INTELLIGENT CONTROL OF ROBOTIC SYSTEMS

Prof. M. Felix Orlando

Department of Electrical Engineering

Indian Institute of Technology Roorkee

Lecture 10: T-S Fuzzy

Good afternoon, everyone. Today, we are going to learn about TS fuzzy control with an example. And fuzzy C-means clustering. The outline of this talk will be as follows. First, we will see TS fuzzy control with an example of a single-link manipulator, where we can achieve the regulation objective, ensuring it settles down to a point for convergence. Then, we will discuss fuzzy control with optimal parameters.

How can we design the fuzzy controller? Designed with optimal parameters. Then, at the end of this lecture, we will explore fuzzy C-means clustering. Now, coming to TS fuzzy control, the major difference between Mamdani-type fuzzy control and TS fuzzy control is that the output is obtained by the linguistic variable in the case of Mamdani. Whereas here, the output is obtained through an expression or a model, precisely a linearized model.

Because we know fully and thoroughly about linearized models, we can derive the non-linearized model. We can clearly observe the non-linearized model. Let us see an example. The continuous TS fuzzy model of a single-link manipulator, whose dynamic model is given by

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -g\sin(x_1) + u$$

is described by four equations or four rules. Because the four rules are the building blocks of the TS fuzzy model for the single-link manipulator.

Rule 1 states that $\text{IF x(t) is around } [0\ 0]^T \text{ THEN } \dot{x}(t) = \begin{pmatrix} 0 & 1 \\ -9.81 & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t).$

Rule 2 states that $\text{IF x(t) is around } \left[\pm\frac{\pi}{6}, 0\right]^T \text{ THEN } \dot{x}(t) = \begin{pmatrix} 0 & 1 \\ -9.37 & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t$

Similarly, for rule 2, we have x1 taking two points, two values precisely. It is plus pi by 6 and minus pi by 6.

With the linear model x dot equal to ax plus bu, where a is 0 1 minus 3 point minus 9.370 and b equal to 0 1 for these two cases. Likewise, rule 3 has the linearized model

$$\text{IF x(t) is around } \left[\pm\frac{\pi}{3}, 0\right]^T \text{ THEN } \dot{x}(t) = \begin{pmatrix} 0 & 1 \\ -8.11 & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t)$$

And these matrices A and B are obtained by linearizing this given non-linear system by finding the Jacobian matrix dou F1 by dou X1

dou f1 by dou x2. Similarly, dou f2 by dou x1 and dou f2 by dou x2 with the point x of t equal to 0 and 0 transpose. With this, we get the Jacobian matrix that comes to be the linearized matrix A for Jacobian rule 1. Likewise, we can go for rule 2 by having the point P plus pi by 6 0, and we have one more rule corresponding to that with point minus pi by 6 and 0.

So, We have a total of 4 rules given here, out of which 3 rules are having plus or minus the equilibrium points being plus or minus pi by 6, 0; plus or minus pi by 3, 0; and plus or minus pi by 2, 0.

So, rule 2 again can be split into 2 and 3, and rule 3 will be 4 and 5, and rule 4 will be 5, 6, and 7, so a total of 1 plus 6 equal to 7 rules here can be found because of the equilibrium point given for the single-link manipulator. So now the question is to design a fuzzy regulator for the system. For that, consider that the local controller gains where j equals 1, 2, 3, 4 for the rules given by, we have specified the k controller gains for these rules.

Since rule 2 and 3, and 4 and 5, then 6 and 7, they do have the same sort of k values because of that plus or minus pi by 6, 0 equilibrium point here, and plus or minus pi by 3, 0 point here, and for plus or minus pi by 2, 0 for these k values. So, if I say this is rule 1 and this is rule 2. 2 and 3 will take these k values. Similarly, rule 4 and 5 will take these two k values, that is k3.

Likewise, rule 6 and 7 will take these. Okay, so we can consider a common Lyapunov matrix P equal to this. That is 1.6201, 0.4722 and 0.4722, 1.6201 in order to solve this problem to design the fuzzy regulator. So, now let us see the solution for this problem. The fuzzy regulator control law is represented by $u(t) = -\sum_{j=1}^{r=7} \sigma_j K_j x(t)$

where kj is the local feedback gain corresponding to rule j, and, as you know, the number of rules here is 7, and $\sigma_j = \frac{\mu_j}{\sum \mu_j}$

And the summation of sigma j for all the rules equals 1, and mu j is a membership function corresponding to the variables associated with each rule.

And it is considered that there are two state variables associated with this signaling manipulator, x_1 and x_2, theta 1 and theta 1 dot, precisely joint angle and joint angular velocity. Here, in this problem, we have considered theta 2, that is x_2 angular velocity theta dot, to be 0 throughout the case. Next, coming to the pseudo code associated with this problem.

So, step 1 here is to initialize the state variables. That is, x_1 of 0 is some value, say 0.4 radian, and x_2 of 0 is 0 velocity. And the desired value x_{1d} equals 0, and x_{2d} equals 0.01. So, it has to come and settle down from 0.4 radian to 0 radian, and hence it is a regulation problem, okay.

And here, this pendulum has been perturbed by an amount theta, which is given by 0.4 radian, which is roughly around 23, 24 degrees angle, and it has to be brought to the vertical 0 angle because here theta equals 0; this is the situation here. So, once we initialize the state variables, then we need to do this in discrete time. So, we have the sampling period assigned.

Let us say delta t equals to 0.01 second. Now, step 3 is computing the error and derivative of error e equals to x_1 of 0 minus x_{1d}, actual value minus desired value of state variable 1, and e dot is x_2 of 0 minus x_{2d}, which equals to We are finding it for the initial case. Then, step 4 computes the membership functions for the input x_1 and the output u for all the rules in this case because your x_2 is considered to be 0. So, let us focus only on the input variable being x_1 and u, the output, in order to move further for finding the membership functions.

So, for all the rules, we are going to find the membership function values for x_1 and u. So, for rule 1, x(t) equals to 0, 0, and for rule 2 It is x(t) equals to pi by 6, 0, and x for rule 3, x(t) can be taken as in my case I have taken like this. We can also take rule 2, pi by 6, 0, rule 3 can be minus pi by 6, 0, but I have taken the positive after the 0. So, rule 1 is 0, 0 point, and rule 2, pi by 6, 0, rule 3, pi by 3, 0, and rule 4, pi by 2, 0.

And rule 5, rule 6, and rule 7 are for minus values for minus pi by 6, 0, minus pi by 3, 0, and rule 7, minus pi by 2, 0. Now, let us talk about rule 1, which is given by where rule 1 corresponds to the membership function of the variable x_1 that is given by the Gaussian membership function. So, that is given by

$$e^{-\frac{(x-0)^2}{\sqrt{2\sigma}}}$$

Where x is the variable here, here it is x_1, and b is the mean value, and sigma is the width of that or the membership corresponding to the crisp value x_1. So, we have obtained for x_1 for the initial case, for rule 1, we have obtained the membership function of the variable x1 as 0.8521. Likewise, we have obtained for rule 2, rule 3, rule 4, rule 5, rule 6, and rule 7. Okay, that is the case. Okay, so we can say that minus pi by 3 and plus pi by 3.

This is it. So, this is the Gaussian function that we have taken, the Gaussian membership function. around plus pi by 3 and another one around minus pi by 3. That is how we have obtained 7 rules from the given 4 rules data. Now, step 5 is control outputs for each rule, and this part is the subpart of the rule.

As for rule 1, if x1 of t is around 0 and 0, then u1 is given by minus kx. So, deliberately, I put kx first, and in the end, I can do minus continuously or commonly for expressing the minus kx as the control value. So, I have taken u1 as the k vector multiplied by x, where x is x1 and x2.

So, for rule 1, u1 is k1, k2 associated with k1, small k1, k2. So, that is multiplied by minus 8.6628 x1 of 0 plus 0.6422 multiplied by x2 of 0. Likewise, rule 2, rule 5 will have the same k, and rule 3, rule 6 have the same k, and rule 4, rule 7 have the same k. So, we got accordingly the values of the control signal or the output signal for each rule. Next,

having obtained the consequent of each rule, now we need to have the defuzzification value that is obtained through the weighted average defuzzification formula or approach. So, step 6 is the final output of the TS Fuzzy logic block, which is the defuzzification approach to get the crisp value from the fuzzy values, precisely using the weighted average defuzzification method that is given by

$$u = (u_1R_1 + u_2(R_2 + R_5) + u_3(R_3 + R_6) + u_4(R_4 + R_7))/(\sum_{i=1}^{7} R_i) = 3.0963$$

and the this u value is given back into the discretized state model in order to obtain x of k plus 1, that is x of k plus 1 equal to x of k plus delta t into x dot of k. Okay, so in that case, we can say that

$$x_1(0 + dt) = x_1(0) + x_2(0)dt = 0.4$$

$$x_2(0 + dt) = x_2(0) + \left(-10sin\left(x_1(0)\right) + u\right)dt = -0.0080$$

So, after completing the entire simulation or the iteration has been completed in order to have the error tolerance, then we have obtained this system states time history with respect to time, where x_1 is the position and x_2 is the velocity. It starts with the initial

value of the state x_1, which is the joint angle, is 23, and it converges to 0. Likewise, the initial velocity also converges to 0, even though it starts from 0. There are some intermediate perturbations, and then it settles down. The control input associated with that is shown here.

It starts with 3.2 and settles down to 0 in the end, since it is a regulation problem where the control input becomes 0 in the end. Now, let us move into the second topic of this talk or lecture, which is fuzzy control with optimal parameters. Let us consider there are two inputs, e, which is error, and e dot, the derivative of error, and one output variable, u. These three are called linguistic variables associated with fuzzy control: error, derivative of error, and the control output. The corresponding linguistic variables

are considered as precisely the membership functions associated with the fuzzy subsets of these linguistic variables: negative, 0, positive. So, the negative fuzzy subset will have a triangular function varying from a1 to a3, and the 0 subset of the fuzzy linguistic variable. Error E varies from A2 to A4. Likewise,

the triangular membership function of the fuzzy subset P, which is positive, of this variable linguistic variable E, error E, varies from A3 to A5.

Now, these values. For each membership function, the triangular membership function must be optimal; that is the objective here. Likewise, the same thing is repeated here for another linguistic variable, which is the derivative of error E dot, which also has the fuzzy subsets negative, 0, and positive with the triangular membership functions. Having the boundaries P1, B3 for the fuzzy subset or membership function for negative, and 0 membership function has B2 to B4, and the positive membership function has B3 to B5. Likewise, the control input or the control input U has the triangular membership functions.

For fuzzy subsets N, which is negative, 0, and positive with the varying parameters, which are nothing but the variables c1, c3 for the negative case of the negative membership function of U, and for 0, c2 and c4, and for positive, it is c3 and c5. So, there are Design variables which are 5 plus 5 plus 5, 15 design variables for this optimization problem. To find the optimal values for these 15 design variables, an optimization technique can be used, precisely a global optimization technique, which is the genetic algorithm, can be utilized. For that, we need an objective function. So, given the pseudo code for this fuzzy control with optimal parameters goes with this technique, pseudo code

given, give the minimum and maximum values for all the design variables associated with each membership functions A1, A2, A3, A4, A5, similarly B1 to B5 and C1.

So that we have a constraint set on these design variables. That each design variable will have the maximum and minimum range associated with that. Now, initialize the parameters associated with the optimization technique. Start the main for loop. Inside the main for loop, we have two important sections or parts, you can say.

The first part is the fuzzy logic control algorithm. It can be either a Mamdani-based fuzzy or a TS fuzzy model, and it finds the fitness function whose value has to be optimized or minimized. For each set of the parameters. So, it states that with these constraints, with the maximum and minimum values associated with the 15 design variables, we can find the U value, which is the control input, and that control input is given to the system model, which is a state-space model, and obtain the system response, get the error, and that error is the objective function which has to be minimized at each design variable set.

So, finally, we come to see that at what design variable set, which is entirely searched within the design variable space, the optimal value that has the minimal objective function will be chosen as the optimal design variable set, and accordingly, each membership function with the ranges a1, a2, a3, a4, a5. Accordingly, b1 to b5 and c1 to c5 can be assigned for each membership function. So, there is a total of 5 plus 5 plus 5.

So, 15 membership functions with these design variables. Now, select the optimal parameters based on the minimum fitness value. Therefore, the output of this code is the optimal parameters associated with the linguistic variables. So, the output of this optimization approach will give the optimal design variables. Using them will lead to fuzzy control with optimal parameters.

Now, let us move on to fuzzy C-means clustering, which is similar to Martin's proposed approach. A researcher in the early 90s, he proposed Kohonen and self-organizing map. In the neural network scheme, which is an unsupervised learning technique. Fuzzy C-means clustering is also similar to that. First of all, let's discuss clustering.

What is clustering Clustering is an unsupervised machine learning technique that groups data points based on their proximity, closeness to one another, forming a specified number of clusters. When I say clusters, there are two types of clusters. One is hard clustering and soft clustering. Let us talk about hard clustering and soft clustering.

Hard clustering is a technique in which the data point is associated with only one cluster strictly. For example, in K-means clustering, Here we assign each data point exactly one of the k means of the k clusters by minimizing the sum of the squared distances between the data points by minimizing the Euclidean distance between them and their respective cluster centroids. Thus strictly each data point belongs to only one cluster. Whereas in soft clustering, as the name indicates, soft clustering

Each data point may belong to multiple clusters with varying degrees of membership grades. The degree of membership, or the membership grade, differs, allowing for overlap between clusters. In the case of hard clusters, each data point here is like this. Whereas, in terms of soft clustering, we have this. So, the data points here belong to multiple clusters.

For example, in Fuzzy C-means clustering, we have soft clustering, which means the data points belong to more than one cluster simultaneously. Now, the algorithm associated with this Fuzzy C-means clustering The first step in clustering is initialization. Initialize the number of clusters, c, to decide how many clusters we want to find in our assigned data. Second, choose the fuzziness parameter, m, and this parameter determines the level of fuzziness. Then, third, initialize

the membership matrix. This matrix contains the membership grades for each data point in each cluster. Obviously, the membership function value is in the interval between 0 and 1, and the sum of the membership grades is equal to 1. So, step 2 is to calculate initial centroids. Calculate the initial centroids, which means the cluster centers.

In the case of 3D, it is 3D data points; it is the centroid. In the case of 2D data points, it is the centroid using the membership matrix that is given by $$v_i = \frac{\sum_{k=1}^{n} u_{ik}^{m} \cdot x_k}{\sum_{k=1}^{n} u_{ik}^{m}}$$

where V_i is the centroid of the cluster I_i. And u_i^k is the membership grade of data point k in cluster i, and x_k is the data point. Now, step 3 is to update the membership function or membership matrix precisely. So, update the membership values of each data point and each cluster using the formula u_i^k = 1 upon summation over j = 1 to c_i.

d_i^k upon d_j^k power 2 upon m minus 1, where u_i^k is the membership grade of data point k in cluster i, and d_i^k is given by the Euclidean distance between x_k and v_i, and d_j^k is the Euclidean distance between x_k and v_j. Where v_j and v_i are the centroids. Now, step 4 goes for recalculating the centroids using the updated membership

matrix that is given by v_i = summation k = 1 to n u_i^k^m multiplied by x_k upon summation k = 1 to n u_i^k^m. Repeat this step until the centroids do not have a significant change. That means the change in the centroid is less than a very minimum tolerance value. Step 5 is to recalculate the centroids.

Iterate between updating the membership matrix and recalculating the centroids until the algorithm converges. The convergence can be determined when the changes in the membership matrix and the centroids fall below a certain tolerance value. Step 6, which is the final step, is to assign data points to a cluster. Once the algorithm converges, assign each data point to the cluster with the highest membership grade so that they are assigned to that cluster. So, the summary here is to initialize, which means to choose a number of clusters and a fuzziness parameter.

Initialize the membership matrix. There are three steps associated with initialization. The second step is to calculate centroids. Compute initial cluster centroids based on the membership matrix. The third step is to update membership.

The fourth step is to recalculate centroids. The fifth step is to iterate. Repeat steps three and four until convergence. This means a change in the centroids matrix. The membership matrix entries can be almost equal to 0.

Finally, assign clusters. Assign each data point to the cluster with the highest membership grade. Let us see a quick example of segmentation of a 2D image or 2D map. Our objective in this example is to segment a given 2D map. So

In that case, an unsegmented map containing all the detailed information, including noise and fine-grained variations in terrain or features, is used to represent the environment. With all the intricacies and variations in data given, which is nothing but an original image, let us say. And here comes the segmented map, which simplifies the original map by reducing the number of distinct regions to the number of clusters defined. It highlights the main regions or features, making it easier to identify and analyze. And this is helpful; the segmentation is helpful in applications such as path planning and obstacle avoidance in robotics.

Again, let us follow these same steps in order to have the segmentation of the given 2D map. First is initialization. Start by defining the number of clusters, initialize the number of clusters, and the fuzziness parameter. Simultaneously, initialize the membership

matrix with random values, ensuring each data point's membership values sum to 1. Step 2 is to load and pre-process the map data.

Load the map data, typically a 2D array where each cell represents a different terrain type or obstacle, and reshape it into a 2D array where each row represents a point. Given the pre-processed map being an alpha cross beta array, This new reshaped map becomes a one-dimensional array, which is nothing but an alpha beta into one array. Alpha cross beta has been reshaped into an alpha beta array. So that 2D data becomes one-dimensional data.

And then let us start the fuzzy C-means clustering approach here. That means, iteratively update the cluster centroids and membership matrix until convergence. So, compute the centroids with the expression given by V_i equal to summation k equal to 1 to n u_i k m into x_k upon summation k equal to 1 to n u_i k m where m is the fuzziness parameter and u_i k is a membership value of the kth data point in the reshaped map which is a one-dimensional array of dimension alpha beta into 1. Then update the membership matrix according to the following formula which is $u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{d_{ik}}{d_{jk}}\right)^{\frac{2}{(m-1)}}}$

Where u_i k again is the membership grade of data point k in cluster i and d_i k is given by the equivalent norm between x_k and v_i and d_j k is the equivalent norm between x_k and v_j.

Repeat the process until the algorithm converges. Step 4 is post-processing, that is after convergence, assign each data point to the cluster with the highest membership value, reshape the data back to the original map, and we get our segmented map. That means first we had a 2D becoming a one-dimensional array, then we process this fuzzy clustering algorithm, then finally convert back into the original 2D array.

So, the further analysis is with the segmented map given here on the left-hand side which is, sorry, in the given unsegmented image we can have the output of this approach for C-means clustering gives you this segmented image. So, that we can easily obtain our features which are the obstacles. So, that we can go for the path planning approach clearly. Thank you. With this, I end up this talk. Thus, we have seen an example of fuzzy TS fuzzy control of a single manipulator towards a regulation problem. Then, we have seen how we can approach the fuzzy controller with optimal parameters. Then, fuzzy C-means clustering has been studied in this lecture. Thank you.