

# Fuzzy Logic and Neural Networks in CS452

Who? Louis Burke

When? July 28, 2017



# The AI Problem

- Approximate the real world in software.
- Analogously approximate a function.
- 5 common solutions:
  - 1 Heuristics
  - 2 Genetics
  - 3 Fuzzy Logic
  - 4 Neural Network
- Worthy mentions:
  - 1 Distributed (we only have 1 system)
  - 2 Cooperative (still just 1 system)
  - 3 Adaptive (I haven't learnt it, but probably very effective)



# Heuristics and Genetics

Heuristics:

- Requires accurate modelling
- Difficult to extend
- Cannot learn
- Easiest to inspect and debug

Genetics:

- Learns too slow!



# Genetics

- Learns too slow!



# Fuzzy vs Neural

	Fuzzy Logic	Neural Networks
Best suited for	Control	Classification
Speed	Fast	Slow
Programming	Manual	Automatic
Learning	No	Yes
Requirements	Rules	Data



# Fuzzy Logic Basics

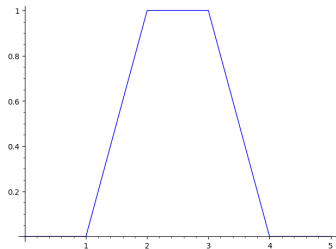
- First turn inputs into fuzzy variables
- Then do logic with those variables
- Then turn fuzzy variables back to outputs



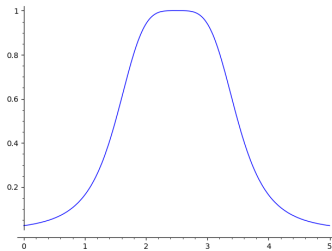
# Fuzzification

- Fuzzy sets have non-integer inclusion
- In crisp sets:  $x \in S$  is boolean
- In fuzzy sets:  $x \in S$  is an element from another set (usually reals)
- Typical membership functions:

Trapezoidal



Bell



# Fuzzifier Comparison

Trapezoidal	Bell
Faster	Slower
Not Differentiable	Differentiable
Less accurate	More accurate

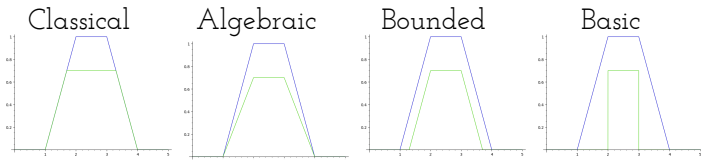
Importance of differentiability is it allows the use of gradient descent algorithms on the membership functions.





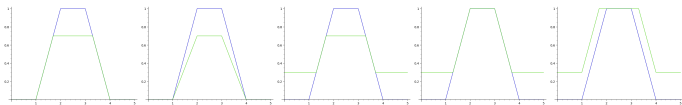
# Fuzzy T-norm

- Fuzzy equivalent to AND operation
- Comes with the equivalent S-norm (OR)
- S-norm computed via DeMorgan
- Typical T-norms:



# Fuzzy Implications

Like T-norm, not obvious. Common implication schemes:



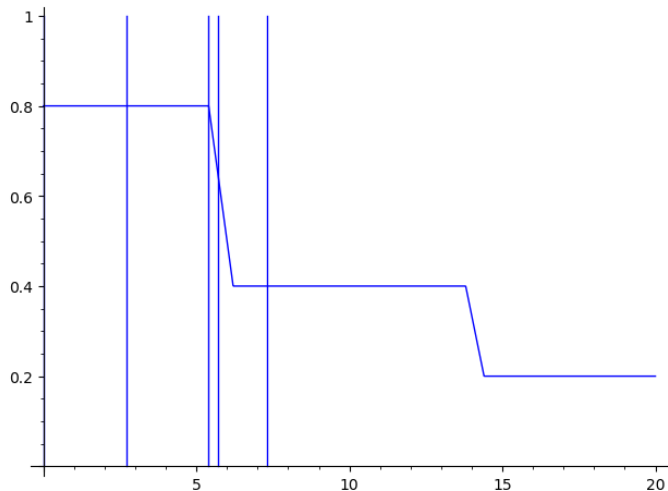
From left to right: Mamdani, Larsen, Zadeh,  
Dienes-Rascher, Lukasiewicz.



# Defuzzification

Last fuzzy operation, turn fuzzy back to non-fuzzy.

Common defuzzification schemes:



From left to right: Min of max (at 0), Mean of max, Max of max, Center of area, Centroid.



# Recommendations

- Use a neural network to learn how long it takes a train to get from switch to switch.
- If that is too slow, simply save last switch-to-switch time?
- Take time since last switch, expected switch-to-switch time, speed, etc to fuzzily determine location.
- Definitely look into adaptive algorithms (especially Least-Mean-Squares) to possibly swap out for the neural network.

