Fuzzy Mamdani Control System for Movie Recommendation System

Kruti Shah

Whiting School of Engineering, Johns Hopkins University, Baltimore, MD, USA

Abstract

Movie recommendation systems play a pivotal role in helping users discover content tailored to their preferences. Traditional recommendation systems often rely on statistical approaches, but there's a fascinating alternative—Fuzzy Mamdani Control Systems. This paper explores the integration of Fuzzy Logic into movie recommendations, utilizing the renowned Mamdani model and comparing it with the traditional recommendation system. The approach studied here is content-based filtering, they are solely based on using item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback. It is pointed out that these reclusive methods rather than being competitive with collaborative methods are complementary.

1. Introduction

Over time, movie recommendation systems have undergone significant evolution, influencing our experiences with digital entertainment. The transition from conventional methods, which are based on statistical techniques and machine learning models, to state-of-the-art approaches such as the Fuzzy Mamdani Controller system has been characterized by significant advancements and prominent investigators' contributions.

Conventional Methods: Content-based filtering dominated movie recommendation systems in the early 2000s. This method included elements like genre, director, actors, and user ratings in its analysis of intrinsic movie qualities and user preferences. Favored by scholars such as Yehuda Koren and Robert Bell, machine learning models, and statistical techniques played a pivotal role in developing recommendation engines predicated on these attributes.

Evolution of Techniques: The emergence of collaborative filtering in the mid-2000s brought about a paradigm shift by placing more emphasis on predictions made by users with similar tastes. Collaborative filtering techniques have become increasingly popular due to the groundbreaking work of researchers such as Carl Kadie, David Heckerman, and John S. Breese. These techniques worked well in situations where there were few explicit movie features.

In the late 2000s, as technology developed, hybrid recommendation systems appeared, combining the benefits of collaborative and content-based filtering. Algorithm advancements and the expansion of big data sources were crucial in improving these systems to provide a wider range of more precise recommendations.

The Fuzzy Mamdani Controller system has become a ground-breaking method of movie recommendation in recent years. Lotfi A. Zadeh invented fuzzy logic, which has grown to be a

mainstay for managing ambiguity and imprecision in decision-making processes. Fuzzy logic, including Mamdani's fuzzy inference system, has been applied to various fields for decision-making and control since the 1970s. Now that fuzzy control techniques have been developed, it is worthwhile to reexamine Mamdani's method of converting linguistic knowledge into a numerical function that may be applied as a control law [2]. This method originally originated with a work by Zadeh [1] from 1973, which described a method for modeling fuzzy if-then rules related to numerical universes. This point of view interpreted an if-then statement that related quantities in many universes as a Cartesian product as opposed to a material implication. The decision taken by Mamdani and Zadeh had a significant influence on the further advances of fuzzy logic in information processing and engineering.

2. Fuzzy Mamdani Control System

The Fuzzy Mamdani Control System is a type of fuzzy logic controller developed by Lotfi A. Zadeh. It's designed to handle complex systems by incorporating the concept of "fuzziness" to model uncertainty and imprecision in decision-making processes. The Mamdani model, specifically, is a fuzzy inference system that uses linguistic variables and rules to make decisions.

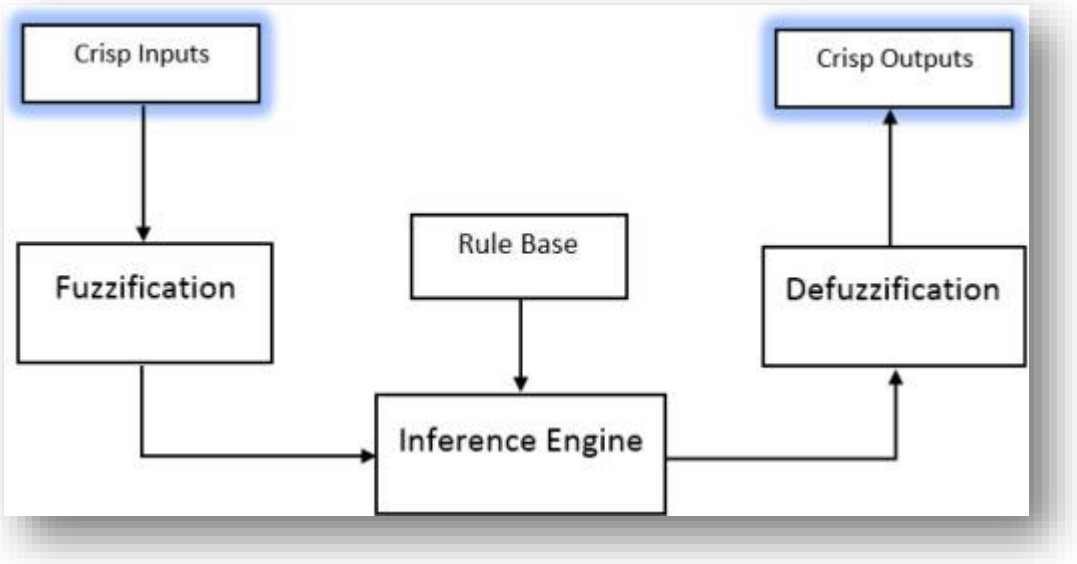


Fig. 1 A schematic of the Mamdani Control System [3]

Here's a step-by-step explanation of the Fuzzy Mamdani Control System:

- Fuzzification:

  The first step involves converting crisp input values (quantitative data) into fuzzy sets. Fuzzy sets allow for the representation of vague or imprecise information. Linguistic variables are used to describe these fuzzy sets. For example, if the input is temperature, fuzzy sets like "cold," "warm," and "hot" might be defined.

- Rule Base:

  The system relies on a set of rules that define the relationship between the fuzzy input variables and the fuzzy output variables. These rules are typically in the form of "if-then" statements and capture the expert knowledge or heuristic understanding of the system. Each rule combines one or more conditions and specifies an action to be taken.

- Inference Engine:

  Fuzzy rules are used to the fuzzy input values during the inference phase to produce fuzzy output values. Based on the input values, this step entails calculating the extent to which each rule is satisfied. A collection of hazy output values connected to every rule is the end outcome.

  Theoretically, the number of rules in the inference system is calculated as follows:[3]

$$R = \prod_{i=1}^{n} FS_i \qquad (1)$$

  where R is the number of rules, n is the number of input variables, and FSi denotes the number of fuzzy sets assigned to the i'th input variable. As implied by Eq. (1), the number of rules directly depends on the number of input variables and fuzzy sets in the system.

- Defuzzification:

  Creating a crisp output from the fuzzy output values is the last stage. We refer to this procedure as defuzzification. Defuzzification can be accomplished using a variety of techniques, including centroid, mean of maximum, and others.

When there are complex relationships between inputs and outputs and conventional control methods may not be able to fully capture the nuances of the system, the Fuzzy Mamdani Control System can be especially helpful. It has found use in domains like artificial intelligence, decision support, and control systems, offering a framework for managing ambiguity and imprecision in a computationally efficient manner.

3. Methodology

The primary challenge in building the Mamdani Control system is related to the formulation of accurate and effective fuzzy rules. Designing a rule base that adequately captures the nuanced relationships between fuzzy input variables (such as user preferences, and movie features) and fuzzy output variables (recommendation scores) is crucial.

3.1 Requirements:

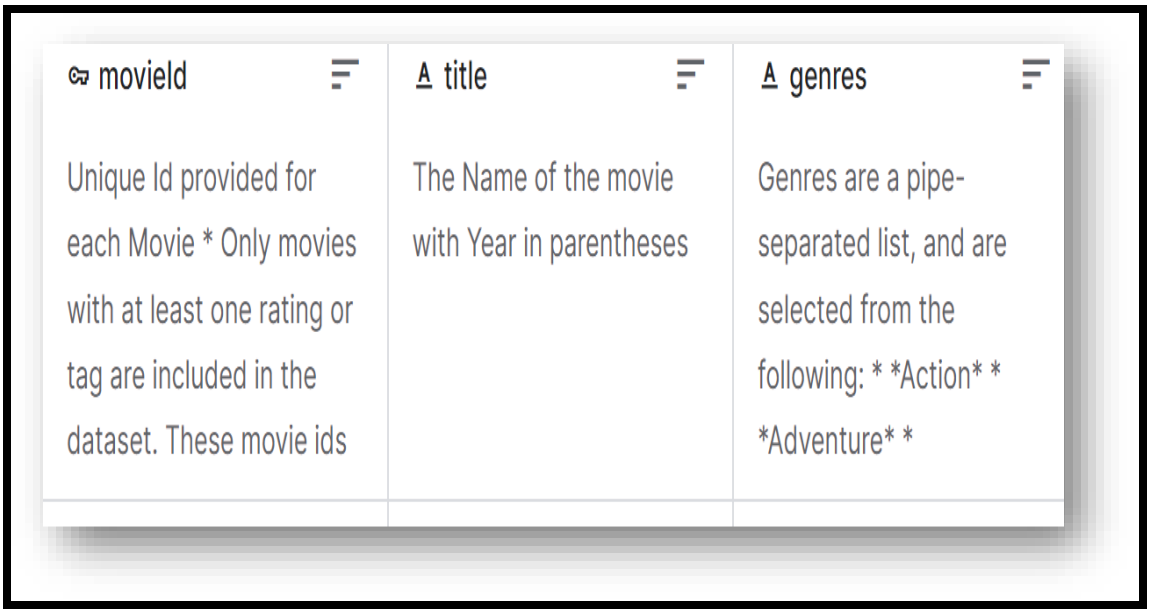
The program requires Python 3.7.4. The **scikit-fuzzy** library [5] needs to be installed which is a collection of fuzzy logic algorithms intended for use in the SciPy Stack, written in the Python computing language.

3.2 Dataset:

The dataset used here for the Movie Recommender System is the 'Movie Lens Dataset'. This dataset is available on Kaggle and can be downloaded from the same.[6]

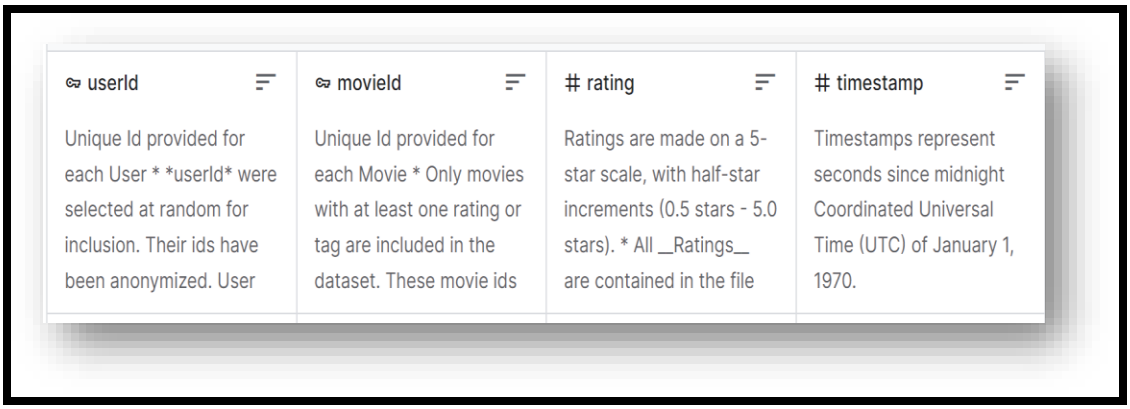
There are 4 .csv files available in this dataset. Only two dataset files are used in this project which are downloaded from the above Kaggle website. The two dataset files are:

3.2.1 movies.csv: Movie information is contained in the file movies.csv. Each line of this file after the header row represents one movie, and has the following format:



movieId	title	genres
Unique Id provided for each Movie * Only movies with at least one rating or tag are included in the dataset. These movie ids	The Name of the movie with Year in parentheses	Genres are a pipe-separated list, and are selected from the following: * *Action* * *Adventure* *

3.2.2 ratings.csv: All ratings for the movies are contained in the file ratings.csv. Each line of this file after the header row represents one rating of one movie by one user, and has the following format:



userId	movieId	# rating	# timestamp
Unique Id provided for each User * *userId* were selected at random for inclusion. Their ids have been anonymized. User	Unique Id provided for each Movie * Only movies with at least one rating or tag are included in the dataset. These movie ids	Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars). * All __Ratings__ are contained in the file	Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

3.3 Data Preprocessing:

- The 'title' has the movie's name and the year it was released. The year of release is scraped off from the title and stored in another column 'year'.
- The ‘genres’ are pipeline separated. The pipeline is removed and it is stored as a list, for eg: ‘Adventure|Animation|Children|Comedy|Fantasy’ is changed to [‘Adventure’, ’Animation’, ‘Children’, ’Comedy’, ‘Fantasy’]
- Later, the 'genres' are each one-hot encoded, and a column for each genre is created. If the genre is present in the ‘genre’ list then the value is 1 otherwise it is 0.
- The timestamp from the ‘ratings.csv’ file is dropped.
- The input data used in this experiment is as below. It is noted that all five movies have 'Action' and 'Adventure' as genres.

	movieId	title	genres	year
0	1	Toy Story	[Adventure, Animation, Children, Comedy, Fantasy]	1995
1033	1274	Akira	[Action, Adventure, Animation, Sci-Fi]	1988
10158	122886	Star Wars: Episode VII - The Force Awakens	[Action, Adventure, Fantasy, Sci-Fi]	2015
10247	134130	Martian, The	[Action, Adventure, Sci-Fi]	2015
10262	135532	The Last Witch Hunter	[Action, Adventure, Fantasy]	2015

3.4 Fuzzy Control System:

The Fuzzy Control System includes three blocks:

a) Fuzzification, b) Inference Engine, c) Defuzzification.

3.4.1 Fuzzification:

For this system, I decided to use two types of input variables (*aka antecedent*): the movie's genres and rating from ratings.csv for that movie. The user's input above shows that all 5 movies have 'Action' or 'Adventure' as their genre. In the interest of time, to narrow the scope of the project and to minimize the number of rules to be coded, I am considering only two genres i.e. 'Action' and 'Adventure'.

The output variable (*aka consequent*) is the 'recommendation percentage'.

```
# Input variables
rating = ctrl.Antecedent(np.arange(0, 6, 1), 'rating')
genre_action = ctrl.Antecedent(np.arange(0, 2, 1), 'genre_action')  # 0 or 1 for 'Action'
genre_adventure = ctrl.Antecedent(np.arange(0, 2, 1), 'genre_adventure')  # 0 or 1 for 'Adventure'

# Output variable
recommendation = ctrl.Consequent(np.arange(0, 101, 1), 'recommendation')
```

Because this is a demonstration case, I wanted to show how you can specify different membership functions. For this, I used three types of membership *Triangular, Gaussian, and Automatic*. The Skit-Fuzzy module in Python allows for the automatic generation of the triangular membership function. For this, you need to specify only the number of the terms and the terms themselves. Now, let's briefly discuss the differences between gaussmf, trimf, and automf.

gaussmf (Gaussian Membership Function): It represents a Gaussian-shaped membership function. It takes three parameters: the mean (center of the curve), the standard deviation (width of the curve), and the input universe.

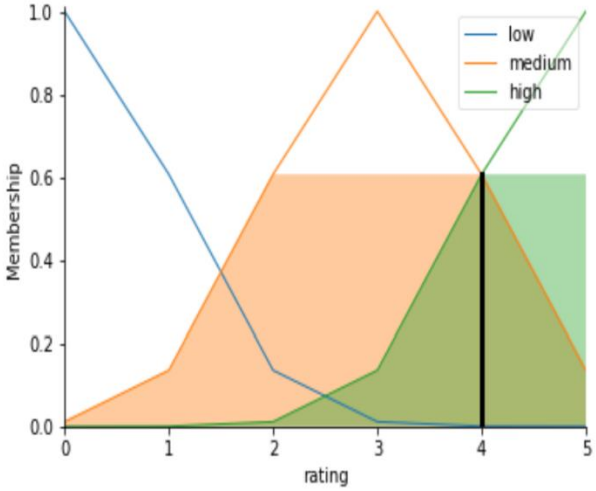


Fig.2 A Gaussian Membership function visualization for rating

trimf (Triangular Membership Function): It represents a triangular-shaped membership function. It takes three parameters: the lower bound, the peak (mode), and the upper bound of the triangle.

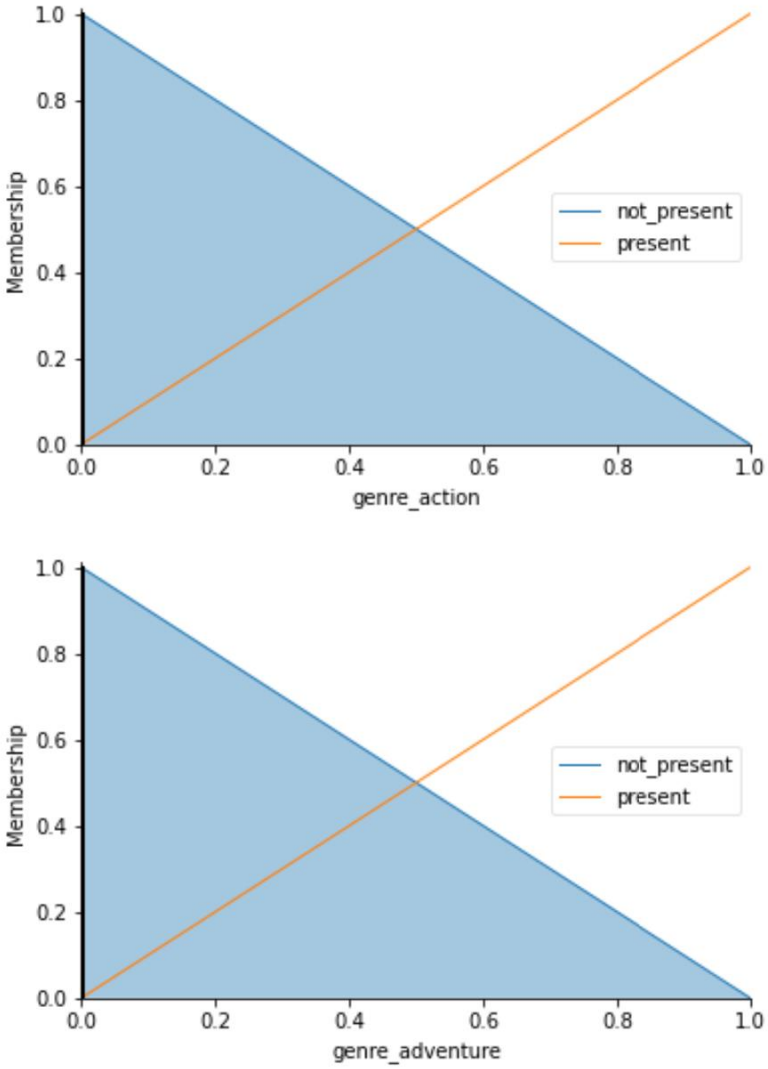


Fig.3 A triangular Membership function visualization for Genre 'Action & 'Adventure'

automf (Automatic Membership Function): It is a convenience function in scikit-fuzzy that automatically generates membership functions based on predefined shapes (like 'poor', 'mediocre', 'average', 'good', 'excellent'). It is often used for quick and simple fuzzy systems where you don't want to manually specify membership function parameters.

In the provided code, I used gaussmf for rating, trimf for binary genre variables, and trimf for the recommendation output. You can choose the appropriate membership function based on the characteristics you want to capture for each variable in your fuzzy system.

```
# Define membership functions using gaussmf
rating['low'] = fuzz.gaussmf(rating.universe, 0, 1)
rating['medium'] = fuzz.gaussmf(rating.universe, 3, 1)
rating['high'] = fuzz.gaussmf(rating.universe, 5, 1)

# Define membership functions using trimf
genre_action['not_present'] = fuzz.trimf(genre_action.universe,[0, 0, 1])
genre_action['present'] = fuzz.trimf(genre_action.universe, [0, 1, 1])

genre_adventure['not_present'] = fuzz.trimf(genre_adventure.universe,[0, 0, 1])
genre_adventure['present'] = fuzz.trimf(genre_adventure.universe, [0, 1, 1])


# Define membership functions using trimf
recommendation['low'] = fuzz.trimf(recommendation.universe, [0, 0, 50])
recommendation['medium'] = fuzz.trimf(recommendation.universe, [10, 50, 90])
recommendation['high'] = fuzz.trimf(recommendation.universe, [50, 100, 100])
```

3.4.2 Inference Engine:

Here, we need to specify the rules by which the fuzzy inputs are connected to the output fuzzy output. The rules look like this; we define the fuzzy relationship between input and output variables.

For this project, we are considering only two genres and ratings i.e., 3 input variables to limit the number of rules. Consider the below rules:

(i) If the rating is low and the 'Action' genre and 'Adventure' genre are not present, then the recommendation is low.
(ii) If the rating is low and the 'Action' genre is not present and the 'Adventure' genre is present, then the recommendation is low.
(iii) If the rating is low and the 'Action' genre is present and the 'Adventure' genre is not present, then the recommendation is low.
(iv) If the rating is low and the 'Action' genre and 'Adventure' genre are present, then the recommendation is medium.

In a similar way, 4 rules for each of the 'medium' and 'high' ratings. The complete list can be seen in the below snapshot of the code.

The rule base of the current system includes 12 rules in total. It is important to ensure the input values activate at least one of the rules in the rule base, otherwise, you will get an empty set. The rules for this system were derived somewhat arbitrarily. In reality, there are rigorous approaches to do that. There is also a way of deriving these rules from the data (e.g., fuzzy adaptive system, fuzzy neural networks).

```
# Rule base
rule1 = ctrl.Rule(rating['low'] & genre_action['not_present'] & genre_adventure['not_present'], recommendation['low'])
rule2 = ctrl.Rule(rating['low'] & genre_action['not_present'] & genre_adventure['present'], recommendation['low'])
rule3 = ctrl.Rule(rating['low'] & genre_action['present'] & genre_adventure['not_present'], recommendation['low'])
rule4 = ctrl.Rule(rating['low'] & genre_action['present'] & genre_adventure['present'], recommendation['medium'])

rule5 =  ctrl.Rule(rating['medium'] & genre_action['not_present'] & genre_adventure['not_present'], recommendation['low'
rule6 = ctrl.Rule(rating['medium'] & genre_action['not_present'] & genre_adventure['present'], recommendation['medium'])
rule7 = ctrl.Rule(rating['medium'] & genre_action['present'] & genre_adventure['not_present'], recommendation['medium'])
rule8 = ctrl.Rule(rating['medium'] & genre_action['present'] & genre_adventure['present'], recommendation['high'])

rule9 =  ctrl.Rule(rating['high'] & genre_action['not_present'] & genre_adventure['not_present'], recommendation['low'])
rule10 = ctrl.Rule(rating['high'] & genre_action['not_present'] & genre_adventure['present'], recommendation['medium'])
rule11 = ctrl.Rule(rating['high'] & genre_action['present'] & genre_adventure['not_present'], recommendation['medium'])
rule12 = ctrl.Rule(rating['high'] & genre_action['present'] & genre_adventure['present'], recommendation['high'])
```

The rules include two operators: AND and *OR*. The *And* operator implies a *Minimum* operation on the respective set and *the OR* operator implies a *Maximum* operation on the set. The below rules use only the AND operator.

```
In [54]:  1 rule1.view()

Out[54]: (<Figure size 432x288 with 1 Axes>, <AxesSubplot:>)
```

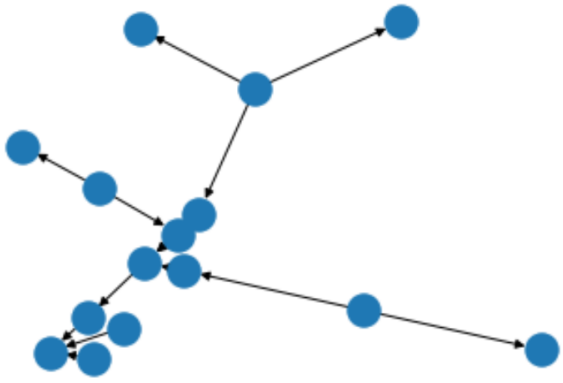


Fig. 4 A visualization of the rule1 in the inference engine

3.4.3 Defuzzification:

Defuzzification is the process of representing a fuzzy set with a crisp number. Internal representations of data in a fuzzy system are usually fuzzy sets. But the output frequently needs to be a crisp number that can be used to perform a function such as commanding a valve to a desired position in a control application or indicating a recommendation percentage as discussed here.[7]
The most commonly used defuzzification method is the center of area method (COA), also commonly referred to as the centroid method. This method determines the center of the area of the fuzzy set and returns the corresponding crisp value. The center of sums (COS) method and the mean of maximum method are two alternative methods in defuzzification.[7]

Now that we have our rules defined, we can simply create a control system via:

```
# Control system
recommendation_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9, rule10, rule11,
```

To simulate this control system, we will create a `ControlSystemSimulation`. Think of this object representing our controller applied to a specific set of circumstances. For recommendation, this might be recommending a movie to Karen. We would create another `ControlSystemSimulation` when we're trying to apply our recommendation_ctrl for Travis for TV shows because the inputs would be different.

```
recommendation_sys = ctrl.ControlSystemSimulation(recommendation_ctrl)
```

We can now simulate our control system by simply specifying the inputs and calling the `compute` method.

Suppose we have a rating = 4.2 and has both Action and Adventure genre present
Pass inputs to the ControlSystem using Antecedent labels with Pythonic API
Note: if you like passing many inputs all at once, use inputs(dict_of_data)

```
recommendation_sys.input['rating']   = 4.5
recommendation_sys.input['genre_action'] = 1
recommendation_sys.input['genre_adventure'] = 1

recommendation_sys.compute()

# Visualize membership functions and the simulation
for var in recommendation_sys.ctrl.antecedents:
    var.view(sim=recommendation_sys)

for var in recommendation_sys.ctrl.consequents:
    var.view(sim=recommendation_sys)

# Show the plots
plt.show()
```

Once computed, we can view the result as well as visualize it.

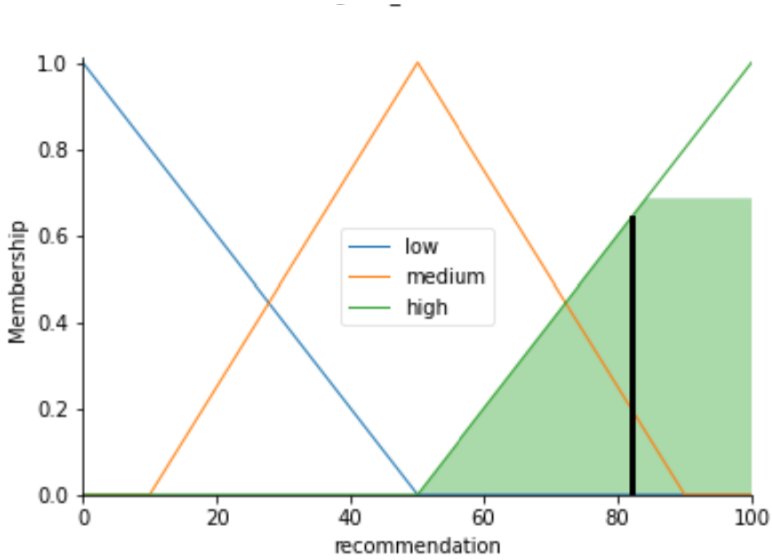
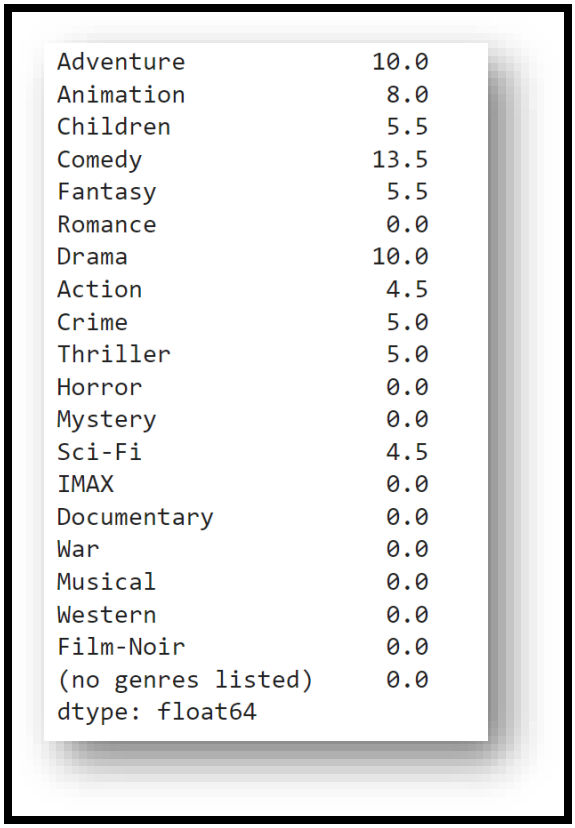


Fig. 5 A Triangular Membership function visualization for recommendation

3.5 Traditional Statistical Approach:

The first step is to create a user profile. This is done by multiplying the one hot encoding value of the genres with the rating. This will be done for each of the shortlisted movies.

Once this multiplication is done we will add up the values of each genre. We get a single value which will be the user preference weightage for a particular genre as given below.



Adventure	10.0
Animation	8.0
Children	5.5
Comedy	13.5
Fantasy	5.5
Romance	0.0
Drama	10.0
Action	4.5
Crime	5.0
Thriller	5.0
Horror	0.0
Mystery	0.0
Sci-Fi	4.5
IMAX	0.0
Documentary	0.0
War	0.0
Musical	0.0
Western	0.0
Film-Noir	0.0
(no genres listed)	0.0

dtype: float64

The above table is called a user profile as this is for a particular user.
The final step in recommender is that we take the one hot-encoded genre table for each movie and multiply the genre with the above score for each genre. We do that for all movies and then we sum up horizontally, that is the sum of all genres for that movie, and then normalize it by dividing by the above score.[8]

Normalized Sum = $$\frac{\sum_{i=1}^{n}(g_i \cdot s_i)}{\sum_{i=1}^{n} s_i} \quad (2)$$

In this equation:

- *n* is the total number of genres.
- *gi* represents the binary-encoded value for genre *i* (0 or 1) for a particular movie.
- *si* represents the score associated with genre *i*.
- The numerator represents the sum of the element-wise multiplication of the genre binary values with their corresponding scores, summed horizontally across all genres.
- The denominator represents the sum of all genre scores.

This equation captures the process of multiplying the binary-encoded genre table with the genre scores, summing horizontally, and then normalizing by dividing by the sum of the genre scores.

Then sort the movie on this score and choose the top 10 movies for recommendation.

4 Results and discussion

4.1 The top ten recommendation results from the Fuzzy Control System:

	movieId	title	genres	year	recommendation_percentage
1	5720	Friend Is a Treasure, A (Chi Trova Un Amico, T...	[Action, Adventure, Comedy]	1981	83.333013
5	31367	Chase, The	[Action, Adventure, Comedy, Crime, Romance, Th...	1994	83.333013
8	108979	Cowboy Bebop	[Action, Adventure, Animation, Crime, Sci-Fi]	1998	82.936013
2	6721	Once Upon a Time in China (Wong Fei Hung)	[Action, Adventure, Drama]	1991	82.779293
3	26012	Samurai III: Duel on Ganryu Island (a.k.a. Bus...	[Action, Adventure, Drama]	1956	82.779293
4	27155	Batman/Superman Movie, The	[Action, Adventure, Animation, Children, Fanta...	1998	82.779293
6	66785	Good, the Bad, the Weird, The (Joheunnom nabbe...	[Action, Adventure, Comedy, Western]	2008	82.779293
7	103210	Fullmetal Alchemist: The Sacred Star of Milos	[Action, Adventure, Animation]	2011	82.779293
9	110216	BloodRayne: The Third Reich	[Action, Adventure, Fantasy, Horror]	2011	82.779293
0	2905	Sanjuro (Tsubaki Sanjûrô)	[Action, Adventure, Drama]	1962	82.628177

4.2 The top ten recommendation results from statistical methods:

	movieId	title	genres	year	recommendation_percentage
0	6350	Laputa: Castle in the Sky (Tenkû no shiro Rapy...	[Action, Adventure, Animation, Children, Fanta...	1986	95.145631
1	27155	Batman/Superman Movie, The	[Action, Adventure, Animation, Children, Fanta...	1998	95.145631
3	52462	Aqua Teen Hunger Force Colon Movie Film for Th...	[Action, Adventure, Animation, Comedy, Fantasy...	2007	95.145631
5	62956	Futurama: Bender's Game	[Action, Adventure, Animation, Comedy, Fantasy...	2008	95.145631
2	27608	Immortel (ad vitam) (Immortal)	[Action, Adventure, Animation, Fantasy, Sci-Fi]	2004	90.291262
4	58404	Justice League: The New Frontier	[Action, Adventure, Animation, Fantasy, Sci-Fi]	2008	90.291262
6	71129	Green Lantern: First Flight	[Action, Adventure, Animation, Fantasy, Sci-Fi]	2009	90.291262
7	80083	Dragon Ball Z: Dead Zone (Doragon bôru Z 1: Or...	[Action, Adventure, Animation, Fantasy, Sci-Fi]	1989	90.291262
8	103659	Justice League: The Flashpoint Paradox	[Action, Adventure, Animation, Fantasy, Sci-Fi]	2013	90.291262
9	108501	Justice League: War	[Action, Adventure, Animation, Fantasy, Sci-Fi]	2014	90.291262

Our analysis reveals a notable discrepancy in the top ten recommendations generated by the Fuzzy Control System and the traditional Statistical approach. This discrepancy is likely attributed to the difference in the number of features considered by each approach.

Specifically, the traditional approach incorporates all 20 genres in its calculations, while the Fuzzy Control System focuses on only two genres. Considering the inclusion of 20 genres and one rating, the traditional approach involves 21 features, resulting in a calculation complexity of (20 x 2) x (3 x 1) = 120.

To mitigate the challenge posed by the increasing number of features in the Fuzzy Control System, several methodologies can be explored. These include Rule Base Reduction [10], Rule Aggregation, Rule Interpolation, Fuzzy Clustering, Machine learning methods [9], and Hierarchical Fuzzy Systems. These methods offer potential avenues for future research aimed at reducing the number of rules and enhancing the efficiency of the Fuzzy Control System.

5 Conclusion

Our study introduces an enhanced approach to recommendation systems through the utilization of an alternative Fuzzy Mamdani Control System. The outcomes of our investigation exhibit significant promise, and we posit that further improvements can be achieved by incorporating additional features and techniques outlined in Section 6.

The Fuzzy Mamdani Control System offers a straightforward implementation, particularly due to the simplicity of rules expressed in plain English. However, a notable drawback in the inference system arises as the number of rules grows exponentially with the inclusion of more features.

In light of these findings, our study not only underscores the potential of the proposed alternative Fuzzy Mamdani Control System but also acknowledges the challenge posed by the escalating number of rules, especially in correlation with an increasing number of features. The insights gained from this research provide a foundation for future investigations aimed at addressing this challenge and optimizing the performance of recommendation systems.

6 Future Study

- We can consider more features of the movie such as all genres, actors, directors, and description of the movie for enhancing recommendation accuracy in movie recommendations.
- Since the greater number of features will pose the exponential growth of the number of rules, implement and adopt rule reduction techniques. [9][10]
- Implement collaborative filtering to incorporate other user preferences who watched similar movies.

7 References

1. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes
2. Abe Mamdani: A Pioneer of Soft Artificial Intelligence
3. A new insight into implementing Mamdani fuzzy inference system for dynamic process modeling: Application on flash separator fuzzy dynamic modeling
4. Mamdani Systems
5. Skfuzzy version 0.3 docs
6. https://movielens.org/

7. https://www.sciencedirect.com/topics/engineering/defuzzification
8. https://youtu.be/-78SKS4wKgw?si=f-L76xvaBs5b3UMD
9. Design of Mamdani fuzzy logic controllers with rule base minimization using genetic algorithm - ScienceDirect
10. A fast belief rule base generation and reduction method for classification problems (sciencedirectassets.com)