Universitat de les Illes Balears

Generation of fuzzy rules from data using an example of driver ratings based on driving style and conversation quality

Professor:
Sebastián Massanet Massanet

Authors:
Adrian Sauter
Laura Häge
Sebastian Kairat

Assignment 1
November 3, 2022

Computational Intelligence

1 Introduction

In the Computer Science domain, (classification) problems tend to be simplified to binary problems. Statements are either true or false, instances either belong to one or another class. This approach however, neglects many real life applications where multiple instances can belong to multiple classes. Fuzzy Logic breaks the hard constraints of traditional Logic and expands it further to handle linguistic, context-dependent terms such as 'small' and 'more'. Fuzzy Logic was first introduced in 1965 by L. A. Zadeh (Zadeh, 1965). Even though, one has to deal with redefining logical operators and sets, one big advantage of Fuzzy Logic is taking (un-)certainty into account thus making models more interpretable, realistic and robust.

To define if an instance x belongs to a fuzzy set A, membership functions are used:

$$\mu_A : X \to [0, 1]$$

If x is completely contained in A then $\mu_A(x) = 1$, if x is not in A at all then $\mu_A(x) = 0$ and the fuzzy case if x is partially in A then $\mu_A(x) \in (0, 1)$. Furthermore, if $\mu_A(x) > \mu_B(x)$ then x has larger parts in A than in B. In this way hierarchies can be defined, using the so called 'degree of membership'.

Fuzzy Logic can be applied to modelling fuzzy rules with linguistic variables as mentioned before. One application example is reasoning fuzzy rule implications such as:

IF it IS warm AND the sun IS shining THEN the beaches ARE crowded

Those linguistic fuzzy rules can be derived from numerical data. In 1997, Nozaki, Ishibuchi and Tanaka introduced an algorithm exactly for that purpose (Nozaki, Ishibuchi, & Tanaka, 1997).

This work is concerned with a real life application of Uber rating predictions. All major ride-sharing and ride-hailing companies rely on rating systems to promote good drivers and passengers. At the end of each trip, riders and drivers have the opportunity to rate each other from 1 to 5 stars, based on their trip experience. All ratings are anonymous and cannot be traced back to individuals. Users can see the average rating of other users, so drivers with a higher average rating have a higher chance to be chosen, and vice versa. For simplification, the focus lies solely on modeling drivers ratings based on two input variables: driver style and conversation quality. The output variable is the respective rating of the driver.

The goal of this work is to derive linguistic fuzzy rules of the rating behavior purely using numerical input data. For the derivation of the fuzzy rules the algorithm of the mentioned paper is applied (Nozaki et al., 1997).

2 Method

The main steps are the acquisition of numerical data for the input and output variables, the implementation of the algorithm and the generation of rule tables, that represent the linguistic fuzzy rules. Afterwards, the rule tables are used to predict the ratings of unseen input variable values.

2.1 Membership Functions

The membership functions were chosen to be normally distributed gaussians, due to the fact that gaussians are continuous, monotonic and often observed in real life applications. Furthermore, the use of gaussians simplifies algorithmic implementation and the adaption of parameters concerning variance.

The first fuzzy input variable is called 'Driver Style' and contains three fuzzy sets with respective linguistic values 'slow', 'good' and 'fast'. Whereas driver style 'good' should result in an expected average rating (0.5), 'fast' should have a positive influence on the rating, since arriving 'fast' at the given location is thought to be desired. Vice versa, driver style 'slow' has a negative impact on the rating. In our standard model, membership functions of slow and fast have their means set to 0.15 and 0.85 respectively, as shown in figure 1a. All membership functions have a standard deviation σ of 0.15, resulting in overlapping membership functions (both extreme membership functions share approximately 2% of their area under the curve.

The second fuzzy input variable 'Conversation Quality' consists of three fuzzy sets and linguistic values as well, called 'boring', 'ok' and 'interesting'. Again, the conversation quality 'ok' should result in an average rating (0.5). While 'entertaining' impacts the rating positively and 'boring' has a negative influence. For this input variable, membership function means are put to the limit (0 for boring and 1 for interesting), with higher σ-values for all membership functions. As shown in figure 1b, σ_{boring} and $\sigma_{interesting}$ are set to 0.3, while σ_{ok} is set to 0.2. In the last part of this exercise we will discuss the effects of changes in parameters for different variables.

For the fuzzy output variable 'Rating', five fuzzy sets with respective linguistic labels were chosen based on common 5-star rating systems. The

five fuzzy sets and linguistic values are called 'terrible', 'bad', 'average','good' and 'perfect', respectively equivalent to 1 to 5 stars. The values represent worst to best rating in ascending order. The value 'average', as the name indicates, is the average expected rating (0.5). As shown in figure 1c, the two extremes have the means of their membership functions set to limit values (0 and 1) again. Between the two, the labels 'bad', 'average' and 'good' are placed evenly. By setting σ equal to 0.1, we mimic the triangle-shaped membership functions used in the original paper (Nozaki et al., 1997), however in our example, the membership functions of two non-neighboring linguistic labels still share around 1.4% of their area under the curve (as compared to zero for the membership functions used in the original paper).

2.2 Data Acquisition

For evaluating the algorithm, mock data was created based on the membership functions of a previously defined fuzzy model using the Fuzzy Logic Toolbox by Matlab as you can see in the attached matlab-file (Fuzzy_final.fis). The fuzzy rule implication defined in the Matlab model were (1-5).

(1) IF driver style IS fast AND conversation IS entertaining THEN the rating IS perfect

(2) IF driver style IS good AND conversation IS entertaining THEN the rating IS good

(3) IF driver style IS good AND conversation IS ok THEN the rating IS average

(4) IF driver style IS good AND conversation IS boring THEN the rating IS bad

(5) IF driver style IS slow AND conversation IS boring THEN the rating IS terrible

Two training data sets and one test data set were created. One data set completely aligns with the expected behavior of the outcome: high values in driver style and conversation quality results in high rating, vice versa (dataset.csv). Another one data set was made to test the influence of outliers on the model: high values in driver style and conversation quality results in a lower than expected rating, and vice versa (dataset_outliers.csv). The exact outliers can be seen in 1. The first three outliers in table 1 result in a rating that deviates two label values, the other four outliers deviate by one

Table 1: Outlier Data

Driver Style	Conversation	Rating
fast (.920)	interesting (.800)	average (.410)
slow(.100)	boring (.220)	average (.530)
average (.300)	ok (.400)	good (.800)
slow(.190)	boring (.210)	average (.201)
average (.400)	interesting (.800)	average (.550)
fast (.800)	interesting (.800)	good (.802)
slow (.011)	boring (.230)	bad (.340)

label value from the expected rating. This data set was used for training and creating the linguistic rule tables. Additionally, a third 'test data set' was created, that does not contain any input combinations of the training data set (dataset_testset.csv). This set was used to generate predictions of the rating.

3 Implementation

Our programming language of choice is Python. To derive fuzzy if-then rule tables with linguistic labels in the consequent part plus a degree of certainty, multiple steps have to be taken in advance:

3.1 Classes

We have to divide y into fuzzy sets of our linguistic labels, in our case the five ratings explained in the Method section. For this, and the generation of the fuzzy input sets "Driver Style" an "Conversation", we build the class ***FuzzyPartition***. It comes with three functions:

create_gaussian(self, mu, sigma) returns the functional value of a gaussian membership function for a fuzzy set, taking μ and σ as parameters.

generate_fuzzy_partition(self, mu_sigma_list) returns the fuzzy set, in our case a list of gaussian membership functions. The parameter *mu_sigma_list* is a list of lists, with every sublist containing the μ and σ values of the corresponding gaussian membership function. We needed to write two functions (***generate_fuzzy_ partition(self, mu_sigma_list)*** and ***create_gaussian(self, mu, sigma)***) in order to generate the fuzzy sets. This

was due to a late binding problem that occurred when trying to generate a list of functions with the lambda functionality directly in one function.

__init__(self, mu_sigma_list, labels, name) creates the FuzzyPartition object, taking a list of gaussian parameter values, a list of linguistic labels and a name for the set as parameters.

The class ***FuzzyExample*** then unites all fuzzy input and output sets in one object, for which it needs all the inputs necessary for each ***FuzzyPartition***. It stores two ***FuzzyPartition*** as inputs and one ***FuzzyPartition*** as the output. Thus, our program is hard-coded to handle problems with two inputs and one output. ***plot_fuzzy_partitions*** plots the membership functions of all fuzzy sets, with one plot for each ***FuzzyPartition***.

3.2 Functions

Apart from classes, several helper functions are necessary, and will be presented here in a hierarchical order:

create_tables(x1, x2) creates an empty (filled with nan) dataframe with x1 columns and x2 rows. The parameters are lists of input-labels.

membership_function(partition, x) takes a list of gaussian membership functions in the form of a partition and an input value x. It calculates the functional value of x for every membership function.

calc_compatibility_degree(x, partition) takes one of the input values (x_1 or x_2) and a fuzzy partition. It first calculates all membership values of x_1 (or x_2), then calculates and returns the compatibility degree of the input with the fuzzy if-then rules, by multiplying its membership values.

sort_by_membership_value(partition, x) takes a list of gaussian membership functions in the form of a partition and an input value x. It applies ***membership_function(partition, x)*** and returns the original indices of the data, sorted by the membership value that corresponds to the gaussian at the position of the index, as well as the corresponding sorted membership values.

calc_b(x, a, partition_in_1, partition_in_2) takes a list x consisting of inputs and and output in the last position, an alpha value a, as well as two partitions of membership functions. It performs the calculations given

in the equations 9 and 10 of the original paper and returns b for the given sample.

get_b_values(partition, labels, b) takes a list of membership functions, their linguistic *labels* and b from ***calc_b*** to calculate the equation 16 of the original paper. It starts by applying ***sort_by _membership_value (partition, b)*** and returns the best fitting label and its degree of certainty as *b_star* and *b_star_cf.* as well as the second best fitting label and its degree of certainty *b_star_star* and *b_star_star_cf.*

fill_table(partition_in_1, partition_in_2, partition_out, x, prim_table, sec_table, a) does the actual calculation for every sample in our data and returns the (eventually updated) primary and secondary rule table as a result. It takes all input and output partitions, the primary and secondary rule tables, as well as α and the datapoint x, which is a list of three numbers, with the first two representing the input values driving style and conversation, and the last representing the output (rating). It starts by calculating b for the data, and then uses ***get_b_values*** for obtaining all B^* and B^{**} values for output and inputs.

The function then checks if there is no previous entry (NaN is a float), and places B^* and B^{**} as well as their respective degree of certainty in the rule tables. If there is an existing entry, it chooses to keep the sample with the higher degree of certainty.

predict_sample(sample, alpha, partition_in_1, partition_in_2, partition_out) performs the calculation of equation 24 in the original paper. It takes a sample (x_p, y_p), alpha, and all three fuzzy partitions as inputs. It starts by calculating all B^* related values for the output and the membership values of all membership functions of the input. $\bar{B}^*$ and $\bar{B}^{**}$ are the center values μ_B of the consequent fuzzy set that best represents the sample. The function then calculates the inferred output $y(x_p)$ and returns it.

In the ***main()*** function, we initialize all parameters as described in the method section, build a ***FuzzyExample*** of it and set alpha to 1. We initialize the dataframes for the rule tables with the input labels, read in our data, and apply ***fill_table*** to all datapoints, filling both rule tables iteratively. It then runs over the test dataset in the same manner to compute the inferred output (with the formula (24) of the paper), compares it to the ground truth, and computes the mean squared error. Finally, it iterates through different values of alpha in the range of [0.1,3] to see how alpha affects outcome and performance.

4 Results

The generated rule tables represent the underlying fuzzy rules that could be derived from the data. When formulating the respective rules from the table, it is clear that the derived rules correspond to the expected outcome.

(1.1) IF driver style IS fast AND conversation IS entertaining THEN the rating IS perfect

(2.1) IF driver style IS good AND conversation IS entertaining THEN the rating IS good

(3.1) IF driver style IS good AND conversation IS ok THEN the rating IS average

(4.1) IF driver style IS good AND conversation IS boring THEN the rating IS bad

(5.1) IF driver style IS slow AND conversation IS boring THEN the rating IS terrible

Furthermore, it can be observed that more fuzzy rules were derived than what was previously defined explicitly in the Matlab model.

(6) IF driver style IS slow AND conversation IS entertaining THEN the rating IS average

(7) IF driver style IS slow AND conversation IS ok THEN the rating IS bad

(8) IF driver style IS fast AND conversation IS boring THEN the rating IS average

(9) IF driver style IS fast AND conversation IS ok THEN the rating IS good

With 56 samples out of which 6 were outliers, the algorithm filled the first rule table with coherent rules, as shown in table 2. Table 3 shows the secondary rules. All primary rules have a degree of certainty higher than 0.5, while all secondary rules have a degree of certainty below that. There were no samples in which the two fuzzy sets fast and boring were the second most certain combination of sets. The scaling parameter *alpha* had no effect on the outcome within the tested range of [0.1,3].

	slow	**average**	**fast**
boring	terrible, 1.0	bad, 0.882	average, 0.546
ok	bad, 0.956	average, 1.0	good, 0.923
entertaining	average, 0.613	good, 1.0	perfect, 1.0

Table 2: Resulting primary rule table for the outlier dataset

	slow	**average**	**fast**
boring	perfect, 0.135	NaN	NaN
ok	average, 0.43	terrible, 0.458	average, 0.135
entertaining	bad, 0.375	perfect, 0.375	good, 0.044

Table 3: Resulting secondary rule table for the outlier dataset

True output	**Inferred output**
0.932	0.783
0.42	0.217
0.557	0.511
0.202	0.223
0.11	0.013
0.637	0.867
0.566	0.533
0.306	0.017
0.575	0.527
0.402	0.233
0.812	0.989
0.843	0.989
0.921	0.989
0.798	0.783
0.99	0.987
0.182	0.217

Table 4: Classification performance on test datapoints new to the algorithm after training with the 56 sample dataset. *Mean Squared Error* = 0.019

5 Discussion

Classification performance clearly outperforms the null model, with very good performance for very high and very low true output values, but some major deviations for some medium output values, as shown in table 4. The greatest deviance is 0.289, which equals more than one linguistic output label

(inferred output is terrible instead of bad). On average, the inferred output differs by 0.138. Even though a Mean Squared Error (MSE) of 0.019 appears to be small, it is always to be considered in comparison with another model, in our case the baseline model of simply predicting 0.5 for every sample. For uniformly distributed data, this would yield $\sigma = 0.25$ and a MSE of 0.0625, with a maximum deviation of 0.5. Having this in mind, the classification performance is decent, but not perfect. However, considering the algorithm had only 56 samples of training data out of which 7 were outliers, the results fit our expectations.

5.1 Rule tables

From the main rule table, we can obtain linguistic knowledge that perfectly aligns with the expected outcome. For both input variables, an increase in quality of service results in a better rating, both extreme ratings can be obtained through the corresponding extreme input values. The obtained degree of certainty for similar input values (e.g. driving style: fast, conversation: entertaining or driving style: average, conversation: ok) is perfect, but the degree of certainty decreases with deviance in the two values. The algorithm struggles to infer outputs for the following inputs: driving style: fast, conversation: boring and driving style: slow, conversation: entertaining. This becomes evident in the main rule table in the form of a lower degree of certainty (0.546 and 0.613 respectively).

The secondary rule table provides no useful information. We expected the output labels in the secondary rule table to differ almost exclusively by one step (e.g. good for fast and entertaining, bad for slow and boring), when trained with outlier free data, the combination of average and ok is actually expected to be two output labels off from the main table output label. However, our data contains outliers, so the one label deviation is only found in 5 out of 7 determined input label combinations. The two outliers are:

1. Slow and boring putting out perfect instead of bad
2. average and ok putting out terrible instead of bad or good

There are two input combinations that were never the second best fitting, average and boring as well as fast and boring. This outcome is to be expected for small datasets, as extreme label values combinations (slow and boring, fast and boring, slow and entertaining, fast and entertaining) are less likely to be obtained. By the way the secondary rule table is set up, extreme

label combinations can only get filled in when the sample reaches its highest confidence with the two medium labels, average and ok. To be filled in the secondary rule table, both input labels have to differ from the main rule table labels.

Both unexpected results in the secondary rule table can be explained: the terrible rating for the average and ok input results from the sample [0.1, 0.2, 0.125], where the output is exactly at the equilibrium point of two output label membership functions, terrible and bad. The algorithm than randomly assigns bad as the highest membership value and terrible as the second highest. Since the best fitting input labels are slow and boring, the two second best fitting input labels are average and ok, and terrible is assigned here. With a sufficient number of outliers, this result can be expected to be overwritten, but our very limited data does not contain the perfectly fitting outlier to overwrite this result. The other unexpected result can be explained by the outlier [0.3, 0.4, 0.8], where the best fitting output label was good, and the second best output labels (average and perfect) were at their equilibrium point. As for input labels, the two best fitting input labels are average and ok, with the second best fitting labels being slow and boring. Thus the algorithm assigned perfect in the second rule table.

It is important to note that difficulties in filling the secondary rule table, as explained above, do not apply and do not affect the main rule table. For the problem chosen here, the main table already suffices in differentiating all label combinations. If the output was parted with $k < 5$ or the input parted with $k > 3$, the secondary rule table would have been necessary to disambiguate.

5.2 Gaussians as membership functions

The use of gaussians over triangle shaped membership functions significantly simplified the implementation, as well as experimentation with different k-values. However, it rendered the α-parameter ineffective, and had no positive effect on performance overall.

5.2.1 Alpha

Changes in the α-parameter did not change anything in the outcome of the algorithm. As explained above, this is due to the choice of gaussian membership functions. Effects similar to the expected change with differing α can be obtained by changing the σ-parameter of the gaussian membership

function, with small values of σ being equivalent to $\alpha > 1$, and high values of σ being similar to $\alpha < 1$. With values of α approaching infinity when applied to a triangle shaped membership function, the result approaches a dirac-δ-distribution, similar to a gaussian function with infinitely small σ and its value at μ normed to 1, vice versa for infinitely small α, where both functions become similar to a uniform distribution.

5.3 Parameter values

There are two different possibilities to meaningfully divide the interval [0,1] evenly between three gaussian membership functions: Either the extreme values have their μ set to 0 and 1, or not. We tried both versions, with our input 'Driving style' having their μ set to 0.15, 0.5 and 0.85 and the μ of 'Conversation' set to 0, 0.5 and 1. There was a slight change in performance between the two variables for some variable combinations, with driving style having a more pronounced effect on output values. The exact combination will be discussed later. However, for most here was no noticeable change in performance between the two variables, for most combinations of membership functions, as long as symmetry was preserved. Even setting the extreme μ values of the input labels to 0.4 and 0.6 did not decrease performance. This is due to the fact that the actual value of a membership function only matters for the obtained degree of certainty, but not for the final linguistic label, as long as the order of the labels is correct and as long as their membership functions do not differ in σ.

The effect of σ for input labels have already been discussed above. Changing σ in the output membership functions affects the classification performance: For small values of σ (0,0.05] the algorithm only predicts values equal to the μ value of an output membership function, because the membership of any other value for any membership function is almost zero. With increasing values of σ, the predicted values converge to the equilibrium point between membership functions. For $\sigma = 1$ this behavior becomes almost exclusive.

Classificaton performance does neither differ significantly between $k_{in} = 2$ and $k_{in} = 3$, nor between $k_{out} = 3$ and $k_{out} = 5$. However, the importance of the secondary rule table should be observable for the combination of $k_{in} = 3$ and $k_{out} = 3$. Here, the abundance of input label combinations that are most compatible with the average output label should be differentiable with the help of the secondary rule table. However, the same complications already mentioned above also apply here, so the predicted behavior is only partly visible with our data.

As a final remark, user ratings are usually set to be 5 stars almost by default. In the case of Uber, not giving 5 stars requires an explanation by the user, else the rating does not count. This difference in rating without any difference in the input variable values can easily be modeled by changing the membership functions of the output to: [(0, 0.1), (0.2, 0.1), (0.4, 0.1), (0.6, 0.1), (1, 0.5)] from terrible to perfect. The resulting rule tables, as shown in table 5 and 6 show two things:

1. The inferred user ratings get shifted towards perfect ratings
2. The effects of driving style are more pronounced than the effects of conversation. This is due to the difference in membership function μ values.

	slow	**average**	**fast**
boring	terrible, 1	average, 0.956	good, 0.995
ok	bad, 1	average, 0.995	perfect, 0.962
entertaining	average, 1	perfect, 0.882	perfect, 1

Table 5: Resulting main rule table for the 'Uber' version

	slow	**average**	**fast**
boring	good, 0.135	NaN	NaN
ok	perfect, 0.783	perfect, 0.738	average, 0.607
entertaining	perfect, 0.475	perfect, 0.783	average, 0.607

Table 6: Resulting secondary rule table for the 'Uber' version

5.4 Changes of input data/ code

As requested, the user can change the following things:

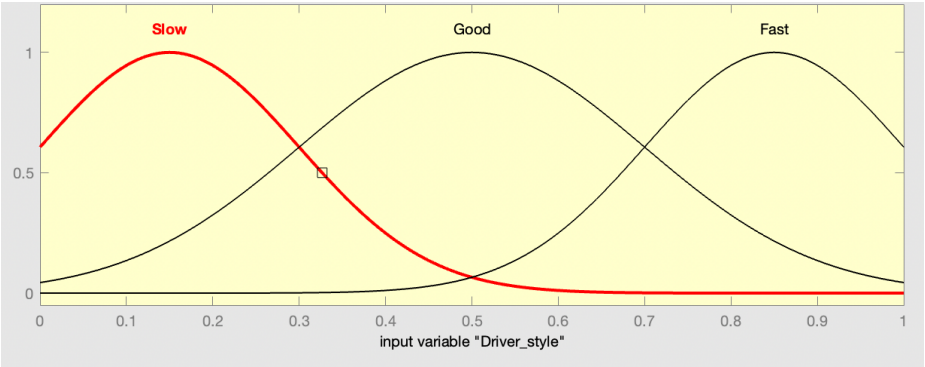
1. input data: The input data can be changed by changing the values in the csv-files. The csv-file which is used to fill the primary and secondary ruletable is loaded into the code in line 257 in the main()-function.
2. K_{in_1}, K_{in_2} and K_{out}: this can be done by adding/ removing labels and items of the corresponding mu_sigma_lists in the lines 242-248 in the main()-function.

3. alpha: The alpha-value can be changed in line 260 in the main()-function. However, a change of the alpha value does not change the obtained ruletables, which is due to the fact that we used gaussians as membership functions as discussed in 5.2.

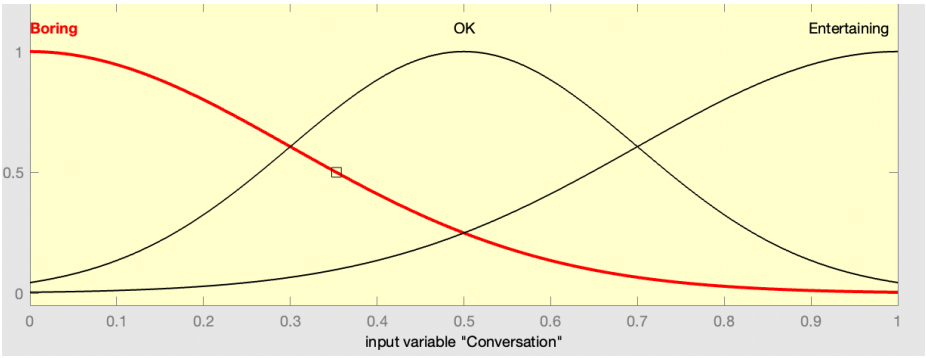
References

Nozaki, K., Ishibuchi, H., & Tanaka, H. (1997). A simple but powerful heuristic method for generating fuzzy rules from numerical data. *Fuzzy sets and systems*, *86*(3), 251–270.

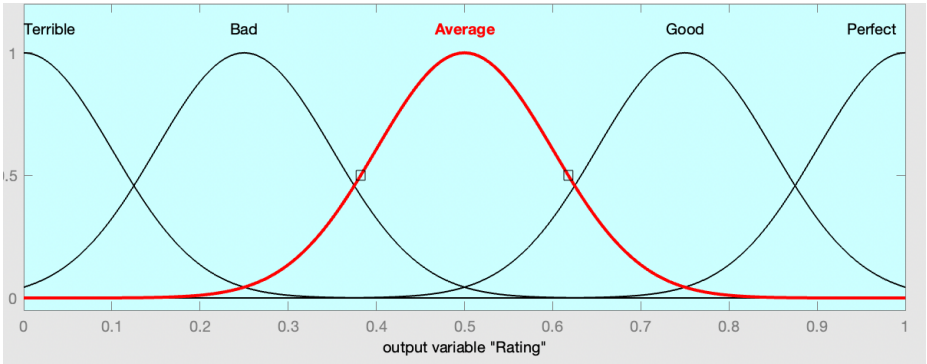
Zadeh, L. A. (1965). Fuzzy sets. *Information and control*, *8*(3), 338–353.



(a) Membership Functions 'Driver Style'



(b) Membership Functions 'Conversation Quality'



(c) Membership Functions 'Rating'

Figure 1: Membership Functions of the Input and Output Variables