**Makalah**

**Fuzzy Logic**

**Fuzzy Associative Memory**

A close up of a sign

Description generated with high confidence

Dwiki Nurkurniawan Satya Putra

NPM. 140810150014

**Program Studi S-1 Teknik Informatika**

**Fakultas Matematika dan Ilmu Pengetahuan Alam**

**Universitas Padjadjaran**

**2018**

# KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Kuasa atas segala limpahan Rahmat, Inayah, Taufik dan Hinayahnya sehingga saya dapat menyelesaikan penyusunan makalah ini dalam bentuk maupun isinya yang sangat sederhana. Semoga makalah ini dapat dipergunakan sebagai salah satu acuan, petunjuk maupun pedoman bagi pembaca dalam pembelajaran maupun penerapan dari mata kuliah *fuzzy logic*.

Harapan penulis adalah makalah ini dapat membantu menambah pengetahuan dan pengalaman bagi para pembaca, sehingga penulis dapat memperbaiki bentuk maupun isi makalah ini sehingga kedepannya dapat lebih baik. Makalah ini penulis akui masih banyak kekurangan karena kurangnya pengalaman. Oleh kerena itu penulis berharap kepada para pembaca untuk memberikan masukan-masukan yang bersifat membangun untuk kesempurnaan makalah ini.

Sumedang, Juni 2018

Penyusun

# BAB I PENDAHULUAN

## Latar Belakang

Sistem pendukung keputusan hingga saat ini masih dibutuhkan dan dikembangkan. Biasanya, suatu sistem pendukung keputusan akan memberikan sebuah output setelah diberikan berbagai macam input yang berbentuk linguistik, seperti: “Jika ruangannya hangat, maka nyalakan kipas”, “Jika sebuah sepeda lebih cepat dari mobil, maka percepat mobil”, dll. Sedangkan dalam kenyataannya, pemrosesan data ini akan lebih mudah jika menggunakan data berbentuk numeris. Hal ini dapat diwujudkan dengan menggunakan sebuah metode yang bernama Fuzzy Associative Memory (FAM). Metode FAM ini akan memproses data dengan menggunakan teori dasar himpunan fuzzy dimana data akan diubah menjadi sebuah nilai dalam rentang [0, 1], dimana nilai tersebut merupakan sebuah data yang merepresentasikan pendapat secara kualitatif.

# BAB II PEMBAHASAN

Fuzzy Associative Memory (FAM) merupakan suatu sistem fuzzy yang memetakan himpunan-himpunan fuzzy ke himpunan-himpunan fuzzy lainnya. FAM merupakan sebuah versi fuzzy dari Bidirectional Associative Memory. Suatu sistem FAM bisa terdiri atas beberapa kumpulan FAM yang berbeda, contoh: (A1 ,B1), (A2, B2), … , (AP, BP).

## 2.1 Fuzzy Hebb FAM

Fuzzy Hebb FAM merupakan sebuah aturan pembelajaran yang dapat digunakan untuk mengkodekan kumpulan fuzzy (A,B) = ((a1, a2, … , an), (b1, b2, … , bp)) ke bentuk matriks FAM secara numeris. Fuzzy Hebb FAM dibagi menjadi 2 aturan pembelajaran, yaitu *correlation-minimum encoding,* dan *correlation-product encoding.*

### 2.1.1 Correlation-minimum encoding

Bentuk dari metode pembelajaran *correlation-minimum encoding* akan memberikan matriks korelasi FAM sebagai berikut:

M = AT ◦ B

Dengan:

mij = min (ai, bj)

dimana m­ij merupakan bobot dari node input ke-i ke node output ke-j.

Jika diasumsikan A dan B adalah himpunan fuzzy normal, dengan H(A) dan H(B) masing-masing merupakan tinggi himpunan fuzzy A dan tinggi himpunan fuzzy B, maka:

**Teorema 1:** Correlation-Minimum Hebb FAM

Jika M = AT ◦ B, maka:

* 1. A ◦ M = B, jika dan hanya jika H(A) ≥ H(B)
  2. B ◦ MT = A, jika dan hanya jika H(B) ≥ H(A)
  3. A’ ◦ M ⸦ B, untuk setiap A’,
  4. B’ ◦ M ⸦ A, untuk setiap B’

### 2.1.2 Correlation-product encoding

Bentuk dari metode pembelajaran *correlation-product encoding* akan memberikan matriks korelasi FAM sebagai berikut:

M = AT B

dengan

mij = ai \* bj

Jika diasumsikan A dan B adalah himpunan fuzzy normal, dengan H(A) dan H(B) masing-masing merupakan tinggi himpunan fuzzy A dan tinggi himpunan fuzzy B, maka:

**Teorema 2:** Correlation-Product Hebb FAM

Jika M = ATB, dengan A dan B adalah vektor nonnull, maka:

* 1. A ◦ M = B, jika dan hanya jika H(A) = 1,
  2. B ◦ MT = A, jika dan hanya jika H(B) = 1,
  3. A’ ◦ M ⸦ B, untuk setiap A’,
  4. B’ ◦ M ⸦ A, untuk setiap B’

## 2.2 Relasi Komposisi

Setelah mendapatkan nilai dari matriks M, maka selanjutnya kita akan mencari nilai B dengan menggunakan relasi komposisi dari matriks A dan M. Dengan cara yang sama, kita dapat mencari nilai A dengan menggunakan komposisi matriks M dan B. Terdapat 2 relasi komposisi, yaitu: *Max-min composition,* dan *max-product composition*

### 2.2.1 Max-min composition

Pada ­*max-min composition*, nilai B diperoleh dengan mencari nilai maksimum dari komposisi matriks A ◦ M sebagai berikut

B = A ◦ M

bj = max1 ≤ i ≤ n min (ai, mij)

Demikian pula, nilai A dapat diperoleh dengan mencari nilai maksimum dari komposisi matriks B ◦ MT sebagai berikut

A = B ◦ MT

ai = max1 ≤ j ≤ p min (bj, mij)

### 2.2.2 Max-product composition

Pada ­*max-product composition*, nilai B diperoleh dengan mencari nilai maksimum dari komposisi matriks AM sebagai berikut

B = AM

bj = max1 ≤ i ≤ n (ai \* mij)

Demikian pula, nilai A dapat diperoleh dengan mencari nilai maksimum dari komposisi matriks BMT sebagai berikut

A = BMT

ai = max1 ≤ j ≤ p (bj \* mij)

## 2.3 Superimposing FAM Rules

Jika kita memiliki beberapa kelompok FAM yang berbeda, seperti (A1, B1), (A2, B2), … , (An, Bn), maka kita dapat mencari nilai Bk’ dengan rumus ­*max-min composition* atau *max-product composition.* Setelah mendapat nilai Bk’ dan melakukan input vektor A, kita dapat mencari nilai dari vektor output B, dengan cara penjumlahan terbobot daris etiap Bk’, yaitu:

dengan bobot wk menunjukkan kekuatan aturan FAM ke-k. Namun pada prakteknya nilai bobot sering ditetapkan dengan 1. Agar nilai tiap anggota output B terletak dalam interval [0, 1], maka perlu dilakukan sebuah proses normalisasi yang dibagi menjadi 2 metode, yaitu:

1. *Winner take all.* Pada metode ini, nilai terbesar akan dianggap sebagai jawaban akhir
2. *Fuzzy centroid deffuzification.* Pada metode ini, output akan dicari dengan menggunakan rumus berikut
   1. , untuk y disket
   2. , untuk y kontinu

## 2.4 Contoh Kasus

Pada suatu persimpangan, digunakan sebuah algoritma untuk menentukan berapa lamanya lampu hijau akan menyala, dimana saat sedang sepi, maka lampu hijau akan menyala selama 5 detik, jika kedaan sedang akan menyala selama 20 detik, dan jika sedang padat akan menyala selama 60 detik. Sedangkan himpunan fuzzy untuk kepadatan berupa X = {0, 5, 8, …, 50} Berikut merupakan aturan fuzzy yang diberikan:

|  |  |
| --- | --- |
| A1 = (1, 0, 0) | B1 = (1, 0, 0) |
| A2 = (0, 1, 0) | B2 = (0, 1, 0) |
| A3 = (0, 0, 1) | B3 = (0, 0, 1) |
| A4 = (0.125, 0.25, 0.875) | B4 = (0, 0.2, 0.8) |
| A5 = (0.875, 0.25, 0.125) | B5 = (0.8, 0.2, 0) |

Dari aturan di atas, maka nilai matriks M dapat dicari untuk setiap aturan, pada kasus ini kita akan menggunakan *correlation-minimum encoding* sehingga didapat nilai M1, M2, M3, M4, M5.

Jika jumlah kendaaran yang sedang menunggu sebanyak 13, maka matriks input A’ akan menjadi A’ = (0.389, 0.700, 0.611), maka nilai B’ dapat diperoleh dengan menggunakan *max-min composition*, sebagai berikut

|  |
| --- |
| B1 = [0.389 0 0] |
| B2 = [0 0.700 0] |
| B3 = [0 0 0.611] |
| B4 = [0 0.200 0.611] |
| B5 = [0.389 0.200 0] |

Dari kumpulan nilai B’ di atas, kita dapat mencari nilai hasil output B. Pada kasus ini bobot setiap aturan akan dianggap sama, yaitu 1, dan untuk menentukan hasil akhirnya kita akan menggunakan metode *winner take all,* sebagai berikut:

b1 = max(0.389, 0, 0, 0, 0.389) = 0.389

b2 = max(0, 0.7, 0, 0.2, 0.2) = 0.7

b3 = max(0, 0, 0.611, 0.611, 0) = 0.611

atau B = (0.389, 0.7, 0.611).

Dikarenakan kita menggunakan metode *winner take all*, maka nilai B\* akan menjadi 20 dikarenakan nilai terbesar terdapat pada b2 (0.7). Apabila kita menggunakan metode *weighted average,* maka nilai B\* akan menjadi:

Jadi, jika kita menggunakan metode *weighted average,* maka nilai B\* akan menjadi 30.944 atau dalam kasus ini lampu hijau akan menyala selama 30.944 detik.

# BAB III PENUTUP

## 3.1 Kesimpulan

Dari paparan di atas, maka dapat disimpulkan bahwa Fuzzy Associative Memory (FAM) merupakan sebuah metode fuzzy untuk memecahkan sebuah masalah yang sering kita hadapi sehari-hari. FAM memiliki beberapa metode di dalamnya, yaitu *correlation-minimum encoding, correlation-product encoding, max-min composition, max-product composition.* Selain itu, FAM dapat menggabungkan banyak aturan ke dalam suatu program untuk menggabungkan beberapa input.

## 3.2 Saran

Menyadari bahwa penulis masih jauh dari sempurna, untuk kedepannya penulis akan lebih fokus dan detail dalam menjelaskan tentang makalah di atas dengan sumber-sumber lain yang lebih banyak dan juga contoh kasus yang lebih mutakhir.

# DAFTAR PUSTAKA

Kusumadewi, Sri dan Hari Purnomo, 2004, “Aplikasi Logika Fuzzy Untuk Pendukung Keputusan”, Yogyakarta: Graha Ilmu.

# LAMPIRAN

Berikut code program untuk FAM dalam bahasa pemrograman Python, library yang digunakan adalah numpy :

|  |
| --- |
| import numpy as np  # Correlation Minimum Encoding  def cme(a, b):  m = np.zeros((len(a), len(b)))  a = a.T  for x in range(len(b)):  for y in range(len(a)):  m[y, x] = min(a[y], b[x]);  return m  # Correlation Product Encoding  def cpe(a, b):  m = np.zeros((len(a), len(b)))  a = a.T  for x in range(len(b)):  for y in range(len(a)):  m[y, x] = a[y] \* b[x]  return m  # Max-min composition  def mmc(array, m, find):  if 'b' in find:  temp = np.zeros(array.shape[0])  hasil = np.zeros(m.shape[1])  for j in range(m.shape[1]):  for i in range(array.shape[0]):  temp[i] = min(array[i], m[i, j])  hasil[j] = max(temp)  elif 'a' in find:  m = m.T  temp = np.zeros(array.shape[0])  hasil = np.zeros(m.shape[1])  for j in range(m.shape[1]):  for i in range(m.shape[0]):  temp[i] = min(array[i], m[i, j])  hasil[j] = max(temp)  return hasil    # Max-Product composition  def mpc(array, m, find):  if 'b' in find:  temp = np.zeros(array.shape[0])  hasil = np.zeros(m.shape[1])  for j in range(m.shape[1]):  for i in range(array.shape[0]):  temp[i] = array[i] \* m[i, j]  hasil[j] = max(temp)  elif 'a' in find:  m = m.T  temp = np.zeros(array.shape[0])  hasil = np.zeros(m.shape[1])  for j in range(m.shape[1]):  for i in range(m.shape[0]):  temp[i] = array[i] \* m[i, j]  hasil[j] = max(temp)  return hasil  # Superimposing FAM rules - max min  def super\_cme(a, b, a\_aksen):  m = np.zeros((len(a), len(a[0]), len(b[0])))  B = np.zeros((len(a), len(b[0])))  b\_ang = np.zeros(len(b[0]))  temp = np.zeros((len(b[0]), len(b[0])))  for x in range(len(a)):  m[x] = cme(a[x], b[x])  B[x] = mmc(a\_aksen, m[x], 'b')  if x < len(b[0]):  for y in range(len(b[0])):  temp[x, y] = B[y, x]  b\_ang[x] = max(temp[x])  return m, b\_ang, max(b\_ang)  # Superimposing FAM rules - contoh 2  def super\_cpe(a, b, a\_aksen):  m = np.zeros((len(a), len(a[0]), len(b[0])))  B = np.zeros((len(a), len(b[0])))  b\_ang = np.zeros(len(b[0]))  for x in range(len(a)):  m[x] = cme(a[x], b[x])  B[x] = np.dot(a\_aksen, m[x])  for x in range(len(a)):  if x < len(b[0]):  for y in range(len(b[0])):  b\_ang[x] += B[x, y];  return m, b\_ang, max(b\_ang)  # KASUS 2  # ==============================  # A = np.array([  # [1, 0, 0, 1, 0, 0, 1, 0, 0],  # [1, 0, 0, 1, 0, 0, 0, 1, 0],  # [1, 0, 0, 1, 0, 0, 0, 0, 1],  # [1, 0, 0, 0, 1, 0, 1, 0, 0],  # [1, 0, 0, 0, 1, 0, 0, 1, 0],  # [1, 0, 0, 0, 1, 0, 0, 0, 1],  # [1, 0, 0, 0, 0, 1, 1, 0, 0],  # [1, 0, 0, 0, 0, 1, 0, 1, 0],  # [1, 0, 0, 0, 0, 1, 0, 0, 1],  # [0, 1, 0, 1, 0, 0, 1, 0, 0],  # [0, 1, 0, 1, 0, 0, 0, 1, 0],  # [0, 1, 0, 1, 0, 0, 0, 0, 1],  # [0, 1, 0, 0, 1, 0, 1, 0, 0],  # [0, 1, 0, 0, 1, 0, 0, 1, 0],  # [0, 1, 0, 0, 1, 0, 0, 0, 1],  # [0, 1, 0, 0, 0, 1, 1, 0, 0],  # [0, 1, 0, 0, 0, 1, 0, 1, 0],  # [0, 1, 0, 0, 0, 1, 0, 0, 1],  # [0, 0, 1, 1, 0, 0, 1, 0, 0],  # [0, 0, 1, 1, 0, 0, 0, 1, 0],  # [0, 0, 1, 1, 0, 0, 0, 0, 1],  # [0, 0, 1, 0, 1, 0, 1, 0, 0],  # [0, 0, 1, 0, 1, 0, 0, 1, 0],  # [0, 0, 1, 0, 1, 0, 0, 0, 1],  # [0, 0, 1, 0, 0, 1, 1, 0, 0],  # [0, 0, 1, 0, 0, 1, 0, 1, 0],  # [0, 0, 1, 0, 0, 1, 0, 0, 1]]  # )  # B = np.array([  # [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],  # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]]  # )  # # b\_aksen = np.array([0.25, 0.75, 0, 0, 0.267, 0.733, 0, 0.75, 0.25])  # b\_aksen = np.array([0,1,0,0,0,1,0,1,0])  # nilaiBDiskret = np.array([  # 148.00, 150.90, 146.50, 143.10, 146.53,  # 142.73, 136.73, 140.77, 135.97, 149.73,  # 153.27, 152.13, 148.00, 150.63, 147.63,  # 141.67, 145.67, 140.20, 142.10, 146.53,  # 142.17, 138.70, 141.40, 138.30, 133.33,  # 138.53, 133.77])    # fam, output, max\_value = super\_cpe(A, B, b\_aksen)  # CONTOH  # ==============================  # A = np.array([  # [1, 0, 0],  # [0, 1, 0],  # [0, 0, 1],  # [0.125, 0.25, 0.875],  # [0.875, 0.25, 0.125]]  # )  # B = np.array([  # [1, 0, 0],  # [0, 1, 0],  # [0, 0, 1],  # [0, 0.2, 0.8],  # [0.8, 0.2, 0]]  # )  # b\_aksen = np.array([0.389, 0.700, 0.611])  # nilaiBDiskret = np.array([5, 20, 60])    # fam, output, max\_value = super\_cme(rulesA, rulesB, b\_aksen)  index = output.tolist().index(max\_value)  print('Matriks FAM\n', fam)  print('Hasil output\n', output)  print('Hasil Defuzzyfication\n', nilaiBDiskret[index]) |