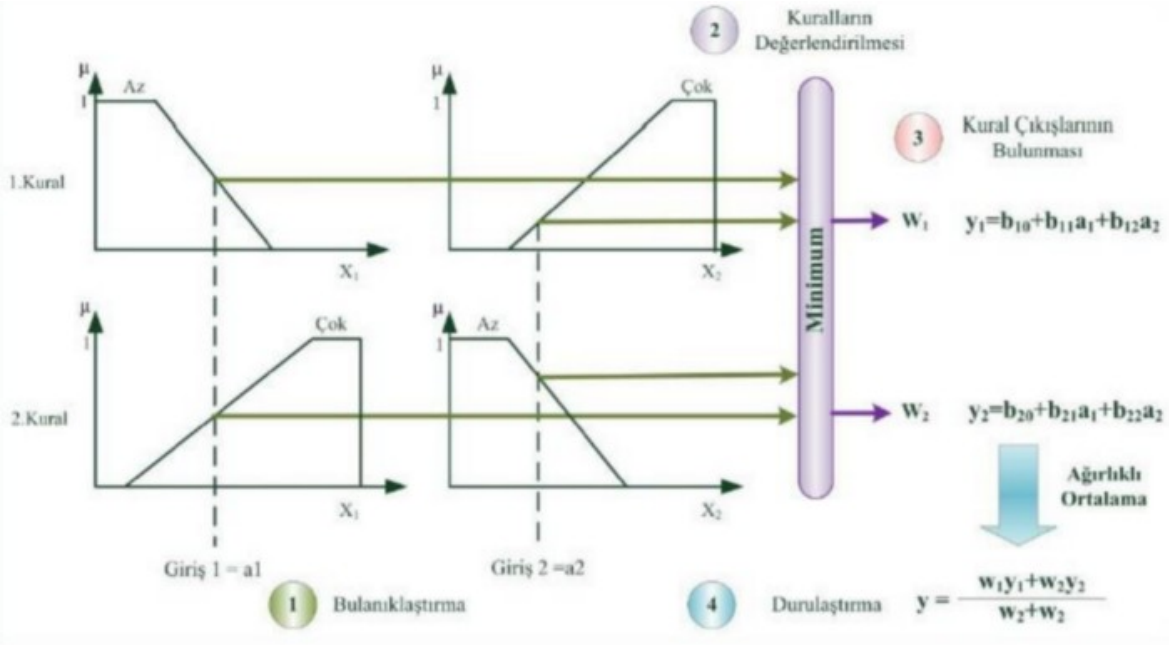
Mamdani çıkarımında son olarak; elde edilen üyelik değerlerinin sonuç kümeleri üzerinde (düşük, orta, yüksek fiyat) kestiği alanlar hesaplar. Bu alanlar toplanır ve bulunan toplam alan değeri, daha sonra durulaştırma yöntemleri (bu yöntemlere ileride değineceğiz) kullanılarak bulanık değerlerden, istediğimiz aralıktaki değerlere ölçeklenir. Şekilde durulaştırma işlemi de gerçekleştirilmiş (Ağırlık merkezi metodu ile) ve son olarak çıkış değerinin 28.4 bin TL olarak hesaplandığı da gösterilmiştir.

4.3.2. Sugeno Çıkarımı

Sugeno çıkarımı özellikle kontrol problemlerinde çokça tercih edilen çıkarım yöntemlerinden birisidir.

Sugeno çıkarımının Mamdani çıkarımından en büyük farkı: Mamdani çıkarımı bulanık değerlerle çıkış verirken, Sugeno çıkarımı çıkış değerini bir fonksiyon şeklinde vermektedir. Bu yüzden Sugeno çıkarımında durulaştırma işlemleri, genellikle ortalama hesaplamak kadar basit işlemlerdir.

Yukarıda bahsettiğimiz Mamdani yönteminde, giriş değerlerinin yanında çıkış değerinin de (fiyat) bir bulanık küme olarak tanımlandığını gördük. Sugeno'da ise, girişler bulanık kümeler şeklinde olabilirken, çıkışın bir fonksiyon şeklinde (genellikle polinom) tanımlanmış olması gerekir. (Şekil 17)



Şekil 17. *Sugeno çıkarım yöntemi.*

Şekilde görüldüğü üzere: y1 ve y2 değerleri, oluşturulan çıkış fonksiyonlarının, giriş değerlerine göre ürettiği sonuçlardır. Daha sonra bu değerler, üyelik değerleri (w1 ve w2) de kullanılarak, durulaştırma işlemine geçilir ve çıkış değeri istenilen şekilde sistemden alınır.

5. Durulaştırma

Yukarıda basit bir bulanık sistem oluşturmak için gerekli olan tasarım parametrelerinden bahsettik. Bu başlıkta; tasarım zincirinin son halkası olan durulaştırma yönteminden bahsedeceğiz. Son kısımda ise klasik bir bulanık mantık problemini, şimdiye dek incelediğimiz yöntemlerle çözmeye çalışacağız.

Bulanık sisteme girilen değerlerin; oluşturduğumuz kural ve bilgilerin süzgecinden geçmeden evvel, cebirsel dünyadan bulanık mantık dünyasına geçiş yapabilmesi için, bulanıklaştırma denilen bir işlemden geçirildiğine değindik. Burada tam olarak; bildiğimiz aralıktaki giriş değerleri, 0 ve 1 aralığındaki bulanık değerlere (üyelik fonksiyonlarının yardımıyla) ölçekleniyor. Sonrasında oluşturulan kurallar ve bu kurallara göre çalışan çıkarım birimleri, bulanık değerleri referans alarak işlemlerini gerçekleştiriyor. Üretilen çıkış değeri buraya kadar bulanık değer olarak geldiği için durulaştırılması gerekiyor.

Bahsedeceğimiz durulaştırma yöntemlerini uygulamak, çıkarım yönteminden tamamen bağımsız bir süreç olmayabilir. Örn. Sugeno çıkarım yönteminde: çıkış için oluşturulan fonksiyonlar, istenilen aralıkta değerler verecek şekilde tasarlanmışsa; bu durumda çıkarım biriminden alınan çıktılar duru değerler olacaktır ve ekstra bir durulaştırma işlemine gerek kalmayacaktır.

Literatürde önerilen çok sayıda durulaştırma yöntemi bulunmaktadır. Bunlardan en çok kullanılanları inceleyelim.

5.1. Durulaştırma Yöntemleri

5.1.1. Ağırlık Merkezi

Özellikle Mamdani çıkarım yönteminde tercih edilen ve sıkça kullanılan durulaştırma yöntemlerinden birisidir. Tetiklenen kurallardan gelen üyelik değerlerinin, bulanık çıkış kümeleri üzerinde kestiği alanlar toplanır. Daha sonra bu alanların geometrik ağırlık merkezi aşağıdaki formül (Şekil 18) ile hesaplanır. Ortaya çıkan değer, artık durulaşmış çıkış değerdir.

$$\mathbf{y_{merkez}} = \frac{\int \mu(\mathbf{y_i})\mathbf{y}\mathbf{dy}}{\int \mu(\mathbf{y})\mathbf{dy}}$$

Şekil 18. *Ağırlık merkezinin hesaplanması*

5.1.2. Ağırlıklı Ortalama

Simetrik üyelik fonksiyonuna sahip çıkış kümelerinde uygulanan bir yöntemdir. Ağırlıklı ortalama yönteminde her bir kuraldan alınan üyelik değeri, bu değerin çıkış kümesi üzerinde kestiği alanla çarpılır. Bu çarpımların toplamının; bütün kurallardan alınan üyelik değerlerinin toplamına oranı bize ağırlıklı ortalamayı vermektedir. Aşağıdaki formülde (Şekil 19) belirtilmiştir.

$$\mathbf{y_{duru}} = \frac{\sum_{\mathbf{i=1}}^{\mathbf{n}} \mu(\mathbf{y_i}) \times \mathbf{y_{alan}}}{\sum_{\mathbf{i=1}}^{\mathbf{n}} \mu(\mathbf{y_i})}$$

Şekil 19. *Ağırlık ortalama değeri ile durulaştırma. n, toplam kural sayıdır.*

5.1.3. Alan Merkezi

Üyelik derecesinin bulanık çıkış kümeleri üzerinde kestiği alanlarda, en büyük üyelik değerini veren çıkış değerleri için aşağıdaki formül (Şekil 20) ile ortalama hesaplanır. Bu ortalamaya alan alan merkezi denilmektedir. Ağırlıklı ortalamaya benzer bir yöntemdir. Fakat bu yöntemde alan hesaplamaya gerek kalmaz.

$$\mathbf{y_{duru}} = \frac{\sum_{\mathbf{i=1}}^{\mathbf{n}} \mu(\mathbf{y_i}) \times \mathbf{y_i}}{\sum_{\mathbf{i=1}}^{\mathbf{n}} \mu(\mathbf{y_i})}$$

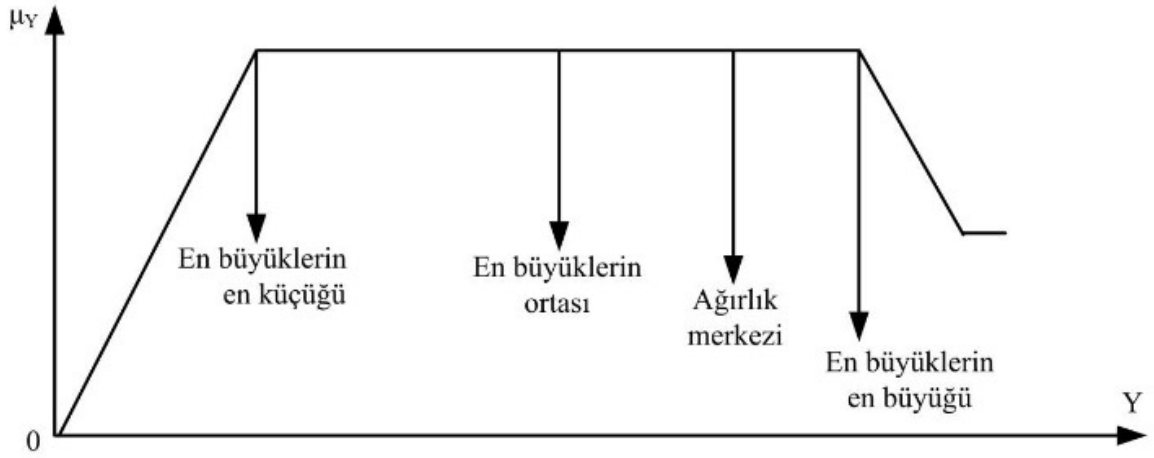
Şekil 20. *Alan merkezi. n, toplam kural sayısıdır.*

5.1.4. En Büyüklerin En Küçüğü ve En Büyüğü

Tüm bulanık çıkış kümelerinin birleşiminde belirlenen en küçük ya da en büyük değerlerin seçilmesi yöntemidir. Doğrudan çıkış aralığı üzerinde, üyelik derecesine göre bir değer seçildiği için; matematiksel işleme gereksinim duyulmaz. (Şekil 21)

5.1.5. En Büyüklerin Ortalaması

4. yönteme benzer bir yöntemdir. Burada da, en büyük üyelik değerini veren çıkış değerlerinin ortalaması hesaplanır. (Şekil 21)



Şekil 21. *Durulaştırma yöntemleri. Y ekseni, çıkış değerler için istenen aralığı tutan eksendir.*

5.2. Çamaşır Makinesi Tasarımı Örneği

Bulanık mantığın sıkça kullanıldığı alanlardan birisi de kontrol sistemleridir. Her koşulda “A girdisine karşılık B çıktısını üret” gibi tam tanımlı ve bütün ihtimalleri değerlendiren tasarımlar mümkün olmadığı için, sistem için gözetilen kriterlere uyacak ve her girdi için en makul çıktıyı üretebilecek sistemlere ihtiyaç duyulur. İnsan mantığının çalışmasını referans alarak çıktılar üretebilen bulanık mantık da, doğal olarak, bu konuda iyi bir alternatif olarak karşımıza çıkar.

Çamaşır makinesi tasarımı, bulanık mantık anlatımında sıkça verilen bir örnektir. Bu tasarım problemi, aslında en basit anlamda bir optimizasyon (iyileme) problemidir. Çünkü geliştiriciden beklenen şey; kirli çamaşır miktarı ve suyun sertlik derecesine göre, makinenin devir sayısını ideal bir değerde tutarak, verimliliğin sağlanmasıdır. Problemin farklı versiyonları da bulunmaktadır.

Problemimizi çözümlemeye; girişleri ve çıkışı belirleyerek başlayalım.

- Girişler: Çamaşır miktarı (kg), sertlik derecesi (ASD)
- Çıkış: Yıkama devri (rpm)

Not: Sertlik birimi olarak Amerikan Sertlik Derecesi (ASD) esas alınmıştır.

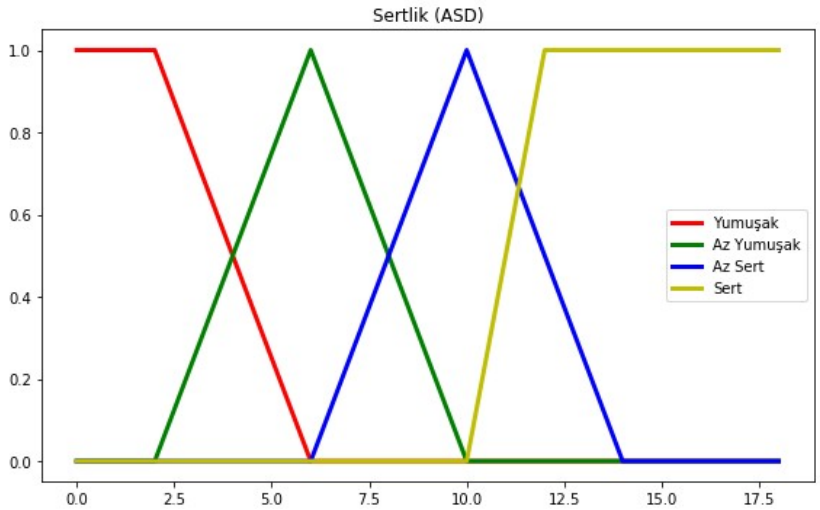
Adım 1: Bulanık kümelerin oluşturulması

Belirlediğimiz giriş ve çıkış değerleri için, dilsel değerleri belirten bulanık kümeleri aşağıdaki gibi oluşturalım:

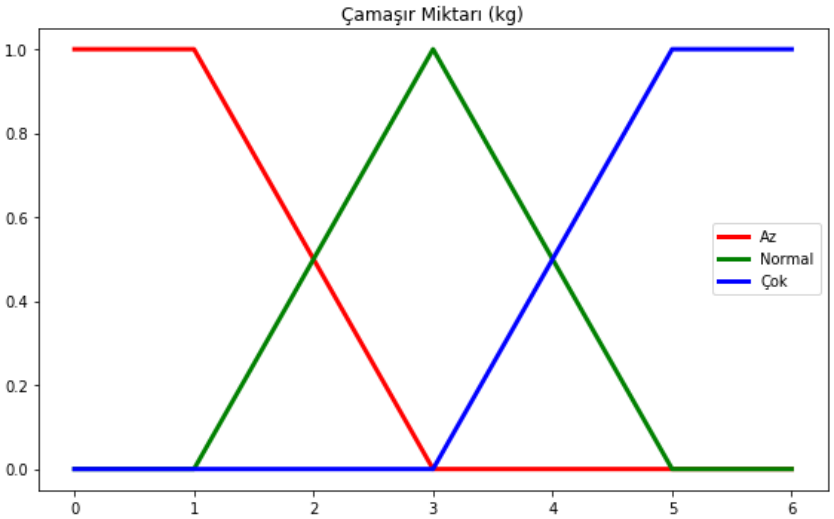
- Sertlik: Yumuşak, Az Yumuşak, Az Sert, Sert
- Çamaşır: Az, Normal, Çok
- Devir: Hassas, Hafif, Normal, Güçlü

Adım 2: Üyelik fonksiyonlarının oluşturulması

Belirtilen bulanık kümeler için üçgen ve yamuk üyelik fonksiyonları kullanılarak, aşağıdaki grafikleri oluşturalım.



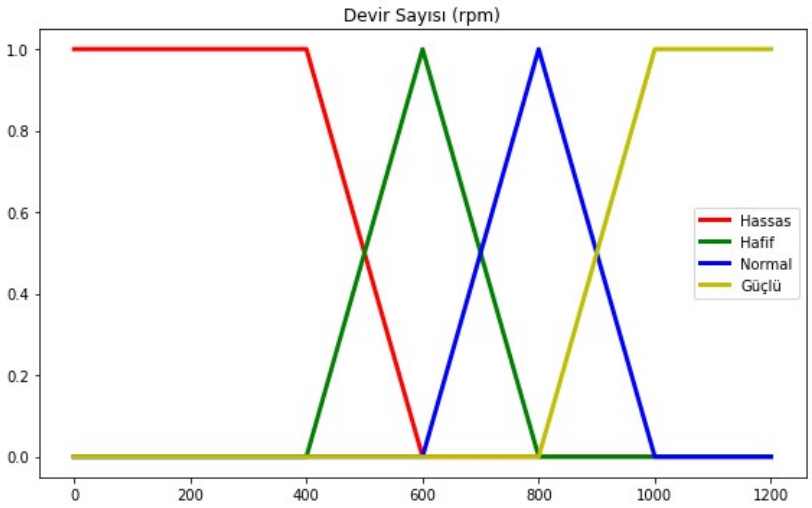
Şekil 22. *Sertlik için üyelik fonksiyonları*



Şekil 23*. Çamaşır miktarı için üyelik fonksiyoları.*

Daha önce de belirttiğimiz gibi; dilsel değerler için aralıkların belirlenmesi, sistemi tanımlayan uzmanların bilgi ve tecrübelerine göre değişebilir.

Son olarak çıkış kümemiz olan devir sayısı kümesi için de üyelik fonksiyonlarını oluşturalım:



Şekil 24. *Devir sayısı için üyelik fonksiyonları.*

Adım 3: Kuralların oluşturulması

Sistemimizin kural tabanı için, aşağıdaki kuralları belirleyelim:

- ***Kural 1:*** *EĞER sertlik YUMUŞAK VE çamaşır AZ İSE, O HALDE devir HASSASTIR.*
- ***Kural 2:*** *EĞER su AZ YUMUŞAK VE çamaşır NORMAL İSE, O HALDE devir HAFİFTİR.*
- ***Kural 3:*** *EĞER su AZ SERT VEYA çamaşır NORMAL İSE, O HALDE devir NORMALDİR.*
- ***Kural 4:*** *EĞER su SERT VEYA çamaşır ÇOK İSE, O HALDE devir GÜÇLÜDÜR.*

Görüldüğü gibi, her bir çıkış değeri için birer kural üretilmiştir. Tercih edilirse, aynı çıkış değerleri için birden fazla kural da üretilebilir. Fakat çözümü olabildiğince basit tutmak için gerek duyulmamıştır.

Adım 4: Giriş değerlerinin alınması ve bulanıklaştırılması

Problemimiz için tanımlanan giriş değerleri şu şekilde olsun:

- ***Sertlik:*** *8 ASD*
- ***Çamaşır miktarı:*** *4.5 kg*

Bulanıklaştırma işlemine geçelim ve giriş değerlerini, üyelik fonksiyonlarını kullanarak, bulanık değerlere dönüştürelim. Sertlik değeri için:

1. µYumuşak(8) = 0
2. µAzYumuşak(8) = 0.5
3. µAzSert(8) = 0.5
4. µSert(8) = 0

8 ASD'lik sertlik değerinin, Az Yumuşak ve Az Sert bulanık kümelerine eşit derecede dahil olduğu görülüyor.

Çamaşır miktarı için:

1. µAz(4.5) = 0
2. µNormal(4.5) =0.25
3. µÇok(4.5) = 0.75

Burada da belirlenen çamaşık miktarının, Normal ve Çok kümelerine farklı derecelerde ait oldukları görülüyor. Bulunan değerler, grafikler üzerinden çıkarılacak doğru denklemleriyle kolaylıkla hesaplanabilir.

Adım 5: Tetiklenen kuralların tespit edilmesi

Dikkat ettiyseniz, 1. ve 2. kuralda olgular 've' bağlacı ile bağlanmışken; diğer kurallarda 'veya' bağlacı kullanılmıştır. Bu da, kurallardan çıkarılacak sonuçlar için, ve bağlacı yerine min; veya bağlacı yerine ise max operatörünün kullanılması gerektiği anlamına gelir. Üyelik değerlerini, kuralların ifade ettiği olgulara göre, ilgili operatörlere uygularsak;

- Kural 1: min(0, 0) = 0
- Kural 2: min(0.5, 0.25) =0.25
- Kural 3: max(0.5, 0.25) = 0.5
- Kural 4: max(0, 0.75) = 0.75

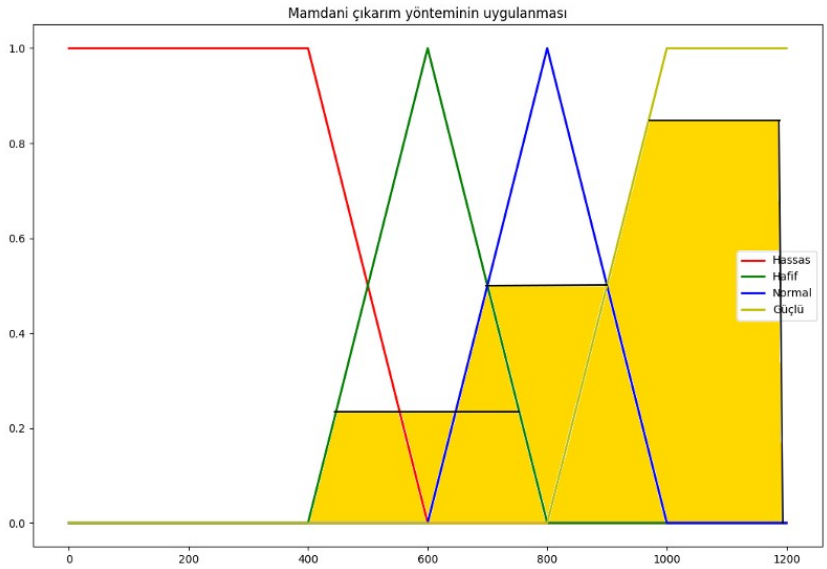
Sonuçlarını elde ederiz. Bu durumda, 1. kural haricindeki tüm kuralların tetiklendiğini söyleyebiliriz.

Adım 6: Çıkarımın gerçeklenmesi

Çıkarım yöntemi olarak Mamdani çıkarımını tercih edebiliriz. Hatırladığınız gibi Mamdani çıkarımını; tetiklenen kurallardan gelen üyelik değerlerinin, çıkış kümeleri üzerinde kestiği bölgelerin hesaplanmasını öneren yöntem olarak belirtmiştik. O halde; kurallardan gelen değerlerin, işaret ettikleri çıkış kümelerine göre y-ekseninde (üyelik değerleri ekseni) keseceği değerler şu şekilde olacaktır (ÇK: Çıkış Kümesi):

- Hassas ÇK için: 0 (Kural1)
- Hafif ÇK için: 0.25 (Kural 2)
- Normal ÇK için: 0.5 (Kural 3)
- Güçlü ÇK için: 0.75 (Kural 4)

Çıkan sonuç aşağıdaki gibi (Şekil 25) olacaktır:



Şekil 25. *Mamdani çıkarımı sonucunda ortaya çıkan alanlar.*

Adım 7: Durulaştırma

Durulaştırma için alan merkezi yöntemini kullanalım. Öncelikle her bir kuraldan gelen çıkış değeri için; Şekil 4.te gösterilen, üyelik fonksiyonlarını kestiği minimum ve maksimum çıkış değerlerini, ardından bu değerlerin ortalamasını hesaplayalım:

- Hassas ÇK için: min = 0, max = 0, ortalama = 0
- Hafif ÇK için: min = 450, max = 750, ortalama = 600
- Normal ÇK için: min = 700, max = 900, ortalama = 800
- Güçlü ÇK için: min = 950, max = 1200, ortalama = 1075

Ortalamaları Formül 3.e göre üyelik dereceleri ile çarpıp toplarsak:

- 600 * 0.25 + 800 * 0.5 + 1075 * 0.75 = 1356.25

değeri hesaplanır. Son olarak; elde ettiğimiz toplamı, üyelik derecelerinin toplamına bölerek durulaştırılmış değeri bulalım:

- Kural değerleri toplamı: 0.25 + 0.5 + 0.75 = 1.5
- Alan merkezi = 1356.25 / 1.5 = 904,16

Alan merkezi olarak hesapladığımız değer, aynı zamanda durulaştırılmış çıkış değeridir. Yani, (belirttiğimiz giriş değerleri olan) 4,5 kg'lık çamaşır ve 8 ASD sertliğe sahip yıkama suyu için, tasarladığımız bulanık sistemin bize önerdiği devir sayısı yaklaşık 904 rpm'dir.

Kullanılan durulaştırma yöntemine göre, farklı sonuçlar ortaya çıkabileceğini belirtelim.,

6. Python ile Bulanık Mantık Modellemesi

Bu bölümde Scikit-Fuzzy'yi kullanarak, örnek bir bulanık sistem tasarımını Python üzerinde gerçeklemeye çalışacağız.

Uygulamanın kaynak kodlarına, notların sonundaki *Kaynakça ve Bağlantılar* kısmındaki adresten ulaşabilirsiniz.

6.1. Scikit-Fuzzy Nedir?

Scikit-Fuzzy, bilimsel ve mühendislik temelli hesaplamalar için kullanışlı araçlar üreten **SciPy** topluluğu tarafından geliştirilen, Python'da yazılmış açık kaynak bir bulanık mantık kütüphanesidir. Bulanık mantıkla ilgili temel seviyedeki çoğu özelliği barındırmasının yanı sıra; bulanık kümeleme gibi ileri seviye makine öğrenmesi konuları için de kullanışlı araçlara sahiptir. Tamamen ücretsiz ve açık kaynak olmasından dolayı, MATLAB'ın *Fuzzy Logic Toolbox* eklentisi gibi ücretli yazılımlar için iyi bir alternatiftir.

6.1.1. Scikit-Fuzzy Kurulumu

Scikit-Fuzzy'nin çalışabilmesi için, aşağıdaki modüllerin ilgili versiyonlarını veya daha güncel olanlarını yüklemiş olmanız gerekir:

- NumPy >= 1.6
- SciPy >= 0.9
- NetworkX >= 1.9

Pip (PIP install packages) üzerinden kurulum için: işletim sisteminizin komut satırı/terminal arayüzünü açın ve aşağıdaki komutu çalıştırın:

```
pip install -U scikit-fuzzy
```

Kaynak dosyası üzerinden kurulum için: bağlantılar kısmında belirtilen[1] adresi kullanarak, kaynak dosyalarını GitHub üzerinden bilgisayarınıza indirin. İndirilen dosyayı arşivden çıkarın. Komut satırı/terminal arayüzü üzerinden, çıkardığınız dosyanın dizinine geçiş yapın. Aşağıdaki komutu çalıştırın:

```
python setup.py install
```

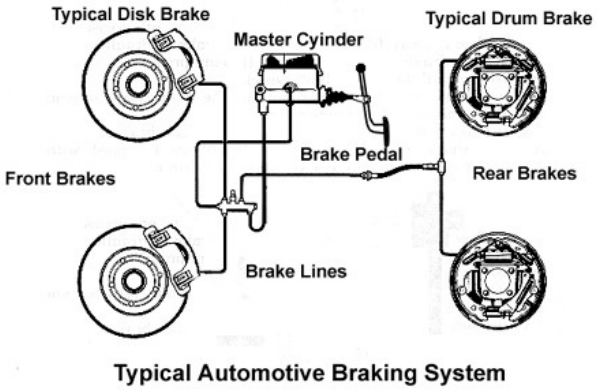
Kurulumları gerçekleştirdikten sonra, herhangi bir Python editörü üzerinden aşağıdaki gibi Scikit-Fuzzy'yi çağırmayı deneyin:

```
import skfuzzy
```

Eğer hata almadıysanız, yükleme işlemi başarıyla tamamlanmış demektir.

6.2. Fren Tasarımı Örneği

Bu tasarım probleminde; belirlenen giriş değerlerine karşılık, aracın uygun bir basınç kuvveti ile fren yapabilmesi için basit bir bulanık sistem tasarlanılması isteniyor.



Şekil 26: *Tipik bir fren sistemi*

Araçların fren sistemi basitçe; aracın hızından kaynaklanan kinetik enerjinin sönümlenerek, ısı enerjisine dönüştürülmesi prensibine göre çalışır. Fren pedalına basıldığı anda, aracın hidrolik mekanizmasında bulunan yağ, yüksek bir basınçla balataları fren diskine doğru sıkıştırır. (Şekil 1) Sıkışma sonucu oluşan sürtünme kuvvetinin etkisiyle araç yavaşlamaya başlar ve belirli bir süre sonra durur. Fren pedalına ne kadar güçlü basılırsa, hidrolik basınçtan kaynaklanan sıkıştırma etkisi de o kadar büyük olacaktır.

Bu bilgiler ışığında, fren sistemini bulanık mantıkla kontrol edebilecek tasarımı gerçekleştirelim. Giriş ve çıkış değerleri aşağıdaki gibi tanımlanmış olsun:

- Girişler: Pedal Basıncı (%), Araç Hızı (mph)
- Çıkış: Fren (%)

Giriş çıkışlar için dilsel değerler de şu şekildedir:

- Pedal Basıncı: Düşük, Orta, Yüksek
- Araç Hızı: Düşük, Orta, Yüksek
- Fren: Zayıf, Güçlü

Belirlenen kurallar ise:

- KURAL 1: EĞER Pedal Basıncı Orta İSE, O HALDE Fren Güçlüdür.
- KURAL 2: EĞER Pedal Basıncı Yüksek VE Araç Hızı Yüksek İSE, O HALDE Fren Güçlüdür.

- KURAL 3: EĞER Pedal Basıncı Düşük VEYA Araç Hızı Düşük İSE, O HALDE Fren Zayıftır.
- KURAL 4: Eğer Pedal Basıncı Düşük İSE, O HALDE Fren Zayıftır.

şeklindedir. Bu aşamadan sonra, bulanık sistemimizi simüle etmek için Python ortamına geçebiliriz.

Öncelikle kullanacağımız kütüphaneleri ekleyelim:

```
import numpy as np
import skfuzzy
as fuzz
import skfuzzy.membership as mf
import matplotlib.pyplot as plt
```

NumPy, Scikit-Fuzzy ve (görselleştirme için) Matplotlib kütüphanelerini ekliyoruz. **‘membership’,** birazdan değineceğimiz üyelik fonksiyonlarını oluşturmaya yarayan bir Scikit-Fuzzy sınıfıdır.

```
x_pedal = np.arange(0, 101, 1)
x_speed = np.arange(0, 101, 1)
x_brake = np.arange(0, 101, 1)
```

Pedal basıncı, araç hızı ve fren değişkenlerinin her biri için birer değişken üretiyoruz. NumPy kütüphanesinden **‘arange’** metodu ile her bir değişken için **[0, 100]** aralığını tanımlıyoruz.

```
pedal_low = mf.trimf(x_pedal, [0, 0, 50])
pedal_med = mf.trimf(x_pedal, [0, 50, 100])
pedal_hig = mf.trimf(x_pedal, [50, 100, 100])

speed_low = mf.trimf(x_pedal, [0, 0, 60])
speed_med = mf.trimf(x_pedal, [20, 50, 80])
speed_hig = mf.trimf(x_pedal, [40, 100, 100])

brake_poor = mf.trimf(y_brake, [0, 0, 100])
brake_strong = mf.trimf(y_brake, [0, 100, 100])
```

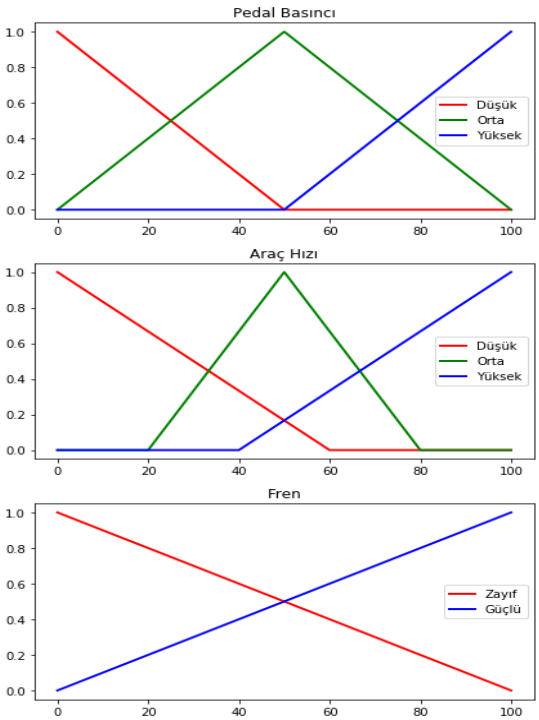
Yukarıda belirlediğimiz dilsel değerlerin her biri için, üyelik fonksiyonlarını oluşturuyoruz. **‘trimf’** metodu, *triangular membership function* yani *üçgen üyelik fonksiyonu* anlamına gelmektedir. İlk parametresi oluşturulacak değişken için tanım aralığını; ikinci parametresi ise üçgenin başlangıç, orta ve bitiş noktalarını (tanım aralığı üzerinde) belirten birer listeden oluşur. Üçgen üyelik fonksiyonlarını bu şekilde oluşturduk. Hatırladığınız gibi, çok sayıda üyelik fonksiyonu tipi bulunuyordu. (gausyen, üstel, sigmoid vb.) Scikit-Fuzzy ile bu üyelik fonksiyonları da oluşturulabilir. Detaylar için Scikit-Fuzzy’nin web sitesindeki dökümana göz atabilirsiniz.

Oluşturduğumuz fonksiyonları, Matplotlib kütüphanesini kullanarak grafiğe dökelim:

```
fig, (ax0, ax1, ax2) = plt.subplots(nrows = 3, figsize =(6, 10))

ax0.plot(x_pedal, pedal_low, 'r', linewidth = 2, label =
'Düşük')
```

```
ax0.plot(x_pedal, pedal_med, 'g', linewidth = 2, label =
'Orta')
ax0.plot(x_pedal, pedal_hig, 'b', linewidth = 2, label =
'Yüksek')
ax0.set_title('Pedal Basıncı')
ax0.legend()

ax1.plot(x_speed, speed_low, 'r', linewidth = 2, label =
'Düşük')
ax1.plot(x_speed, speed_med, 'g', linewidth = 2, label =
'Orta')
ax1.plot(x_speed, speed_hig, 'b', linewidth = 2, label =
'Yüksek')
ax1.set_title('Araç Hızı')
ax1.legend()

ax2.plot(y_brake, brake_poor, 'r', linewidth = 2, label =
'Zayıf')
ax2.plot(y_brake, brake_strong, 'b', linewidth = 2, label =
'Güçlü')
ax2.set_title('Fren')
ax2.legend()

plt.tight_layout()
```

Buraya kadar elde edilen görsel çıktımız şu şekilde olacaktır:



Giriş değerlerini aşağıdaki değerlerle tanımlayalım:

```
input_pedal = 40
input_speed = 75
```

Belirlediğimiz giriş değerleri için, yukarıda üyelik fonksiyonlarıyla oluşturduğumuz bulanık kümelere olan üyelik derecelerini hesaplayalım:

```
pedal_fit_low = fuzz.interp_membership(x_pedal, pedal_low,
input_pedal)
pedal_fit_med = fuzz.interp_membership(x_pedal, pedal_med,
input_pedal)
pedal_fit_hig = fuzz.interp_membership(x_pedal, pedal_hig,
input_pedal)

speed_fit_low = fuzz.interp_membership(x_speed, speed_low,
input_speed)
speed_fit_med = fuzz.interp_membership(x_speed, speed_med,
input_speed)
speed_fit_hig = fuzz.interp_membership(x_speed, speed_hig,
input_speed)
```

‘interp_membership’ metodu, üyelik değerlerinin hesaplanmasında kullanılan bir metottur. İlk parametesi ilgili değişken için tanım aralığını, ikinci parametresi ilgili bulanık kümenin tanım aralığını, üçüncü parametesi ise giriş değerini almaktadır. Giriş değerlerini bulanıklaştırdık. Artık kurallarımızın sahneye çıkma vakti geldi:

```
rule1 = np.fmin(pedal_fit_med, brake_strong)
rule2 = np.fmin(np.fmin(pedal_fit_hig, speed_fit_hig),
brake_strong)
rule3 = np.fmin(np.fmax(pedal_fit_low, speed_fit_low),
brake_poor)
rule4 = np.fmin(pedal_fit_low, brake_poor)
```

NumPy kütüphanesine ait **‘fmax’** ve **‘fmin’** metotları, elemanter düzeyde karşılaştırma yaparak, listelerin minimum ve maksimum elemanlarını bulan metotlardır. Örneğin; [1, 2, 5] ve [2, 0, 3] listeleri *fmin* metoduna verilirse, bu metodun döndürdüğü yeni liste şu şekilde olur: [1, 0, 3]. Yani; her bir elemanı, eş sıralı diğer elemanla karşılaştırıp min yada max olanı hesaplıyorlar. Burada ise, bulanık mantık kullanımında anlamlı olması açısından; *fmax* metodunu *veya* bağlacı yerine, *fmin* metodunu da *ve* bağlacı yerine kullanıyoruz. En dışarıda *fmin* metodunu kullanmamızın nedeni ise: hesaplanan üyelik değerinin üstündeki değerleri almamak içindir.

İlgili girişlere göre hesaplanan kural çıkışlarını, aşağıdaki gibi *fmax* metoduna veriyoruz. Burada amaçladığımız şey: aynı çıkışı veren bulanık kümelerin **birleşim kümesini** hesaplamak. (Hatırlayın, *MAX* operatörü bulanık kümelerde birleşimi ifade ediyordu)

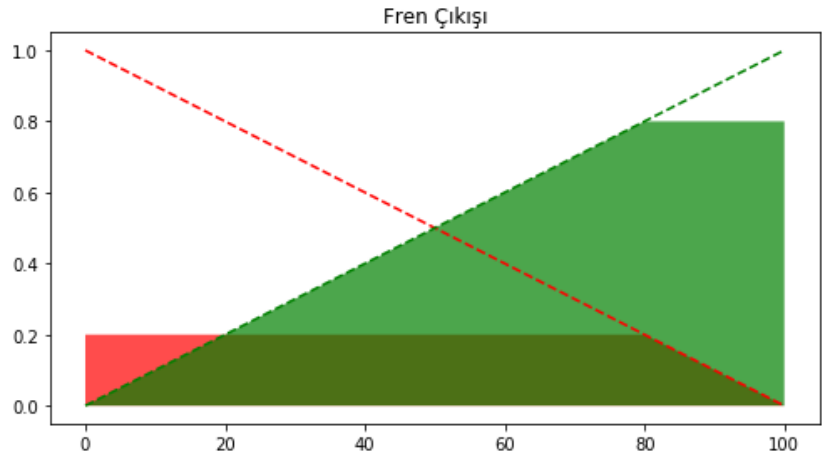
```
out_strong = np.fmax(rule1, rule2)
```

```
out_poor = np.fmax(rule3, rule4)
```

Girişlerin çıkış kümeleri üzerinde kestiği bölgeleri, yine Matplotlib kütüphanesini kullanarak görselleştirelim:

```
brake0 = np.zeros_like(y_brake)

fig, ax0 = plt.subplots(figsize = (7, 4))
ax0.fill_between(y_brake, brake0, out_poor,
facecolor = 'r', alpha = 0.7)
ax0.plot(y_brake, brake_poor, 'r', linestyle = '--')
ax0.fill_between(y_brake, brake0, out_strong,
facecolor = 'g', alpha = 0.7)
ax0.plot(y_brake, brake_strong, 'g', linestyle = '--')
ax0.set_title('Fren Çıkışı')

plt.tight_layout()
```

Görsel çıktımız aşağıdaki gibidir:



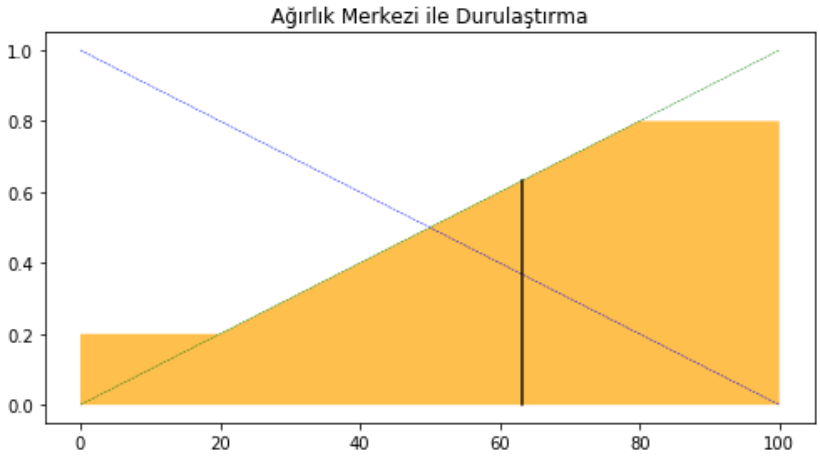
Son aşama olarak durulaştırma işlemine geçiyoruz. Öncelikle, çıkış kümeleri üzerinde gösterilen alanların yine *fmax* metodunu ile birleşimini hesaplayalım ve bir tek çıkış kümesi şeklinde durulaştırma fonksiyonuna devredelim.

```
out_brake = np.fmax(out_poor, out_strong)

defuzzified = fuzz.defuzz(y_brake, out_brake, 'centroid')

result = fuzz.interp_membership(y_brake, out_brake, defuzzified)
```

‘out_brake’, çıkış kümemizin son halidir. **‘defuzz’** metodu ise, durulaştırma fonksiyonudur. Birinci parametre çıkış değişkeninin tanım aralığını, ikinci parametre giriş değerleriyle elde edilen alanı, üçüncü parametre ise durulaştırma yöntemini belirtir. **‘centroid**’ yöntemi, önceki yazıda da bahsettiğimiz, ağırlık merkezi yöntemidir. Diğer yöntemlere yine Scikit-Fuzzy’nin dökümanı üzerinden ulaşabilirsiniz. **‘defuzzified’** değikeni, artık bulanık sistemden son çıkan durulaşmış

değeri tutmaktadır. **‘result’** değişkeni ise; hesaplanan ağırlık merkezinin, tam olarak hangi üyelik derecesine denk geldiğini tutmaktadır.

Hesaplanan çıkış değeri ve grafik üzerinde gösterimi ise aşağıdaki gibidir:



Fren için çıkış değeri: %63.2

Tasarladığımız bulanık sistem; %40 pedal basıncına ve 75 mph araç hızına karşın, frenleme için yaklaşık %63’lük bir oran ile hidrolik sistemi tahrik edecek bir çıkış değeri önermiştir.
Tasarımımızın ne kadar iyi çalıştığı tartışmaya açıktır. Sistemin geliştirilmesi, titizlikle oluşturulmuş kurallar ve bulanık küme tanımlamalarıyla sağlanabilir.

Kaynaklar ve Bağlantılar

[1] Dr. A. Merve ACILAR, Necmettin Erbakan Üniversitesi, Bilgisayar Mühendisliği Bölümü, Bulanık Mantık dersi notları

[2] Scikit-Fuzzy web sayfası: https://pythonhosted.org/scikit-fuzzy/

[3] Fren Tasarımı uygulaması kaynak kodları: https://github.com/atasoglu98/Fuzzy-Brake-Example

Görsel Kaynaklar

Şekil 1	http://4.bp.blogspot.com/_Q-3_PaJrcLs/TRrjDz5QBLI/AAAAAAAABy0/Q_tvPHHiAK8/s1600/venn-diagram-universal-set-E-.PNG
Şekil 2.A	https://www.matematiktutkusu.com/forum/ekstra/matimage/matabirlesimb.png
Şekil 2.B	https://www.matematiktutkusu.com/forum/ekstra/matimage/matakesisimb.jpg
Şekil 2.C	https://www.mathgoodies.com/sites/default/files/lesson_images/subset_example1_0.png
Şekil 2.D	https://upload.wikimedia.org/wikipedia/commons/thumb/5/5a/Venn0010.svg/250px-Venn0010.svg.png
Şekil 3	http://www.aforgenet.com/articles/fuzzy_computing_basics/Fig1.jpg
Şekil 9	http://zone.ni.com/images/reference/en-XX/help/370401J-01/loc_eps_shapesmf.gif
Şekil 15	http://www.mmsrn.com/wp-content/uploads/2018/04/p-q-mantiksal-onermeler.png
Resim 2	https://nthambazale.com/wp-content/uploads/2010/01/mamdani.jpg
Görsel	https://pixabay.com/tr/photos/launderette-%C3%A7ama%C5%9F%C4%B1r-makinesi-708176/
Şekil 26	https://image.cpsimg.com/sites/carparts-mc/assets/classroom/images/brake_system.gif