fuzzylite
A Fuzzy Logic Control Library in C++

Juan Rada-Vilela

www.fuzzylite.com
Introduction
Introduction

Are you familiar with...?
Introduction

Are you familiar with...?
Introduction

Are you familiar with...?

(Lord of the Rings)
You should know that...

- Many of them use **Fuzzy Logic Control**
- Over **50 000** patents involve **Fuzzy Logic Control** *
- Over **$10 000M** in product sales using **Fuzzy Logic Control**
- ... a **single** product: a blood pressure monitor *
- Profits estimated in **billions ($$$)** using **Fuzzy Logic Control** †

Can you believe that...?

State-of-the-art **FLC libraries** have **strong** limitations

---

* [http://goo.gl/oYWGkM](http://goo.gl/oYWGkM)
† [http://goo.gl/VDgOk9](http://goo.gl/VDgOk9)
State of the Art

Limitations

- Matlab and Fuzzy Logic Toolbox: **Costly** license
- Octave and Fuzzy Logic Toolkit: **Restrictive** license
- jFuzzyLogic: **Unfortunate** design choices
- Others: do **not** even bother…

**fuzzylite**: A Fuzzy Logic Control Library in C++

- **Free and open source**
- **Commercial friendly** license
- Mostly **fortunate** design choices
- **More** features
- Very **easy** to use
- Linux, Mac OS X, Windows, and others…
Objectives

Overall Goal

Introduction to Fuzzy Logic Controllers

Specific Objectives

- **Design** of Fuzzy Logic Controllers
- **Operation** of Fuzzy Logic Controllers
- **Examples** of Fuzzy Logic Controllers
- **Description** of fuzzylite
Design

Juan Rada-Vilela

fuzzylite: A Fuzzy Logic Control Library in C++
Fuzzy Logic Controller

Definition

Controller

if condition then action

Inputs

Outputs

Juan Rada-Vilela

fuzzylite: A Fuzzy Logic Control Library in C++
Example

Problem

How much to **tip** at a restaurant?

Solution

**Controller**

- if **Service** is **poor** then **Tip** is **cheap**
- if **Service** is **good** then **Tip** is **average**
- if **Service** is **great** then **Tip** is **generous**
Linguistic Variables

Quality of Service

\[ S(2.5) = \text{poor} \]

Tip

\[ T(17.5) = \text{average} + \text{generous} \]

Linguistic variables with **crisp sets** as certainty \( \in \{0, 1\} \)
Linguistic Variables

Service

\[ \tilde{S}(2.5) = 1.0/\text{poor} \]

Tip

\[ \tilde{T}(17.5) = 0.2/\text{average} + 0.8/\text{generous} \]

Linguistic variables with **fuzzy sets** as certainty \( \mu \in [0.0, 1.0] \)
Design of a Fuzzy Logic Controller

**Input: Service**

![Input Graph]

**Output: Tip**

![Output Graph]

**Rules**

- If Service is poor then Tip is cheap
- If Service is good then Tip is average
- If Service is great then Tip is generous

Juan Rada-Vilela

fuzzylite: A Fuzzy Logic Control Library in C++
Operation of a Fuzzy Logic Controller

![Graph showing tip as a function of service quality]

Juan Rada-Vilela

fuzzylite: A Fuzzy Logic Control Library in C++
Operation
Stages of a Fuzzy Logic Controller

- Inputs
- Fuzzification
- Inference
- Defuzzification
- Outputs
Stages of a Fuzzy Logic Controller

- Inputs
- Fuzzification
- Inference
- Defuzzification
- Outputs
Fuzzification

Definition

**Fuzzification:** Converts **crisp** input values into a **fuzzy** set

Example

<table>
<thead>
<tr>
<th>Service</th>
<th>Fuzzification</th>
</tr>
</thead>
</table>
| ![Service Graph](image) | \[\tilde{S}(x) = \sum_{i \in S} \mu_i(x) / i\]  
\(\mu_i : \text{membership function of term } i\)  
\[\tilde{S}(1.0) = 0.4/\text{poor} + 0.0/\text{good} + 0.0/\text{great}\]  
\[\tilde{S}(2.5) = 1.0/\text{poor} + 0.0/\text{good} + 0.0/\text{great}\]  
\[\tilde{S}(7.0) = 0.0/\text{poor} + 0.2/\text{good} + 0.8/\text{great}\] |
Stages of a Fuzzy Logic Controller

- Inputs
- Fuzzification
- Inference
- Defuzzification
- Outputs
Inference

**Definition**

Inference: activates the rules to generate **fuzzy outputs**

- **Fuzzy Inputs** → **Activation of Antecedents**
- e.g. if **Service** is **great**
- **Modification of Consequents**
- e.g. then **Tip** is **generous**
- **Accumulation of Outputs** → **Fuzzy Outputs**
Inference

Activation of the Antecedents

Example

\[
\tilde{S}(2.5) = 1.0/\text{poor} + 0.0/\text{good} + 0.0/\text{great}
\]

Activation

\[
\begin{align*}
\text{if (Service is poor)} &= 1.0 \\
\text{if (Service is good)} &= 0.0 \\
\text{if (Service is great)} &= 0.0
\end{align*}
\]
Inference
Activation of the Antecedents

Example

if Service is poor then Tip is cheap
if Service is good then Tip is average
if Service is great then Tip is generous

\( \tilde{S}(7.0) = 0.0/\text{poor} + 0.2/\text{good} + 0.8/\text{great} \)

Activation

if (Service is poor) = 0.0
if (Service is good) = 0.2
if (Service is great) = 0.8
Inference
Modification of Consequents

Example

\[
\text{if Service is } \text{poor} \quad \text{then Tip is cheap}
\]
\[
\text{if Service is } \text{good} \quad \text{then Tip is average}
\]
\[
\text{if Service is } \text{great} \quad \text{then Tip is generous}
\]

\[
\tilde{\mathcal{S}}(2.5) = 1.0/\text{poor} + 0.0/\text{good} + 0.0/\text{great}
\]

Modification

\[
\text{then Tip is } (1.0 \otimes \text{cheap})
\]
\[
\text{then Tip is } (0.0 \otimes \text{average})
\]
\[
\text{then Tip is } (0.0 \otimes \text{generous})
\]

\(\otimes\) : Activation Operator
Inference
Modification of Consequents

Example

\[ \{ (1.0 \otimes \text{cheap}), (0.0 \otimes \text{average}), (0.0 \otimes \text{generous}) \} \]
Inference
Modification of Consequents

Example

\{ (0.0 \otimes \text{cheap}), (0.2 \otimes \text{average}), (0.8 \otimes \text{generous}) \}

\begin{align*}
\otimes : & \min(\mu_i, \mu_j(x)) \\
\otimes : & \prod(\mu_i, \mu_j(x))
\end{align*}
Inference
Accumulation of Consequents

Example

\[ \tilde{S}(7.0) = 0.0/\text{poor} + 0.2/\text{good} + 0.8/\text{great} \]

Activation

- if (Service is poor) = 0.0
- if (Service is good) = 0.2
- if (Service is great) = 0.8

Modification

- then Tip is (0.0 \times \text{cheap})
- then Tip is (0.2 \times \text{average})
- then Tip is (0.8 \times \text{generous})

Accumulation

\[ \tilde{T}_{7.0} = (0.0 \times \text{cheap}) \oplus (0.2 \times \text{average}) \oplus (0.8 \times \text{generous}) \]

\[ \oplus : \text{Accumulation Operator} \]
Inference
Modification of Consequents

Example

\{ (0.0 \otimes \text{cheap}), (0.2 \otimes \text{average}), (0.8 \otimes \text{generous}) \}

\[ \mu(y) \]

\[ \otimes : \min(\mu_i, \mu_j(x)) \]

\[ \otimes : \prod(\mu_i, \mu_j(x)) \]
Inference
Accumulation of Consequents

Example

\[ \tilde{T}_{7.0} = (0.0 \otimes \text{cheap}) \oplus (0.2 \otimes \text{average}) \oplus (0.8 \otimes \text{generous}) \]

\[ \otimes : \min(\mu_i, \mu_j(x)) \]
\[ \oplus : \max(\mu_i, \mu_j(x)) \]

Juan Rada-Vilela
Stages of a Fuzzy Logic Controller

- Inputs
- Fuzzification
- Inference
- Defuzzification
- Outputs

Juan Rada-Vilela

fuzzylite: A Fuzzy Logic Control Library in C++
Defuzzification

**Definition**

Defuzzification: converts the fuzzy outputs into crisp values

**Example**

**Centroid**

![Graph showing defuzzification using the centroid method]

**Maxima**

![Graph showing defuzzification using the maxima method]
Stages of a Fuzzy Logic Controller

- Inputs
- Fuzzification
- Inference
- Defuzzification
- Outputs
fuzzylite
**Features**

fuzzylite is a library (.so, .dylib, .dll)

qtfuzzylite links to fuzzylite

**Main Features**

- **Mamdani**, Takagi-Sugeno and Tsukamoto FLCs
- 17+ linguistic terms
- 13 fuzzy logic operators
- Seven defuzzifiers
- Six types of hedges (e.g. *very*, *somewhat*, *not*)
- Import and export using FCL, FIS, C++
- Extend and incorporate new components
## Linguistic Terms

<table>
<thead>
<tr>
<th>Basic</th>
<th>Extended</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangle</td>
<td>Gaussian</td>
<td>Ramp</td>
</tr>
<tr>
<td>Trapezoid</td>
<td>Bell</td>
<td>Sigmoid</td>
</tr>
<tr>
<td>Rectangle</td>
<td>Sigmoid D.</td>
<td>S-Shape</td>
</tr>
<tr>
<td>Discrete</td>
<td>Constant, Linear, Custom</td>
<td>Z-Shape</td>
</tr>
</tbody>
</table>

\[
f(x) = c \\
f(x, y) = ax + by + c
\]
## Abstract Model

### InputVariable
- **input**: scalar
- **terms**: vector<Term*>

### Engine
- **inputs**: vector<InputVariable*>
- **outputs**: vector<OutputVariable*>
- **ruleBlocks**: vector<RuleBlock*>
- **hedges**: vector<Hedge*>
- **process()**: void

### RuleBlock
- **rules**: vector<Rule*>
- **tnorm**: TNorm*
- **snorm**: SNorm*
- **activation**: TNorm*
- **fireRules()**: void

### OutputVariable
- **terms**: vector<Term*>
- **output**: Accumulated*
- **defuzzifier**: Defuzzifier*
- **defuzzify()**: scalar
Conclusions and Future Work
Conclusions

- FLCs are a **powerful** alternative to traditional control algorithms
  - **Easy** to design
  - **Easy** to operate
  - **Easy** to maintain over time
  - **Many** algorithms to tune FLCs
Conclusions

- Important to **recognize** when to utilize FLCs

![Graph showing Tip vs. Service and Ramp term]
Future Work

- **Type-2** Fuzzy Logic Controllers
- Adaptive **Neuro-Fuzzy** Inference System (ANFIS)
- Fuzzy C-Means **clustering** algorithm
- … and there are still **many** more things to do!
Donations

support **fuzzylite** with a donation
fuzzylite
A Fuzzy Logic Control Library in C++

Juan Rada-Vilela

www.fuzzylite.com