Fuzzy Systems and Evolutionary Computing

Introduction

Andrea Campagner, PhD, Post-Doc Researcher

IRCCS Istituto Ortopedico Galeazzi, Milan, Italy
MUDI Lab, Department of Computer Science, University of Milano-Bicocca, Milan, Italy

Contacts: onyris93@gmail.com; andrea.campagner@unimib.it
https://andreacampagner.github.io/

13 March 2023

Table of Contents

Requirements

Requirements

- Python 3
- NumPy, SciPy, Matplotlib
- Jupyter Notebooks

Take the time to familiarize with the libraries (also, ask me questions!): you don't need to implement everything during the class!

Installation tip

Suggested installation method:
Anaconda (`https://www.anaconda.com/`)

Code and Examples

Important!

All code and examples we will see in the class are available on GitHub:

`https://github.com/AndreaCampagner/Fuzzy-Systems-and-Evolutionary-Computing`

Use it as a reference (code on GitHub will be a bit more complex, but more "production-ready")

Introduction to the Course

Objective

Giving you a broad introduction to two forms of *soft computing*: **Fuzzy Systems** and **Evolutionary Computing**

Objective of the Lab

Hands-on teaching: Learning by programming $\rightarrow$ **Implement a Python (mini-)library** for fuzzy and evolutionary computing

Soft Computing

- What is soft computing?
- Branch of computer science devoted to the study and implementation of algorithms and methods that tolerate **uncertainty**: *imprecision*, *approximation*, *partial truth*, *vagueness*

Soft Computing

- What is soft computing? **Approximate Reasoning** + **Function Approximation/Random Search**
- **Approximate Reasoning**: relaxes classical logical to allow for uncertainty (in both specification and computation)... Examples: **Fuzzy systems**, *probability theory*, *imprecise probabilities*, ...
- **Function Approximation/Random Search**: automatize optimization and search through approximate but general-purpose heuristics... Examples: **Evolutionary computing**, *neural networks*

Soft Computing: Fuzzy Systems

In the first part of the course, we will focus on fuzzy systems:

- *fuzzy set theory*
- *fuzzy logic*
- *possibility theory*
- *fuzzy clustering*
- *fuzzy control*

Fuzzy Systems

- Fuzzy systems: computational systems inspired by the fact that human reasoning is often imprecise and vague
- Many forms (and applications) of fuzzy systems:
  - **Fuzzy control**: control of *physical systems* (e.g., braking systems, air conditioners, robotics, ...)
  - **Fuzzy clustering**: discover *imprecise* relationships in data
  - **Fuzzy ML**: represent and manage uncertainty in Machine Learning problems (e.g., weakly supervised learning)
- Rely on *Fuzzy set theory* and *Fuzzy logic*

Fuzzy Set Theory

- We start from standard set theory: a universe of discourse U, given a set S each object x either $x \in S$ or $x \notin S$
- Is it the same in *natural language*?
- No: Think of the **set of tall persons** T: some people are definitely tall ($x \in T$), some are definitely not ($x \notin T$)...
- Others are *tall to a degree*...

Fuzzy Set Theory (cont.)

- Fuzzy set theory allows for modeling this *to a degree* by relaxing classical set theory
- A **fuzzy set** is a function $F : U \to [0, 1]$
- $F(x)$ represents the *degree of membership* of x to the set F: The higher $F(x)$ the more x *belongs to* F...
- Many interpretations[1]: degrees of truth, degrees of typicality, ... We will adopt an *agnostic* approach!

[1] `https://plato.stanford.edu/entries/logic-fuzzy/`

Fuzzy Logic

- **Fuzzy logic** extends classical Boolean logic to *propositions* (Propositional logic) and *predicates* (First-order logic) that are interpreted in terms of *fuzzy sets*
- Formulas like in classical logic but propositions/predicates have values in $[0, 1]$ rather than $\{0, 1\}$...
- Connectives ($\wedge$, $\vee$, $\neg$) have to be re-defined appropriately: t-norms (e.g. max), t-conorms (e.g. min), negations (e.g. $\neg p = 1 - p$)

Implementing Fuzzy Sets

- From the computational point of view: **what is** a fuzzy set?
- It is a data structure: **representation of data** + **operations**
- Representation of data should enable efficient (fast, not too much space) implementation of the operations

Implementing Fuzzy Sets (cont.)

We distinguish between:

- Discrete fuzzy sets (finite number of objects)
- Continuous fuzzy sets (uncountably infinite number of object) $\rightarrow$ In particular: fuzzy number (each α-cut is closed and convex)

Implementing Fuzzy Sets (cont.)

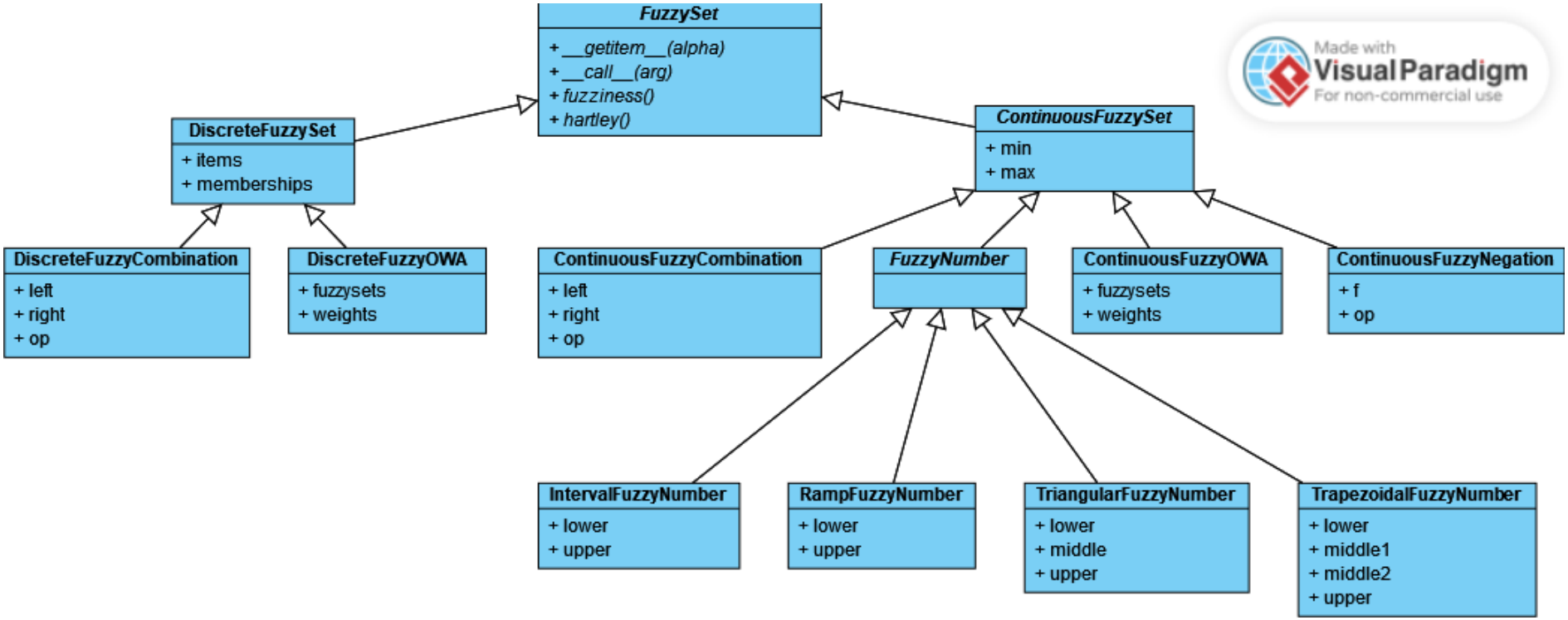
Which operations should a fuzzy set support?

Implementing Fuzzy Sets (cont.)

Which operations should a fuzzy set F support?

- Taking the membership value of an object x, $F(x)$
- Taking an α-cut $F^\alpha = \{x \in U : F(x) \geq \alpha\}$
- Logical operations: t-norms $F \wedge G$, t-conorms $F \vee G$, negations $\neg F$
  $\rightarrow$ Note: each such operation gives a new fuzzy set!
- Possibly also: entropy (fuzziness, non-specificity)

Hierarchy of Classes



Note: we override *__getitem__*() (i.e. []) to implement α-cuts and *__call__*() (i.e. ()) to implement taking membership values.

Discrete Fuzzy Sets

Ideally:

- Taking membership values: $O(1)$ (but maybe start with $O(n)$?)
- Taking α-cuts: $O(n)$

Suggestions:

- Use numpy arrays
- Note: the two objectives above conflict! How can you deal with it?
  $\rightarrow$ Hint: you need to use more memory...

Discrete Fuzzy Sets (cont.)

- Fuzziness: recall, defined as

$$\sum_x F(x) \log_2(\frac{1}{F(x)}) + (1 - F(x)) \log_2(\frac{1}{1 - F(x)})$$

- Hartley entropy: is a measure of non-specificity (useful when you interpret membership degrees as sort of degrees of belief), defined as

$$\sum_{\alpha^{(i)}=\alpha^{(1)}}^{\alpha^{(n)}} (\alpha^{(i)} - \alpha^{(i+1)}) \log_2(|F^{\alpha^{(i)}}|)$$

where $\alpha^{(1)}, \ldots, \alpha^{(n)}, \alpha^{(n+1)} = 0$ are the membership degrees in decreasing order

Continuous Fuzzy Sets

They are too generic to implement most operations (including getting memberships)... however we can implement:

- Computing the fuzziness
- Computing (approximate) α-cuts

Note: all (bounded below) continuous fuzzy sets have a minimum element x^m s.t. $F(x^m) > 0$ and a maximum element x^M s.t. $F(x^M) > 0$

Suggestions:

- Fuzziness is defined as

$$\int F(x)\log_2(\frac{1}{F(x)}) + (1-F(x))\log_2(\frac{1}{1-F(x)})dx$$

- You can't rely on formulas to get α-cuts: search for elements that belong to F^α in $[x^m, x^M]$

You don't need to implement everything from zero:

- Integrals can be evaluated using scipy.integrate.quad
- You can generate a regular grid of values using numpy.linspace

Fuzzy Numbers

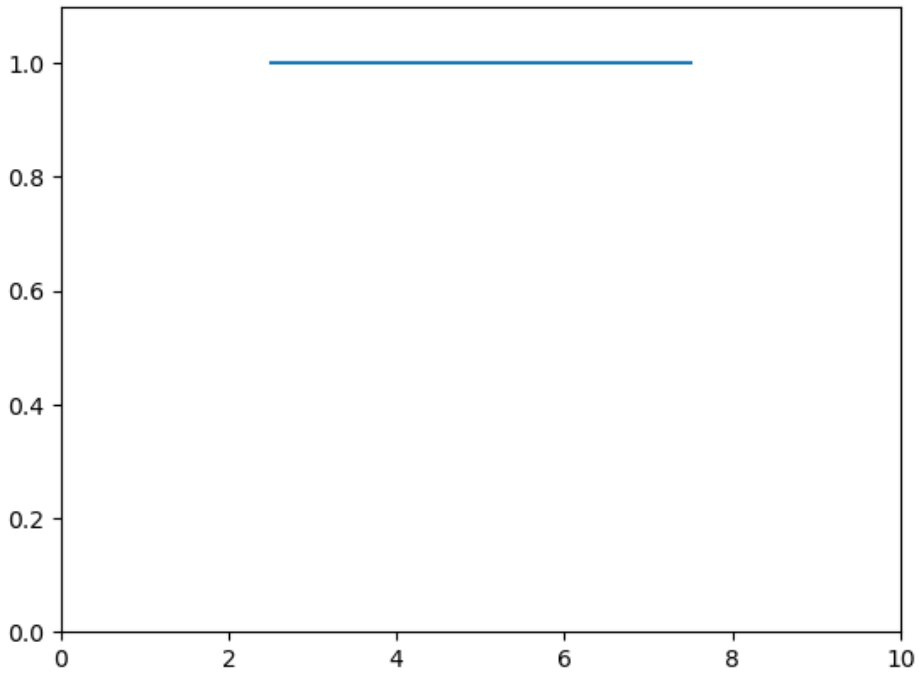
A special class of continuous fuzzy numbers with good properties

- Each α-cut is a bounded interval
- Typically defined (piecewise) by a closed formula

We consider some examples...

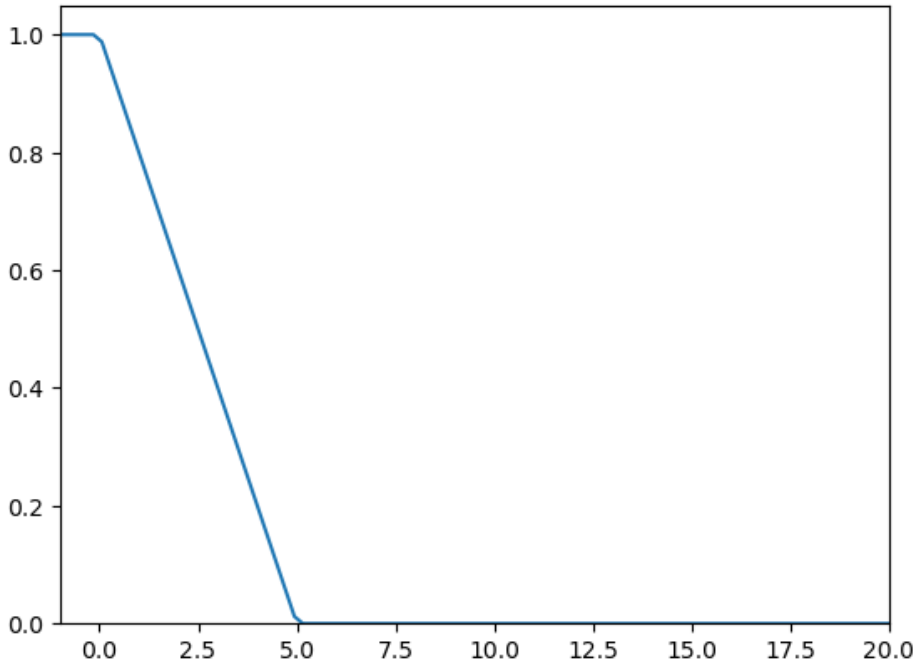
Fuzzy Numbers (cont.)

Intervals: $F_{a,b}(x) = \begin{cases} 1 & x \in [a, b] \\ 0 & otherwise \end{cases}$, with $a \leq b$



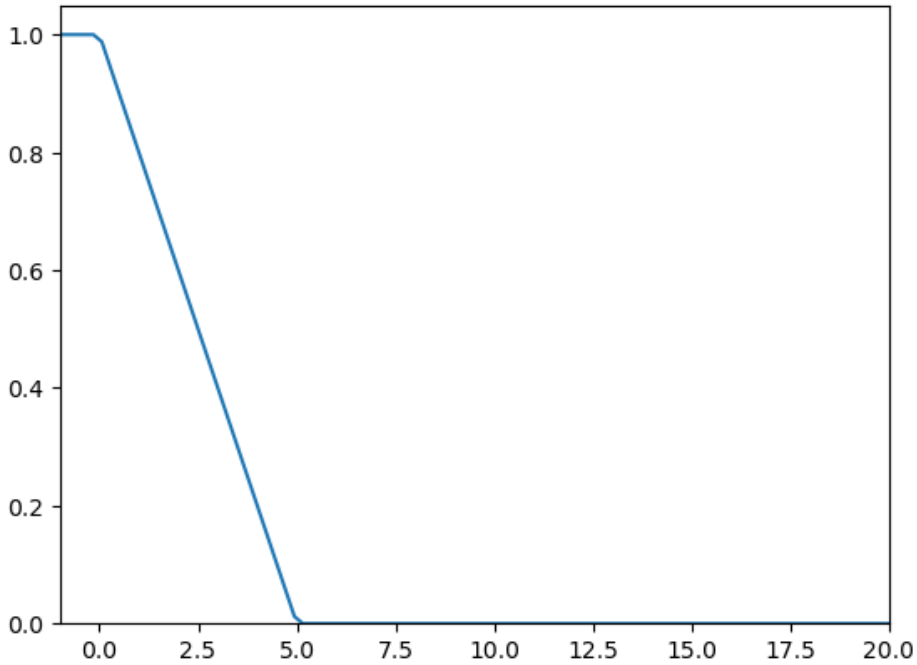
Fuzzy Numbers (cont.)

$$\text{Ramp: } F_{a,b}(x) = \begin{cases} 1 & x > b \\ \frac{x-a}{b-a} & x \in [a,b] \\ 0 & x < a \end{cases} \rightarrow \text{Bonus: Consider } b < a$$



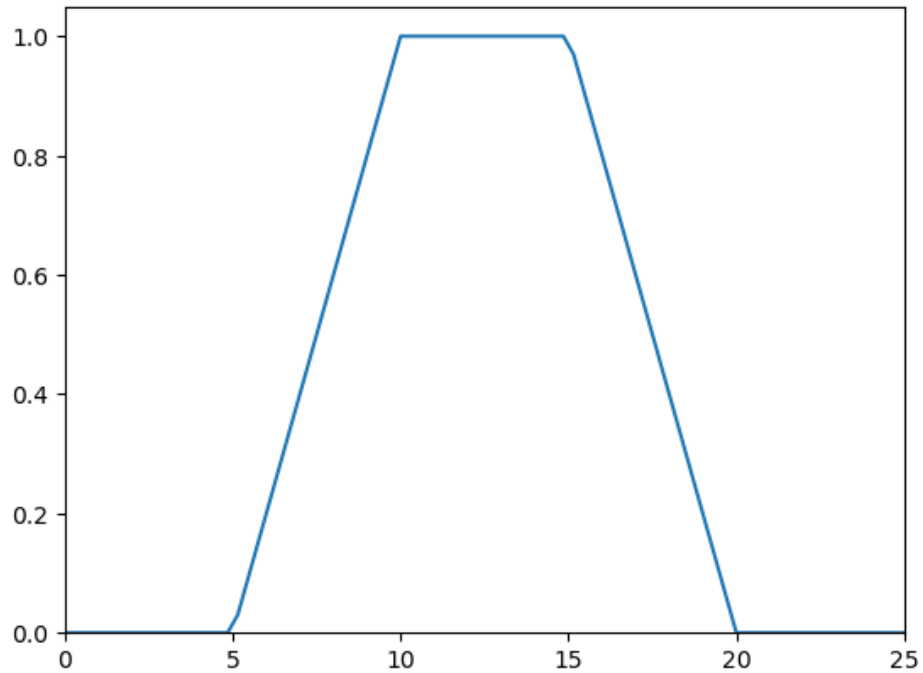
Fuzzy Numbers (cont.)

Triangular: $F_{a,m,b} = \begin{cases} 0 & x \notin [a, b] \\ \frac{x-a}{m-a} & x \in [a, m) \\ \frac{b-x}{b-m} & x \in [m, b] \end{cases}$



Fuzzy Numbers (cont.)

Trapezoidal: $F_{a,m_1,m_2,b} = \begin{cases} 0 & x \notin [a, b] \\ \frac{x-a}{m_1-a} & x \in [a, m_1) \\ 1 & x \in [m_1, m_2] \\ \frac{b-x}{b-m_2} & x \in (m_2, b] \end{cases}$



Fuzzy Numbers (cont.)

For each of the above ones (not necessarily not all now!), you should implement:

- Taking membership values (i.e., *__call__*())
- Taking α-cuts (i.e., *__getitem__*()): since α-cuts are intervals, you can simply return the minimum and maximum
- Hartley entropy, here defined as :

$$\int \log_2(|F^\alpha|) d\alpha$$

Remember: each α-cut is an interval!

Combination Rules

We consider different combination rules:

- T-norms (e.g., minimum, product, ...)

$$T(x) = F(x) \cap G(x) \tag{1}$$

- T-conorms (e.g., maximum, probabilistic sum, ...)

$$T(x) = F(x) \cup G(x) \tag{2}$$

- Negation (e.g., $1 - \cdot$)

$$N(x) = F^c(x) \tag{3}$$

- OWA (ordered weighted averages)

$$OWA(x; F_1, \ldots F_n; w_1, \ldots w_n) = \sum_{i=1}^{n} w_i F^{(i)}(x), \tag{4}$$

where $F^{(1)}(x) \leq \ldots \leq F^{(n)}(x)$

Combination Rules (cont.)

We will use some classes to abstract their functionality... operators are then defined by functions (one function for each operator) that simply construct objects of the appropriate type!

Combination Rules (cont.)

We start with unary operators (e.g. negations)

- ContinuousFuzzyNegation: for binary operators, when the argument is a ContinuousFuzzySet
- Do you really need a supporting class for the negation of a DiscreteFuzzySet ?

Accept two arguments:

- Fuzzy set to be negated
- Operation

Notice: a ContinuousFuzzyNegation is a ContinuousFuzzySet (think of it!)

Combination Rules (cont.)

We start with binary operators (e.g. t-norms, t-conorms)

- DiscreteFuzzyCombination: for binary operators, when both arguments are DiscreteFuzzySet
- ContinuousFuzzyCombination: as above, when both are ContinuousFuzzySet

They will accept three arguments:

- Left operand
- Right operand
- Operation

Notice: a DiscreteFuzzyCombination is a DiscreteFuzzySet and a ContinuousFuzzyCombination is a ContinuousFuzzySet (think of it!)

Combination Rules (cont.)

And we get to OWAs

- DiscreteFuzzyOWA: when both arguments are DiscreteFuzzySet
- ContinuousFuzzyOWA: when both are ContinuousFuzzySet

They will accept two arguments:

- An array of FuzzySet
- An array of weights (number)

Notice: a DiscreteFuzzyOWA is a DiscreteFuzzySet and a ContinuousFuzzyOWA is a ContinuousFuzzySet (think of it!)

Bonus Exercise (Fuzzy Numbers)

Bonus Exercise: implement the GaussianFuzzyNumber

$$F_{\mu,\sigma}(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5}$$

For α-cuts you should solve for x!

Bonus Exercise (Combination Rules)

Bonus Exercise: re-implement the combination of DiscreteFuzzySet (i.e., DiscreteFuzzyCombination) without using DiscreteFuzzyCombination!

Bonus Exercise (Combination Rules)

Bonus Exercise: implement the weighted average

$$WA(x; F_1, ...F_n; w_1, ...w_n) = \sum_{i=1}^{n} w_i F_i(x) \tag{6}$$

- Constraint: You are only allowed to use DiscreteFuzzyCombination and ContinuousFuzzyCombination
- Hint: Is the weighted average associative?