Using Learning Automata for Tuning Fuzzy Membership Functions in Learning Driver Preferences

Narges Afshordi and **Mohammad Reza Meybodi**[1]

Abstract. With the growth of car navigation systems technology comes a variety of enhancements aiming to increase user comfort and satisfaction. One such application is the appearance of methods for learning a driver's preferences in making a choice between several routes. A driver may know his/her basic and most important factors in making such a decision, but may have these factors weighing in differently. Hence, machine learning methods can be applied to model the driver's preferences, thus predicting the result of the decision process. This paper proposes a new method which combines a fuzzy expert system approach with learning automata.

1 INTRODUCTION

Car navigation systems have made great progress in the recent years, making route planning faster and easier. These systems are normally designed to find the *optimum* route between a given source and destination. Often, the *optimum* route is either the route with the shortest distance or the shortest time, meaning only one parameter such as distance is taken in as a deciding factor. This can be somewhat limiting, as the driver may want to prefer a route that has less traffic to one that is longer in distance. A survey [1] carried out in London, Paris and Munich showed that users find a system that chooses a route making a tradeoff between route time and route distance just as appealing as one that would choose only based on route time, if not more desirable.

Due to this, more research has been channeled towards introducing different factors into the process, leading to multi-objective route planning. Another step forward and we are able to interact with each user as an individual and learn his/her preferences. To do this, we must be able to apply a learning method that would adaptively keep track of the user's choices, and deductively be able to make choices similar to those of the user in future route selections.

1.1 Literature review

Different methods of modeling the user's behaviour have been proposed. Rogers et al [2, 3] introduced a method that comprises of a simple perceptronic model. For each given source and destination, the user is given several routes. A cost, which is the weighted sum of its attributes, is assigned to each route. If the user chooses the route with the least cost, no change is needed for the time being. However, if the user chooses another route, the perceptron alters the weights being applied, changing them so that the chosen route will have the least cost among all routes. A user model consists of the weights' vector consistent with the user.

Fuzzy logic methods have also been used in route selection. An important work was contributed by Pang et al [4] in their Fuzzy-Neural approach to solving this problem. Their system initially asked the user to indicate the importance of different parameters. From this information a set of fuzzy rules is made, which is implemented by using a fuzzy neural network. This network takes the real values of the different parameters, fuzzifies them, applies the rules, and assigns a score to the route as the output. The network is tested against user choices. If the system's choice differs from that of the user, the weights are recalculated. The user model in this method is the weights for the neural network connections.

On the other hand, methods have been proposed for tuning fuzzy membership functions in fuzzy logic controllers. Berenji and Khedkar [5] proposed the GARIC structure which uses a fuzzy-neural network to implement a fuzzy rule base. It also makes use of another neural network serving as an internal critic in the task of tuning the membership functions. This internal critic enables the system to use reinforcement learning methods as a means for reducing the controller's error.

1.2 The problem

This paper proposes a new method for learning a driver's preferences in making a choice between several available routes. It uses a fuzzy expert system approach, linked together with learning automata. The proposed system is made of three major elements. The first deals with the map and finding routes between a given source and destination. The second element is a fuzzy expert system and the final part consists of learning automata, introduced to tune the membership functions of the fuzzy variables, and to enhance the scoring precision of the fuzzy expert system. The learning automata can be considered as embedded in the fuzzy expert system because their functionality is closely interlocked. Feedback from the environment causes changes in the learning automata, which in turn results in changes in the membership functions.

The new method proposed in this paper is different as it makes use of reinforcement learning methods to deal with an environment that is not always transparent and clear. Such methods had not been used for adaptive routing prior to this. Furthermore, many of the previous endeavors using fuzzy reasoning focused on being able to describe and model user behavior, rather than finding a way to predict it. They were also less intent on exploring the different parameters that may be involved in the process of making the decision. The new system also shows flexibility as it can build a new and individualized

[1] Department of Computer Engineering and Information Technology, Amir Kabir University of Technology, email: afshordi@cic.aut.ac.ir, mmeybodi@aut.ac.ir

rule base for each user, in contrast with more rigid and fixed rule bases that are usually used.

Implementation results show that the proposed method shows better prediction ability and less error when dealing with new situations. The rest of the paper is as follows, section 2 will deal with search methods, and section 3 with the fuzzy expert system. Then section 4 will explain the mechanism of the learning automata. The results are discussed in section 5, and concluding notes are mentioned in section 6.

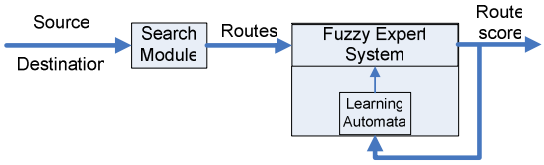

Figure 1. General structure of the system

2 SEARCH MODULE

Our system is designed and implemented on the city of Tehran. The city map is first vectorized, and is then stored in the form of a directed graph, with each edge representing a strip of road between two junctions, and each node representing a junction. As Tehran is a relatively large city, the graph in question is also quite big, consisting of more than 28,000 nodes. This graph is used for searching. Each edge has several attributes associated with it. One of these is the road type which shows whether the edge is on a highway, a main road, or a minor road.

The source of a trip may be set to be the place the car is in at that moment, meaning it is discovered through the use of the map matching module of the car navigation system. It may also be entered by the user, along with the destination. There are many algorithms for finding the k-shortest routes in a graph [6]. In this system a generalized version of the Dijkstra algorithm [7] is used to find several routes between the source and destination. This algorithm first uses Dijkstra to find one route. Then all edges on that route are given a relatively high cost, so as prevent the second route from passing through them again, unless absolutely necessary. The resulting routes will be given to the fuzzy expert system.

3 FUZZY EXPERT SYSTEM

The second element is a fuzzy expert system that gives each route a score. These systems have been accepted as efficient decision support systems. But first, as with any multi-objective optimization process, the first step is specifying the different parameters. Previous methods have used several parameters for making this choice. Route distance and route time are very common characteristics, but there are also a number of other parameters. Rogers et al [2] used number of turns and number of intersections. Pang et al [3] made use of the degree of congestion, toll, degree of difficulty and scenery.

Some parameters are more appropriate for out of city trips. For example, scenery might not be much of an option in a busy and crowded city, such as Tehran. Considering the different aspects of each parameter, the ones chosen for being used are:

- **Distance:** The total distance of a route can be calculated as the addition of the distance of its constituting edges.
- **Time:** The time for a route can also be estimated. Time is clearly a very important parameter.
- **Congestion:** Cost is assigned to congested subparts of a route. If there is no congestion, the cost will be 0.
- **Number of intersections:** Intersections between main roads are only taken into account, as passage through them requires more focus and attention on the driver's part.
- **Degree of passage through highways:** This parameter shows what fraction of the route passes through highways.
- **Degree of passage through main roads:** This parameter is similar to the last one, only it shows what fraction of the route passes through main roads.
- **Degree of passage through minor roads:** This parameter is of interest, as minor roads such as alleys are usually less congested, and can provide a fast way out of traffic.

These parameters are included and computable for a route in the system, but this does not mean the user has to use all of them. A user usually has a fairly good idea about the parameters most critical in his/her decision making. This information is valuable because the system needs to be able to learn quickly. Each trip that is supported by the system provides it with only one set of data, meaning that learning needs to occur using relatively limited amounts of data. Thus, any prior information about a specific user will be useful in reducing learning time and effort. In order to get this information, a new user is asked to indicate the degree of importance he/she links to each characteristic. The choice is between these:

- **Very important:** This means that this characteristic should have a lot of influence in making the rules, and that changes is such a characteristic are important.
- **Important:** The characteristic is important, but not as much as a very important one.
- **Slightly important:** This characteristic is not important. Only great changes in it will cause a difference in the result.
- **Don't care:** This characteristic has no influence on the rules.

Each one of these characteristics is considered to be a linguistic variable. Therefore the importance of these characteristics helps in the building the fuzzy rule set. Before explaining the mechanism of the rules however, it is important to specify the different labels for each linguistic variable, which are available in Table 1. There is also an output characteristic that will show the *route quality*. Unlike the input variables, this one has 5 labels, which are: very bad, bad, average, good, and very good.

In order to build the rule base, we need a set of rules for making the rules. They are explained here:

Table 1. The labels for the route parameters

Parameter	*Value #1*	*Value #2*	*Value #3*	*Value #4*
Distance	Very short	Short	Long	Very long
Time	Very short	Short	Long	Very Long
Congestion	None	Light	Heavy	Very heavy
Number of intersections	A few	Several	Many	Too many
Highway* passage	None	Some	A lot	Total

* The same labels are applied to the parameters: *Main road passage*, *Minor road passage.*

1. If the input parameter is marked as *very important*, then even slight changes in its value will result in change in the output. Poor values for it will cause poor values in the output, and vice versa.
2. If the input variable is marked as *important*, a poor value for this variable will result in the worst cast. However the best value for this variable will not invoke the best output.
3. If the input variable is marked as *slightly important*, a poor value for this variable will result in a poor (but not worst) value for the output. For better values of it, the route will only be regarded as average.

Table 2. The rules for extracting the fuzzy rules

Input importance	*Output for value#1*	*Output for value#2*	*Output for value#3*	*Output for value#4*
Very important	Very good	Good	Bad	Very bad
Important	Good	Good	Bad	Very bad
Slightly important	Average	Average	Bad	Bad

An example of applying the rules in Table 2 is given here. If the distance variable is very important, the rules below will be extracted:

- If route distance is *very short*, route is *very good*.
- If route distance is *short*, route is *good*.
- If route distance is *long*, route is *bad*.
- If route distance is *very long*, route is *very bad*.

The majority of the rules have a simple antecedent, consisting of only one condition. There is however an exception. In situations where there are two or more variables marked as *important*, there will be a rule with two conditions in the antecedent, joined by the junction AND. This rule will make sure that the best values for both of these variables will result in the best value for the output.

To apply the fuzzy rule base, several points need to be cleared. The min operator is used for determining the degree of support for a rule with more than one condition in the antecedent. If there arises a situation in which there are two or more rules being fired at the same time, a method similar to that of Berenji and Khedkar's [5] is used for conflict resolution. For defuzzification purposes, smallest of maximum (SOM) method is used.

The variables being used in the problem have a membership function for each of their labels. The type of membership function chosen for them are triangular shaped. Triangular membership functions are both simple and have been proved to be sufficient in scores of application domains [5]. It is worth mentioning that the first and last membership function for each variable is a trapezoidal membership function, to allow for complete membership of extremely small or large values.

In addition Gaussian membership functions are also separately implemented and tested at a later stage.

4 The learning automata

The layout of the membership functions is very important. The reason for tuning the membership functions is to arrive at a layout which fits the driver. A lot of people might consider a route with a short time to be a good one, but their idea of what is a short route, and what is not a short route, can be very different. The learning automata will help in finding that exact layout for each person. All input and output variables are normalized to the [0, 1] range. The general layout for each input variable is chosen from a set of 10 different layouts. There is another set of 10 layouts for the output variable, route quality. The layout is chosen randomly at each iteration of execution, with each layout having a fixed probability of 0.1.

For each variable, its membership functions are further customized by choosing the position of the center. The center position is considered to be relative to the start, and is chosen from the set below

$$\mathbf{R} = \{0.2, 0.4, 0.5, 0.6, 0.8\} \quad (1)$$

Figure 2 shows several membership functions with the same start and end points, but with different center positions chosen from the set **R**.

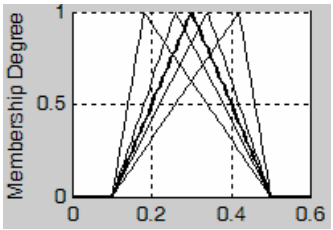

Figure 2. Membership functions with different centers

The choice of the center is very important and this is where the learning automata are put into action. For each variable label, there will be a variable structure stochastic automaton, providing a choice for the center. The choice of one of the members of R is an action for the automata. As **R** has 5 members, each member of **R** has an initial action probability of 0.2.

At the beginning of dealing with a new user, the system asks for the importance of each variable and builds the rule base. In the first trip, the system provides the user with several routes. If

the route chosen by the user is also the system's chosen one, nothing is done at this moment. If otherwise, the scores attributed to the routes are reversed, so that the user's choice has the best score. Then the learning automata have a set of data that it can use to tune the membership functions.

The general algorithm for the system is given below:

1. Set ε = 0.01.
2. Choose layouts for the input and output variables, and a center for each membership function randomly.
3. Take in the source and destination as inputs.
4. Search module provides the fuzzy expert system with possible routes, $\{r_1,...,r_n\}$
5. Apply fuzzy rules and acquire a score for the routes, $\{s_1,..., s_n\}$.
6. If the user chooses the route with the best score go to 3.
7. If the user chooses another route, replace the score for the user's route and the best route. The resulting pairs $\{(r_1, s_1), (r_2, s_2)\}$ are tuning data.
8. Choose layouts and centers for the input and output variables randomly.
9. Apply fuzzy rules to the tuning data.
   a. If (error < ε) go to 3.
   b. If (error > ε) and feedback is positive reward the chosen center positions.
   c. If (error > ε) and feedback is don't do anything.
   d. Go to 8.

Several issues should be cleared at this point about the algorithm. Randomly choosing a layout is not exactly random. For choosing a layout a random number is generated between 0 and 1. Each of the ten possible layouts has a 0.1 chance of being chosen, therefore the space [0,1] is divided into ten smaller intervals. A layout is chosen depending on whether the random number falls into its interval or not. A similar process is followed for choosing a center for a membership function. As mentioned, the initial probability for each center position is 0.2, however this will change in the course of step 9.

The error is the average for the error on each of the tuning data. The error on each data is the difference between the score it should have and the score returned by the fuzzy expert system.

To assess the feedback, the degree of support for a rule is compared to the degree of membership in the output variable. These should be ideally equivalent. If this difference is below a certain threshold, positive feedback is given to the learning automata, and vice versa.

4.1 Learning method

There are several widely common linear methods used for updating the action probabilities of a learning automaton. L_{P-R} is one such method[8]. If there is a positive feedback the chosen action will be rewarded, and the remaining actions that were not chosen will be penalized. On the other hand, if there is a negative feedback, the chosen action will be penalized and the other actions will be rewarded. Rewarding in these updating methods always means increasing that action's probabilities, while penalizing will mean decreasing it. This process will continue until one of the action probabilities reaches 1, causing the learning automaton to converge.

This method was applied to this problem, but the learning automata did not always converge, and showed poor and slow results when it did.

L_{R-I} is another linear method[8], where the chosen action is rewarded in the case of positive feedback and the other actions are penalized. The difference with the previous method is when the system receives negative feedback. This method chooses inaction in this case, making no changes to the action probabilities. Equation 2 shows the updated probability for an action that has resulted in a positive feedback, where *a* is the reward coefficient, chosen from (0, 1). Equation 3 shows the updated probability for other actions that were not chosen.

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)] \tag{2}$$

$$p_j(n+1) = (1-a)p_j(n), \; j \neq i \tag{3}$$

When a negative feedback occurs, the probability for any action will remain at its previous value.

$$p_i(n+1) = p_i(n) \tag{4}$$

This method has shown to be efficient in this problem.

5 RESULTS

Implementation was done using MATLAB, and all tests were run on a 1.60GHz processor. An acceptable result consists of a permutation of membership functions and their centers, so that the error is less than ε. If there are no learning automata, this permutation can still be reached through numerous samplings. In this case it normally takes an average 711.5 epochs of sampling, and an average time of 9.71 seconds. The results show that the learning automata fully converge after an average of around 200 epochs. However, the condition for reaching a result is not necessarily convergence, and may happen even sooner, meaning that the maximum probability will not be unity, but very close to it. The learning automata method reaches an answer in an average 166.7 epochs, taking 2.29 seconds.

Figure 3 shows the results for a user who has chosen distance, time and congestion to have influence on the route choice. The GARIC structure proposed in [5] was also implemented and fitted to the problem. In comparison, the learning automata method is both faster and also shows smaller errors. As mentioned before, the same method was also implemented using Gaussian membership functions. In this version the learning automata tuned the membership function through making changes is *c* and *δ*. The first method is faster in comparison to this version too, and yields better results. Figure 4 shows Gaussian membership functions after learning.

6 CONCLUSION

A new approach for tuning fuzzy membership functions in a fuzzy expert system was introduced in this paper, providing a new alternative for modeling each individual user in an environment with a small number of available data. The method was applied and implemented to the problem of learning a driver's preferences in choosing a route in the city of Tehran. Implementation results show better performance in comparison with older methods.

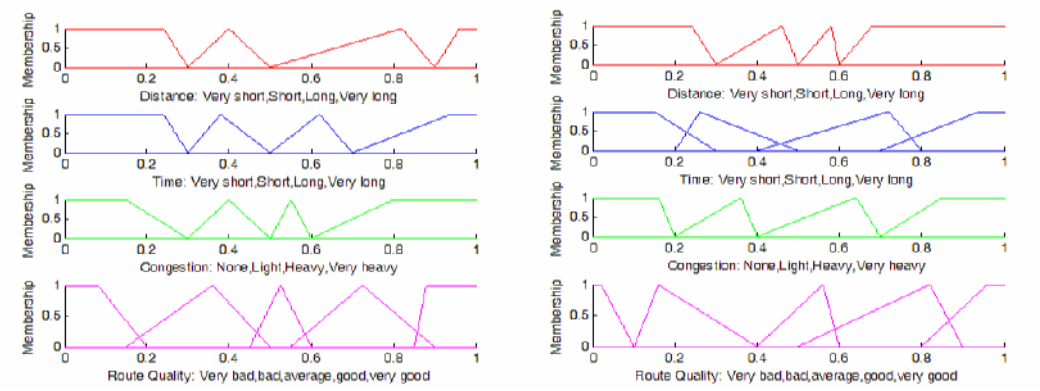

Figure 3. The membership functions for input and output variables: (a) is before learning, and (b) is after learning

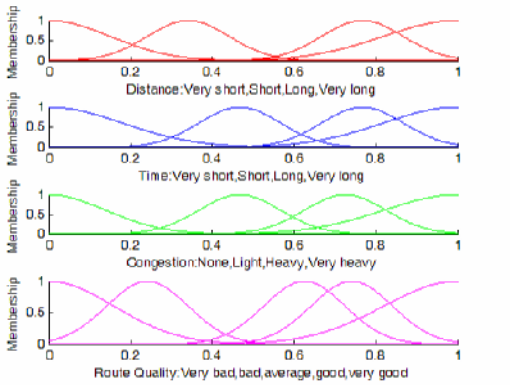

Figure 4. Gaussian membership functions after learning

REFERENCES

[1] P. W. Bonsall and T. Parry, 'Drivers' requirements for route guidance,' in *Proc. 3rd Int. Conf. Road Traffic Control*, May 1990, pp. 1–5.

[2] S. Rogers and P. Langley, 'Interactive Refinement of Route Preferences for Driving,' *Spring Symposium on Interactive and Mixed-Initiative Decision-Theoretic Systems,* March 1998, pp. 109-113.

[3] S. Rogers, C. Fiechter, and P. Langley, 'An Adaptive Interactive Agent for Route Advice', *Autonomous Agents '99, ACM*, 1999.

[4] G.K.H. Pang, K. Takahashi, T. Yokota, and H. Takenaga, 'Adaptive Route Selection for Dynamic Route Guidance System Based on Fuzzy-Neural Approaches,' *IEEE Transactions on Vehicular Technology*, Vol. 48, No. 6, November 1999, pp. 2028-2041.

[5] H.R. Berenji, and P. Khedkar, 'Learning an Tuning Fuzzy Logic Controllers Through Reinforcements,' *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, September 1992.

[6] D. Eppstein, 'Finding the k Shortest Paths,' *35th IEEE Symp. Foundations of Computer Science.*, Santa Fe, 1994, pp. 154-165.

[7] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to algorithms*. Cambridge, MA: MIT Press.

[8] K. S. Narendra, and K. S. Thathachar, *Learning Automata: An Introduction*. New York: Prentice-Hall, 1989.