Aprend. Maq. e Model. Conhecim. Incerto
Apoio à Decisão com Lógica *Fuzzy*
Prof. Wagner Igarashi

Henrique R. Favaro[1], **Felipe Piassa**[1]

[1]Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá, PR – Brazil

ra115408@uem.br, ra112647@uem.br

Resumo. *Tomadas de decisão, quando tratando-se de múltiplas variáveis envolvidas no processo, podem ser algo desafiador para um sistema realizar, não sendo tão simples como escolher entre 0 e 1. Dessa forma, com base nos conceitos da lógica Fuzzy, aliados ao poder computacional das máquinas atuais, sistemas podem ser criados para realizar decisões com base em inúmeras variáveis, com grande confiabilidade. Sendo assim, o presente trabalho visa implementar e apresentar um sistema de gerenciamento com base nos conceitos da lógica Fuzzy aplicado à um cenário muito recorrente em empresas.*

1. Introdução

Dentro do campo de aprendizagem de máquinas, muito se discute sobre as melhores maneiras de implementar sistemas que auxiliem em diversas tarefas do cotidiano, sejam estes sistemas simples, como o de um aspirador de pó para limpar a casa, ou, até mesmo sistemas inteligentes que visam aprender com o decorrer das ações e aprimorarem seus resultados cada vez mais.

Neste contexto, a lógica *Fuzzy* se encaixa como uma ferramenta de grande utilidade dentro de sistemas complexos, pois possibilita a tomada de decisão com base em diversas variáveis, tratando de valores inteiros, como números entre 0 e 9, para cada uma das variáveis, e, apresentando como resultado uma combinação que gera um valor totalmente diferentes do inicial. Tal valor pode então ser aplicado à algum sistema que toma a decisão final do que será realizado.

No presente estudo, apenas abordada-se a lógica em si, quanto às suas variáveis de entrada, os processos realizados durante a transformação dos valores, e a geração de resultado final. Este valor final poderia então ser inserido em um outro sistema, para realizar tarefas de maneira totalmente autônoma.

Quanto ao cenário de aplicação, o presente estudo aborda a aplicação desta lógica para tomada de decisão em um sistema de Avaliação de Desempenho de Funcionários, podendo auxiliar um gestor à decidir quais funcionários irão permanecer na equipe, receber bonificação, ou, até mesmo serem promovidos. Os valores de entrada são notas de conceito em diferentes pontos avaliados pelo suposto gestor, onde na saída tem-se uma nota geral de desempenho para cada indivíduo.

2. Modelagem do Problema Avaliação de Desempenho de Funcionários

Para realizar a avaliação de desempenho de funcionários utilizando-se lógica *Fuzzy*, 4 variáveis foram selecionadas, consideradas como valores de entrada para a avaliação. Estas variáveis são representadas em valores de 1 a 10, onde cada funcionário recebe uma nota correspondente à sua dedicação.

As notas recebidas correspondem ao Número de Projetos Concluídos dentro do Prazo, Pesquisa de Satisfação dos Funcionários, Cumprimento de Metas Individuais e Participação em Treinamentos. Após a definição dessas variáveis, é possível a realização do cálculo de desempenho, estando em um intervalo de valores entre 1 e 100.

Os valores entre 1 e 20 são considerados desempenho baixo, de 20 a 40 está a fatia intermediária entre baixo e médio, de 40 a 60 temos o desempenho médio, de 60 a 80 tem-se o desempenho entre médio e alto, e, valores maiores que 80 são considerados desempenho alto.

Esse valor de desempenho é obtido quando a fusificação das notas obtidas é realizada, realizando-se então a fusificação do desempenho e obtendo-se um valor resultante final. Este valor pode ser utilizado por um gestou ou software responsável, para fins de realizar controle sobre os funcionários da empresa, tarefa essencial quando visa-se manter qualidade de equipe e resultados otimistas.

O conjunto das variáveis, ainda que estas sejam numeradas, pode ser representado de maneira linguística, onde, para todas estas tem-se os valores Baixo, Médio e Alto, indicando a participação de um funcionário em cada quesito avaliado, bem como seu desempenho.

Em relação às regras de produção, as quais selecionam valores de entrada e geram os valores de desempenho, estas foram criadas de maneira automática, contendo todas as possibilidades possíveis para a combinação das 4 variáveis, onde verifica-se a incidência de cada nível linguístico para selecionar o nível de saída (baixo, médio ou alto). Quando os níveis acontecem em mesma quantidade, seleciona-se médio como o nível de saída. Podendo-se notar a construção destas regras por meio da Figura 1.

Por fim, para o método de defusificação, após a comparação entre todos os disponíveis, selecionou-se o *Centroid* com uma melhor apresentação de resultados, não causando grande desvio nos resultados de maneira significativa.

3. Implementação do Software e Ferramentas Utilizadas

Visando a implementação e aplicação da lógica citada, no cenário de auxílio à tomada de decisão, utilizou-se de um programa desenvolvido na linguagem Python somado às bibliotecas NumPy[1] e scikit-fuzzy[2], para a criação de arranjos numéricos representando o intervalo de pertinência para cada variável, bem como a aplicação desses intervalos e a aplicação de lógica *Fuzzy* pela segunda biblioteca.

A linguagem Python foi escolhida por apresentar tais ferramentas de maneira simplificada, sem uma grande carga de trabalho durante implementação das funções necessárias utilizadas no processamento. Também, uma vasta quantidade de exemplos pode

[1] https://numpy.org/

[2] https://pythonhosted.org/scikit-fuzzy/

```
# Listas de variáveis e seus conjuntos
variaveis = [num_projetos, satisfacao, cumprimento_metas, participacao_treinamentos]
conjuntos = ['baixo', 'medio', 'alto']

# Lista de todas as combinações possíveis
combinacoes = list(itertools.product(conjuntos, repeat=len(variaveis)))

rules_file = open(self.RULES_FILE, 'w')
# Criar regras dinamicamente
rules = []
for combo in combinacoes:
    antecedentes = []
    for var, conj in zip(variaveis, combo):
        antecedentes.append(var[conj])
    consequente = functools.verify_max(combo)
    antecedente = antecedentes[0] & antecedentes[1] & antecedentes[2] & antecedentes[3]
    rule = ctrl.Rule(antecedente, desempenho[consequente])
    rules.append(rule)
    rules_file.write(str(rule) + '\n')

rules_file.close()
```

Figura 1: Criação das Regras de Produção

ser observada em consultas na mesma linguagem, facilitando o entendimento quanto ao funcionamento da biblioteca e suas configurações necessárias.

Para utilização do programa, basta executar o arquivo **main.py**, sendo solicitado o método de defusificação e o tipo de entrada de dados, que pode ser manual ou por meio de uma planilha do tipo *.csv*.

Ainda, o programa implementado foi desenvolvido dentro da ferramenta Visual Studio Code[3], em um Notebook da linha Dell modelo Inspiron, contando com um processador I5 de 12° geração, uma placa gráfica GTX 1050 TI, 16GB de memória RAM e sistema operacional Ubuntu 22.04-LTS.

3.1. Código desenvolvido em Python

O código desenvolvido pode ser analisado em duas partes, iniciando-se pela função principal, a qual é chamada logo que o programa é executado. Esta função tem como objetivo coletar o método de defusificação, bem como o modelo de dados de entrada. Após esta etapa, a função principal irá instanciar um objeto da classe FuzzyCore, classe esta que possui a implementação das funções que realizam o processamento dos dados. Por fim, a função principal guarda os resultados obtidos em arquivos de saída, denominados com o tipo de defusificação escolhido, no formato *.out*.

A classe FuzzyCore, citada anteriormente, contém a definição dos intervalos para cada uma das variáveis, a definição do intervalo para a variável resultado, bem como a definição de todas as regras de produção. Ainda, esta classe é responsável por instanciar todas as ferramentas utilizadas provenientes da biblioteca sckit-fuzzy, apresentada inicialmente.

Ambos os códigos desenvolvidos, função principal e classe FuzzyCore, podem ser observados pelas Figuras 2 e 3, respectivamente.

[3]https://code.visualstudio.com/

```
def main():
    defussify_method = input("Digite o método de defuzzificação ('centroid', 'bisector', 'mom', 'som', 'lom'): ")
    if(defussify_method not in ['centroid', 'bisector', 'mom', 'som', 'lom']):
        print("Método de defuzzificação inválido")
        return

    log_file_name = f"log_{defussify_method}_results.out"

    fuzzyCore = fuzzyCoreClass.fuzzy_core(defuzzify_method=defussify_method, log_file=log_file_name)
    simulador = fuzzyCore.get_simulator()
    log_file = fuzzyCore.get_log_file()

    logger = open(log_file, 'w')

    input_type = input("Digite o tipo de entrada ('manual' ou 'csv'): ")
    if input_type == 'manual':
    elif input_type == 'csv':
        satisfacao_arr, num_projetos_arr, cumprimento_metas_arr, participacao_treinamentos_arr = funcTools.get_csv_inputs(CSV_INPUT_FILE)

        for satisfacao, num_projetos, cumprimento_metas, participacao_treinamentos in zip(satisfacao_arr, num_projetos_arr, cumprimento_metas_arr, participacao_treinamentos_arr):
            simulador.input['satisfacao'] = satisfacao
            simulador.input['num_projetos'] = num_projetos
            simulador.input['cumprimento_metas'] = cumprimento_metas
            simulador.input['participacao_treinamentos'] = participacao_treinamentos

            # Computação do resultado
            simulador.compute()

            input_print = f"INPUT: [{format(satisfacao, '2')},{format(num_projetos, '2')},{format(cumprimento_metas, '2')},{format(participacao_treinamentos, '2')}]"
            result_print = f"DESEMPENHO: {format(round(simulador.output['desempenho'], 2), '5')},"
            logger.write(f"{input_print} | {result_print}\n")
        logger.close()
    else:
        print('Entrada inválida')
        return

    print('Resultado salvo em ' + log_file)
    print("Regras utilizadas salvas em " + fuzzyCore.get_rules_file())
    print('Gráfico salvo em ' + fuzzyCore.get_graph_file() + '.png')
    print("Fim da execução.")

    return
```

Figura 2: Função Principal

Optou-se pelo desenvolvimento desta forma, dividindo-se o programa em duas partes principais, para que a carga de trabalho ficasse bem definida dentro de cada arquivo, ou seja, o arquivo **main.py** apenas inicializa o processo e faz coleta de *inputs* do usuários, enquanto o arquivo **fuzzy_core.py** realiza todo o processamento com os dados recebidos da função principal, facilitando assim as possíveis manutenções e entendimento do código como um todo.

4. Execução de Experimentos

Visando executar o sistema implementado, pode ser executada a função principal, com o comando ***python3 main.py***, escolhendo-se as diferentes opções de defusificação e dados de entrada, como representado pela Figura 4, onde tem-se a utilização do sistema para todos os tipos de defusificação possíveis pela biblioteca utilizada.

Para validar o funcionamento do sistema, foram realizadas execuções com dados de entrada gerados de maneira randômica, os quais foram adicionados à uma planilha para facilitar a execução de dados em massa. Os resultados obtidos foram armazenados em um arquivo de saída *.out* para uma melhor análise, sem que fosse necessária uma leitura de dados diretamente no terminal.

Dessa forma, pode-se observar os dados de entrada utilizados, bem como os dados de saída gerados, por meio das Figuras 5a e 5b respectivamente.

Ambas as figuras representam apenas as primeiras 40 entradas de dados, porém, para o experimentos, foram utilizados 150 exemplos de entrada, o que pode ser observado acessando-se o repositório[4] disponível, contendo o código fonte e os dados utilizados.

Pode-se notar, por meio dos dados de entrada, que 4 colunas valoradas foram utilizadas. A primeira representando a nota do funcionário em relação à Pesquisa de Satisfação, a segunda representando a nota em relação ao Número de Projetos entregues dentro do prazo, a terceira contendo a nota em relação às metas, e, por fim, a quarta

[4]https://github.com/MrRique15/fuzzy_logic

```python
class fuzzy_core:
    def __init__(self, defuzzify_method='centroid', log_file=None):

        self.RULES_FILE = 'used_rules.txt'
        self.LOG_FILE =  log_file if log_file is not None else 'log.txt'
        self.GRAPH_FILE_NAME = 'graph'

        # Criação das variáveis de entrada
        satisfacao = ctrl.Antecedent(np.arange(0, 11, 1), 'satisfacao')
        num_projetos = ctrl.Antecedent(np.arange(0, 11, 1), 'num_projetos')
        cumprimento_metas = ctrl.Antecedent(np.arange(0, 11, 1), 'cumprimento_metas')
        participacao_treinamentos = ctrl.Antecedent(np.arange(0, 11, 1), 'participacao_treinamentos')

        # Criação da variável de saída
        desempenho = ctrl.Consequent(np.arange(0, 101, 1), 'desempenho', defuzzify_method=defuzzify_method)

        # Fuzificação das variáveis de entrada
        # Definição das funções de pertinência para cada variável
        satisfacao['baixo'] = fuzz.trapmf(satisfacao.universe, [0, 0, 2, 4])
        satisfacao['medio'] = fuzz.trapmf(satisfacao.universe, [2, 4, 6, 8])
        satisfacao['alto']  = fuzz.trapmf(satisfacao.universe, [6, 8, 10, 10])

        num_projetos['baixo'] = fuzz.trapmf(num_projetos.universe, [0, 0, 2, 4])
        num_projetos['medio'] = fuzz.trapmf(num_projetos.universe, [2, 4, 6, 8])
        num_projetos['alto']  = fuzz.trapmf(num_projetos.universe, [6, 8, 10, 10])

        cumprimento_metas['baixo'] = fuzz.trapmf(cumprimento_metas.universe, [0, 0, 2, 4])
        cumprimento_metas['medio'] = fuzz.trapmf(cumprimento_metas.universe, [2, 4, 6, 8])
        cumprimento_metas['alto']  = fuzz.trapmf(cumprimento_metas.universe, [6, 8, 10, 10])

        participacao_treinamentos['baixo'] = fuzz.trapmf(participacao_treinamentos.universe, [0, 0, 2, 4])
        participacao_treinamentos['medio'] = fuzz.trapmf(participacao_treinamentos.universe, [2, 4, 6, 8])
        participacao_treinamentos['alto']  = fuzz.trapmf(participacao_treinamentos.universe, [6, 8, 10, 10])

        # Definição das funções de pertinência para a variável de saída
        desempenho['baixo'] = fuzz.trapmf(desempenho.universe, [0, 0, 20, 40])
        desempenho['medio'] = fuzz.trapmf(desempenho.universe, [20, 40, 60, 80])
        desempenho['alto']  = fuzz.trapmf(desempenho.universe, [60, 80, 100, 100])

        # Listas de variáveis e seus conjuntos
        variaveis = [num_projetos, satisfacao, cumprimento_metas, participacao_treinamentos]
        conjuntos = ['baixo', 'medio', 'alto']

        # Lista de todas as combinações possíveis
        combinacoes = list(itertools.product(conjuntos, repeat=len(variaveis)))
```

(a) Parte 1

```python
        rules_file = open(self.RULES_FILE, 'w')
        # Criar regras dinamicamente
        rules = []
        for combo in combinacoes:
            antecedentes = []
            for var, conj in zip(variaveis, combo):
                antecedentes.append(var[conj])
            consequente = functools.verify_max(combo)
            antecedente = antecedentes[0] & antecedentes[1] & antecedentes[2] & antecedentes[3]
            rule = ctrl.Rule(antecedente, desempenho[consequente])
            rules.append(rule)
            rules_file.write(str(rule) + '\n')

        rules_file.close()

        # Criação do sistema de controle
        self.desempenho_ctrl = ctrl.ControlSystem(rules)
        self.plot_graph(desempenho, satisfacao)

    def get_simulator(self):
        desempenho_simulador = ctrl.ControlSystemSimulation(self.desempenho_ctrl)
        return desempenho_simulador

    def get_log_file(self):
        return self.LOG_FILE

    def get_rules_file(self):
        return self.RULES_FILE

    def get_graph_file(self):
        return self.GRAPH_FILE_NAME + '.png'

    def plot_graph(self, desempenho, satisfacao):
        # Plot
        desempenho.view(sim=self.get_simulator())
        plt.savefig(self.GRAPH_FILE_NAME + '_desempenho' + '.png')

        satisfacao.view(sim=self.get_simulator())
        plt.savefig(self.GRAPH_FILE_NAME + '_inputs' + '.png')
```

(b) Parte 2

Figura 3: Classe FuzzyCore

```
henrique@henrique-dell-notebook:~/Desktop/CONTENT/UEM/UEM_Aulas/AprendMaquinas/trab1_fuzzy$ python3 main.py
Digite o método de defuzzificação ('centroid', 'bisector', 'mom', 'som', 'lom'): centroid
Digite o tipo de entrada ('manual' ou 'csv'): csv
Resultado salvo em -> outputs/log_centroid_results.out
Regras utilizadas salvas em -> outputs/used_rules.txt
Gráficos salvos em -> outputs/graph.png.png
Fim da execução.
henrique@henrique-dell-notebook:~/Desktop/CONTENT/UEM/UEM_Aulas/AprendMaquinas/trab1_fuzzy$ python3 main.py
Digite o método de defuzzificação ('centroid', 'bisector', 'mom', 'som', 'lom'): bisector
Digite o tipo de entrada ('manual' ou 'csv'): csv
Resultado salvo em -> outputs/log_bisector_results.out
Regras utilizadas salvas em -> outputs/used_rules.txt
Gráficos salvos em -> outputs/graph.png.png
Fim da execução.
henrique@henrique-dell-notebook:~/Desktop/CONTENT/UEM/UEM_Aulas/AprendMaquinas/trab1_fuzzy$ python3 main.py
Digite o método de defuzzificação ('centroid', 'bisector', 'mom', 'som', 'lom'): mom
Digite o tipo de entrada ('manual' ou 'csv'): csv
Resultado salvo em -> outputs/log_mom_results.out
Regras utilizadas salvas em -> outputs/used_rules.txt
Gráficos salvos em -> outputs/graph.png.png
Fim da execução.
henrique@henrique-dell-notebook:~/Desktop/CONTENT/UEM/UEM_Aulas/AprendMaquinas/trab1_fuzzy$ python3 main.py
Digite o método de defuzzificação ('centroid', 'bisector', 'mom', 'som', 'lom'): som
Digite o tipo de entrada ('manual' ou 'csv'): csv
Resultado salvo em -> outputs/log_som_results.out
Regras utilizadas salvas em -> outputs/used_rules.txt
Gráficos salvos em -> outputs/graph.png.png
Fim da execução.
henrique@henrique-dell-notebook:~/Desktop/CONTENT/UEM/UEM_Aulas/AprendMaquinas/trab1_fuzzy$ python3 main.py
Digite o método de defuzzificação ('centroid', 'bisector', 'mom', 'som', 'lom'): lom
Digite o tipo de entrada ('manual' ou 'csv'): csv
Resultado salvo em -> outputs/log_lom_results.out
Regras utilizadas salvas em -> outputs/used_rules.txt
Gráficos salvos em -> outputs/graph.png.png
Fim da execução.
```

Figura 4: Execuções do Sistema

	A	B	C	D
1	3	8	6	8
2	2	1	10	5
3	5	10	9	1
4	10	3	7	4
5	1	6	7	4
6	8	10	3	4
7	9	4	8	2
8	10	1	5	5
9	7	9	2	1
10	9	3	9	1
11	8	10	9	2
12	9	6	6	7
13	4	5	3	1
14	10	3	9	3
15	8	5	5	10
16	6	7	4	4
17	5	1	1	10
18	8	5	8	9
19	7	5	9	9
20	3	4	10	5
21	10	7	9	4
22	6	8	8	1
23	4	7	3	4
24	7	1	3	8
25	3	10	8	5
26	10	5	7	6
27	10	7	8	4
28	8	10	4	9
29	3	6	4	7
30	6	6	2	5
31	7	3	10	7
32	1	7	6	4
33	4	8	9	10
34	1	9	2	10
35	2	9	2	10
36	1	6	8	7
37	10	2	8	9
38	1	7	7	2
39	1	7	5	6
40	8	4	2	1

(a) Dados de Entrada

```
You, há 2 horas | 1 author (You)
1  INPUT: [ 3, 8, 6, 8] | DESEMPENHO: 62.44,
2  INPUT: [ 2, 1,10, 5] | DESEMPENHO: 15.56,
3  INPUT: [ 5,10, 9, 1] | DESEMPENHO: 84.44,
4  INPUT: [10, 3, 7, 4] | DESEMPENHO: 62.44,
5  INPUT: [ 1, 6, 7, 4] | DESEMPENHO:  50.0,
6  INPUT: [ 8,10, 3, 4] | DESEMPENHO: 62.44,
7  INPUT: [ 9, 4, 8, 2] | DESEMPENHO: 84.44,
8  INPUT: [10, 1, 5, 5] | DESEMPENHO:  50.0,
9  INPUT: [ 7, 9, 2, 1] | DESEMPENHO: 37.56,
10 INPUT: [ 9, 3, 9, 1] | DESEMPENHO: 62.44,
11 INPUT: [ 8,10, 9, 2] | DESEMPENHO: 84.44,
12 INPUT: [ 9, 6, 6, 7] | DESEMPENHO:  50.0,
13 INPUT: [ 4, 5, 3, 1] | DESEMPENHO:  50.0,
14 INPUT: [10, 3, 9, 3] | DESEMPENHO: 62.44,
15 INPUT: [ 8, 5, 5,10] | DESEMPENHO:  50.0,
16 INPUT: [ 6, 7, 4, 4] | DESEMPENHO:  50.0,
17 INPUT: [ 5, 1, 1,10] | DESEMPENHO: 15.56,
18 INPUT: [ 8, 5, 8, 9] | DESEMPENHO: 84.44,
19 INPUT: [ 7, 5, 9, 9] | DESEMPENHO: 62.44,
20 INPUT: [ 3, 4,10, 5] | DESEMPENHO:  50.0,
21 INPUT: [10, 7, 9, 4] | DESEMPENHO: 62.44,
22 INPUT: [ 6, 8, 8, 1] | DESEMPENHO: 84.44,
23 INPUT: [ 4, 7, 3, 4] | DESEMPENHO:  50.0,
24 INPUT: [ 7, 1, 3, 8] | DESEMPENHO:  50.0,
25 INPUT: [ 3,10, 8, 5] | DESEMPENHO: 62.44,
26 INPUT: [10, 5, 7, 6] | DESEMPENHO:  50.0,
27 INPUT: [10, 7, 8, 4] | DESEMPENHO: 62.44,
28 INPUT: [ 8,10, 4, 9] | DESEMPENHO: 84.44,
29 INPUT: [ 3, 6, 4, 7] | DESEMPENHO:  50.0,
30 INPUT: [ 6, 6, 2, 5] | DESEMPENHO:  50.0,
31 INPUT: [ 7, 3,10, 7] | DESEMPENHO: 62.44,
32 INPUT: [ 1, 7, 6, 4] | DESEMPENHO:  50.0,
33 INPUT: [ 4, 8, 9,10] | DESEMPENHO: 84.44,
34 INPUT: [ 1, 9, 2,10] | DESEMPENHO:  50.0,
35 INPUT: [ 2, 9, 2,10] | DESEMPENHO:  50.0,
36 INPUT: [ 1, 6, 8, 7] | DESEMPENHO: 62.44,
37 INPUT: [10, 2, 8, 9] | DESEMPENHO: 84.44,
38 INPUT: [ 1, 7, 7, 2] | DESEMPENHO: 37.56,
39 INPUT: [ 1, 7, 5, 6] | DESEMPENHO:  50.0,
40 INPUT: [ 8, 4, 2, 1] | DESEMPENHO: 15.56,
```

(b) Resultados - Centroid

Figura 5: Dados utilizados e resultados obtidos

fazendo referência a participação em treinamentos deste mesmo funcionário. Com todas estes valores disponíveis, é possível a aplicação dentro do sistema, obtendo-se o valor resultante de desempenho, observado nos dados de resultado.

Referências

Du, Y., Liu, Y., Liu, D., Liu, R., Zhang, W., and Yu, L. (2021). Python-based fuzzy control of ore feeding system.

Majumdar, S. and Brick, A. (2022). Recognizing handwriting styles in a historical scanned document using scikit-fuzzy c-means clustering. *arXiv preprint arXiv:2210.16780*.

Russell, S. J. and Norvig, P. (2010). *Artificial intelligence a modern approach*. London.