

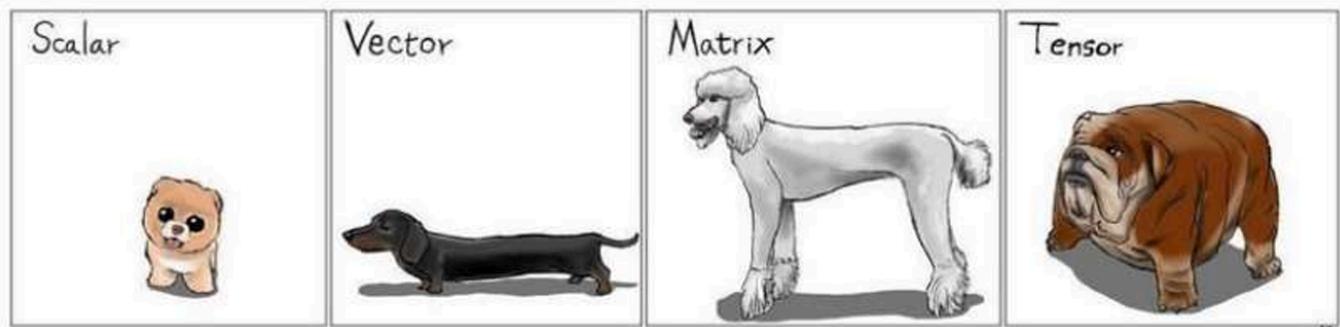
Mantık kavramı

Mantık, insan zihni gibi doğrulu ve yanlışı ayırt etmek için kullanılan bir düşünme disiplinidir. Mantık kavramı doğru (1) veya yanlıştan (0) oluşan bir yargının sonucudur. Örneğin; "İstanbul, Türkiye'nin doğusundadır." cümlesi bir önermedir ve bu önerme yanlıştır. "23 Nisan, Ulusal Egemenlik ve Çocuk Bayramı'dır." önermesi ise doğru bir önermedir. Mantık kavramı insan beyninin problem çözme teknikleri ile oldukça benzerdir. İnsan beyni ilk olarak problemin kaynağı olan faktörleri tespit ettikten sonra, problemin çözümü için çözüm aşamalarını mantıksal olarak gerçekleştirir.

Yapay Zeka Matemetiği

Günlük hayatta bir insanın yaşı, **boyutsuz bir skalar değer** olarak ifade edilir. Buna karşılık, bir insanın fotoğrafı; piksel değerlerinden oluşan ve **iki boyutlu sayısal bir yapı** olan **matris** ile temsil edilir. Matrislerin satır veya sütunlarını oluşturan her bir tek boyutlu yapı ise **vektör** olarak adlandırılır.

Tensör kavramı; birden fazla boyutu olan ve daha karmaşık yapıları temsil etmek için kullanılır. Örneğin, bir insanın damarlarındaki kan akışı incelenirken, damarın hacimsel ve yüzeysel alan değişimlerinin birlikte ifade edilmesi gerekmektedir. Bu tür çok boyutlu fiziksel büyüklüklerin modellenmesinde tensörler kullanılır. Benzer şekilde, **zekâ küpü** yapısı da tensöre örnek olarak verilebilir.



9

5 3 7

Sayı

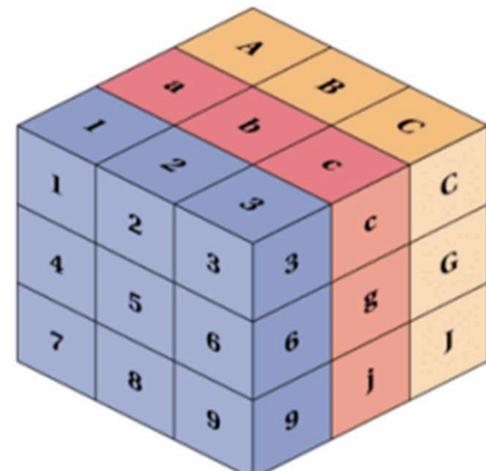
5
1.5
2

Satır vektör

$$\begin{bmatrix} 4 & 19 & 8 \\ 16 & 3 & 5 \end{bmatrix}$$

Sütün vektör

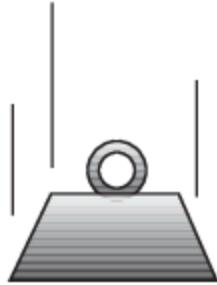
Matris



Tensör

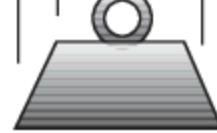
Precision and Significance in the Real World

A 1500 kg mass
is approaching
your head at
45.3 m/s



Precision

**LOOK
OUT!!**



Significance

Bulanık mantık

Klasik mantıkta:

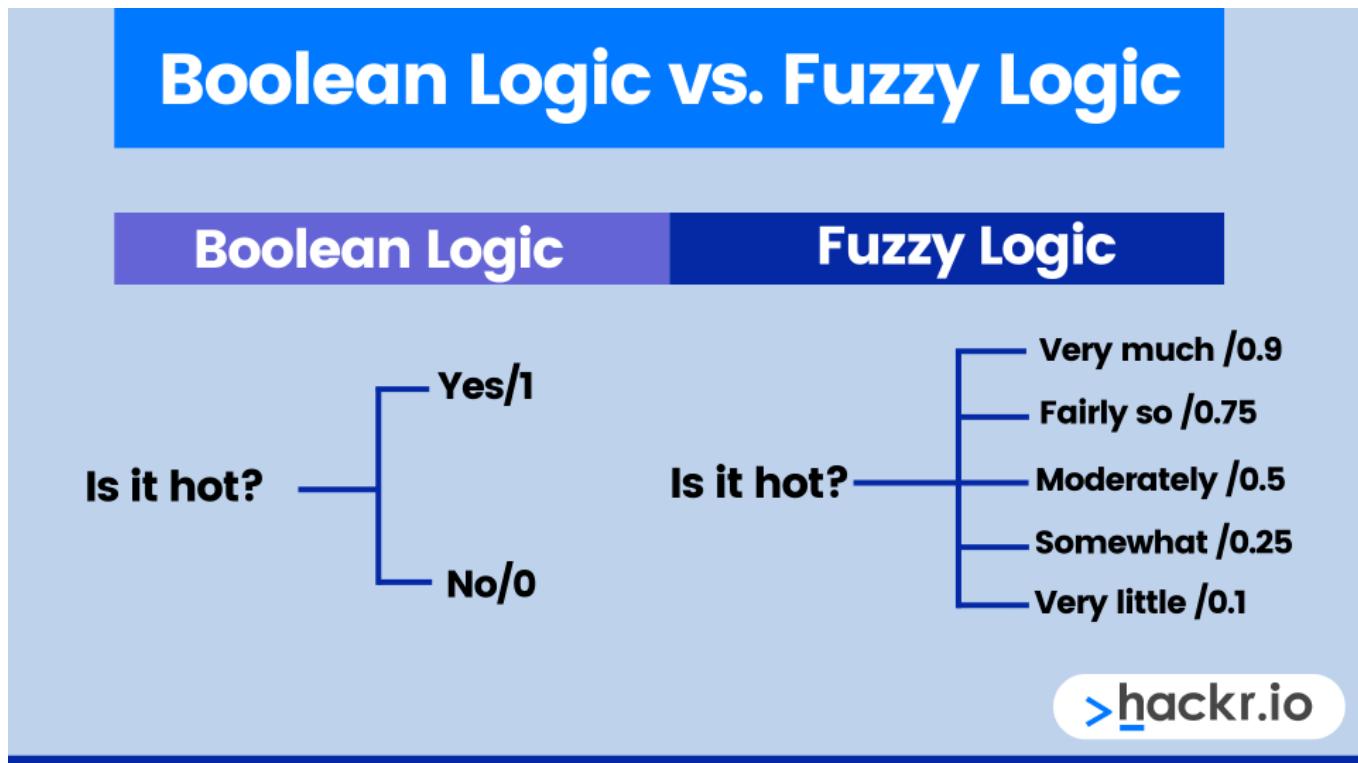
- Doğru / Yanlış
- 0 / 1

Bulanık mantıkta ise:

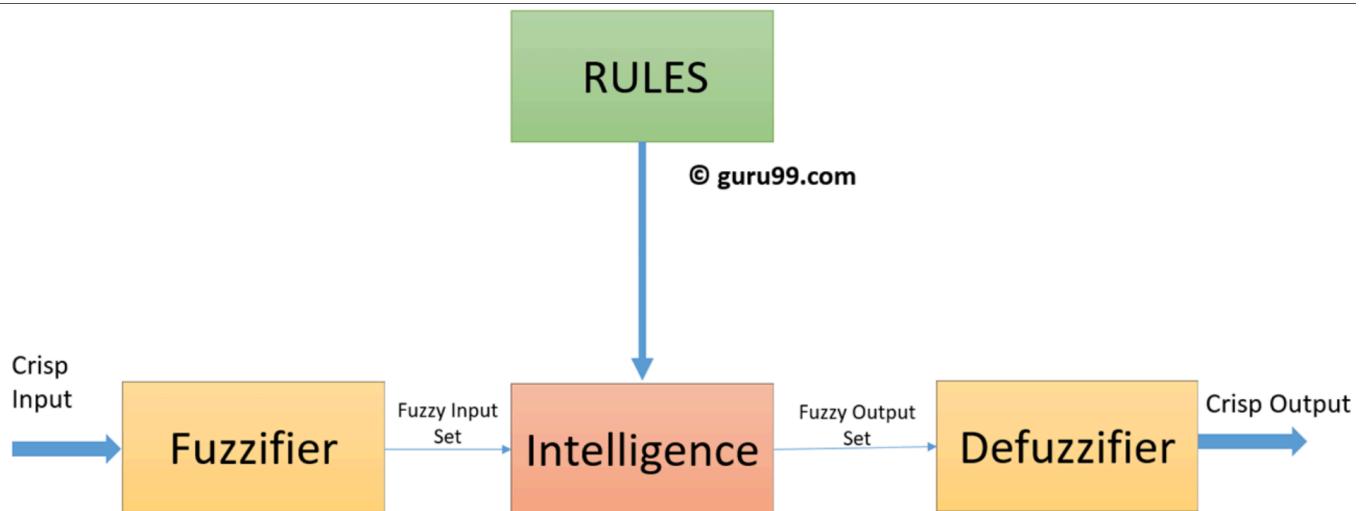
- Kısmen doğru
- 0 ile 1 arasında değerler

Örnek: "Su sıcak mı?"

- Klasik: sıcak / değil
- Fuzzy: %30 sıcak, %70 ılık



Fuzzy ile adım adım



Fuzzification (Bulanıklaştırma)

Kesin (crisp) sayısal girişleri, bulanık kümelere dönüştürür.

Örnek:

Giriş:

Sıcaklık = 30 °C

Tanımlı bulanık kümeler:

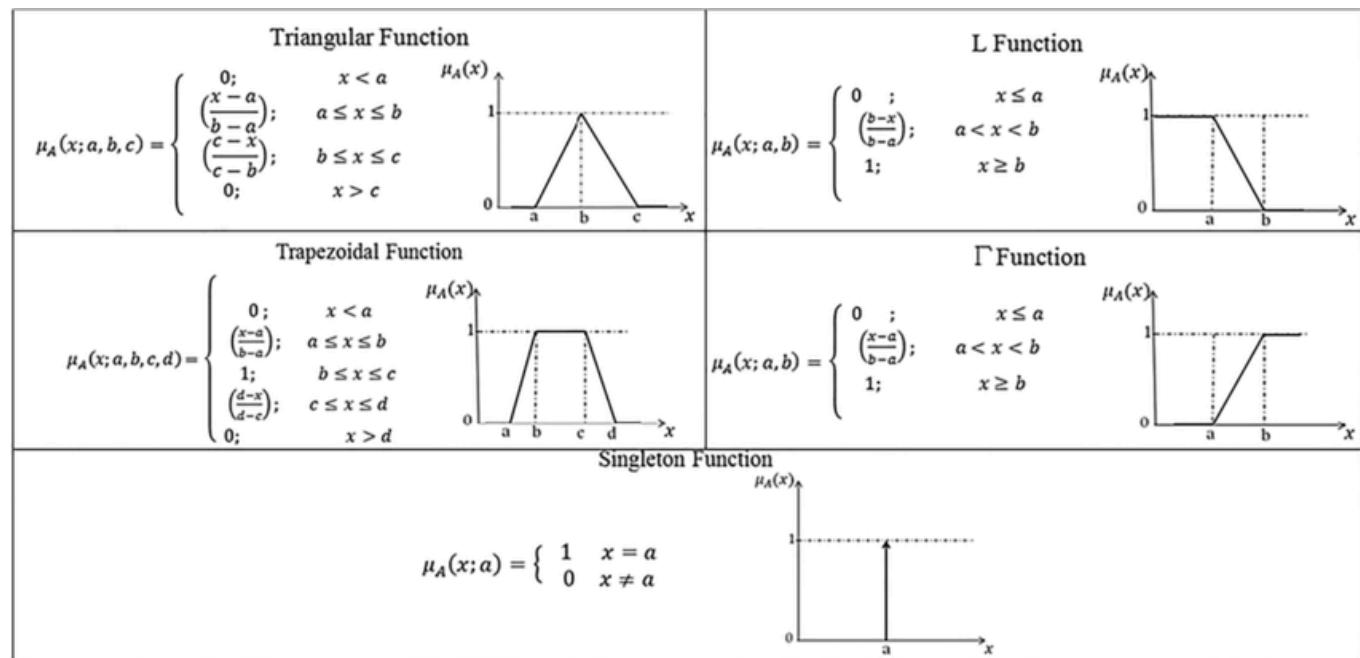
- Soğuk
- İllik
- Sıcak

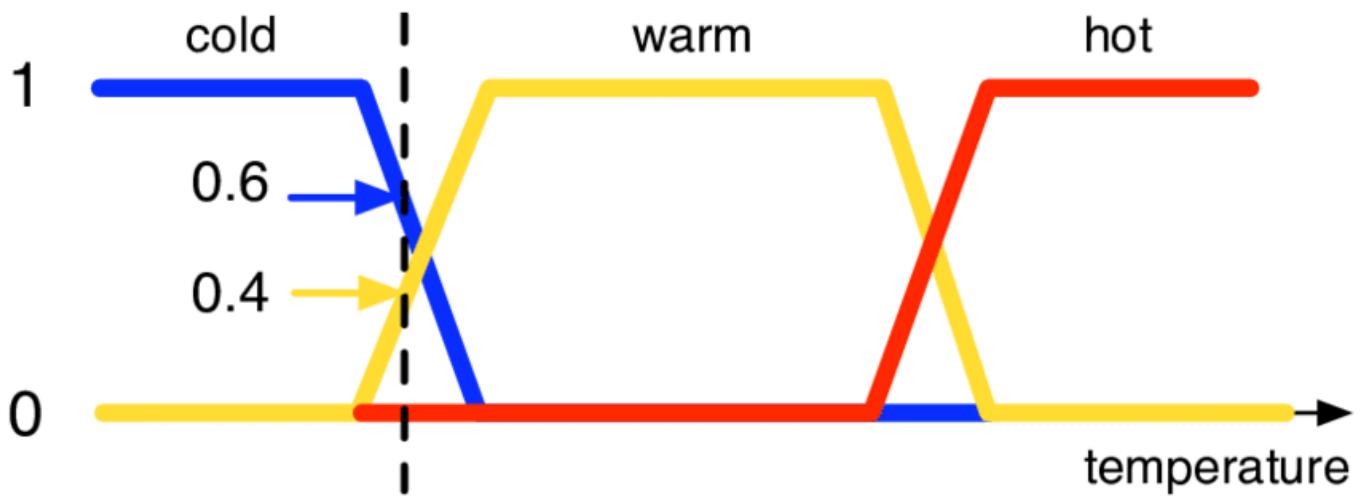
Sonuç: $\mu(\text{Soğuk}) = 0.0$ $\mu(\text{İllik}) = 0.6$ $\mu(\text{Sıcak}) = 0.4$

Yani 28°C hem **illik** hem **sıcak**, ama farklı oranlarda.

Membership Function (Üyelik Fonksiyonları)

Bir değerin **bir fuzzy kümeye ne kadar ait olduğunu** belirler.





Rule Base (Kural Tabanı / Rule Table)

IF – THEN şeklinde yazılmış insan mantığına yakın kurallar.

Örnek Kurallar

IF sıcaklık IS hot AND hız IS düşük THEN güç IS orta

IF sıcaklık IS cold THEN güç IS yüksek

Rule Table Örneği

Sıcaklık \ Hız	Düşük	Orta	Yüksek
Soğuk	Yüksek	Orta	Düşük
İllik	Orta	Orta	Düşük
Sıcak	Düşük	Düşük	Çok Düşük

Bu tablo IF-THEN kurallarının tabloya dökülmüş hali.

Inference Mechanism (Çıkarım Mekanizması)

- Kuralları değerlendirir
 - Girişlerin etkisini çıktı fuzzy kümelerine yansıtır
 - En yaygın yöntemler
 - Mamdani (En yaygın)
 - AND → min
 - OR → max
- Örnek: IF A AND B → $\min(\mu_A, \mu_B)$

Aggregation (Birleştirme)

Birden fazla kuraldan gelen çıktılar **tek bir fuzzy çıktı** haline getirilir.

`max(çıktı1, çıktı2, çıktı3)`

Defuzzification (Durulaştırma)

- **Bulanık çıkışı, tek bir sayısal değere** çevirir.
- En yaygın yöntem: Centroid (Ağırlık Merkezi)

FORMÜL KOY

En kararlı ve en çok kullanılan yöntemdir.

Düzenleme yöntemleri

- Mean of Maximum (MoM)
- Largest of Maximum (LoM)
- Smallest of Maximum (SoM)

Önerilen Youtube videoları.

https://www.youtube.com/watch?v=J_Q5X0nTmrA

<https://www.youtube.com/watch?v=RuuZ5vKgoql>

Neden Fuzzy Kullanılır?

- Matematiksel modeli zor sistemler
- İnsan mantığına yakın kontrol
- Gürültülü sensör verileri
- AUV, robot, iklimlendirme, hız kontrolü

Kod yazımı ve kurulum



Fuzzy logic uygulamamız için bu kütphaneyi enviromentimize eklememiz gerekecek. aşağıdaki kod ile eklenebilir.

```
conda activate yapayzeka_env
pip install scikit-fuzzy
```

```
(yapayzeka_env) C:\Users\User>pip install scikit-fuzzy
Collecting scikit-fuzzy
  Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl.metadata (2.6 kB)
  Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl (920 kB)
    ...
    920.8/920.8 kB 3.8 MB/s 0:00:00
Installing collected packages: scikit-fuzzy
Successfully installed scikit-fuzzy-0.5.0
```

Kütüphaneler

```
import numpy as mat import skfuzzy as bulanik from skfuzzy import control as kontrol
```

- `numpy` → Sayısal aralıklar (universe) için
- `skfuzzy` → Membership function (trimf)
- `skfuzzy.control` → Fuzzy sistem (Antecedent, Consequent, Rule)

Fuzzy Değişkenlerin Tanımlanması (Antecedent / Consequent)

Giriş Değişkenleri (Antecedent)

```
bulasik_miktari = kontrol.Antecedent(mat.arange(0, 100, 1), 'bulaşık miktarı')
kirlilik = kontrol.Antecedent(mat.arange(0, 100, 1), 'kirlilik seviyesi')
```

- **Bulaşık miktarı** → 0–100 arası

- **Kirlilik seviyesi** → 0–100 arası
 - Bunlar **crisp girişlerdir**, birazdan fuzzification yapılacak.
-

Çıkış Değişkeni (Consequent)

```
yıkama = kontrol.Consequent(mat.arange(0, 180, 1), 'yıkama süresi')
```

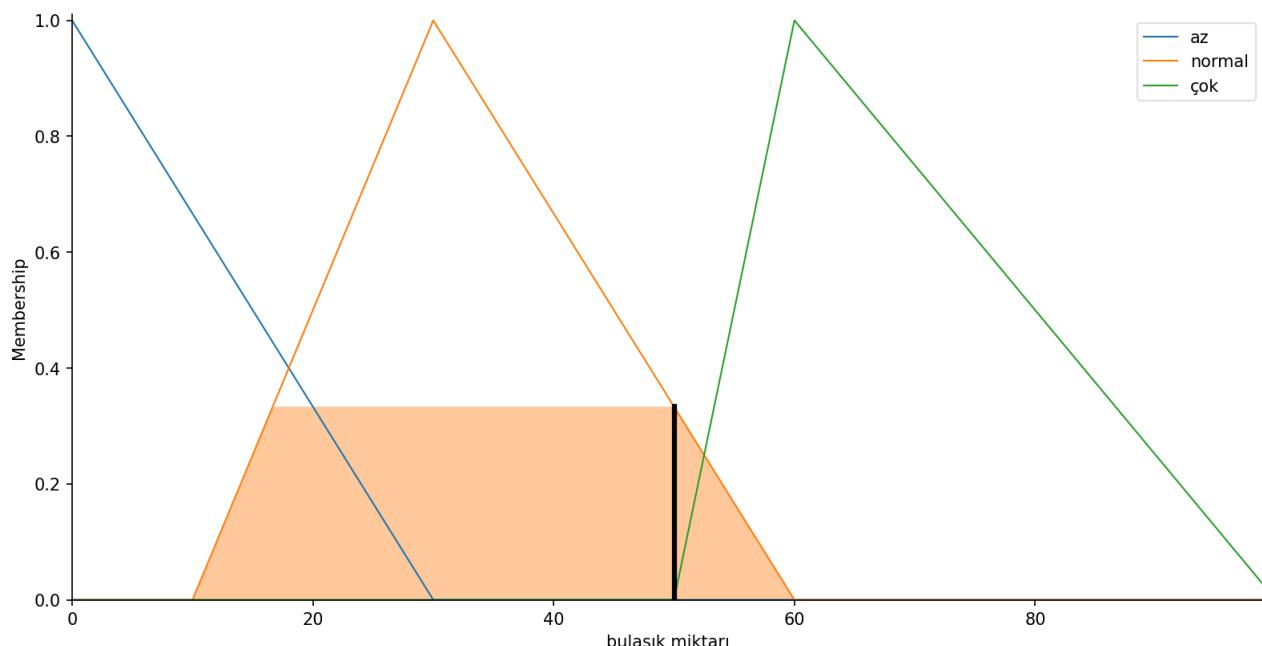
- Yıkama süresi → 0–180 dakika
 - Bu değişken **fuzzy çıkış** olacak, sonra defuzzification ile tek sayıya indirgenecek.
-

Membership Function Tanımları

Bulaşık Miktarı

```
bulasik_miktari['az'] = bulanik.trimf(bulasik_miktari.universe, [0, 0, 30])
bulasik_miktari['normal'] = bulanik.trimf(bulasik_miktari.universe, [10, 30, 60])
bulasik_miktari['çok'] = bulanik.trimf(bulasik_miktari.universe, [50, 60, 100])
```

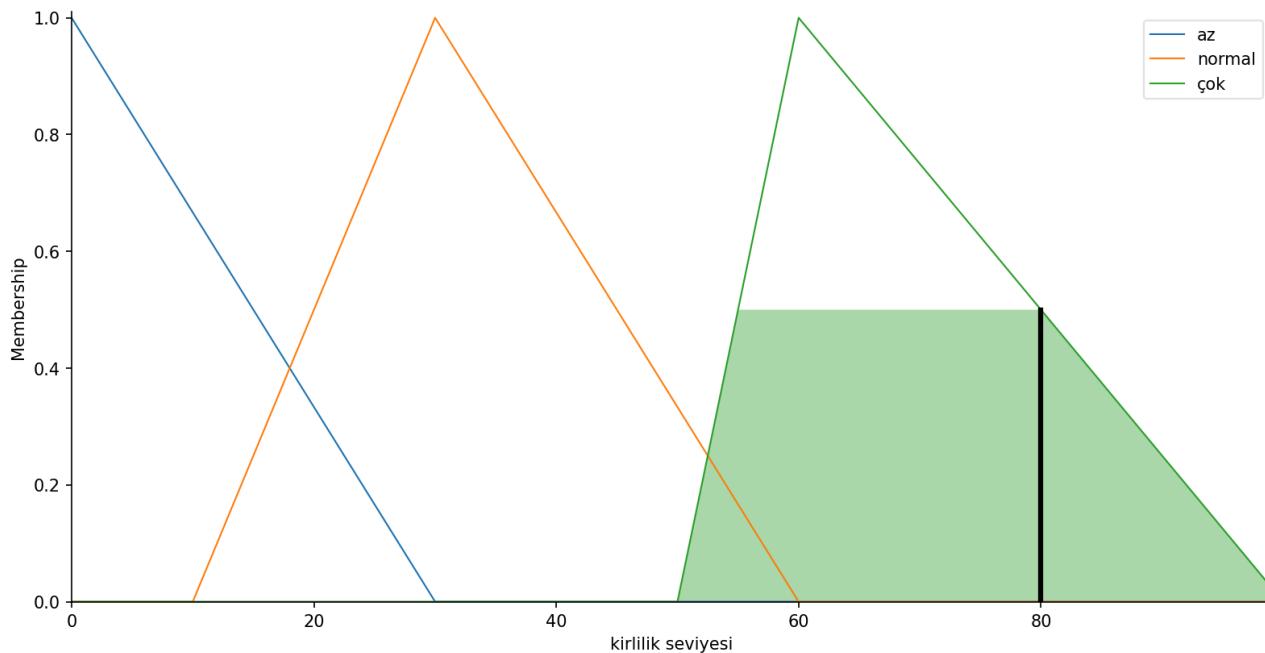
- **az** → 0–30
- **normal** → 10–60
- **çok** → 50–100



Kümeler **üst üste biniyor**, bu fuzzification için gerekli.

Kirlilik Seviyesi

```
kirlilik['az'] = bulanik.trimf(kirlilik.universe[0, 0, 30])
kirlilik['normal'] = bulanik.trimf(kirlilik.universe, [10, 30, 60])
kirlilik['çok'] = bulanik.trimf(kirlilik.universe, [50, 60, 100])
```



- Bulaşık miktarıyla **aynı yapı**
 - Sistem simetrik ve anlaşılır
-

Yıkama Süresi (Çıkış)

```
yıkama['kısa'] = bulanik.trimf(yıkama.universe, [0, 0, 50])
yıkama['normal'] = bulanik.trimf(yıkama.universe, [40, 50, 100])
yıkama['uzun'] = bulanik.trimf(yıkama.universe, [60, 80, 180])
```

- **kısa** → 0–50 dk
- **normal** → 40–100 dk
- **uzun** → 60–180 dk

Yine örtüşme var → yumuşak geçiş.

Rule Base (Kural Tabanı)

Burada **9 adet IF–THEN kuralı** tanımlanmış.

```
kural1 = kontrol.Rule(bulasik_miktari['az'] & kirlilik['az'], yikama['kisa'])
```

Eğer bulaşık az VE kirlilik az ise yıkama süresi kısa

Tüm Kuralların Mantığı

	Giriş parametreleri		Çıkış parametresi
Kurallar	Bulaşık Miktarı	Kirlilik	Yıkama zamanı
Kural-1	Az	Az	Kısa
Kural-2	Normal	Az	Normal
Kural-3	Çok	Az	Normal
Kural-4	Az	Normal	Normal
Kural-5	Normal	Normal	Uzun
Kural-6	Çok	Normal	Uzun
Kural-7	Az	Çok	Normal
Kural-8	Normal	Çok	Uzun
Kural-9	Çok	Çok	Uzun

Fuzzy Kontrol Sisteminin Oluşturulması

```
sonuc = kontrol.ControlSystem([kural1, kural2, ..., kural9])
```

- Rule base burada sisteme yükleniyor

- Henüz hesaplama yok
-

Simülasyon Nesnesi

```
model_sonuc = kontrol.ControlSystemSimulation(sonuc)
```

- Giriş verilerini alıp
 - Inference + Defuzzification yapan yapı
-

Girişlerin Verilmesi (Crisp Input)

```
model_sonuc.input['bulaşık miktarı'] = 50  
model_sonuc.input['kirlilik seviyesi'] = 80
```

- Bulaşık miktarı = **50**
 - Kirlilik = **80**
-

Fuzzification + Inference + Defuzzification

```
model_sonuc.compute()
```

İçeride olanlar:

1. Fuzzification

- 50 → az / normal / çok üyelikleri
- 80 → normal / çok üyelikleri

2. Inference (Mamdani)

- AND → min
- Kurallar aktive edilir

3. Aggregation

- Çıktı fuzzy kümeleri birleştirilir

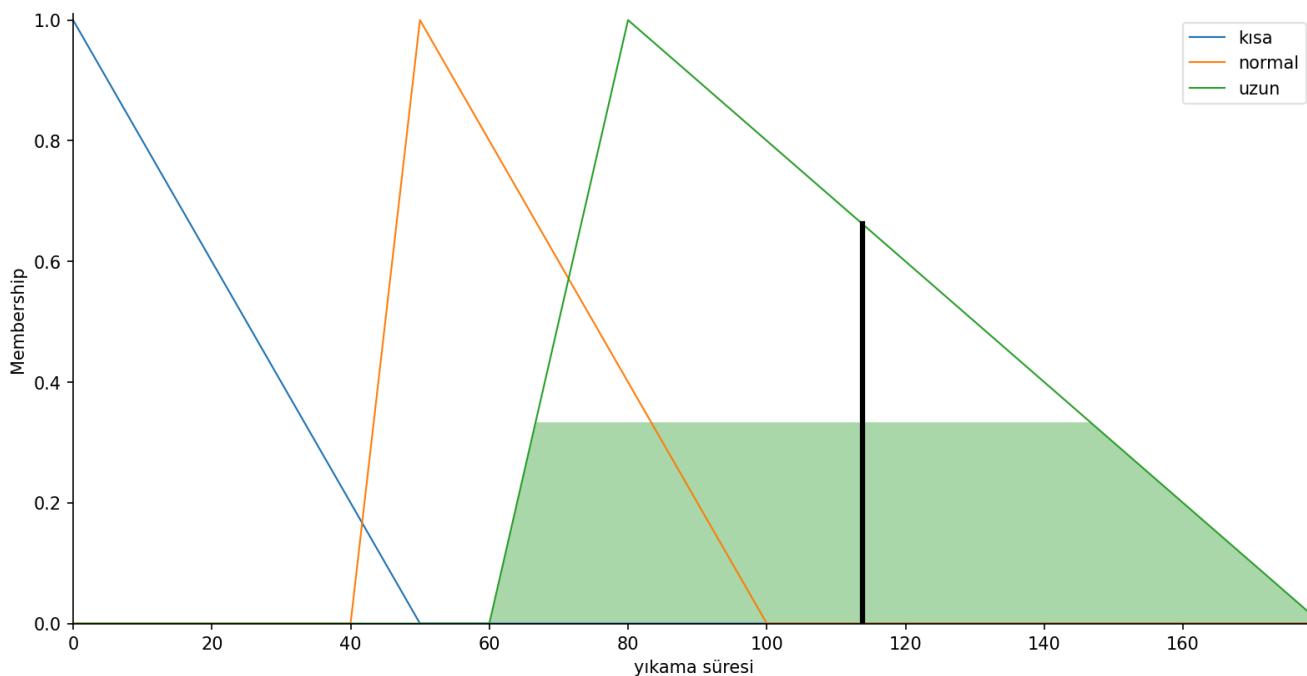
4. Defuzzification

- Centroid yöntemi
- Tek bir sayı üretilir

Çıkışın Alınması

```
print(model_sonuc.output['yıkama süresi'])
```

dakika cinsinden net yıkama süresini verir.



Bu Kodun Fuzzy Adımlarla Eşleşmesi

Fuzzy Adım	Kodda Neresi
Fuzzification	<code>trimf, input[...]</code>
Membership Function	<code>bulanik.trimf()</code>
Rule Base	<code>kontrol.Rule()</code>
Inference	<code>compute()</code>
Defuzzification	<code>compute()</code>
Crisp Output	<code>output[...]</code>

```

import numpy as mat
import skfuzzy as bulanik
from skfuzzy import control as kontrol

bulasik_miktari = kontrol.Antecedent(mat.arange(0, 100, 1), 'bulasik miktari')
kirlilik = kontrol.Antecedent( mat.arange(0, 100, 1), 'kirlilik seviyesi')
yikama = kontrol.Consequent(mat.arange(0, 180, 1), 'yikama süresi')

bulasik_miktari['az'] = bulanik.trimf(bulasik_miktari.universe, [0, 0, 30])
bulasik_miktari['normal'] = bulanik.trimf(bulasik_miktari.universe, [10, 30, 60])
bulasik_miktari['çok'] = bulanik.trimf(bulasik_miktari.universe, [50, 60, 100])

kirlilik['az'] = bulanik.trimf(kirlilik.universe, [0, 0, 30])
kirlilik['normal'] = bulanik.trimf(kirlilik.universe, [10, 30, 60])
kirlilik['çok'] = bulanik.trimf(kirlilik.universe, [50, 60, 100])

yikama['kisa'] = bulanik.trimf(yikama.universe, [0, 0, 50])
yikama['normal'] = bulanik.trimf(yikama.universe, [40, 50, 100])
yikama['uzun'] = bulanik.trimf(yikama.universe, [60, 80, 180])

kural1 = kontrol.Rule(bulasik_miktari['az'] & kirlilik['az'], yikama['kisa'])
kural2 = kontrol.Rule(bulasik_miktari['normal'] & kirlilik['az'], yikama['normal'])
kural3 = kontrol.Rule(bulasik_miktari['çok'] & kirlilik['az'], yikama['normal'])
kural4 = kontrol.Rule(bulasik_miktari['az'] & kirlilik['normal'], yikama['normal'])
kural5 = kontrol.Rule(bulasik_miktari['normal'] & kirlilik['normal'], yikama['uzun'])
kural6 = kontrol.Rule(bulasik_miktari['çok'] & kirlilik['normal'], yikama['uzun'])
kural7 = kontrol.Rule(bulasik_miktari['az'] & kirlilik['çok'], yikama['normal'])
kural8 = kontrol.Rule(bulasik_miktari['normal'] & kirlilik['çok'], yikama['uzun'])
kural9 = kontrol.Rule(bulasik_miktari['çok'] & kirlilik['çok'], yikama['uzun'])

sonuc = kontrol.ControlSystem([kural1, kural2, kural3, kural4, kural5, kural6, kural7, kural8,kural9])

model_sonuc = kontrol.ControlSystemSimulation(sonuc)

model_sonuc.input['bulasik miktari'] = 50
model_sonuc.input['kirlilik seviyesi']=80

```

```
model_sonuc.compute()  
print (model_sonuc.output['yükama süresi'])
```

Bulanık mantık Nerede neden kullandık ki şimdi

Bu çalışmada, PID katsayıları (K_p , K_i , K_d) için aynı ayrı bulanık mantık denetleyiciler tasarlanmıştır [7]. Her bir katsayı, bulanık mantık ile optimize edilerek kontrol performansı artırılmıştır. MATLAB/Simulink kullanılarak kontrol modeli oluşturulmuş ve test edilmiştir. Sonuçlar, bulanık mantık denetleyicilerin PID kontrolünde esnek ve adaptif bir performans sağladığını göstermektedir. Bu yaklaşım, manuel PID parametre ayarlamalarına olan ihtiyacı ortadan kaldırılmıştır. Şekil 2.3.2, tasarlanan kontrol modelini ve blok diyagramlarını göstermektedir.

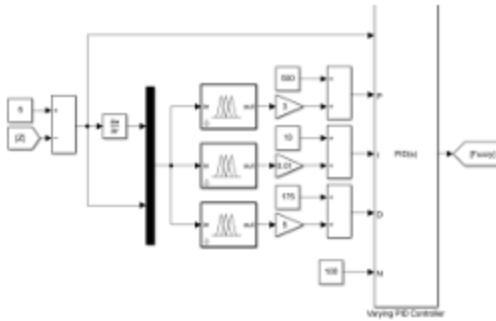


Figure 2.3.2

Üyelik fonksiyonlarının tanımlanması, bulanık mantık denetleyici tasarımları için kritik bir adımdır. Bu çalışmada, giriş değişkenleri için -6 ile $+6$ arasında tanımlanan üçgen üyelik fonksiyonları kullanılmıştır. Beş dil değişkeni tanımlanmıştır: NB (Negative Big), NS (Negative Small), ZO (Zero), PS (Positive Small), ve PB (Positive Big). ZE (Zero) üyelik fonksiyonu, daha yüksek hassasiyet sağlamak için keskin bir yapıdadır.

Bulanık mantık denetleyicilerde, giriş ve çıkış arasındaki ilişki IF-THEN kuralları ile oluşturulur[4]. Kurallar, sistemin kontrol performansını doğrudan etkileyen önemli bir faktördür. Ancak, dil değişkenlerinin sayısının çok fazla olması kural setinde karmaşılığa ve artan işlem yüküne neden olabilir. Bu nedenle, optimum bir denge sağlanarak beş dil değişkeni ile kontrol tasarımları yapılmıştır. Bu çalışmada oluşturulan kural tablosu[5], sistem performansını artırmak amacıyla dikkatle hazırlanmıştır ve Tablo 2.3.1'de gösterilmektedir.

Üyelik fonksiyonlarının üçgen formu, tasarımın basitliğini koruyarak hesaplama karmaşılığını azaltmıştır. Bu yaklaşım, bulanık mantık denetleyicinin hem giriş hem de çıkış değişkenleri için etkili bir kontrol sağlamıştır.

Table 1. Fuzzy rules table M_{μ}

\tilde{Z}	SC				
	NB	NS	ZO	PS	PB
NB	PB	PB	PS	PS	NB
NS	PB	PS	PS	NS	NB
ZO	ZD	ZD	ZD	ZD	ZD
PS	NB	NS	PS	PS	PB
PB	NS	NS	PS	PB	PB

Table 2. Fuzzy rules table M_{μ}

\tilde{Z}	SC				
	NB	NS	ZD	PS	PB
NB	PB	PB	PS	PS	NB
NS	PB	PS	NS	PS	NS
ZD	PS	ZD	ZD	ZD	NS
PS	PB	PS	NS	PS	PB
PB	NS	NS	PS	PB	PB

Table 3. Fuzzy rules table M_{μ}

\tilde{Z}	SC				
	NB	NS	ZD	PS	PB
NB	PB	PS	NS	NS	NS
NS	PB	PS	NS	PS	PS
ZD	PS	ZD	ZD	ZD	NS
PS	PB	PS	NS	PS	PB
PB	NS	NS	PS	PB	PB

Table 2.3.1

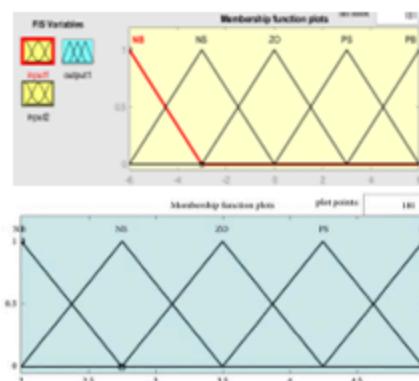


Figure 2.3.3

Bu çalışmada kullanılan üyelik fonksiyonları [6], FIS editörü üzerinde Şekil 2.3.3'de gösterilmektedir. Belirtilen kurallar ve üçgen üyelik fonksiyonları kullanılarak tasarlanan bulanık mantık denetleyicinin performansı, giriş ve çıkış sinyallerinin karşılaşırılmasıyla değerlendirilmiştir. Bulanık mantık denetleyici ile kontrol edilen sistemin kaplı çevrim cevabı ise Şekil 2.3.4'da sunulmaktadır.

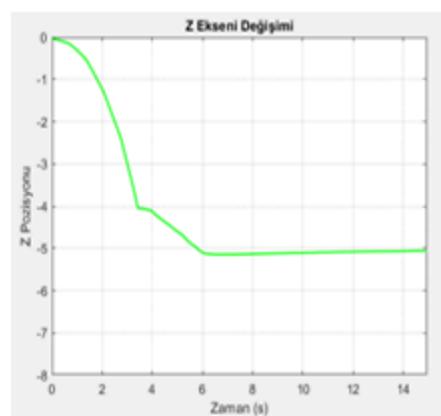


Figure 2.3.4

Fuzzy Logic burada neden devreye girer?

Bulanık mantık, PID katsayılarını:

- Matematiksel modele gerek duymadan
- İnsan sezgisine dayalı
- Gerçek zamanlı
- Adaptif

olarak belirler. PID'i "aklılı" hale getirir.

Fuzzy Logic neden Yapay Zekâdır?

Çünkü fuzzy:

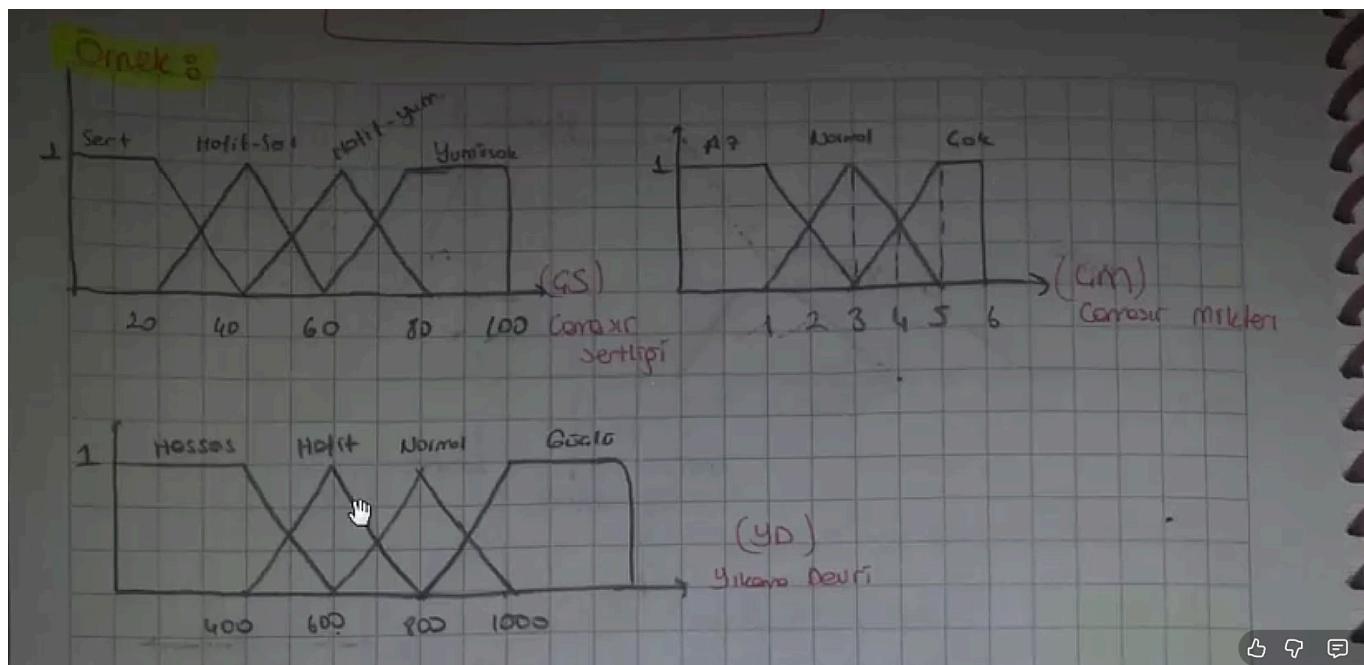
- Kesin kurallara bağlı değildir
- İnsan uzman bilgisini kullanır
- Belirsiz verilerle karar verir
- Doğal dil ile çalışır

Bu özellikler **yapay zekânın temel tanımıdır**.

Fuzzy Logic, ilk endüstriyel yapay zekâ yöntemlerinden biridir.

Youtube hesap örneği

https://www.youtube.com/watch?v=_H2NPo41vsg&t=776s



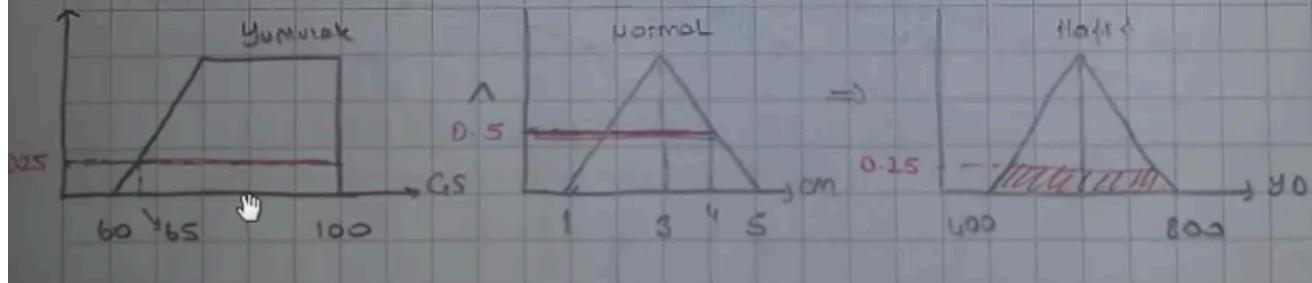
* Ginosur Miktarı : 6 kg olsun. \rightarrow Normal ve Çok
 * Ginosur settingi : ≈ 65 \rightarrow Hafif yum ve yumusak

* Normalde 2×2 'den 4 kural olmas gerekiyor. Bir 2 kural yeter.

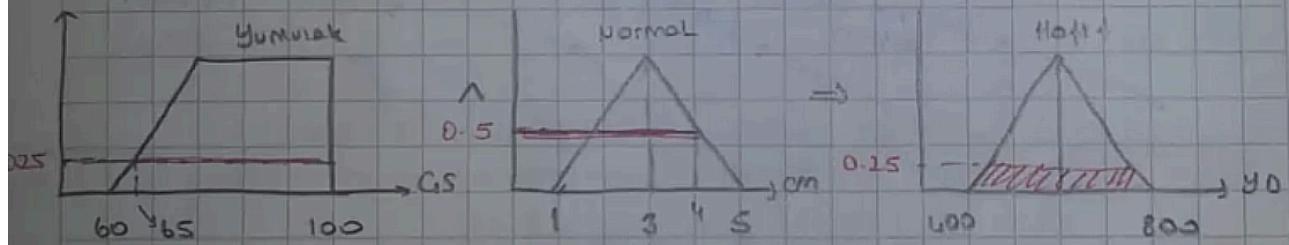
K1: GS "yumusak" ve CM "normal" ise YD "hafif" dir.

R2: GS "Hafif-yumusak" ve CM "normal" ise YD "normal" dir.

Kural 1:



Kural 1:



Kural 2:

