Нечёткий классификатор

# Постановка задачи

Очень часто люди попадают в неприятные ситуации, когда ошибаются с выбором одежды по погоде. Ориентируясь по одному только термометру они одевают тёплую одежду при низкой температуре и лёгкую при высокой. Однако, часто бывает, что людей знобит при довольно высокой температуре, или им наоборот становится жарко в холодную погоду. Это происходит из-за того, что на восприятие человеком температуры влияет ещё и влажность воздуха.

На основании относительной влажности воздуха и температуры определить, что будет чувствовать человек, находящийся в помещении: холод, жару или тепло.

# Теоретический раздел

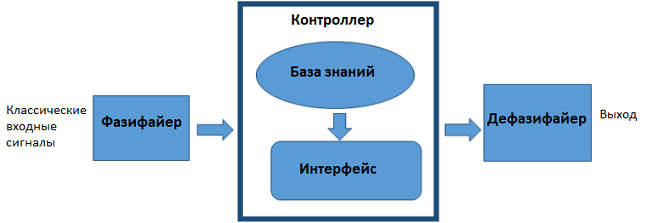
Нечеткая логика – это логическая или управляющая система n-значной логической системы, которая использует степени состояния («степени правды») входов и формирует выходы, зависящие от состояний входов и скорости изменения этих состояний. Это не обычная «истинная или ложная» (1 или 0), булева (двоичная) логика, на которой основаны современные компьютеры. Она в основном обеспечивает основы для приблизительного рассуждения с использованием неточных решений и позволяет использовать лингвистические переменные.

Нечеткая логика была разработана в 1965 году профессором Лотфи Заде в Калифорнийском университете в Беркли. Первым приложением было выполнение обработки компьютерных данных на основе естественных значений.

Если говорить проще, состояниями нечеткой логики могут быть не только 1 или 0, но и значения между ними, то есть 0.15, 0.8 и т.д. Например, в двоичной логике, мы можем сказать, что мы имеем стакан горячей воды (то есть 1 или высокий логический уровень) или стакан холодной воды, то есть (0 или низкий логический уровень), но в нечеткой логике, мы можем сказать, что мы имеем стакан теплой воды (ни горячий, ни холодный, то есть где-то между этими двумя крайними состояниями). Четкая логика: да или нет (1, 0). Нечеткая логика: конечно, да; возможно, нет; не могу сказать; возможно да и т.д.

## Базовая архитектура нечеткой логической системы

Система нечеткой логики состоит из следующих модулей:



Фазифаер (Fuzzifier или оператор размытия). Он принимает измеренные переменные в качестве входных данных и преобразует числовые значения в лингвистические переменные. Он преобразует физические значения, а также сигналы ошибок в нормализованное нечеткое подмножество, которое состоит из интервала для диапазона входных значений и функций принадлежности, которые описывают вероятность состояния входных переменных. Входной сигнал в основном разделен на пять состояний, таких как: большой положительный, средний положительный, малый, средний отрицательный и большой отрицательный.

Контроллер. Он состоит из базы знаний и механизма вывода. База знаний хранит функции принадлежности и нечеткие правила, полученные путем знания работы системы в среде. Механизм вывода выполняет обработку полученных функций принадлежности и нечетких правил. Другими словами, механизм вывода формирует выходные данные на основе лингвистической информации.

Дефазифаер (Defuzzifier или оператор восстановления чёткости). Он выполняет обратный процесс фазифаера. Другими словами, он преобразует нечеткие значения в нормальные числовые или физические сигналы и отправляет их в физическую систему для управления работой системы.

## Принцип работы системы нечеткой логики

Нечеткая операция предполагает использование нечетких множеств и функций принадлежности. Каждое нечеткое множество представляет собой представление лингвистической переменной, которая определяет возможное состояние вывода. Функция принадлежности является функцией общего значения в нечетком множестве, так что и общее значение, и нечеткое множество принадлежат универсальному множеству.

Степени принадлежности в этом общем значении в нечетком множестве определяют выход, основанный на принципе IF-THEN. Принадлежность назначается на основе предположения о выходе с помощью входов и скорости изменения входных данных. Функция принадлежности в основном представляет собой графическое представление нечеткого множества.

# Программа

import numpy as np

import skfuzzy as fuzz

*# New Antecedent/Consequent objects hold universe variables and membership*

*# functions*

temperature = ctrl.Antecedent(np.arange(0, 11, 1), 'temperature')

humidity = ctrl.Antecedent(np.arange(0, 11, 1), 'humidity')

feeling = ctrl.Consequent(np.arange(0, 26, 1), 'feeling')

*# Auto-membership function population is possible with .automf(3, 5, or 7)*

temperature.automf(3)

humidity.automf(3)

*# Custom membership functions can be built interactively with a familiar,*

*# Pythonic API*

feeling['Cold'] = fuzz.trimf(feeling.universe, [0, 0, 13])

feeling['Warm'] = fuzz.trimf(feeling.universe, [0, 13, 25])

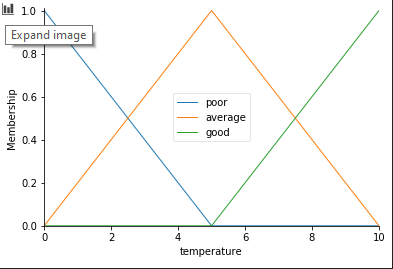
feeling['Hot'] = fuzz.trimf(feeling.universe, [13, 25, 25])

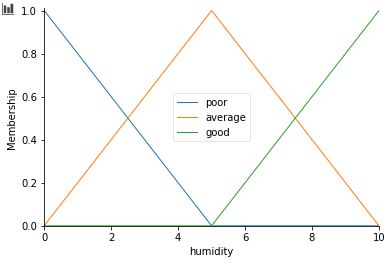
temperature.view()

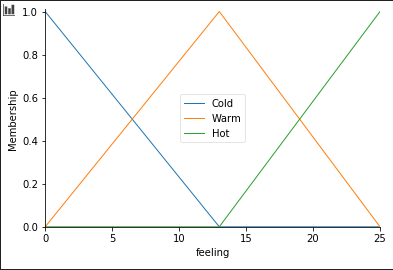
humidity.view()

feeling.view()

# Результаты







# Выводы

Нечеткие системы позволяют эмулировать сложное, интуитивное поведение на основе на разреженной системы правил с минимальными нагрузками. Обратите внимание на наше membership  функций были грубыми, определялись только целыми числами, но fuzz.interp\_membership позволяет увеличивать эффективное разрешение настолько, насколько требуется. Эта система может реагировать на сколь угодно малые изменения входных данных при минимальных нагрузках.