Data-Driven Design of a Tree Fuzzy Inference System

A Case Study in Mortgage Approval

Prof. D.Sc. BARSEKH-ONJI Aboud

Faculty of Engineering
Universidad Anáhuac México Sur

November 3, 2025

Agenda

Agenda

1. Project Overview
2. System Design Data Generation
3. Automated Learning and Tuning
4. Results and Conclusion

The Challenge: Complex, Human-like Decisions

The Problem

How can we model a complex decision-making process, like mortgage approval, that relies on imprecise information and expert knowledge?

Inputs are often "fuzzy":

- What is a "good" credit score?
- What constitutes a "high" income?
- When is a down payment "adequate"?

The Problem with Single FIS

A single FIS with many inputs becomes unmanageable. The number of rules can grow exponentially! ($3^4 = 81$ rules for our case).

The Challenge: Complex, Human-like Decisions

The Problem

How can we model a complex decision-making process, like mortgage approval, that relies on imprecise information and expert knowledge?

Solution: Fuzzy Inference Systems (FIS)

- Use linguistic variables and IF-THEN rules.
- Manage uncertainty and complexity.

The FIS Tree Approach

Break the problem down into smaller, interconnected modules. This is a modular, hierarchical approach that is easier to design and understand.

System Architecture: A Two-Level FIS Tree

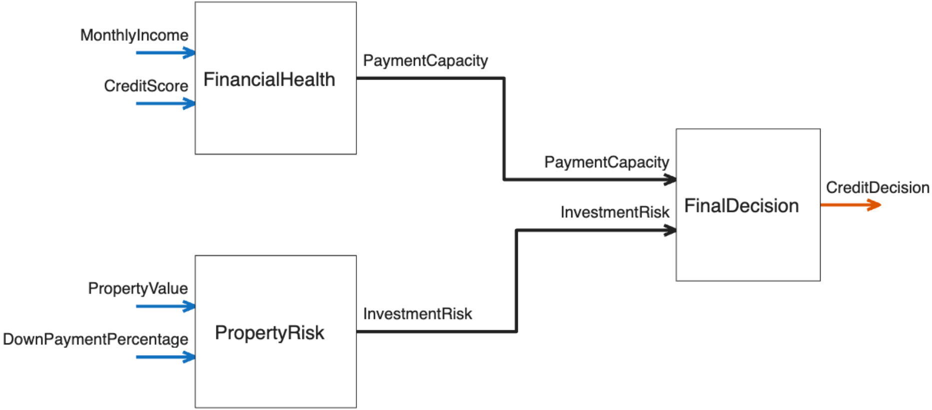

Figure 1: FIS Tree for Mortgage Approval

Agenda

1. Project Overview
2. System Design Data Generation
3. Automated Learning and Tuning
4. Results and Conclusion

Building the Initial FIS Tree

Goal

Programmatically create the initial FIS tree structure in MATLAB based on expert knowledge. This serves as our baseline model.

The project includes three main scripts:

1. **generateData.m**: Creates a synthetic dataset for training and validation.
2. **buildFisTree.m**: Constructs the initial tree structure shown previously.
3. **learnFis.m**: Uses the data to automatically learn rules.
4. **tueFis.m**: Uses the data to automatically tune parameters of the learned system.

GitHub Repository

https://github.com/AboudOnji/FUZZY-LOGIC-CREDIT-APPROVAL-SYSTEM

Data Generation: ‘generateData.m‘

Why Generate Data?

Machine learning requires data. We create a synthetic dataset that mimics real-world scenarios to train and validate our system.

- **Method:** We define 5 ”applicant archetypes” (e.g., Ideal Applicant, High-Risk, Young Professional) to create realistic and diverse data points.
- **Output:** A matrix named `trainingData` with 200 samples and 5 columns:

Col 1	**Col 2**	**Col 3**	**Col 4**	**Col 5**
MonthlyIncome	CreditScore	PropertyValue	DownPayment%	ExpectedDecision

Key Idea

The first four columns are the system’s inputs. The fifth column is the ”correct answer” or target output we want the system to learn.

Building the Tree: `buildFisTree.m`

Process

We create the three FIS components (`fis1_Health`, `fis2_Risk`, `fis3_FinalDecision`) and connect them using the modern `fistree` function.

This approach ensures a clear, modular, and verifiable structure.

Agenda

1. Project Overview
2. System Design Data Generation
3. Automated Learning and Tuning
4. Results and Conclusion

Why Automate? From Expert Knowledge to Data-Driven Design

Limitations of Manual Design

- Rules can be subjective.
- Tedious to define and tune.
- May not be optimal.

The Data-Driven Solution

- Use algorithms to **learn** the best rules from data.
- Automatically **tune** membership function parameters to minimize error.

Our Tools

We use the `tunefis` function in MATLAB with Particle Swarm Optimization for both learning and tuning.

Stage 1: Learning the Rules (‘learnFisTree.m‘)

Goal

Automatically generate a small, effective set of fuzzy rules from the data, replacing our initial manual rules.

Stage 2: Tuning the Parameters (‘tuneFisTree.m‘)

Goal

Take the system with the newly learned rules and fine-tune all of its parameters (membership functions) to maximize accuracy.

Agenda

Evaluating Performance: Root Mean Squared Error (RMSE)

How do we measure success?

We calculate the Root Mean Squared Error (RMSE) between the system's predictions and the expected outputs from our dataset. **A lower RMSE is better.**

System Stage	**RMSE**
Initial Manual FIS	~ 15.8
After Rule Learning	~ 9.2
After Full Tuning	~ 4.5

Result

The fully automated, data-driven approach significantly improved the model's accuracy compared to the initial manual design.

Visualizing the Tuning Process

What does "tuning" look like?

The optimization algorithm physically shifts and reshapes the membership functions to better fit the patterns in the data.

Conclusion and Key Takeaways

What We Accomplished

- We modeled a complex problem using a modular FIS Tree.
- We created a synthetic dataset to train and validate our model.
- We used modern MATLAB tools (`fistree`, `tunefis`) to automatically learn rules and tune parameters from data.
- The data-driven approach resulted in a significantly more accurate system.

Conclusion and Key Takeaways

Broader Applications

This workflow of combining expert knowledge (initial structure) with data-driven optimization can be applied to many other fields, including:

- Medical diagnosis
- Financial risk analysis
- Process control and automation
- Customer churn prediction