Comparison of Behaviour of Topic Modelling Techniques-LDA and LSA

Jyothi Yendamuri
Sid:11467683
yendamurij@uni.coventry.ac.uk

Nikhil Bodduluri
Sid :11270094
boddulurin@coventry.ac.uk

Abstract:

In this paper, different topic modelling techniques such as Latent Dirichlet Allocation (LDA), Latent Semantic Analysis (LSA) are studied and implemented on the dataset taken from Kaggle that containing huge amount of unstructured Textual data. Text mining is done on the unstructured data; reviews collected by the company about their various food products. Topic modelling will identify the hidden topics and form clusters of reviews depending upon the similarity between them. This model will help company to improve their facilities depending upon customer reviews. The results of both the Topic modelling techniques-LDA and LSA will be compared.

Keywords: Topic modelling, Text Mining, Latent Dirichlet Allocation (LDA), Latent Semantic Analysis (LSA).

Introduction:

There are many methods of text mining used to identify patterns in unstructured textual data. In this study, LDA and LSA are used which are very efficient method of topic modelling. These techniques analyze the text and try to identify underlying topics automatically. The main assumption of these algorithms is each document contains multiple topics each having some specific weight. Topic modelling can also be thought of as clustering; here clusters of words are built rather than texts[1]. These techniques classifies text in a document to a particular topic. Topic modelling is an unsupervised learning as it automatically form groups of words in a document without knowing any predefined list of labels. Upon feeding data to the model, model gives different sets of words, and each set of words describes the topic[3]. These techniques are quite helpful as texts are automatically categorised without being assigned any label. LDA model initially builds a topic per document model and words per topic model[2]. LSA being a computational model; generally works on the notion of words having same meaning appear in similar context.These techniques to extract topics from textual information is effective in case of large volumes of text for topic extraction.

Problem and Dataset description:

The dataset has been taken from Kaggle. It is publicly available and free to use. The dataset can be found here. This dataset is the reviews given by the customer about the food products to the company. Depending upon the usefulness of the review, the company would like to work on the review received for all categories such as pet food, coffee reviews, spices review, etc. The dataset has 10 attributes: Id, ProductId, UserId, ProfileName, HelpfulnessNumerator, HelpfulnessDenominator, Score, time, Summary, Text. Id is the unique identification number given to the review. ProductId is the identification number of the product about which review has been given. UserId is the identification of user giving the review about the product. ProfileName is the name of the person giving the review. HelpfulnessNumerator, HelpfulnessDenominator, score are the columns showing the importance of the review. Time tells the time when review was given. Text contains the reviews given by customers.

The column of interest is last column, Text, as it contains the textual data (reviews of the customer) on which topic modelling techniques are applied. It has 568,454 rows which means number of reviews received are more than 568K. The aim is to categorise the reviews based on some topics so that company can analyse the reviews according to specific topic rather than going through all the feedbacks and then divide into groups. The techniques applied on the column is – Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA).

Nowadays, getting feedbacks from the customers and improving based on the reviews received is the key to success for every business. It is very often used by all the companies in order to meet the expectations of the customers and build good cutomer-relations. Feedbacks received by the companies are huge collection of unstructured data. Scanning all these feedbacks manually is very daunting and time-consuming as the data is highly unstructured. So, these topic modelling techniques play significant role in doing such tasks saving a lot of time and human efforts.

Methods:

Latent Dirichlet Allocation (LDA):

It is a generative probabilistic model of any large text document. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words[4]. The generative process of LDA can be given as, for the M number of documents, number of words to be N, number of topics K, the model gives output according to two probabilities: *psi*, the distribution of words for each topic *K* and *phi*, the distribution of topics for each document *i*.

LDA works on two basic principles: (i) Every document has multiple topics which means every document contains words belonging to various topics in particular proportions. For instance, if number of topics are 2, first document may contain 80% words belonging to topic 0 and 20% words from topic 1.

(ii) Every topic has multiple words which means there are many words belonging to each topic. Topic can be thought of as category having several words. For instance, “food” topic may have words like ‘delicious’, ‘sweet’, ‘sour’, etc. Similarly, different topics contains different words. However, it is also possible that one word can simultaneously come in two topics.

In LDA, both of these principles are considered at the same time while modelling. Sometimes, words are also considered in groups, known as phrases. In order to form model with phrases, n-grams such as bigrams and trigrams can be used to make the model more efficient. There are three major parameters of the model: (i) Number of topics which should be given prior so that model will categorise based on the given number of topics (ii) Alpha parameter is the document-topic density. The value of alpha determines the number of topics in a document. If the value of alpha is high, it means there are more topics present in a document whilst less value shows lower number of topics. (iii) Beta parameter is the topic-word density. If value of beta parameter is high, it means there are more words in a topic whilst less value of beta shows lower number of words.

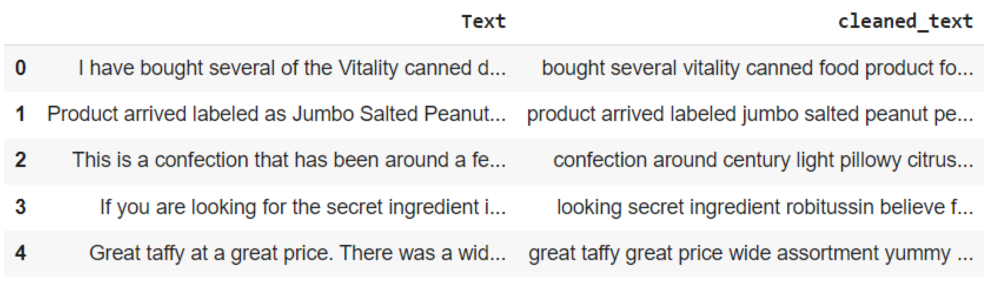
Latent Semantic Analysis (LSA):

Latent Semantic analysis is one of the foundational techniques of topic modelling in which relations between words are extracted statistically. In this process, texts are given as input using which document-topic and topic-term matrix is formed. It accepts TF-IDF matrix which is very sparse. Hence, in order to reduce dimension, singular value decomposition is used to obtain most important words in a document. This newly generated matrix can be used by models to compute similarities between topics. Albeit LSA is quick and efficient to use, it still has some drawbacks like it requires large sets of documents and vocabulary to get good results; representation is not much effective[5]. LSA is a dimension-reduction technique whereas no such step is required for LDA.

Experimental Setup:

The model is implemented in Google colab using python language. The steps of execution are described in this section. Initially all the required libraries are imported that are required for making topic models, pre processing, visualizations. The next step after importing libraries is to load the dataset for which model has to be implemented. The dataset has more than 568K reviews. So, only 5000 reviews are selected in order to save computational time. Moreover, as discussed above, dataset has 10 features which are not of any use for this model. So, all the columns except textual column containing reviews are dropped. So, the new dataset has 1 textual column only with 5000 rows.

The next crucial step is Pre-Proecssing. It is very significant as models do not accept textual unstructured data directly. This data has to be preprocessed before it is passed to model to get good results. So, various preprocessing techniques are applied such as removal of stop words, tokenization, lemmatization. The result is pre-processed data which looks like:



	Text	cleaned_text
0	I have bought several of the Vitality canned d...	bought several vitality canned food product fo...
1	Product arrived labeled as Jumbo Salted Peanut...	product arrived labeled jumbo salted peanut pe...
2	This is a confection that has been around a fe...	confection around century light pillowy citrus...
3	If you are looking for the secret ingredient i...	looking secret ingredient robitussin believe f...
4	Great taffy at a great price. There was a wid...	great taffy great price wide assortment yummy ...

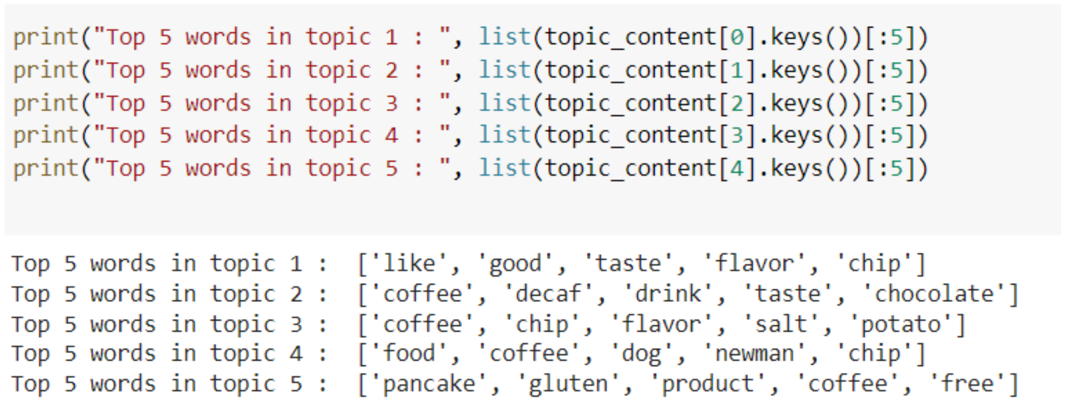
Fig 1: Pre-processed data

In Fig1, text column shows the data before pre processing and cleaned_text shows the reviews after preprocessing. Now, the unprocessed data will be dropped as it is of no use longer. The next step is to generate the document-term matrix so that similarities can be found out by the LSA model. Ranking was given to the words using tfidf vectorization. It was observed that like is the most common word and oreo is the least occurring word with lowest rank. In order to implement LSA, SVD was done on the vectorized text.

The next model to be implemented was LDA. The two major inputs of LDA model is the dictionary and corpus. So, dictionary and corpus was made using preprocessed data and gensim package. The number of topics given for both the models was 5. Afterwards, LDA model was designed passing all these parameters.

Results:

Top 5 words of each topic designed by LSA model were fetched and displayed below:

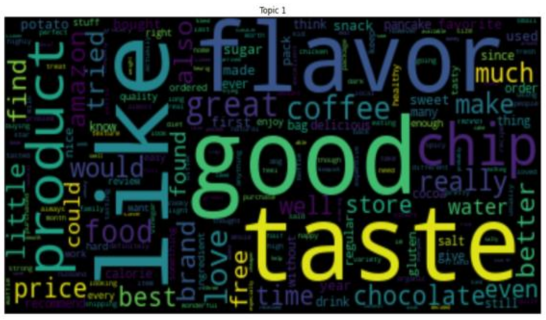


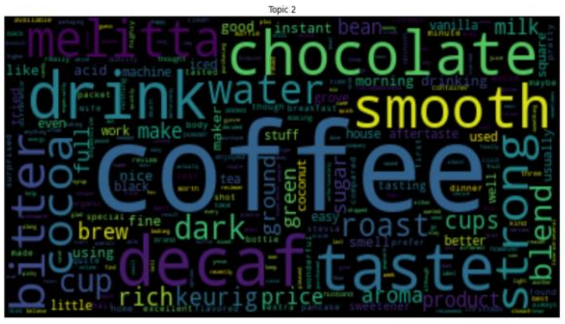
```
print("Top 5 words in topic 1 : ", list(topic_content[0].keys())[:5])
print("Top 5 words in topic 2 : ", list(topic_content[1].keys())[:5])
print("Top 5 words in topic 3 : ", list(topic_content[2].keys())[:5])
print("Top 5 words in topic 4 : ", list(topic_content[3].keys())[:5])
print("Top 5 words in topic 5 : ", list(topic_content[4].keys())[:5])
```

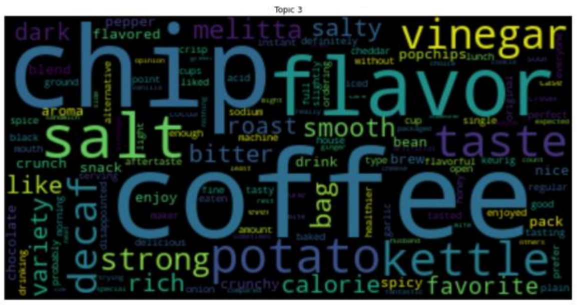
```
Top 5 words in topic 1 :  ['like', 'good', 'taste', 'flavor', 'chip']
Top 5 words in topic 2 :  ['coffee', 'decaf', 'drink', 'taste', 'chocolate']
Top 5 words in topic 3 :  ['coffee', 'chip', 'flavor', 'salt', 'potato']
Top 5 words in topic 4 :  ['food', 'coffee', 'dog', 'newman', 'chip']
Top 5 words in topic 5 :  ['pancake', 'gluten', 'product', 'coffee', 'free']
```

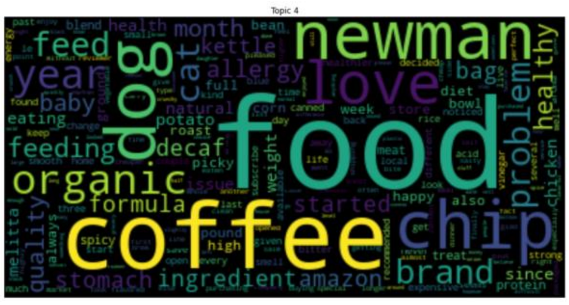
Fig 2: Displaying Top 5 words from each topic

From fig2, it can be observed that there are many words which are getting repeated in more than one topic like “coffee”, “chip”, etc. Word Clouds of all the 5 topics are generated by displaying maximum 500 words in each word cloud.









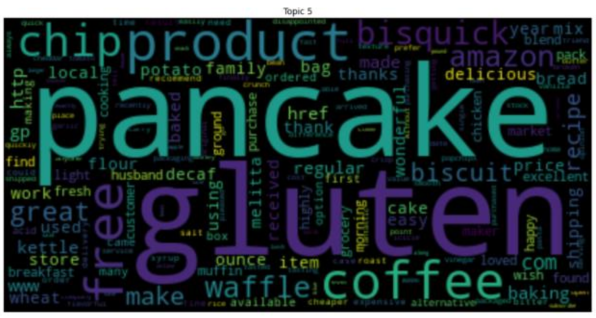
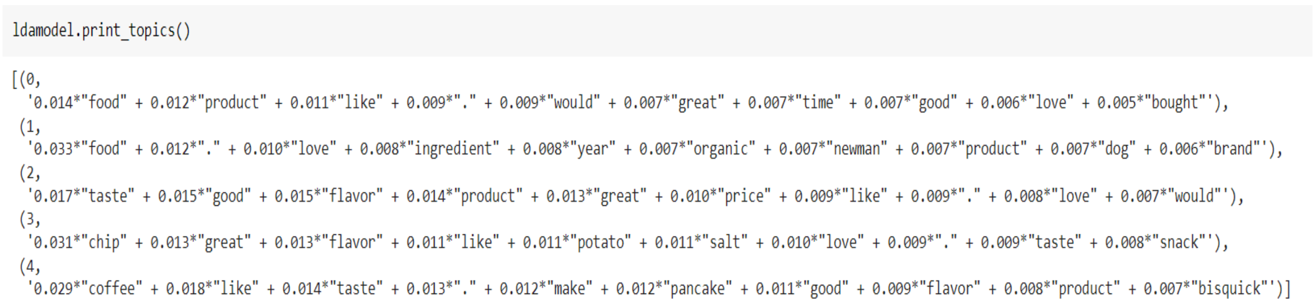


Fig 3: Displaying word clouds for each topic

The topics given by LDA model are:



```
ldamodel.print_topics()
```

```
[(0,
  '0.014*"food" + 0.012*"product" + 0.011*"like" + 0.009*"." + 0.009*"would" + 0.007*"great" + 0.007*"time" + 0.007*"good" + 0.006*"love" + 0.005*"bought"'),
 (1,
  '0.033*"food" + 0.012*"." + 0.010*"love" + 0.008*"ingredient" + 0.008*"year" + 0.007*"organic" + 0.007*"newman" + 0.007*"product" + 0.007*"dog" + 0.006*"brand"'),
 (2,
  '0.017*"taste" + 0.015*"good" + 0.015*"flavor" + 0.014*"product" + 0.013*"great" + 0.010*"price" + 0.009*"like" + 0.009*"." + 0.008*"love" + 0.007*"would"'),
 (3,
  '0.031*"chip" + 0.013*"great" + 0.013*"flavor" + 0.011*"like" + 0.011*"potato" + 0.011*"salt" + 0.010*"love" + 0.009*"." + 0.009*"taste" + 0.008*"snack"'),
 (4,
  '0.029*"coffee" + 0.018*"like" + 0.014*"taste" + 0.013*"." + 0.012*"make" + 0.012*"pancake" + 0.011*"good" + 0.009*"flavor" + 0.008*"product" + 0.007*"bisquick"')]
```

Fig4: Displaying topics designed by LDA model

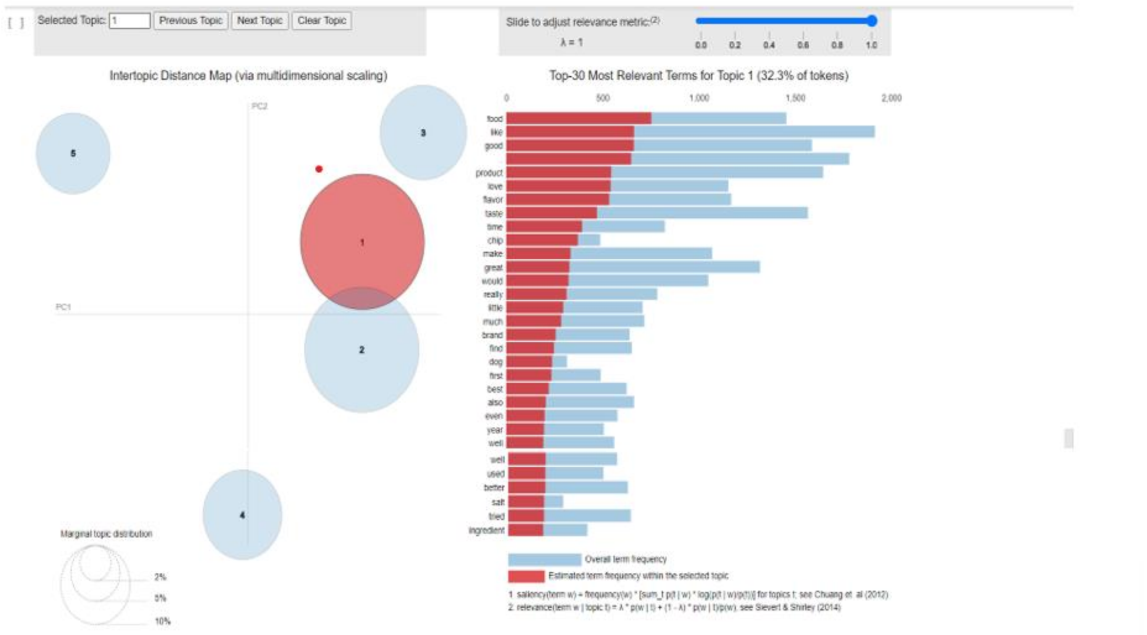


Fig5 Visualising LDA results

From Fig5, it is clear that LDA model was successful in building all the 5 clusters efficiently. There is only little bit overlapping in cluster 1 and 2.

It is not easy to compare the results of both LSA and LDA models by just looking at the topics. Visualisations make comparison better,hence, T-SNE clustering was used. The clusters formed by both the models are shown below and results are compared for both the models.

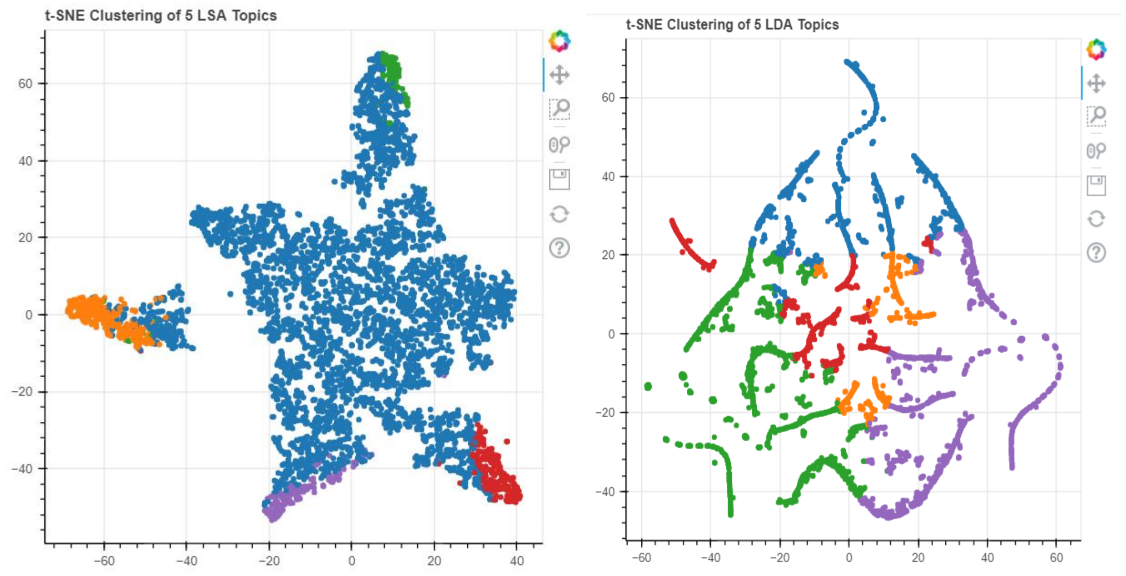


Fig 6: Comparing t-sne clusters of (a) LSA (b) LDA model

From fig6, it is clear that clusters formed by LDA model are clear and precise whereas results of LSA are not desirable.

Social, ethical, legal and professional considerations:

Since the dataset has been taken from publicly available source Kaggle, all the social, ethical, legal and professional ethics are considered. There is no information used which is belonging to any user personally such as userid, name. All these information were ignored and not taken into consideration while designing the model.

Conclusions:

This model uses two topic modelling techniques-LSA and LDA to make groups of the textual data , reviews received by a food company from the customers for different food items. Between both the techniques, LDA outperformed the LSA technique. The performance of both the models were evaluated using T-SNE clustering method. Plots of both the models were analysed and it was concluded that clusters formed by LDA model can be clearly observed whilst for LSA, there was significant overlap among all the clusters. The reason behind highly unexpected results of LSA can be the size od the dataset as it works efficiently when dataset is huge. The model can be used by company to analyse the reviews for each topic. Depending upon the reviews, company can improve their facilities.

Evolutionary and Fuzzy Systems

Jyothi Yendamuri

Sid:11467683

yendamurij@uni.coventry.ac.uk

Nikhil Bodduluri

Sid :11270094

boddulurin@coventry.ac.uk

Part-1: Implementation of the Fuzzy Logic Controller:

In this part-1, the aim is to design the Smart assistive home. Fuzzy logic is very helpful for designing where automation is required. So, Smart assisstive home can be implemented using Fuzzy Logic Controller. The steps of implementation are:

(i)Inputs and membership functions:

There are 4 input variables used which are shown below:

1. Temperature: It is measured in degree Celsius. The range of input variable is -10 to 70. The membership functions are: VL, L, M, H, VH. The types of membership function used is Trapezoidal for VL and VH whilst triangular for L, M, and H. The details are shown below in figure:

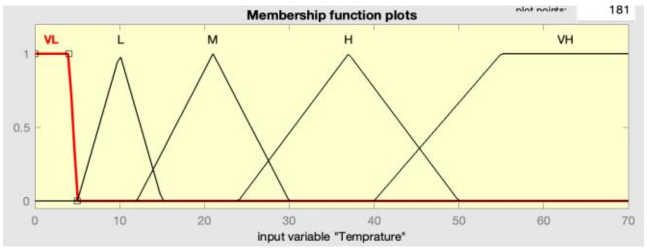


Fig7 Temperature membership function

2. Lighting: It is measured in Lux. Its value lies between 0 to 500 Lux. There should be enough illuminance in the house to make it comfortable. This input variable has 5 membership functions among which VL and VH are trapezoidal and L, M and H are triangular.

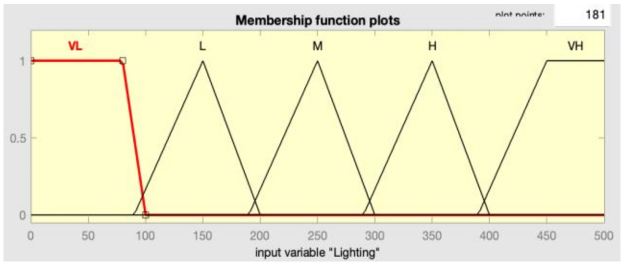


Fig8 Lighting membership function

3. AQI: It measures air quality of the house. It is measured in ppm. It also has 5 membership functions among which VL and VH are trapezoidal and L, M and H are triangular. Its value lies between 0-1000ppm.

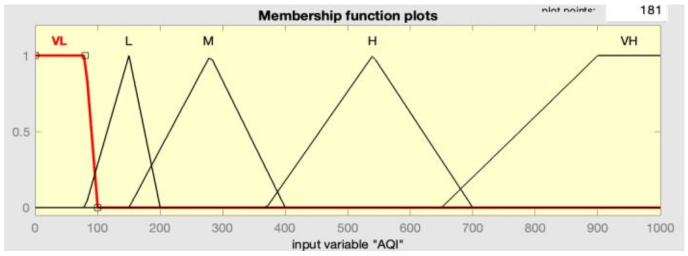


Fig9 AQI membership function

4. Blood_Pressure: It is the mesure of blood pressure of the person. Its value ranges between 60 to 200. It has 3 membership functions- L , M and H. L and H are trapezoidal whilst M is triangular.

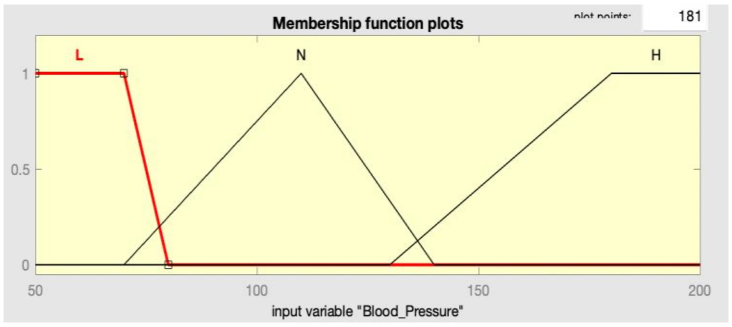
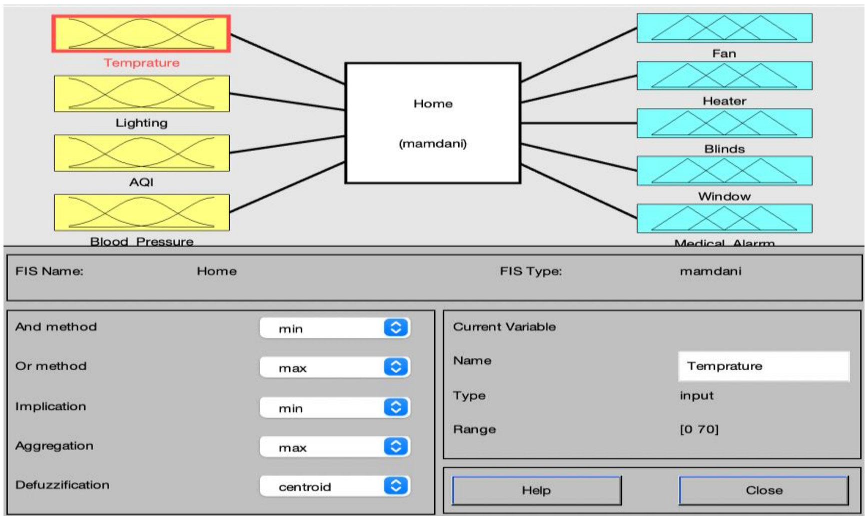


Fig10 Blood_Pressure membership function

(ii) Fuzzy Inference System:

In this fuzzy logic controller, Mamdani fuzzy inference system is used. It will accept inputs variables and their membership function and produce fuzzy sets as output. The output is produced by using fuzzy rules designed for the system.



FIS Name: Home
FIS Type: mamdani

And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

Current Variable
Name: Temprature
Type: input
Range: [0 70]

Fig11 GUI of Fuzzy Inference

(iii) Output and membership functions:

There are five outputs designed for the fuzzy logic controller which are explained below:

1. Fan: Fan is used to control the temperature of the home. If the temperature is medium of high, cooling fans can be used to make feel comfortable. It has 4 membership functions- Off, L, M, H. The value ranges between 0 to 100%. Off and H are trapezoidal whilst L and M are triangular.

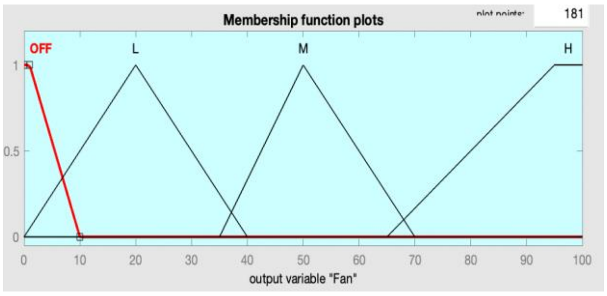


Fig12 Fan membership function

2. Heater: Heater is also used to control the temperature of the room. If the temperature is too low, then it can be controlled using heaters. It has 4 membership functions- Off, L, M, H. The value ranges between 0 to 100%. Off and H are trapezoidal whilst L and M are triangular.

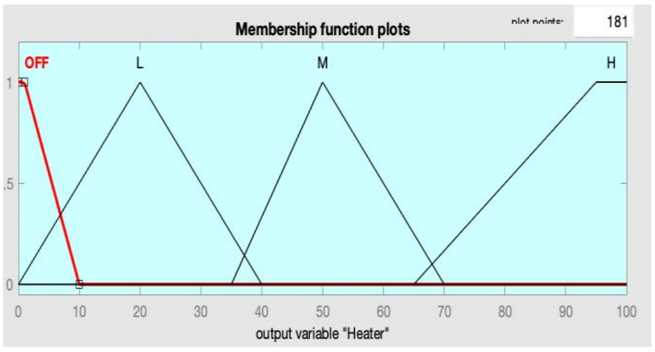


Fig13 Heater membership function

3. Blinds: Blinds are used to control the lighting into the house. If the lighting is less, blinds can be opened so that there should be enough light in the house. It has 3 membership functions- Open, half, close. The value ranges between 0 to 100%. Open and close are trapezoidal whilst half is triangular.

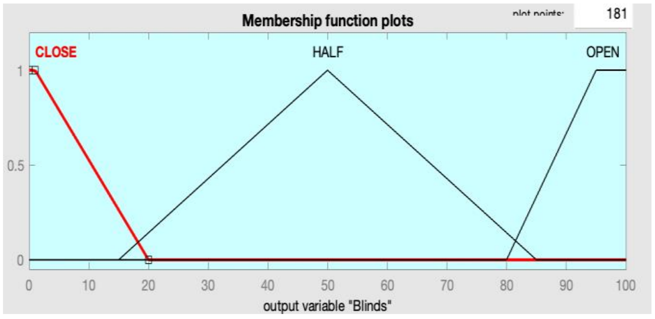


Fig14 Blinds membership function

4. Window: Window can be opened if air quality is not good. High AQI means air quality is not good inside home, so, windows can be opened so that there can be fresh air inside home. The range is 0 to 100%. It has 3 membership functions among which Open and close are trapezoidal and Half is triangular.

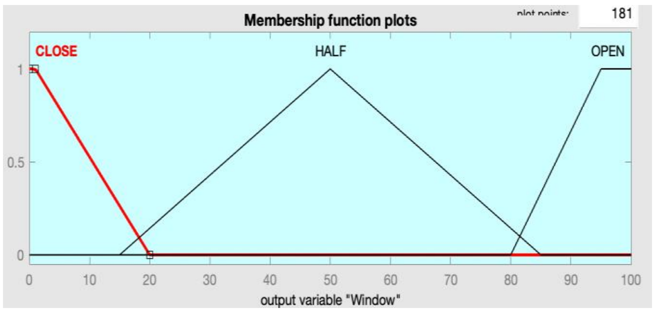


Fig15 Window membership function

5. Medical Alarm: Medical warnings can be given if person's blood pressure is not normal. It has two membership functions- On and Off. Both of them are trapezoidal membership functions.

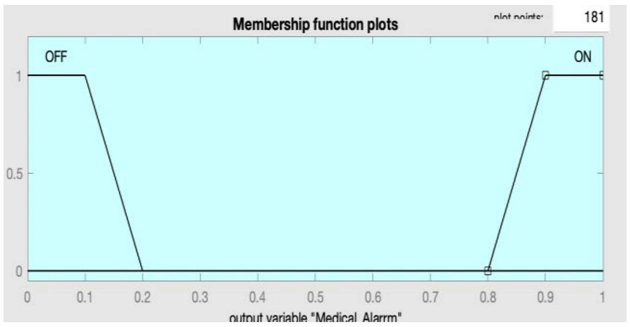


Fig16 Medical_Alarm membership function

(iv) Fuzzy Rules:

Inputs are converted to outputs based on some set of rules which are given while designing the FLC. These rules are significant in designing FLC. In this step, multiple variables are combined forming fuzzy rules. Here, t-norm rules are used that makes use of AND operator to combine multiple variables. There are total 18 rules designed for the system.

Temperature	**VL**	**L**	**M**	**H**	**VH**
Fan	Off	Off	L	M	H
Heater	H	M	Off	Off	Off

Table1: Temperature input vs Fan and Heater Output

Lighting	**VL**	**L**	**M**	**H**	**VH**
Blinds	Open	Open	Half	Close	Close

Table2: Lighting input vs Blinds Output

AQI	**VL**	**L**	**M**	**H**	**VH**
Window	Close	Close	Half	Open	Open

Table3: AQI input vs Window Output

Blood_Pressure	**L**	**N**	**H**
Medical Alarm	On	Off	On

Table4: Blood_Pressure input vs Medical Alarm Output

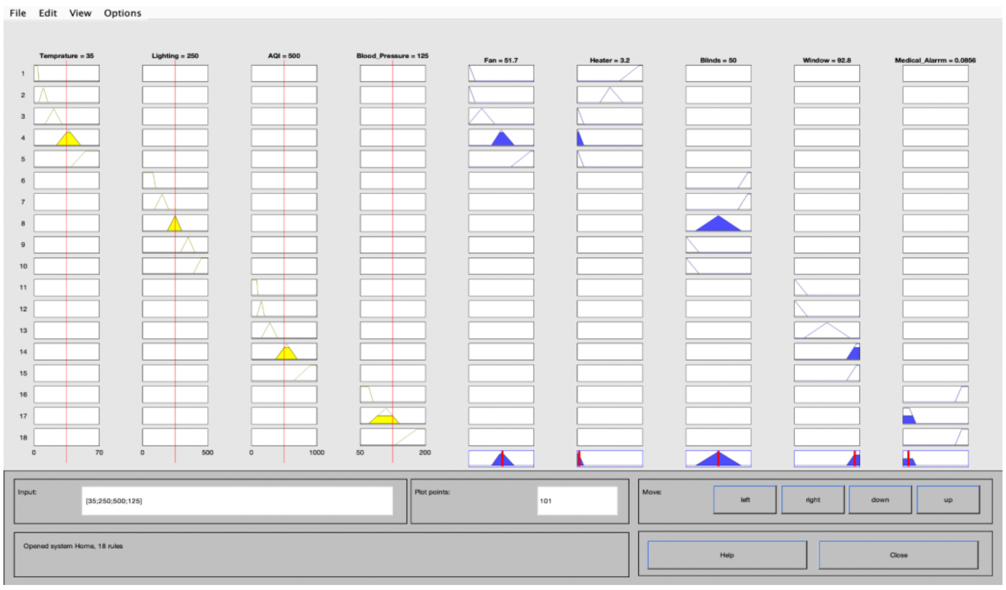
(v) Aggregation:

This step combines fuzzy sets to form a single fuzzy sets. It takes output of each rule and aggregates it, resulting into single fuzzy set. It accepts maximum values while aggregating fuzzy sets.

(vi) Defuzzification:

The last step of FLC is defuzzification in which output produced by the aggregation step is converted into crisp output. The aggregated result is a fuzzy set. This fuzzy set is converted into a single number in this step. There are many methods available for defuzzification such as MoM, CoG, bisector, etc. In this FLC, CoG has been used for defuzzification.

After this step, rules can be viewed as:



Input: [35;250;500;125]

Plot points: 101

Opened system Home, 18 rules

Fig17 Fuzzy Inference Rule

Plots of inputs vs output can be seen in figure below:

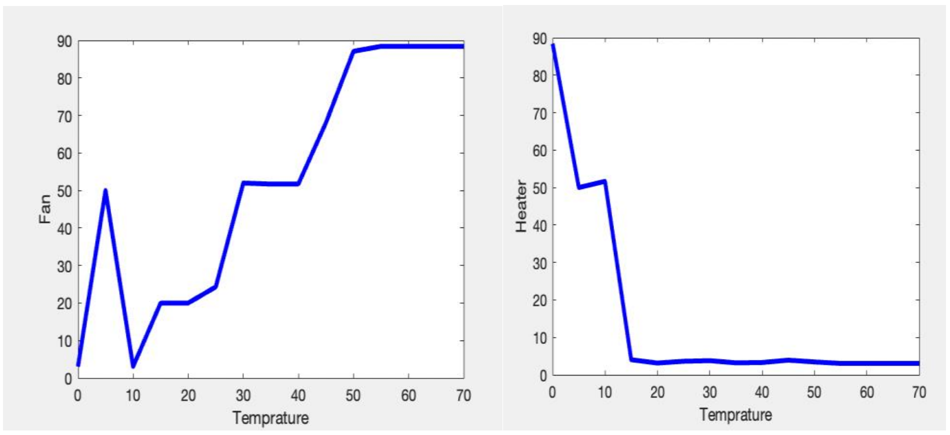
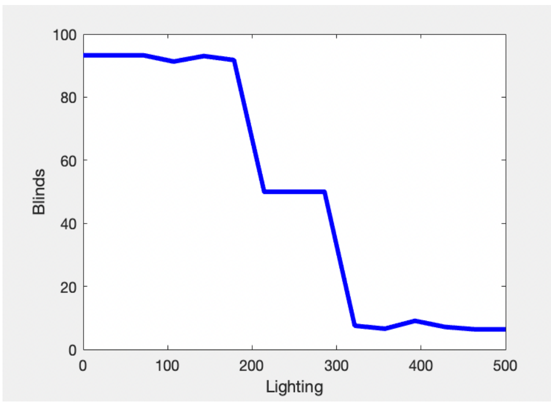


Fig18 Temperature vs (a) Fan (b) Heater



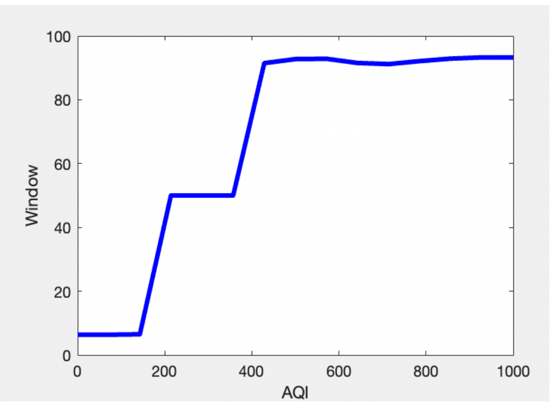


Fig19 Lighting vs Blinds

Fig 20 AQI vs Window

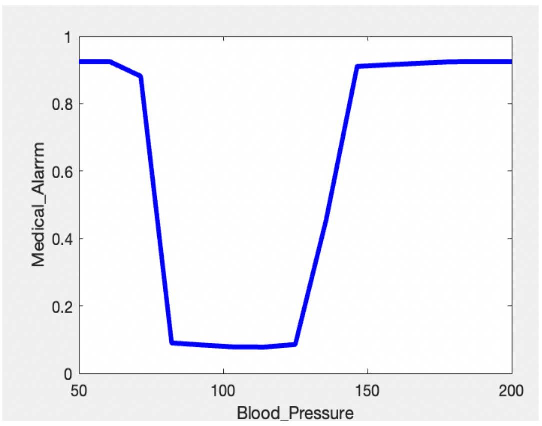


Fig21 Blood_Pressure vs Medical_Alarm

Part-2: Optimizing the FLC developed for Part 1:

Genetic algorithms can be used to optimize the Fuzzy Logic Controller developed in Part-1.

The first step is to convert all the parameters of the membership functions in the binary form. Each digit 0 or 1 is known as gene and combined is known as chromosome. Chromosomes together are known as populations.

Trapezoidal membership function has 4 parameters whereas the triangular membership function has 3 parameters. In part-1, there are 4 input membership functions and 5 output membership functions.

So, the total number of chromosomes for inputs are: [(4+3+3+3+4) + (4+3+3+3+4) + (4+3+3+3+4) + (4+3+3+4)]= 65 chromosomes.

So, the total number of chromosomes for outputs are: [(4+3+3+4) + (4+3+3+4) + (4+3+4) + (4+3+4) + (4+4)]= 58 chromosomes.

Then, fitness function is used to find how fit each chromosome is for mating. This function will give fitness score to each chromosome selecting which offspring can be created.

The next step is to decide the stopping criteria. All the parameters such as crossover rate, mutation rate, number of populations should be selected at this step so that genetic operations such as Selection, Crossover and Mutation can be performed.

Depending upon the fitness score, the chromosomes with high fitness scores will be selected using Roulette wheel method. These chromosomes will be used for producing new chromosomes.

After selecting the fit chromosomes, these chromosomes will be used for mating to produce new chromosomes. The next step is crossover where some parts of any 2 chromosomes are exchanged to produce offspring.

The last step after crossover is mutation where any gene is selected randomly from the chromosome and changed. The algorithm keeps on running unless optimised results are obtained or stopping criteria is reached.

Using Sugeno in place of Mamdani:

As discussed in Part-1, there are two methods of Fuzzy inference: Mamdani and Sugeno. In Part-1, Mamdani Inference system has been used. If Sugeno has been used, the output membership function would have been linear or constant instead of fuzzy sets. The results of Sugeno are more efficient. So, it is useful for mathematical analysis whereas for human-based rule implementation, mamdani inference system is useful.

Part-3: Comparing different optimization techniques on CEC’2005 functions

In this part, two Optimisation techniques- Genetic Algorithm Optimisation and Particle Swarm Optimisation are applied on two basic CEC’2005 benchmark functions- F9: Shifted Rastrigin’s Function and F10: Shifted Rotated Rastrigin’s Function. The optimisation algorithm is run for 15 iterations. The aim is to find the global minima for parameters.

(i) Shifted Rastrigin's Function:

Formula:

$$F_9(\mathbf{x}) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias_9 ,\ \mathbf{z} = \mathbf{x} - \mathbf{o} ,\ \mathbf{x} = [x_1, x_2, ..., x_D]$$

D: dimensions

$\boldsymbol{o} = [o_1, o_2, ..., o_D]$: the shifted global optimum

f_bias$_9$= - 330

Results with Genetic Algorithm:

Dimension	Best Performance	Worst Performance	Mean	Standard Deviation
2D	**-329.9923**	**-326.8197**	**-328.5240**	**0.8326**
10D	**-330.0000**	**-328.9743**	**-329.5302**	**0.5129**

Table5: Shifted Rastrigin's Function Results Summary with Genetic Algorithm

Results with Particle Swarm optimisation Algorithm:

Dimension	Best Performance	Worst Performance	Mean	Standard Deviation
2D	**-330.0000**	**-328.9717**	**-329.8234**	**0.3595**
10D	**-328.0101**	**-318.0605**	**-323.8312**	**3.1508**

Table6: Shifted Rastrigin's Function Results Summary with PSO

(ii) Shifted Rotated Rastrigin's Function: Formula:

$$F_{10}(\mathbf{x}) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias_{10} ,\ \mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M} , \mathbf{x} = [x_1, x_2, ..., x_D]$$

D: dimensions

$\mathbf{o} = [o_1, o_2, ..., o_D]$: the shifted global optimum

$\mathbf{M}$: linear transformation matrix, condition number=2

f_bias$_{10}$= - 330

Results with Genetic Algorithm:

Dimension	Best Performance	Worst Performance	Mean	Standard Deviation
2D	**-329.3025**	**-320.2820**	**-327.0233**	**2.1948**
10D	**-302.9415**	**-244.8916**	**-280.7450**	**16.2211**

Table7: Shifted Rotated Rastrigin's Function Results Summary with Genetic Algorithm

Results with Particle Swarm optimisation Algorithm:

Dimension	**Best Performance**	**Worst Performance**	**Mean**	**Standard Deviation**
2D	**-330.0000**	**-328.9743**	**-329.5302**	**0.5129**
10D	**-323.0353**	**-283.3498**	**-310.8615**	**11.0887**

Table8: Shifted Rotated Rastrigin’s Function Results Summary with PSO

From above tables, it is clear that for function F9- Shifted Rastrigin’s Function, PSO outperformed GA when number of dimensions are 2. On the other hand, when dimensions are 10, standard deviation is less for Genetic Algorithm. It is closer to target global minimum.

For function F10- Shifted Rotated Rastrigin’s Function, the performance of Particle Swarm optimisation Algorithm is better in both the cases when number of dimensions =2 or 10.

From the above results, it can be concluded that these two algorithms work really well as results of both the algorithms are nearly close to target values given in literature of these functions which means both the algorithms are suitable for optimising any real-world problem.

References

[1] Kapadia, S. (2019). *Topic Modeling in Python: Latent Dirichlet Allocation (LDA)*. Medium. Retrieved 13 December 2021, from https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0.

[2]. Li, S. (2018). *Topic Modeling and Latent Dirichlet Allocation (LDA) in Python*. Medium. Retrieved 13 December 2021, from https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24.

[3] *pyLDAvis: Topic Modelling Exploration Tool That Every NLP Data Scientist Should Know - neptune.ai*. neptune.ai. (2021). Retrieved 13 December 2021, from https://neptune.ai/blog/pyldavis-topic-modelling-exploration-tool-that-every-nlp-data-scientist-should-know.

[4]. Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, *3*, 993-1022.

[5]. Joyce, Xu.(2018). Topic Modeling with LSA,PLSA, LDA & lda2Vec https://medium.com/nanonets/topic-modeling-with-lsa-psla-lda-and-lda2vec-555ff65b0b05

Appendix:

The link to the dataset is: https://www.kaggle.com/hj5992/nlp-topic-modelling

Task 1: The direct link to the code for Task1:

https://colab.research.google.com/drive/1ry6v-DtBFZcT5EnpF1_DINDGBrILZz5a#scrollTo=vFZkbHPkWppM

OR

https://drive.google.com/file/d/13WnycghzpAaI7F2GBglslFunTSamM3Sg/view?usp=sharing

Task-2:

FLC Code Part-1:

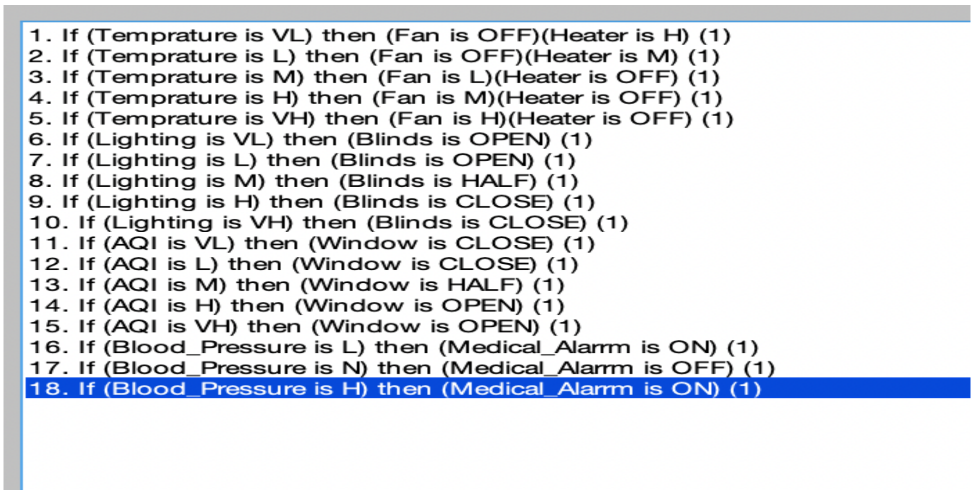
```
[System]

Name='Home'

Type='mamdani'

Version=2.0

NumInputs=4

NumOutputs=5

NumRules=18

AndMethod='min'

OrMethod='max'

ImpMethod='min'

AggMethod='max'

DefuzzMethod='centroid'

[Input1]

Name='Temprature'

Range=[0 70]

NumMFs=5

MF1='VL':'trapmf',[-10 0 4 5]

MF2='M':'trimf',[12 21 30]

MF3='VH':'trapmf',[40 55 70 100]

MF4='L':'trimf',[5 10 15]

MF5='H':'trimf',[24 37 50]

[Input2]

Name='Lighting'

Range=[0 500]
```

```
NumMFs=5
MF1='VL':'trapmf',[-10 0 80 100]
MF2='H':'trimf',[290 350 400]
MF3='L':'trimf',[90 150 200]
MF4='M':'trimf',[190 250 300]
MF5='VH':'trapmf',[390 450 500 510]
[Input3]
Name='AQI'
Range=[0 1000]
NumMFs=5
MF1='VL':'trapmf',[-10 0 80 100]
MF2='M':'trimf',[150 280 400]
MF3='VH':'trapmf',[650 900 1000 1100]
MF4='L':'trimf',[80 150 200]
MF5='H':'trimf',[370 540 700]
[Input4]
Name='Blood_Pressure'
Range=[50 200]
NumMFs=3
MF1='L':'trapmf',[40 50 70 80]
MF2='N':'trimf',[70 110 140]
MF3='H':'trapmf',[130 180 200 240]
[Output1]
Name='Fan'
Range=[0 100]
NumMFs=4
MF1='OFF':'trapmf',[-10 0 1 10]
MF2='M':'trimf',[35 50 70]
MF3='H':'trapmf',[65 95 100 110]
MF4='L':'trimf',[0 20 40]
[Output2]
Name='Heater'
Range=[0 100]
NumMFs=4
```

```
MF1='OFF':'trapmf',[-10 0 1 10]
MF2='M':'trimf',[35 50 70]
MF3='H':'trapmf',[65 95 100 110]
MF4='L':'trimf',[0 20 40]
[Output3]
Name='Blinds'
Range=[0 100]
NumMFs=3
MF1='CLOSE':'trapmf',[-10 0 1 20]
MF2='HALF':'trimf',[15 50 85]
MF3='OPEN':'trapmf',[80 95 100 110]
[Output4]
Name='Window'
Range=[0 100]
NumMFs=3
MF1='CLOSE':'trapmf',[-10 0 1 20]
MF2='HALF':'trimf',[15 50 85]
MF3='OPEN':'trapmf',[80 95 100 110]
[Output5]
Name='Medical_Alarrm'
Range=[0 1]
NumMFs=2
MF1='ON':'trapmf',[0.8 0.9 1 1.2]
MF2='OFF':'trapmf',[-1 0 0.1 0.2]
[Rules]
1 0 0 0, 1 3 0 0 0 (1) : 1
4 0 0 0, 1 2 0 0 0 (1) : 1
2 0 0 0, 4 1 0 0 0 (1) : 1
5 0 0 0, 2 1 0 0 0 (1) : 1
3 0 0 0, 3 1 0 0 0 (1) : 1
0 1 0 0, 0 0 3 0 0 (1) : 1
0 3 0 0, 0 0 3 0 0 (1) : 1
0 4 0 0, 0 0 2 0 0 (1) : 1
0 2 0 0, 0 0 1 0 0 (1) : 1
```

```
0 5 0 0, 0 0 1 0 0 (1) : 1

0 0 1 0, 0 0 0 1 0 (1) : 1

0 0 4 0, 0 0 0 1 0 (1) : 1

0 0 2 0, 0 0 0 2 0 (1) : 1

0 0 5 0, 0 0 0 3 0 (1) : 1

0 0 3 0, 0 0 0 3 0 (1) : 1

0 0 0 1, 0 0 0 0 1 (1) : 1

0 0 0 2, 0 0 0 0 2 (1) : 1

0 0 0 3, 0 0 0 0 1 (1) : 1
```

Snippet of fuzzy rules:



```
1. If (Temprature is VL) then (Fan is OFF)(Heater is H) (1)
2. If (Temprature is L) then (Fan is OFF)(Heater is M) (1)
3. If (Temprature is M) then (Fan is L)(Heater is OFF) (1)
4. If (Temprature is H) then (Fan is M)(Heater is OFF) (1)
5. If (Temprature is VH) then (Fan is H)(Heater is OFF) (1)
6. If (Lighting is VL) then (Blinds is OPEN) (1)
7. If (Lighting is L) then (Blinds is OPEN) (1)
8. If (Lighting is M) then (Blinds is HALF) (1)
9. If (Lighting is H) then (Blinds is CLOSE) (1)
10. If (Lighting is VH) then (Blinds is CLOSE) (1)
11. If (AQI is VL) then (Window is CLOSE) (1)
12. If (AQI is L) then (Window is CLOSE) (1)
13. If (AQI is M) then (Window is HALF) (1)
14. If (AQI is H) then (Window is OPEN) (1)
15. If (AQI is VH) then (Window is OPEN) (1)
16. If (Blood_Pressure is L) then (Medical_Alarm is ON) (1)
17. If (Blood_Pressure is N) then (Medical_Alarm is OFF) (1)
18. If (Blood_Pressure is H) then (Medical_Alarm is ON) (1)
```

Rule Evaluation:

Parameter	Value
Temperature	40 C
Lighting	200 Lux
AQI	550 ppm
Blood_Pressure	180
Fan	51.7
Heater	3.3
Blinds	50
Window	93.2
Medical_Alarm	0.92

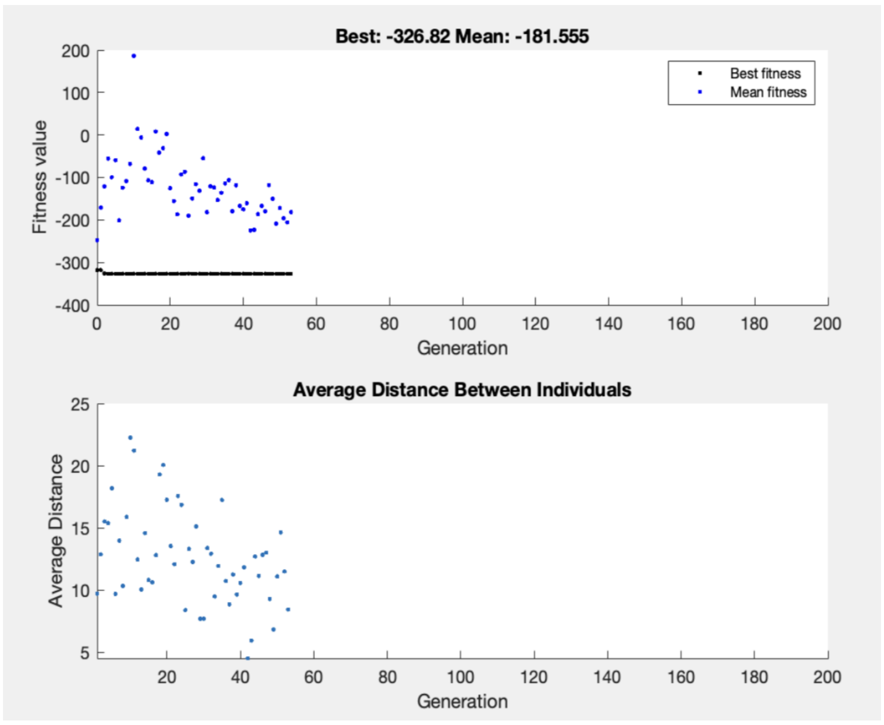
Parameter	Value
Temperature	0 C
Lighting	50 Lux
AQI	800 ppm
Blood_Pressure	80
Fan	3.06
Heater	88.4
Blinds	93.3
Window	92.3
Medical_Alarm	0.09

Part-3:

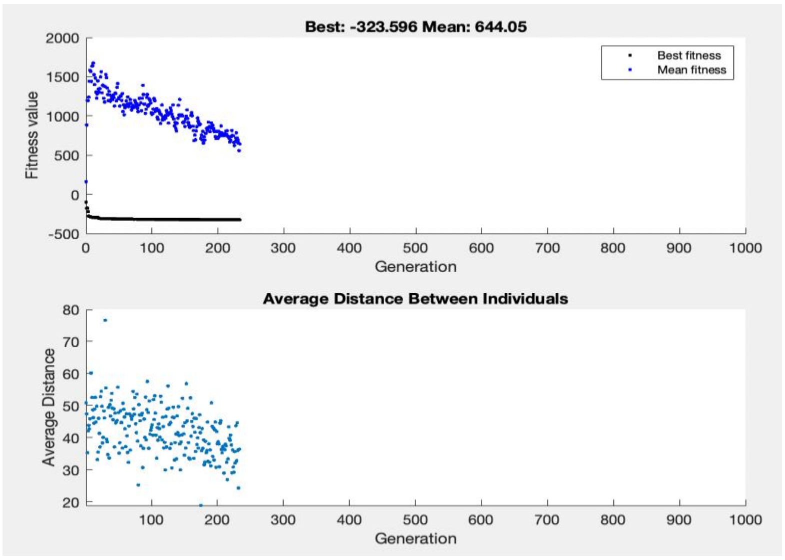
Plot of CEC function for Genetic algorithm tool:

F9:

D=2



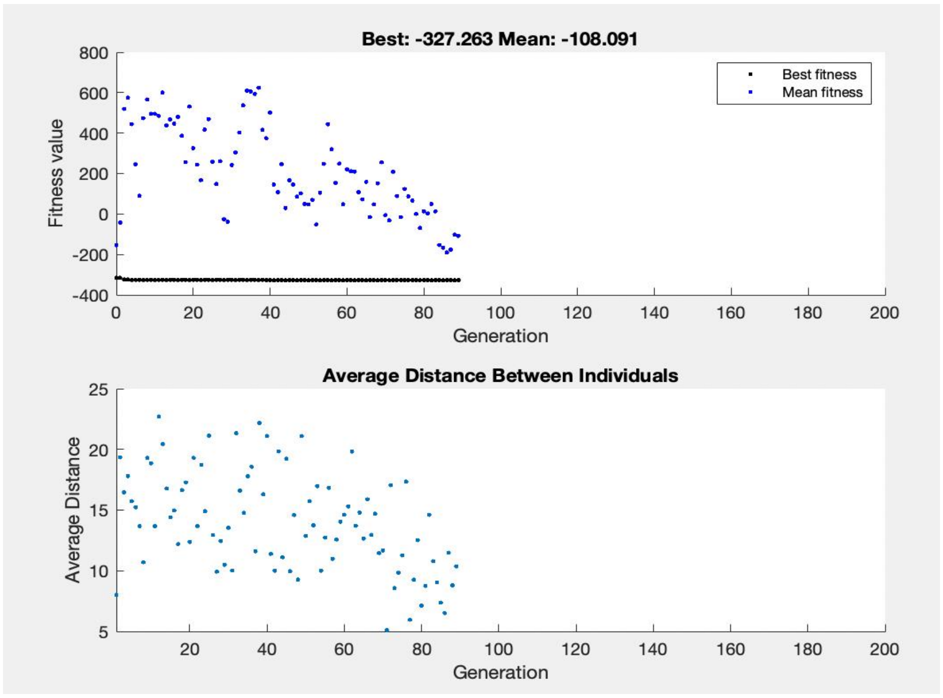
D=10



Best: -323.596 Mean: 644.05
Best fitness
Mean fitness
Fitness value
Generation
Average Distance Between Individuals
Average Distance
Generation

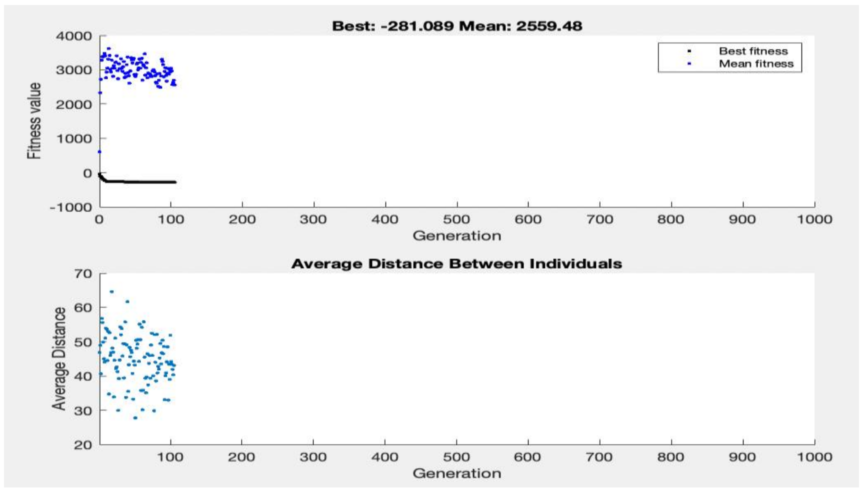
F10:

D=2



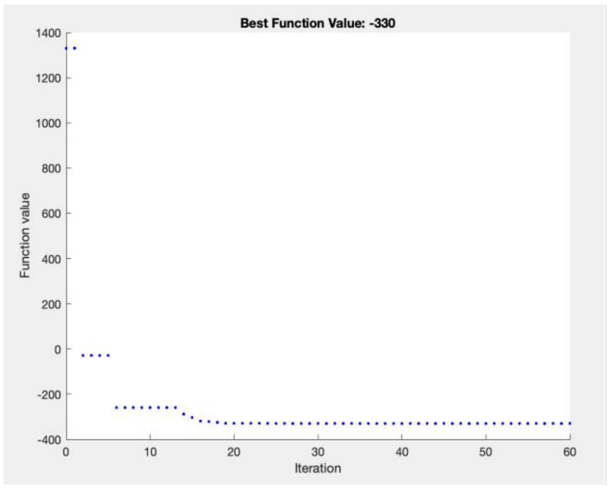
Best: -327.263 Mean: -108.091
Best fitness
Mean fitness
Fitness value
Generation
Average Distance Between Individuals
Average Distance
Generation

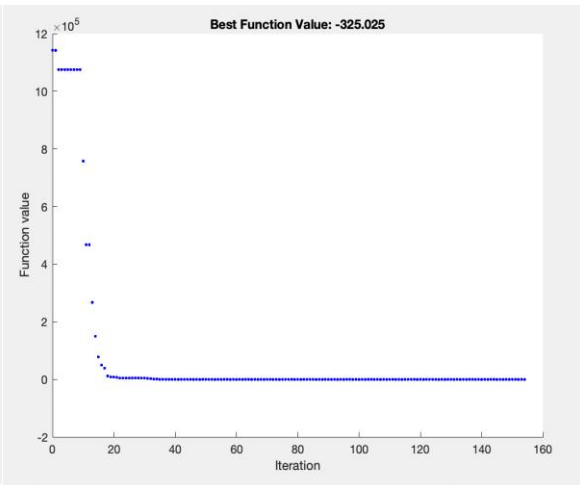
D=10



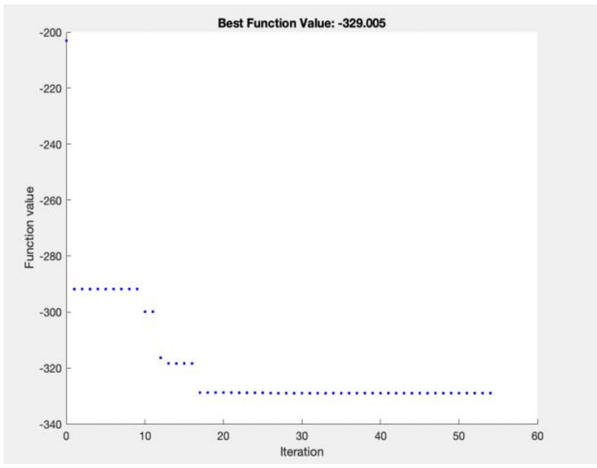
Particle Swarm Optimisation:

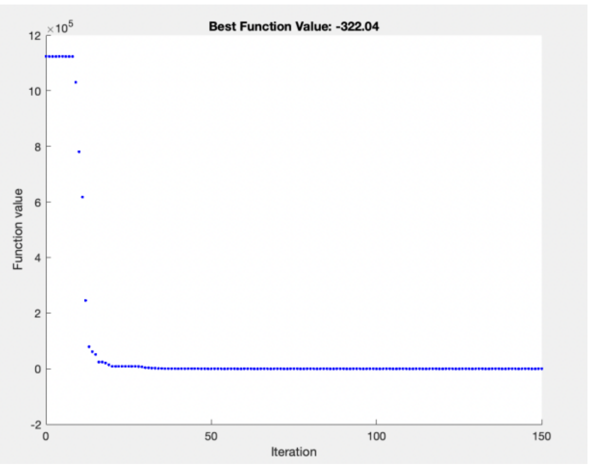
F9: **D=2** **D=10**





F10: **D=2** **D=10**





Algorithm for GA optimisation and PSO:

```
clc, clear, close all

rng default

global initial_flag

initial_flag = 0;

options =
optimoptions('ga','PlotFcn',{@gaplotbestf,@gaplotdistance});

for i=1:15

   initial_flag = 0;

   % use the below for 2D function 9

   %[ga_x,ga_val,ga_exit_flag,ga_output]
=ga(@(x)benchmark_func(x,9),2,options)

   % use the below for 10D function 9

   [ga_x,ga_val,ga_exit_flag,ga_output]
=ga(@(x)benchmark_func(x,9),10,options)

   % use the below for 2D function 10

   %[ga_x,ga_val,ga_exit_flag,ga_output]
=ga(@(x)benchmark_func(x,10),2,options)

   % use the below for 10D function 10

   %[ga_x,ga_val,ga_exit_flag,ga_output]
=ga(@(x)benchmark_func(x,10),10,options)

   ga_main_x(i,:) = ga_x

   ga_main_val(i) = ga_val

   ga_main_exit_flag(i) = ga_exit_flag

   ga_main_output(i) = ga_output

   % save visualizations to file

   fname = sprintf('filename(%d).fig', i) ;

   savefig(fname)

end

ga_val_max = max(ga_main_val)

ga_val_min = min(ga_main_val)

ga_val_mean = mean(ga_main_val)

ga_val_std = std(ga_main_val)
```

PSO:

```
clc, clear, close all

rng default

global initial_flag

%% Particle Swarm Optimization 15 Iterations

options = optimoptions('particleswarm','PlotFcn',{@pswplotbestf}); %rng default

initial_flag = 0;

for i=1:15

  %use the below for 10D function 10

%[swarm_x,swarm_val,swarm_exit_flag,swarm_output] =
particleswarm(@(x)benchmark_func(x,10),10,[-100,-100],[100,100],options)

  % use the below for 2D function 10

%[swarm_x,swarm_val,swarm_exit_flag,swarm_output] =
particleswarm(@(x)benchmark_func(x,10),2,[-100,-100],[100,100],options)

  % use the below for 10D function 9

[swarm_x,swarm_val,swarm_exit_flag,swarm_output] =
particleswarm(@(x)benchmark_func(x,9),10,[-100,-100],[100,100],options)

%[swarm_x,swarm_val,swarm_exit_flag,swarm_output] =
particleswarm(@(x)benchmark_func(x,9),2,[-100,-100],[100,100],options)

swarm_main_x(i,:) = swarm_x

swarm_main_val(i) = swarm_val

swarm_main_exit_flag(i) = swarm_exit_flag

swarm_main_output(i) = swarm_output

  % save visualizations to file

fname = sprintf('filename(%d).fig', i) ;

  savefig(fname)

end

%% Particle Swarm calculations

swarm_val_max = max(swarm_main_val)

swarm_val_min = min(swarm_main_val)

swarm_val_mean = mean(swarm_main_val)

swarm_val_std = std(swarm_main_val)
```