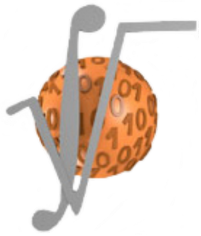


UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL
CURSO DE MESTRADO EM MODELAGEM COMPUTACIONAL

Generalização do Processo de Agregação do Sistema Fuzzy Multi-rótulo Takagi-Sugeno-Kan baseado na Integral de Choquet

por

Karina Maria Vargas Condori

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal do Rio Grande - FURG, como requisito parcial para obtenção do Grau de Mestre. Área concentração: Modelagem Computacional.

Orientadora: Profª. Drª. Graçaliz Pereira Dimuro
Coorientador: Profº. Dr. Julian Moises Sejje Suarez

Rio Grande, dezembro, 2024

Ficha catalográfica

B928r Condori, Karina Maria Vargas.

Generalização do Processo de Agregação do Sistema Fuzzy Multi-rótulo Takagi-Sugeno-Kan baseado na Integral de Choquet / Karina Maria Vargas Condori. – 2024.
72f.

Dissertação (mestrado) – Universidade Federal do Rio Grande – FURG, Programa de Pós-Graduação em Modelagem Computacional, Rio Grande/RS, 2024.
Orientadora: Dra. Graçaliz Pereira Dimuro.
Coorientador: Dr. Julian Moises Sejje Suarez.

1. Integral de Choquet 2. Funções de agregação 3. Sistemas de Inferência Difusa 4. Classificação Multi-etiqueta I. Dimuro, Graçaliz Pereira II. Suarez, Julian Moises Sejje III. Título.

CDU 004.932

Karina Vargas Condori

“Generalização do Processo de Agregação do Sistema Fuzzy Multi-rótulo Takagi-Sugeno-Kan baseado na Integral de Choquet”

Dissertação apresentada ao Programa de Pós - Graduação em Modelagem Computacional da Universidade Federal do Rio Grande - FURG, como requisito parcial para obtenção do Grau de Mestre. Área de concentração: Modelagem Computacional.

Aprovada em: 04/12/2024

BANCA EXAMINADORA

Profa. Dra. Graçaliz Pereira Dimuro (Orientadora – FURG)

Prof Dr. Julian Moises Sejje Suarez (Coorientador – FURG)

Prof. Dr. Rodrigo de Bem (FURG)

Prof. Dr. Benjamin Bedregal (UFRN)

Profa. Dra. Heloisa de Arruda Camargo (UFSCAR)

Profa. Dra. Diana Francisca Adamatti (FURG)

Rio Grande - RS
2024

Dedicatória

Aos meus pais,
pelo amor incondicional,
pelo apoio constante
e por me ensinarem o valor da perseverança.
Aos meus irmãos,
por serem minha inspiração
e meus melhores amigos.
Aos meus professores e mentores,
por me guiarem neste caminho
e compartilharem comigo sua sabedoria.
E finalmente,
a todos aqueles que acreditaram em mim,
esta conquista também é de vocês.

Agradecimentos

À minha família, por sempre acreditarem em mim e apoiarem incondicionalmente a realização dos meus sonhos.

À minha orientadora, Prof ª Dra. Graçaliz Pereira Dimuro, pela inspiração, pelas ideias, pela incansável dedicação e paciência, e pelos valiosos ensinamentos compartilhados ao longo deste trabalho.

Ao meu coorientador, Prof º Dr. Julian Moises Sejje Suarez, pela imensa paciência, pelo constante apoio, pelo companheirismo e pela generosa partilha de conhecimentos.

Aos colegas da pós-graduação, por compartilharem ideias, por sua amizade e pelo apoio durante os estudos e a escrita desta dissertação.

Ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal do Rio Grande, por fornecer um ambiente acadêmico estimulante e recursos essenciais para a realização deste trabalho.

À Universidade Federal do Rio Grande, pela infraestrutura e pela assistência estudantil oferecida, que foram cruciais durante minha jornada acadêmica.

E a todos que, de alguma forma, passaram pela minha vida, me incentivaram e contribuíram para que eu chegasse até aqui. Esta conquista é também de vocês.

Epígrafe

"O sucesso é a soma de pequenos esforços repetidos dia após dia."
— Robert Collier

Resumo

A classificação multi-rótulo é uma tarefa fundamental no aprendizado de máquina, pois permite que uma instância pertença a múltiplas categorias simultaneamente, o que é essencial em diversos problemas reais, como o reconhecimento de imagens em Engenharia e Ciências Médicas. Esse tipo de classificação possibilita uma representação mais precisa e abrangente das relações complexas entre dados e categorias, superando as limitações das classificações binária e multi-classe.

Esta dissertação apresenta o **Sistema Fuzzy Multi-Rótulo Takagi-Sugeno-Kang-Choquet (ML-TSKC FS)**, um modelo inovador de classificação multi-rótulo que generaliza o Sistema Fuzzy Multi-Rótulo Takagi-Sugeno-Kang (ML-TSK FS) ao incorporar a **Integral de Choquet**. O modelo proposto utiliza a integral de Choquet, definida em termos de medidas fuzzy, para agregar informações no cálculo da força de ativação nos antecedentes das regras, capturando interações complexas entre atributos e a incerteza presente nos dados. Essa abordagem permite um tratamento mais refinado da informação, tornando o modelo especialmente eficaz em cenários onde os rótulos possuem dependências complexas.

Para avaliar o desempenho do ML-TSKC FS, exploramos a aplicação da integral de Choquet com cinco medidas fuzzy distintas, realizando um estudo comparativo em doze conjuntos de dados de classificação multi-rótulo. A metodologia inclui validação cruzada e testes estatísticos para garantir a robustez dos resultados. Os resultados mostram que o modelo ML-TSKC FS oferece melhorias significativas em termos de precisão e capacidade de generalização, quando comparado ao modelo padrão ML-TSK FS e a outros métodos de referência, incluindo algoritmos tradicionais, redes neurais e sistemas fuzzy.

Concluímos que o uso da Integral de Choquet no processo de agregação aumenta a robustez e a precisão do modelo ao lidar com incertezas e dependências complexas, tornando o ML-TSKC FS uma alternativa promissora para problemas de classificação multi-rótulo em aplicações práticas.

Palavras-Chave: Multi-rótulo Classificação, Integral de Choquet, Medida Fuzzy, Sistema de Inferência Neuro-Fuzzy, Sistema Fuzzy Multi-rótulo Takagi-Sugeno-Kang.

Abstract

Multi-label classification is a fundamental task in machine learning, as it allows an instance to belong to multiple categories simultaneously, which is essential in various real-world problems, such as image recognition in Engineering and Medical Sciences. This type of classification enables a more precise and comprehensive representation of the complex relationships between data and categories, overcoming the limitations of binary and multi-class classification.

This dissertation presents the **Takagi-Sugeno-Kang-Choquet Multi-Label Fuzzy System (ML-TSKC FS)**, an innovative multi-label classification model that extends the Takagi-Sugeno-Kang Multi-Label Fuzzy System (ML-TSK FS) by incorporating the **Choquet Integral**. The proposed model uses the Choquet integral, defined in terms of fuzzy measures, to aggregate information in the calculation of activation strength in rule antecedents, capturing complex interactions between attributes and the uncertainty present in the data. This approach enables a more refined information processing, making the model especially effective in scenarios where labels exhibit complex dependencies.

To evaluate the performance of ML-TSKC FS, we applied the Choquet integral with five distinct fuzzy measures in a comparative study on twelve multi-label classification datasets. The methodology includes cross-validation and statistical tests to ensure robust results. The results show that the ML-TSKC FS model provides significant improvements in terms of accuracy and generalization capacity when compared to the standard ML-TSK FS model and other reference methods, including traditional algorithms, neural networks, and fuzzy systems.

We conclude that using the Choquet Integral in the aggregation process increases the model's robustness and accuracy in handling uncertainties and complex dependencies, making ML-TSKC FS a promising alternative for multi-label classification problems in practical applications.

Keywords: Multi-Label Classification, Choquet Integral, Fuzzy Measure, Neuro-Fuzzy Inference System, Multi-Label Takagi-Sugeno-Kang Fuzzy System

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AM Média Aritmética

ANFIS Sistema Adaptativo de Inferência Neuro-Fuzzy

AP Precisão Média (Average Precision)

CV Cobertura (Coverage)

DNN Deep Neural Network (Rede Neural Profunda)

FCM Fuzzy C-Means

FIS Sistema de Inferência Fuzzy

FNN Fuzzy Neural Network (Rede Neural Fuzzy)

FS Fuzzy System (Sistema Fuzzy)

GM Média Geométrica

HL Perda de Hamming (Hamming Loss)

HM Média Harmônica

ML-TSK FS Sistema Fuzzy Multi-Rótulo Takagi-Sugeno-Kang

ML-TSKC FS Sistema Fuzzy Multi-Rótulo Takagi-Sugeno-Kang-Choquet

OWA Operador de Ponderação Ordenada

PR Produto

QM Média Quadrática

RL Perda de Ranking (Ranking Loss)

RNA Redes Neurais Artificiais

SVM Support Vector Machine (Máquina de Vetores de Suporte)

TSK Takagi-Sugeno-Kang

Capítulo 1

INTRODUÇÃO

No mundo atual, a quantidade de informação disponível é imensa e cresce de forma exponencial. Para lidar e entender essa avalanche de dados, é crucial desenvolver modelos de aprendizado de máquina eficientes e precisos. Uma técnica que tem ganhado destaque nesse campo é a **classificação multi-rótulo**, que amplia as capacidades das abordagens tradicionais de classificação, permitindo que uma instância seja associada a múltiplas categorias simultaneamente. Por exemplo, um artigo de notícias pode ser classificado como *política*, *economia* e "internacional" ao mesmo tempo [Wei et al. 2022]. Isso é especialmente útil em tarefas como categorização de textos e reconhecimento de imagens, onde os dados do mundo real muitas vezes não podem ser limitados a uma única categoria.

Essa flexibilidade permite que os modelos lidem com a complexidade inerente aos dados reais, superando as limitações das abordagens binárias ou multi-classes, que atribuem cada instância a um único rótulo [Tsoumakas e Katakis 2007]. Além disso, a classificação multi-rótulo considera as potenciais interações entre os rótulos. Por exemplo, na medicina, um paciente pode apresentar sintomas que correspondem a múltiplas doenças relacionadas. Ignorar essas interações pode levar a diagnósticos menos precisos [Herrera et al. 2016]. Com essa perspectiva, é possível capturar como a presença de um rótulo pode influenciar a presença de outro, oferecendo uma visão mais precisa e contextualizada dos problemas do mundo real [Read et al. 2011].

No entanto, essas abordagens mais detalhadas apresentam desafios [Zhang, Ling et al. 2013]. Além disso, a incerteza e a imprecisão comuns em tarefas de classificação multi-rótulo agravam esses desafios. Conjuntos de dados com ruído ou incompletos

podem criar ambiguidades na classificação, onde métodos que não consideram a incerteza inerente aos dados podem falhar em produzir modelos robustos e confiáveis [Gibaja e Ventura 2015].

1.1 Motivação e Justificativa

Motivados pelos desafios apresentados, torna-se necessário explorar novas abordagens que contemplem tanto a complexidade, interdependência entre rótulos, incerteza e a imprecisão dos dados. Foi nesse contexto que os **modelos neuro-fuzzy** apareceram como uma alternativa promissora [Lou et al. 2021]. Esses modelos combinam as vantagens das redes neurais, que são eficientes para aprender representações complexas de grandes volumes de dados, com a lógica fuzzy, conhecida por sua eficácia no tratamento de incertezas e imprecisões [Zadeh 1965].

Para ilustrar isso, considere o problema de diagnosticar doenças a partir de exames laboratoriais. Muitas vezes, os resultados de exames clínicos apresentam variações sutis e sobreposição de valores entre diferentes condições médicas. Por exemplo, níveis moderadamente elevados de glucose e pressão arterial podem ser indicativos de diversas condições, como pré-diabetes, diabetes tipo 2 ou até síndrome metabólica, dependendo de outros fatores. Um modelo neuro-fuzzy consegue lidar com essa incerteza nos dados de entrada ao permitir graus de pertinência a diferentes diagnósticos, ajustando-se a cada cenário clínico específico e emulando o raciocínio humano.

A integração de redes neurais e lógica fuzzy oferece uma maneira de superar as limitações das técnicas convencionais, proporcionando uma estrutura robusta para modelar dados complexos e ambíguos [Lin et al. 1991; Jang e Jyh-Shing 1993; Kasabov, Song e Qun 2002; Lou et al. 2021].donde

Contudo, um aspecto fundamental dos modelos neuro-fuzzy está no tratamento da informação dos atributos durante a ativação das regras, ou seja, na forma como o peso dessas regras é calculado no processo de inferência.

Classificadores notáveis, como o ANFIS (Sistema Adaptativo de Inferência Neuro-Fuzzy [Jang e Jyh-Shing 1993], DENFIS (Sistema Evolutivo Dinâmico de Inferência Neuro-Fuzzy) [Kasabov, Song e Qun 2002], HYFIS (Sistema Híbrido de Inferência Neuro-Fuzzy) [Kim e Kasabov 1999], e ML TSK FS (Sistema Fuzzy Multi-Rótulo Takagi-Sugeno-Kang) [Lou et al. 2021], utilizam operações de agregação tradicionais, como os operadores mínimo e produto, para esse cálculo. Embora esses operadores tenham sido importantes no início do desenvolvimento dos sistemas de inferência fuzzy, eles não conseguem representar adequadamente a complexidade das interações entre atributos e rótulos. Isso ocorre porque assumem relações simples ou independentes, enquanto contextos de classificação multi-rótulo exigem ferramentas capazes

de modelar interações mais complexas e interdependências entre os dados.

Portanto, há uma necessidade crítica de explorar novas abordagens de agregação que possam superar essas limitações, fornecendo um meio mais eficaz de considerar as interações complexas entre atributos e a incerteza dos dados. O desenvolvimento de tais métodos representa um passo fundamental para melhorar a eficácia dos modelos neuro-fuzzy em aplicações de classificação multi-rótulo.

Por outro lado, estudos recentes têm mostrado que o uso da **Integral de Choquet** como método de agregação no cálculo da **agregação das regras** em sistemas de inferência fuzzy tem gerado bons resultados, conforme demonstrado em diversos estudos [Lucca, Sanz, Dimuro et al. 2019; Marco-Detchart et al. 2021; Wieczynski, Dimuro et al. 2020; Ferrero-Jaurrieta et al. 2023]. Esses estudos exploram a aplicação da Integral de Choquet em áreas como processamento de imagem, memória de longo prazo, tomada de decisão multicritério, classificação, interfaces cerebrais, reconhecimento de padrões e gerenciamento de projetos.

A **Integral de Choquet** é uma ferramenta matemática sofisticada que vai além dos operadores tradicionais de agregação, como mínimo e máximo, usados em sistemas fuzzy convencionais. Enquanto esses operadores se baseiam nos graus individuais de pertinência das variáveis, a Integral de Choquet, definida em termos de uma medida fuzzy, captura o relacionamento entre os atributos, considerando a importância relativa de cada combinação possível de variáveis [Lucca, Sanz, Dimuro et al. 2019; Wieczynski, Dimuro et al. 2020].

1.2 Pergunta de Pesquisa

À luz do exposto, formula-se a seguinte pergunta de pesquisa:

A inclusão da Integral Discreta de Choquet na agregação de atributos para calcular a ativação das regras em um modelo de classificação multi-rótulo neuro-fuzzy pode levar a uma melhoria no desempenho do modelo original?

1.3 Objetivos

Objetivo Geral

O objetivo principal deste trabalho foi:

- Propor uma generalização do modelo ML-TSK FS, desenvolvendo um novo modelo denominado ML-TSKC FS, que utiliza a Integral de Choquet no processo de agregação de atributos para o cálculo da ativação das regras (Antecedentes) para melhorar a performance e robustez na classificação multi-rótulo.

Objetivos Específicos

1. Estudar os métodos de agregação usados em modelos neuro-fuzzy para classificação.
2. Implementar a Integral de Choquet no cálculo da ativação das regras no modelo original, gerando o novo modelo generalizado proposto ML-TSKC FS.
3. Avaliar o desempenho do modelo proposto em comparação com abordagens existentes na literatura, utilizando conjuntos de domínios diferentes.
4. Validar a eficácia do modelo através de testes estatísticos.

1.4 Metodologia

Para alcançar os objetivos propostos, foi seguida uma metodologia estruturada em várias etapas:

- **Revisão Bibliográfica**: Estudo dos modelos neuro-fuzzy existentes e das técnicas de agregação utilizadas na classificação.
- **Desenvolvimento do Modelo**: Implementação da integral de Choquet no algoritmo do modelo ML-TSK FS.
- **Implementação Computacional**: Adaptação do código do modelo original ML-TSK FS, utilizando ferramentas de programação adequadas para obter o código da versão proposta.
- **Experimentação**: Estudos comparativos do modelo proposto ML-TSKC FS com o modelo original ML-TSK FS e modelos consolidados na Literatura.
- **Análise de Resultados**: Avaliação de desempenho do modelo proposto ML-TSKC FS com métricas padrão em classificação multi-rótulo e realização de testes estatísticos comparativos.

1.5 Estrutura do Trabalho

O trabalho organiza-se da seguinte maneira:

- Capítulo 1: Apresenta o contexto da pesquisa, motivação e justificativa, além dos objetivos geral e específicos.
- Capítulo 2: Discute a classificação multi-rótulo, aprendizado de máquina, redes neurais artificiais, sistemas de inferência fuzzy, e a Integral de Choquet, fornecendo a base teórica necessária.

- Capítulo 3: Descreve o desenvolvimento do modelo proposto, Sistema Fuzzy Multi-Rótulo Takagi-Sugeno-Kang Choquet (ML-TSKC-FS). Detalha a arquitetura do modelo proposto, descrevendo cada uma das camadas e o método de aprendizagem utilizado.
- Capítulo 4: Apresenta as ferramentas e técnicas usadas na análise experimental, os experimentos realizados e os resultados obtidos, tanto em comparação com o modelo ML-TSK FS quanto com outros modelos da literatura.
- Capítulo 5: Resume as principais conclusões derivadas da pesquisa e aponta possíveis direções para futuros trabalhos, destacando as contribuições do modelo ML-TSKC FS.

Capítulo 2

PRELIMINARES

Este capítulo apresenta os conceitos teóricos essenciais que fundamentam este trabalho. Primeiramente, discute-se a classificação multi-rótulo. Em seguida, apresenta-se a fundamentação dos sistemas neuro-fuzzy, introduzindo os princípios básicos das redes neurais artificiais e dos sistemas de inferência fuzzy. A seguir, explora-se a Integral de Choquet, que desempenha um papel central na agregação de dados. Esses conceitos estabelecem a base para o entendimento do desenvolvimento proposto neste trabalho.

2.1 CLASSIFICAÇÃO MULTI-RÓTULO

Nesta seção, abordaremos a classificação multi-rótulo no contexto do aprendizado de máquina supervisionado, também explicaremos suas diferenças em relação às classificações tradicionais e suas aplicações práticas.

2.1.1 Aprendizado de Máquina

O aprendizado de máquina, ou *Machine Learning* (ML), é uma subdisciplina da inteligência artificial que permite a criação de modelos capazes de ***aprender*** a partir dos dados, ajustando seu comportamento com base em padrões e características extraídas automaticamente, sem necessidade de programação explícita para cada tarefa [Mitchell 1997]. Esse aprendizado é especialmente útil em problemas onde há grandes quantidades de dados e a complexidade é alta, tornando a solução por métodos tradicionais impraticável [Alpaydin 2010].

O principal objetivo do aprendizado de máquina é desenvolver algoritmos que possam realizar inferências ou previsões a partir de novos dados, imitando a capacidade humana de aprendizado por meio da experiência. Esse processo tem ampla aplicação, desde a análise de dados e reconhecimento de padrões até a automação de tarefas em áreas como visão computacional, processamento de linguagem natural e medicina [Mitchell 1997; Herrera et al. 2016].

Tipos de Aprendizado de Máquina

Existem três tipos principais de aprendizado de máquina, cada um adaptado a diferentes tipos de problemas, conforme mostrado na Figura 2.1.

Figura 2.1: Aprendizado de máquina

- **Aprendizado Supervisionado**: Neste tipo, o modelo é treinado com dados rotulados, onde cada entrada tem uma saída conhecida. O objetivo é aprender a mapear as entradas para as saídas corretas e realizar previsões precisas. As principais tarefas incluem:
  - *Classificação*: A tarefa de categorizar dados em classes, como identificar se um e-mail é **spam** ou **não spam.**
  - *Regressão*: Focada em prever valores numéricos contínuos, como o preço de uma casa com base em suas características [Alpaydin 2010].
- **Aprendizado Não Supervisionado**: Aqui, o modelo trabalha com dados não rotulados, buscando identificar padrões e relações subjacentes nos dados. As principais tarefas incluem:
  - *Clusterização*: Agrupamento de dados similares, como segmentar clientes com base em seu comportamento de compra.

  - *Redução de Dimensionalidade*: Simplificação dos dados, preservando as características mais importantes para facilitar a análise e visualização [Kosko 1992; Herrera et al. 2016].

- **Aprendizado por Reforço**: Nesse tipo de aprendizado, um agente interage com o ambiente e aprende por tentativa e erro, buscando maximizar uma recompensa acumulativa. É amplamente aplicado em controle de robôs e jogos, onde decisões sequenciais são fundamentais [Mitchell 1997].

Dentre as várias tarefas de aprendizado supervisionado, a classificação é uma das mais importantes e aplicáveis. A capacidade de um modelo de aprendizado de máquina de classificar dados com precisão, permite tomadas de decisão automáticas e precisas, sendo, portanto, uma ferramenta poderosa para transformar dados complexos em informações úteis. [Mitchell 1997; Suthaharan 2016].

2.1.2 Classificação

A classificação de dados é uma tarefa fundamental no aprendizado supervisionado. O processo de classificação envolve a atribuição de uma ou mais categorias predefinidas a instâncias de dados com base em características observadas. Cada instância é classificada em uma categoria específica, com o objetivo de distinguir entre diferentes classes ou grupos nos dados de entrada. [Mitchell 1997; Herrera et al. 2016]. A seguir, uma definição formal do processo de classificação.

Definição 1. *(Classificação) Seja $\mathcal{X}$ o espaço de entrada e $\mathcal{Y}$ o espaço de saída. Definimos o conjunto de treinamento $\mathcal{D} = \{(x_i, y_i) \mid 1 \leq i \leq n\}$, onde x_i pertence a $\mathcal{X}$ e y_i pertence a $\mathcal{Y}$. O objetivo da classificação é aprender um modelo, representado por uma aplicação $f_p : \mathcal{X} \to \mathcal{Y}$, onde p representa os parâmetros que determinam o modelo específico. Para um novo dado de entrada x, usamos f_p para prever a saída correspondente $\hat{y} = f_p(x)$.*

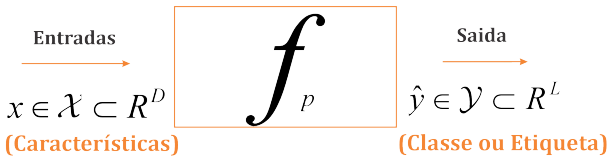


Figura 2.2: Classificador

Com base na definição 1 e na Figura 2.2, podemos descrever a classificação como um processo em que um modelo de aprendizado de máquina é treinado para associar um conjunto de características de entrada, representado pelo espaço $\mathcal{X}$, a uma variável de saída discreta, representada pelo espaço $\mathcal{Y}$, que contém as classes ou categorias possíveis. Para isso, utiliza-se um conjunto de treinamento $\mathcal{D} =$

$\{(x_i, y_i) \mid 1 \leq i \leq n\}$, onde cada x_i pertence ao espaço de entrada $\mathcal{X}$ e cada y_i pertence ao espaço de saída $\mathcal{Y}$. O objetivo de um modelo de classificação é estimar uma função, chamada de *classificador*, $f_p : \mathcal{X} \to \mathcal{Y}$, onde p representa os parâmetros que definem o modelo específico (ver Figura 2.2). Assim, dada uma nova instância de entrada x, o modelo utiliza a função f_p para prever a saída correspondente $\hat{y} = f_p(x)$. Esse processo busca minimizar os erros de classificação e maximizar a precisão ao aplicar o modelo em novos dados [Alpaydin 2010; Mitchell 1997].

A seguir, abordaremos os principais tipos de classificação em aprendizado de máquina, incluindo: classificação binária, multi-classe e multi-rótulo.

2.1.3 Tipos de Classificação

De acordo com [Herrera et al. 2016], os tipos de classificação no aprendizado supervisionado são determinados pela natureza das saídas que o modelo deve prever. Esses tipos incluem classificação binária, multi-classe e multi-rótulo, cada um deles direcionado a problemas específicos e características dos dados. Essas variações garantem que os modelos possam se adaptar a diferentes graus de complexidade e aplicações, desde problemas simples até os que envolvem múltiplas categorias simultâneas. A seguir, exploramos cada tipo de classificação em detalhe.

- **Classificação Binária:** Consiste em atribuir uma das duas categorias possíveis a cada item em um conjunto de dados. Este tipo de classificação é simples e fundamental, onde o objetivo é que o modelo aprenda a distinguir entre duas opções opostas, como *sim* ou *não*. Um exemplo clássico de classificação binária é a detecção de e-mails de spam, onde o modelo classifica os e-mails como *spam* ou *não spam* como mostrado na Figura 2.3.

Figura 2.3: Exemplo de classificação binária

A Tabela 2.1 apresenta a notação usada para a classificação binária. Neste caso, cada linha representa um exemplo onde o vetor de entrada é classificado como spam ou não spam, dependendo das características observadas em $\mathbf{x}$. A variável y indica a categoria, assumindo valores em $\mathcal{L} = \{0, 1\}$, onde 1 representa *spam* e 0 representa *não spam*.

		Binário
Exemplo	Características	$y \in \mathcal{L} = \{0, 1\}$
1	$\mathbf{x}_1$	1
2	$\mathbf{x}_2$	0

Tabela 2.1: Notação de classificação binária

- **Classificação Multi-classe:** É um tipo de classificação em que os dados são atribuídos a uma das várias categorias possíveis. Diferente da classificação binária, aqui o modelo é treinado com várias classes, e o objetivo é reconhecer e categorizar novas instâncias em uma dessas classes. Um exemplo típico é o reconhecimento de imagens, onde o modelo deve classificar uma imagem em uma das várias categorias possíveis, como **pássaro**, **gato** ou **cão** como mostrado na Figura 2.4.

Figura 2.4: Exemplo de classificação multi-classe

A Tabela 2.2 apresenta a notação usada para a classificação multi-classe. Nesse caso, cada linha representa um exemplo onde o vetor de entrada é classificado em uma das várias categorias possíveis, dependendo das características observadas em $\mathbf{x}$. A variável y indica a categoria da classe, assumindo valores em $\mathcal{L} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$, onde cada valor representa uma classe diferente, como **pássaro**, **gato**, **peixes**, **cão** ou **coelho**.

		Multi-classe
Exemplo	Características	$y \in \mathcal{L} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$
1	$\mathbf{x}_1$	λ_2
2	$\mathbf{x}_2$	λ_4
3	$\mathbf{x}_3$	λ_3
4	$\mathbf{x}_4$	λ_1
5	$\mathbf{x}_5$	λ_3

Tabela 2.2: Notação de classificação multi-classe

2.1.4 Classificação Multi-rótulo: Definição e Aplicações

A classificação multi-rótulo é uma evolução dos modelos tradicionais de classificação, desenvolvida para lidar com problemas onde uma instância pode pertencer a múltiplas classes simultaneamente. Essa abordagem difere das classificações binária e multi-classe tradicionais, que assumem que uma instância pertence a apenas uma categoria. No contexto multi-rótulo, uma instância pode estar associada a vários rótulos, o que aumenta a complexidade e requer técnicas específicas para uma análise eficiente [Herrera et al. 2016; Zhang, Zhou e Tsoumakas 2009].

Formalmente, o problema de classificação multi-rótulo é definido como uma extensão dos métodos de classificação tradicionais. Considerando um conjunto de dados $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, onde $\mathbf{x}_i$ representa a instância de entrada e $\mathbf{y}_i = \{y_{i1}, y_{i2}, \ldots, y_{iL}\}$ é um vetor de rótulos binários, em que $y_{ij} = 1$ indica que o rótulo j está presente na instância i, enquanto $y_{ij} = 0$ indica sua ausência [Suthaharan 2016].

Essa definição contrasta com a classificação binária, onde $\mathbf{y}_i$ possui apenas dois valores possíveis, e com a classificação multi-classe, onde $\mathbf{y}_i$ pertence a uma única classe entre várias. A classificação multi-rótulo, portanto, permite a associação de uma instância a múltiplas classes simultaneamente, o que requer modelos específicos para lidar com as correlações e dependências entre rótulos [Herrera et al. 2016].

Figura 2.5: Exemplo de classificação multi-rótulo

A Figura 2.5 exemplifica um caso de classificação multi-rótulo, onde uma imagem pode conter múltiplos rótulos, como *gato* e *cão*, capturando a complexidade do contexto. Em seguida, aprofundaremos na definição formal e detalharemos o funcionamento do classificador multi-rótulo.

Exemplo	Características	Multi-rótulo				
		y_1	y_2	y_3	y_4	$y \subseteq \mathcal{L} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$
1	$\mathbf{x}_1$	1	1	0	1	$\{\lambda_1, \lambda_2, \lambda_4\}$
2	$\mathbf{x}_2$	0	0	0	1	$\{\lambda_4\}$
3	$\mathbf{x}_3$	0	1	1	1	$\{\lambda_2, \lambda_3, \lambda_4\}$
4	$\mathbf{x}_4$	1	0	1	0	$\{\lambda_1, \lambda_3\}$
5	$\mathbf{x}_5$	0	1	1	0	$\{\lambda_2, \lambda_3\}$

Tabela 2.3: Notação de classificação multi-rótulo

A Tabela 2.3 apresenta a notação usada para a classificação multi-rótulo. Nesse caso, cada linha representa um exemplo onde o vetor de entrada é classificado em múltiplas categorias simultaneamente, dependendo das características observadas em $\mathbf{x}$. A variável y indica os rótulos atribuídos a cada exemplo, assumindo um subconjunto de valores em $\mathcal{L} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$. Isso permite que um único exemplo pertença a várias classes ao mesmo tempo, como *pássaro*, *gato*, *peixes*, *cão* ou *coelho*, conforme mostrado na Tabela 2.3.

Definição 2. *(Classificador Multi-Rótulo) Seja $\mathcal{X} = \mathbb{R}^A$ o espaço de características e $\mathcal{Y} = \mathbb{R}^L$ o espaço de rótulos. Dado o conjunto de treinamento $\mathcal{D}$, o objetivo é encontrar uma função $f_p : \mathcal{X} \to \mathcal{Y}$ chamado de classificador multi-rótulo que, para uma nova instância x, preverá o vetor de rótulos apropriados.*

Na definição de classificador multi-rótulo o parâmetro p representa os parâmetros de aprendizagem do modelo f_p. A seguir esse processo de aprendizagem é descrito.

Processo de Treinamento de um Classificador Multi-rótulo

O processo de treinamento de um classificador multi-rótulo pode ser dividido em várias etapas fundamentais. A imagem a seguir ilustra essas etapas, desde a organização do conjunto de dados até a classificação final de uma nova instância:

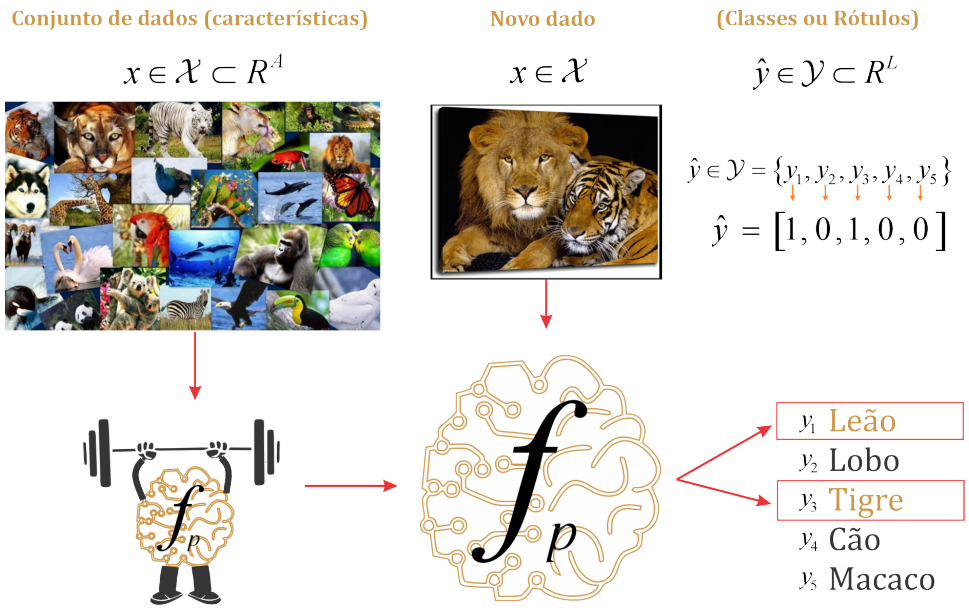


Figura 2.6: Esquema do Processo de Treinamento e Predição em Classificação Multi-rótulo

- **Conjunto de Dados e Pré-processamento**:O processo começa com a organização do **conjunto de dados**, representado na Figura 2.6 à esquerda. Cada instância $x \in \mathcal{X} \subset \mathbb{R}^A$ representa uma imagem caracterizada por um conjunto

de atributos ou características. Para melhorar a qualidade do treinamento, são aplicadas etapas de pré-processamento, como:

  - **Normalização e Padronização**: Garantir que os atributos tenham escalas comparáveis.
  - **Divisão do Conjunto de Dados**: Separar em conjuntos de treinamento, validação e teste, assegurando que o modelo seja avaliado em dados não vistos.

- **Inicialização e Ajuste de Parâmetros**: Após o pré-processamento, o modelo é inicializado e os hiperparâmetros são ajustados. Isso envolve:

  - **Inicialização dos Parâmetros do Modelo**: Definir valores iniciais para os pesos ou parâmetros específicos do modelo (ex.: pesos iniciais para redes neurais).
  - **Ajuste de Hiperparâmetros**: Configurar valores como taxa de aprendizado ou regularização usando métodos como *grid search*.

- **Iteração de Treinamento (Aprendizado**: O classificador f_p, representado na Figura 2.6 pela **mente** em treinamento, aprende a mapear as instâncias x para seus respectivos multi-rótulos $\hat{y}$:

$$f_p(x) = \hat{y} \in \mathcal{Y} \subset \mathbb{R}^L$$

- **Durante o treinamento**:

  - **Função de Custo**: Calcula-se o erro entre as previsões do modelo e os valores reais, guiando o ajuste dos parâmetros.
  - **Retropropagação e Atualização de Parâmetros**: Ajuste dos pesos para minimizar o erro, usando algoritmos como retropropagação para redes neurais.

- **Predição em Novos Dados**: Após o treinamento, o modelo é aplicado a novos dados, como ilustrado na parte central da Figura 2.6. Ao receber uma nova instância x, o classificador gera uma predição multi-rótulo $\hat{y}$, onde $\hat{y} = [1, 0, 1, 0, 0]$ representa as etiquetas atribuídas à imagem.

- **Interpretação dos Resultados**: Cada valor em $\hat{y}$ indica a presença (1) ou ausência (0) de cada rótulo. Na imagem, os rótulos correspondentes a **Leão** (y_1) e **Tigre** (y_3) são ativados, indicando que o classificador associou a imagem a esses animais. Esse resultado é essencial para contextos onde múltiplas classes podem descrever uma única instância.

- **Validação e Teste**: Para avaliar a performance do modelo em dados reais, é usado um conjunto de teste. Métricas multi-rótulo são aplicadas para verificar a capacidade do modelo de generalizar para novos dados.

Em seguida, exploraremos algumas técnicas comuns para a construção de classificadores multi-rótulo, com foco em abordagens baseadas em transformação de problema e adaptação de algoritmo.

2.1.5 Técnicas Comuns de Classificação Multi-Rótulo

As técnicas para resolver problemas de classificação multi-rótulo, podem ser divididas principalmente em técnicas de *transformação de problema* e *adaptação de algoritmos* [Herrera et al. 2016]. Essas abordagens ajudam a resolver os desafios de associar múltiplos rótulos a uma mesma instância, seja transformando o problema em tarefas mais simples ou adaptando algoritmos existentes para lidar diretamente com múltiplos rótulos.

Transformação de Problema: Converte a classificação multi-rótulo em outras formas de aprendizado que podem ser resolvidas por métodos de classificação binária tradicionais. Os principais métodos de transformação de problema incluem:

- **BR (Binary Relevance)**: Trata cada rótulo como um problema separado de classificação binária. O método proposto por [Łęski 2002] utiliza uma abordagem ε-insensível para melhorar a generalização ao resolver sistemas de desigualdades lineares.
- **CC (Classifier Chains)** : Também divide o problema em múltiplas classificações binárias, mas considera a dependência entre rótulos ao usar o resultado da previsão de cada rótulo como entrada para o próximo rótulo. Essa abordagem foi introduzida por [Read et al. 2011] e permite capturar a correlação entre rótulos, melhorando o desempenho.
- **ML-KNN (Multi-Label k-Nearest Neighbor)**: Um método baseado em vizinhos mais próximos, que prevê rótulos para uma nova instância com base nos k vizinhos mais próximos. Essa técnica clássica foi adaptada para multi-rótulo por [Zhang e Zhi-Hua 2007].
- **MLSF (Multi-Label Selection of Features)**: O método MLSF combina aprendizado de meta-rótulos com um processo de seleção de características, considerando as correlações entre rótulos para identificar características relevantes e melhorar a precisão, conforme descrito por [Sun, Kudo e Kimura 2016].

Adaptação de Algoritmos: Nessa abordagem, ajustam-se algoritmos tradicionais para que lidem diretamente com problemas multi-rótulo, sem a necessidade de transformá-los em classificações binárias. Alguns dos métodos mais populares de adaptação incluem:

- **BP-MLL (Backpropagation for Multi-Label Learning)**: Este método foi um dos primeiros a introduzir o aprendizado de correlação entre rótulos, assumindo que prever rótulos relacionados juntos é mais preciso do que prever rótulos isoladamente. Desenvolvido por [Min-Ling e Zhi-Hua 2006], o BP-MLL utiliza redes neurais específicas para aprendizado multi-rótulo.

- **C2AE (Canonical Correlation Analysis Autoencoder)**: Utiliza análise de correlação canônica com uma estrutura de autoencoder para aprender mapeamentos de características que sejam eficazes, conforme descrito por [Yeh et al. 2017]. Esse método é particularmente útil para problemas multi-rótulo complexos.

- **JBNN (Joint Binary Neural Network)**: Em vez de uma função softmax, o JBNN utiliza múltiplas funções logísticas para modelar rótulos diferentes, capturando a correlação entre rótulos por meio de uma função de perda de entropia cruzada binária conjunta [He e Xia 2018].

- **HNOML (Hybrid Noise-tolerant Multi-Label Learning)**: Proposto por [Zhang, Yu et al. 2019], o HNOML aborda tanto o ruído nos rótulos quanto nas características através de regularização bi-espacial e enriquecimento de rótulos, melhorando a robustez e o desempenho em conjuntos de dados com ruídos.

- **ML-TSK FS (Multi-Label Takagi-Sugeno-Kang Fuzzy System)**: Esse modelo aplica o processo de inferência Takagi-Sugeno-Kang (TSK) com base em regras fuzzy, conforme [Lou et al. 2021]. Ele é projetado para aprender as relações entre características e rótulos enquanto minimiza a perda por regressão, proporcionando previsões mais precisas.

Esses métodos representam algumas das técnicas mais proeminentes na classificação multi-rótulo, ilustrando tanto as abordagens de transformação de problema quanto de adaptação de algoritmos. A escolha da técnica depende da complexidade do problema, da correlação entre rótulos e da necessidade de interpretação dos resultados. Em seguida, apresentamos algumas das aplicações onde são usados os classificadores multi-rótulo.

Aplicações da Classificação Multi-rótulo

Na vida real, muitos domínios produzem dados que não podem ser adequadamente descritos por apenas um único rótulo. Em vez disso, uma única instância frequentemente representa múltiplas características ou está associada a múltiplos conceitos simultâneos. Esse tipo de complexidade motiva o uso e o estudo de classificadores multi-rótulo, pois eles são capazes de lidar com situações em que uma instância pode pertencer a várias classes de forma simultânea.

Referência	Domínio	Descrição	Rótulos
[Blockeel, Džeroski e Grbović 1999]	Modelagem ambiental	Predição de qualidade da água em rios com múltiplos parâmetros	Vários parâmetros de qualidade
[Grady e Funka-Lea 2004]	Medicina	Segmentação multi-rótulo de tecidos em imagens médicas	Diferentes tecidos/órgãos
[Boutell et al. 2004]	Imagens	Classificação de cenas com múltiplos objetos	Rótulos como *praia*, *urbano*
[Katakis, Tsoumakas e Vlahavas 2008]	Redes sociais	Detecção de spam em sistemas de marcadores sociais	Rótulos para spam/não spam
[Briggs, Lakshminarayanan et al. 2012]	Áudio	Identificação de espécies de aves em gravações	Até 13 espécies por gravação
[Ratnarajah e Qiu 2014]	Medicina	Segmentação de estruturas cerebrais em neonatos	Diferentes estruturas cerebrais
[Schulz, Mencía e Schmidt 2016]	Ciências sociais	Classificação de incidentes urbanos em tweets	Tipos de incidentes (acidente, incêndio, etc.)
[Haobo Wang et al. 2021]	Comércio	Detecção de fraude em e-commerce	Rótulos: fraude, segurança, risco, autenticação
[Deniz, Erbay e Coşar 2022]	Comércio eletrônico	Classificação de opiniões de clientes	Opiniões positivas, negativas, neutras

Tabela 2.4: Aplicações da classificação multi-rótulo em diferentes domínios

A Tabela 2.4 resume algumas dessas aplicações, destacando a diversidade de domínios onde a classificação multi-rótulo se mostra indispensável. Como é possível observar, esses dados vêm de diversas áreas, como modelagem ambiental, redes sociais, áudio, comércio eletrônico e afins, ilustrando a versatilidade e a importância do paradigma multi-rótulo. Essas aplicações demonstram a diversidade de rótulos atribuídos a uma mesma instância reflete a complexidade dos dados do mundo real,

tornando o paradigma multi-rótulo uma ferramenta essencial para capturar a totalidade de informações presentes em cada observação. A seguir, apresentamos os principais desafios enfrentados pela classificação multi-rótulo.

2.1.6 Desafios da Classificação Multi-rótulo

Os desafios da classificação multi-rótulo devem-se à complexidade intrínseca dos dados e às características únicas deste tipo de problema. Em contraste com a classificação binária ou multi-classe, a classificação multi-rótulo exige que o modelo lide com a possibilidade de que cada instância pertença a múltiplas classes simultaneamente, o que gera questões específicas que precisam ser abordadas para que o modelo seja eficiente. Abaixo estão os principais desafios:

- **Correlações entre Rótulos**: Um dos maiores desafios na classificação multi-rótulo é modelar as correlações entre rótulos, pois, ao contrário da classificação tradicional, os rótulos em um cenário multi-rótulo geralmente não são independentes. Frequentemente, a presença de um rótulo aumenta a probabilidade de ocorrência de outro. Por exemplo, uma música classificada como *rock* pode também ser classificada como *alternativa*, devido às características compartilhadas entre esses gêneros [Herrera et al. 2016].

- **Desequilíbrio de Classes e Dimensionalidade Alta**: Outro desafio recorrente em tarefas de classificação multi-rótulo é o desequilíbrio de classes, que ocorre quando alguns rótulos aparecem muito mais frequentemente que outros no conjunto de dados. Esse problema se agrava em cenários de alta dimensionalidade, onde o número de rótulos é elevado, aumentando a complexidade do modelo e o risco de sobreajuste [Mitchell 1997].

- **Imprecisão, Incerteza e Ambiguidade na Classificação Multi-rótulo**

  Em contextos de classificação multi-rótulo, é comum enfrentar problemas de imprecisão, incerteza e ambiguidade nos dados, que podem comprometer a precisão dos modelos. Esses problemas são próprios dos dados multi-rótulo e do processo de atribuir vários rótulos a uma mesma instância.

  - **Imprecisão**: Ocorre quando os dados de entrada possuem características que podem ser interpretadas de diversas maneiras, dificultando a determinação exata dos rótulos. Esse problema é particularmente relevante em tarefas de análise de sentimentos, onde uma frase pode conter elementos positivos e negativos ao mesmo tempo, gerando interpretações ambíguas [Herrera et al. 2016].
  - **Incerteza**: Refere-se à probabilidade de que uma instância pertença a múltiplos rótulos com diferentes graus de confiança. Para lidar com esse

tipo de incerteza, métodos baseados em teorias de probabilidade e lógica fuzzy são amplamente utilizados, permitindo modelar a associação com cada rótulo e suavizar transições entre rótulos [Suthaharan 2016; Keller et al. 2016].

– **Ambiguidade**: Surge em casos onde há sobreposição de características entre rótulos, dificultando a distinção pelo classificador. Esse é um problema comum em tarefas de classificação de imagens complexas, onde múltiplas categorias podem estar representadas visualmente em uma mesma imagem, criando desafios para a separação de rótulos [Mitchell 1997; Alpaydin 2010].

Esses desafios exigem o uso de técnicas avançadas, como modelos probabilísticos, redes neurais adaptativas e sistemas fuzzy, que são projetados para manejar altos níveis de incerteza e ambiguidade nos dados [Keller et al. 2016]. A seguir, abordaremos os sistemas neuro-fuzzy, uma técnica eficaz para lidar com complexidades e incertezas em classificação multi-rótulo.

2.2 SISTEMAS NEURO-FUZZY

Os Sistemas Neuro-Fuzzy (SNF) combinam as capacidades das Redes Neurais Artificiais (RNA) e dos Sistemas de Inferência Fuzzy (SIF). Essa integração busca melhorar a habilidade de aprendizado das redes neurais com a interpretabilidade dos sistemas fuzzy, criando modelos que podem lidar com dados complexos e adaptar-se a mudanças. [Kosko 1992; Jang, Jyh-Shing et al. 1997; Chen e Pham 2000].

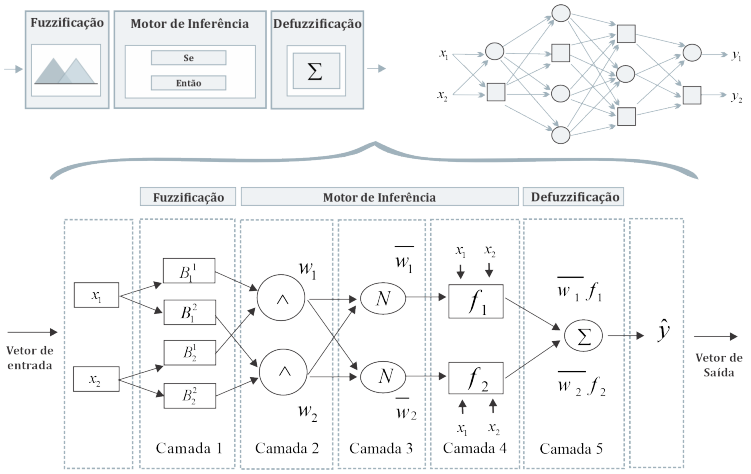


Figura 2.7: Arquitetura de um Sistema Neuro-fuzzy

A Figura 2.7 ilustra a arquitetura de um sistema neuro-fuzzy, onde as camadas das redes neurais interagem com o sistema de inferência fuzzy para criar um sistema

híbrido que combina aprendizado e interpretabilidade. Nas próximas seções, exploraremos mais profundamente as Redes Neurais Artificiais e os Sistemas de Inferência Fuzzy.

2.2.1 Fundamentos de Redes Neurais Artificiais (RNA)

As Redes Neurais Artificiais (RNA) são inspiradas no funcionamento do cérebro humano e surgiram com os trabalhos pioneiros de Fitch e Pitts [Fitch 1944]. Compostas por neurônios artificiais conectados por meio de sinapses ponderadas, as RNA têm a capacidade de aprender padrões complexos a partir de grandes conjuntos de dados, sem necessidade de programação explícita para cada tarefa.

Componentes Fundamentais das RNA

As redes neurais utilizam componentes fundamentais que são cruciais para sua capacidade de aprendizado e adaptação. Entre esses componentes temos, o neurônio biológico, o bias e a função de ativação.

- **Neurônio Biológico e Artificial**: Um neurônio artificial, assim como um neurônio biológico, processa entradas e gera uma saída com base em uma função de ativação.

Figura 2.8: Comparativo entre o neurônio biológico e o neurônio matemático.

A Figura 2.8 destaca a analogia entre o neurônio biológico e o neurônio matemático, mostrando como entradas ponderadas são processadas por uma função de ativação para gerar uma saída.

Os componentes principais de um neurônio matemático incluem:

- $\boldsymbol{x}$: Vetor de entradas, onde cada componente x_i representa uma característica.
- $\boldsymbol{w}$: Vetor de pesos, onde cada w_i indica a importância da entrada correspondente.

- $\sum$**:** Soma ponderada das entradas.
- **f:** Função de ativação.
- **b:** Bias, que ajusta a entrada da função de ativação.

A operação do neurônio pode ser descrita pelas seguintes equações [Haykin 2001, p.37]:

$$u_k = \sum_{i=1}^{n} x_i w_{ki}$$

$$y_k = f\left(u_k + b_k\right)$$

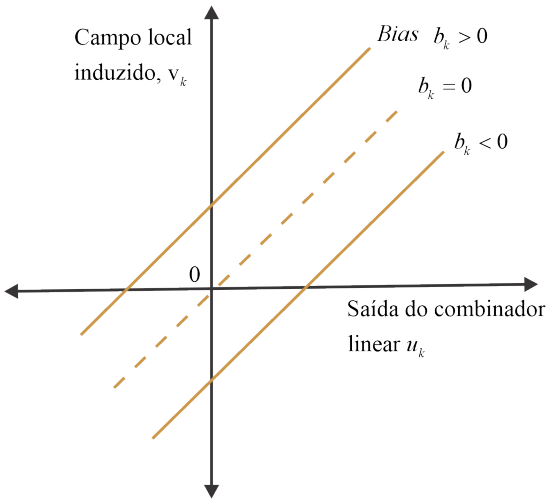


Figura 2.9: Efeito do bias sobre o campo local induzido.

A Figura 2.9 ilustra o impacto do bias: valores negativos diminuem o potencial de ativação, enquanto valores positivos aumentam, ajustando a função de ativação para otimizar a resposta do neurônio.

- **Funções de Ativação**: São elementos essenciais em redes neurais, pois introduzem não-linearidade nas saídas dos neurônios, permitindo que a rede capture padrões complexos nos dados. A escolha da função de ativação depende do tipo de rede neural e do problema específico. A Tabela 2.5 resume as principais funções de ativação, suas fórmulas, redes neurais associadas, e gráficos que ilustram o comportamento de cada função.

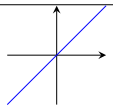
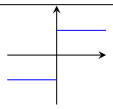
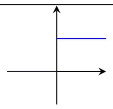
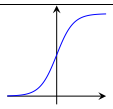
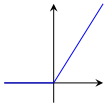
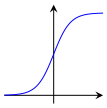
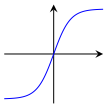
Tipo	**Função $f(s)$**	**Rede Neural**	**Gráfico**
Linear	$f(s) = s$	Hopfield, BSB	
Sinal	$f(s) = \begin{cases} +1 & \text{se } s \geq 0 \\ -1 & \text{se } s < 0 \end{cases}$	Perceptron	
Degrau	$f(s) = \begin{cases} +1 & \text{se } s \geq 0 \\ 0 & \text{se } s < 0 \end{cases}$	Perceptron, BAM	
Sigmoidal	$f(s) = \frac{1}{1+e^{-s}}$	Perceptron, redes profundas	
ReLU	$f(s) = \begin{cases} s & \text{se } s \geq 0 \\ 0 & \text{se } s < 0 \end{cases}$	Redes Convolucionais, redes profundas	
Softmax	$f(s_i) = \frac{e^{s_i}}{\sum_j e^{s_j}}$	Classificação multi-classe, camada de saída de redes profundas	
Tangente Hiperbólica	$f(s) = \tanh(s) = \frac{1-e^{-2s}}{1+e^{-2s}}$	Perceptron, Hopfield, BAM, BSB	

Tabela 2.5: Funções de Ativação e Redes Neurais

Arquitetura de RNA

Uma rede neural é composta por múltiplas camadas: camada de entrada, camadas ocultas e camada de saída, cada uma desempenhando um papel essencial no processamento dos dados. Essas camadas determinam a complexidade e a capacidade de generalização da rede.

Entre as arquiteturas mais conhecidas estão:

- **Perceptron Simples:** O perceptron simples é o modelo mais básico de uma rede neural, composto por uma única camada de nós de entrada conectados diretamente a uma saída. Ele aplica uma função de ativação ao resultado de uma combinação linear ponderada das entradas. Embora seja eficiente para problemas linearmente separáveis, suas limitações aparecem ao lidar com dados que não seguem essa característica.

- **Perceptron Multicamadas (MLP - Multi-Layer Perceptron):** O perceptron multicamadas é uma extensão do perceptron simples, incorporando uma ou mais camadas ocultas entre as entradas e a saída. Com funções de

ativação não lineares, como ReLU ou sigmóide, ele pode aprender relações complexas e não lineares nos dados. Essa arquitetura é amplamente usada em tarefas de classificação, regressão e previsão.

- **Redes Neurais Feedforward (FFNN)** As redes neurais feedforward generalizam o perceptron multicamadas, permitindo o fluxo unidirecional da informação, da entrada até a saída. Elas são simples de treinar e entender, sendo amplamente aplicadas em problemas de previsão, classificação e modelagem de dados estáticos. Por sua simplicidade estrutural, são a base para redes mais avançadas.

- **Redes com Funções de Base Radial (RBF):** As redes de função de base radial utilizam funções de base radial como ativação para modelar dados não lineares. Essas funções medem a distância entre os dados de entrada e um ponto central, proporcionando flexibilidade na classificação e regressão. São ideais para problemas onde é necessária uma interpolação precisa ou a aproximação de funções complexas, sendo amplamente aplicadas em aprendizado supervisionado.

- **Redes Convolucionais (CNN - Convolutional Neural Networks):** As CNNs são especializadas no processamento de dados com estrutura espacial, como imagens e vídeos. Utilizam camadas convolucionais para extrair características locais e camadas de pooling para reduzir a dimensionalidade, preservando informações essenciais. Elas são amplamente usadas em visão computacional, como no reconhecimento de imagens e na detecção de objetos.

- **Redes Recorrentes (RNN - Recurrent Neural Networks):** As RNNs são projetadas para lidar com sequências temporais ou dados ordenados, utilizando conexões que introduzem memória nas iterações. Variedades como LSTM e GRU resolvem problemas de longo prazo, como o desaparecimento do gradiente em sequências longas. Essas redes são amplamente aplicadas em tradução automática, séries temporais e processamento de linguagem natural.

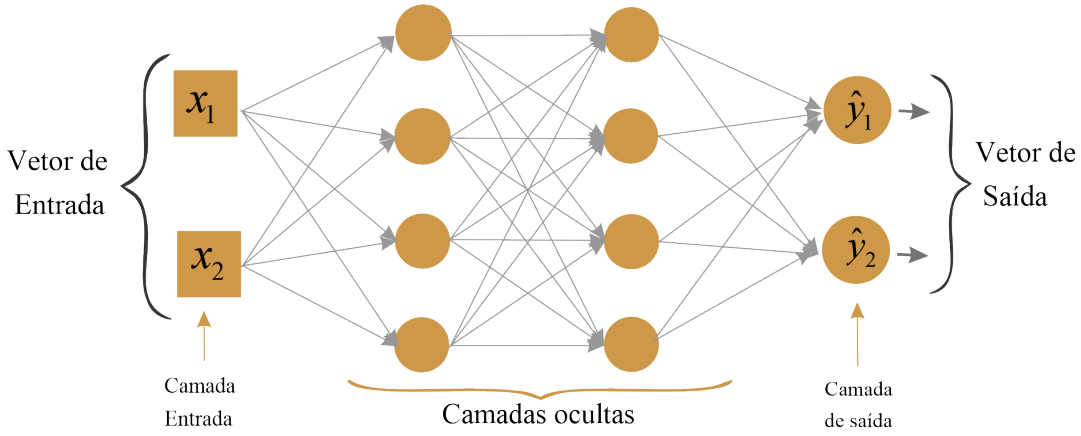


Figura 2.10: Arquitetura de uma Rede Perceptron Multicamadas.

A Figura 2.10 exemplifica uma arquitetura MLP, onde várias camadas ocultas ajudam a capturar padrões complexos em dados para tarefas de classificação.

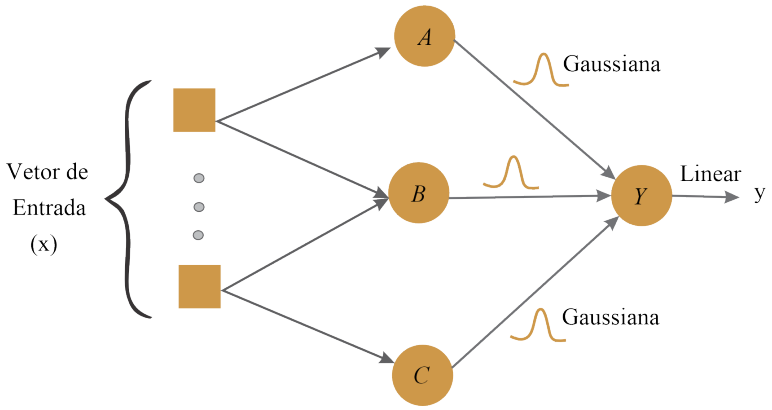


Figura 2.11: Arquitetura de uma Rede com Funções de Base Radial.

A Figura 2.11 apresenta uma rede RBF, usada principalmente para aproximação de funções e problemas de classificação onde a posição dos dados no espaço de entrada é relevante.

Aprendizado de RNA

O aprendizado consiste em ajustar os pesos das conexões com base nos dados de entrada, o que pode resultar na criação, modificação ou remoção dessas conexões.. Quando o peso de uma conexão é zero, ela é considerada inexistente. O aprendizado continua até que esses pesos se estabilizem, indicando que a rede conseguiu *aprender* ou captar o padrão desejado. Cada rede possui critérios próprios para ajustar esses pesos, conhecidos como *regras de aprendizado*, que definem o tipo de aprendizado: supervisionado ou não supervisionado e por reforço.

- **Supervisionado:** A rede é treinada com pares de entrada e saída conhecidos.

- **Não Supervisionado:** A rede identifica padrões por si mesma, ajustando-se de forma autônoma aos dados disponíveis.
- **Por Reforço:** A rede ajusta-se com base em recompensas e penalidades.

Algoritmo de treinamento

Os algoritmos de treinamento para redes neurais podem ser classificados em duas categorias principais: algoritmos supervisionados e algoritmos não supervisionados, cada um com objetivos e métodos distintos que refletem a forma como a rede processa e aprende com os dados. Dentro dos algoritmos de aprendizado supervisionado, o algoritmo de retropropagação é o método mais amplamente utilizado para treinar redes neurais.

- **Algoritmo de Retropropagação (backpropagation)**: O algoritmo de retropropagação permite que a rede aprenda de maneira eficiente ajustando os pesos camada por camada, a partir da minimização do erro com base no método do gradiente.
  - **Correção de erro**: a rede ajusta os pesos imediatamente após a apresentação de cada padrão de entrada, corrigindo o erro na saída de forma direta e pontual. Esse método oferece um ajuste responsivo, pois a rede corrige seus parâmetros constantemente.

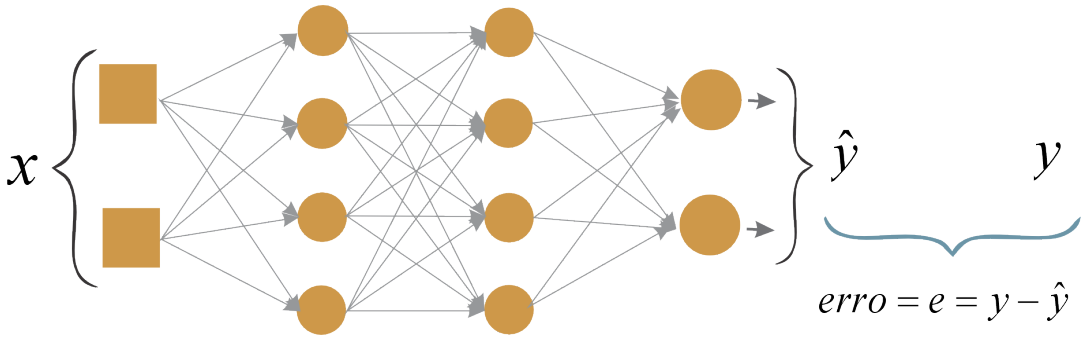


Figura 2.12: Correção de erro

  - **Método do gradiente**: os pesos são atualizados para minimizar o erro quadrático médio, considerando todos os padrões de entrada. Esse processo de minimização leva os pesos na direção oposta ao gradiente da função de erro, resultando em uma abordagem mais estável e robusta, que visa otimizar a precisão global da rede ao longo de todo o conjunto de dados.

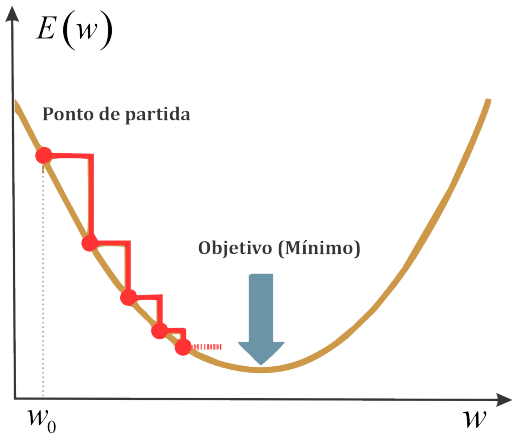


Figura 2.13: Gradiente descendente

Passos da Retropropagação

1. **Inicialização dos Pesos**:
   - Os pesos w_{ij} são inicializados aleatoriamente para cada conexão entre os neurônios.
2. **Propagação Direta (Forward Pass)**:
   - Calcula-se a saída y_j de cada neurônio aplicando a função de ativação σ à soma ponderada das entradas:

$$z_j = \sum_i w_{ij} \cdot x_i + b_j$$

$$y_j = \sigma(z_j)$$

3. **Cálculo do Erro**:
   - O erro total E é calculado comparando a saída prevista $\hat{y}$ com a saída desejada y, usando uma função de perda (exemplo: erro quadrático):

$$E = \frac{1}{2} \sum_k (y_k - \hat{y}_k)^2$$

4. **Retropropagação do Erro (Passo Reverso)**:
   - Calcula-se o gradiente do erro em relação a cada peso. Para o neurônio de saída k:

$$\delta_k = (y_k - \hat{y}_k) \cdot \sigma'(z_k)$$

   - Propaga-se esse erro para ajustar cada peso das camadas anteriores.
5. **Atualização dos Pesos**:

- Ajusta-se os pesos w_{ij} usando a taxa de aprendizado η:

$$w_{ij} \leftarrow w_{ij} - \eta \cdot \delta_j \cdot y_i$$

6. **Repetição**:

   - Repete-se o processo para todas as amostras até que o erro seja minimizado ou satisfaça um critério de parada.

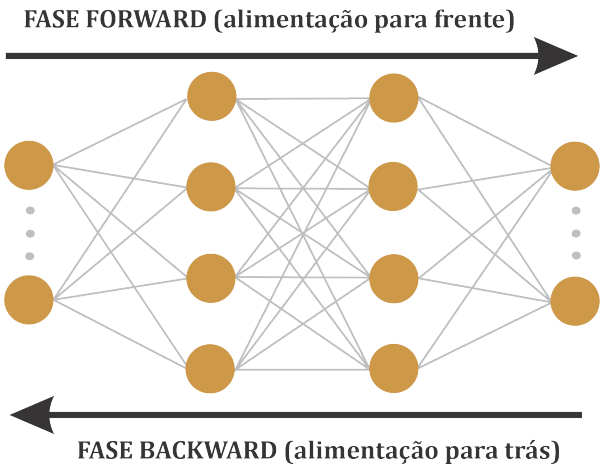


Figura 2.14: Algoritmo de Retropropagação.

A Figura 2.14 mostra o fluxo do algoritmo de retropropagação, onde o erro é propagado de volta na rede para ajustar os pesos e melhorar a precisão.

Estruturas das Redes Neurais Artificiais

As redes neurais podem ser classificadas pela arquitetura e pelo método de aprendizado [Haykin 2001]. As principais arquiteturas são:

- **Feedforward (Alimentação Direta):** Nesta arquitetura, o sinal é propagado em uma única direção, da camada de entrada até a camada de saída, sem ciclos ou feedback.
- **Rede Recorrente (RNN):** Diferente das redes feedforward, as redes recorrentes possuem conexões de feedback, permitindo que informações de passos anteriores influenciem os estados futuros da rede.
- **Rede Convolucional (CNN):** Utilizadas para análise de imagens e dados com estrutura espacial.
- **Rede de Funções de Base Radial (RBF):** Com funções de ativação radial, aplicáveis a problemas de classificação e regressão.

Vantagens das RNAs

- **Capacidade de Aprendizado Adaptativo:** As redes neurais possuem uma habilidade notável de aprendizado adaptativo, ajustando-se a partir de grandes volumes de dados para melhorar o desempenho de tarefas específicas sem necessidade de reprogramação manual para cada caso [Haykin 2001].
- **Reconhecimento de Padrões Complexos:** São especialmente eficazes em tarefas como reconhecimento de padrões e classificação, onde conseguem identificar correlações complexas e sutis nos dados que métodos tradicionais não conseguem capturar [Oliveira 2007].
- **Robustez e Tolerância a Falhas:** A arquitetura distribuída das redes neurais confere a elas uma alta tolerância a falhas, mantendo um bom desempenho mesmo quando algumas conexões ou neurônios apresentam problemas [Jang e Jyh-Shing 1993].
- **Versatilidade em Diferentes Aplicações:** São amplamente utilizadas em áreas como visão computacional, processamento de linguagem natural e sistemas de recomendação, provando-se versáteis para diversas aplicações de inteligência artificial [Kasabov 2002].
- **Capacidade de Generalização:** Redes neurais são capazes de generalizar bem ao serem expostas a novos dados, o que as torna ideais para aplicações em ambientes dinâmicos e incertos, onde os dados estão em constante mudança [Lee e Lin 1991].
- **Integração com Outros Sistemas Inteligentes:** Combinadas com a lógica fuzzy, as redes neurais são capazes de formar sistemas neuro-difusos, unindo a capacidade de aprendizado adaptativo com a habilidade de lidar com incertezas e imprecisões em dados reais [Zadeh 1965].

Com a compreensão das estruturas das redes neurais artificiais, seguimos agora para a próxima seção, onde exploraremos os fundamentos dos Sistemas de Inferência Fuzzy.

2.2.2 Fundamentos dos Sistemas de Inferência Fuzzy (SIF)

Os sistemas de inferência fuzzy são amplamente reconhecidos por sua capacidade de lidar com incertezas e complexidades em uma vasta gama de aplicações, como controle de processos, modelagem e previsão. Ao contrário dos sistemas tradicionais, que operam com limites rígidos e definições exatas, os sistemas fuzzy utilizam graus de verdade para representar informações, proporcionando uma abordagem mais flexível e adequada ao mundo real [Chen e Pham 2000]. Essa flexibilidade permite

que os sistemas fuzzy capturem nuances e variações sutis, tornando-se valiosos em contextos onde a incerteza e a gradualidade das transições prevalecem [Ross 2004; Cox 1994; Keller et al. 2016].

Componentes Fundamentais dos SIF

Para compreender adequadamente os sistemas de inferência fuzzy, é essencial começar pela definição de conceitos fundamentais como:

- **Conjuntos Fuzzy e Função de Pertinência** A teoria dos conjuntos fuzzy foi introduzida por Lotfi Zadeh em 1965, para lidar com a incerteza e a vaguidade inerentes a muitos conceitos do mundo real. Ao contrário dos conjuntos clássicos, onde um elemento pertence totalmente ou não ao conjunto, os conjuntos fuzzy permitem que os elementos tenham graus variáveis de pertinência, que variam entre 0 e 1, capturando nuances e transições graduais que são comuns em termos linguísticos, como *temperatura agradável* ou *altura moderada* [Zadeh 1965].

  Definição 3. *(Conjunto Fuzzy) Um* ***conjunto fuzzy*** *A em um universo X é representado por:*

  $$A = \{(x, \mu_A(x)) \mid x \in X\}$$

  onde $\mu_A : X \to [0, 1]$ é a ***função de pertinência*** *que associa a cada elemento $x \in X$ um grau de pertencimento $\mu_A(x)$, que varia entre 0 e 1. Esse valor indica o quão fortemente o elemento x pertence ao conjunto A, de acordo com os critérios a seguir: $\mu_A(x) = 0$: x não pertence ao conjunto. $\mu_A(x) = 1$: x pertence completamente ao conjunto. $0 < \mu_A(x) < 1$: x pertence parcialmente ao conjunto, com o grau de pertencimento proporcional ao valor de $\mu_A(x)$.*

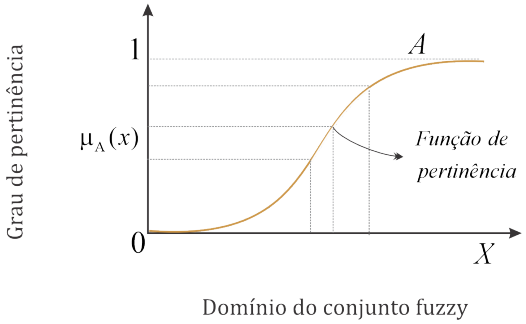


Figura 2.15: Componentes de um Conjunto Fuzzy

A Figura 2.15 ilustra a definição de um **conjunto fuzzy**. A função de pertinência $\mu_A(x)$ associada ao conjunto fuzzy A atribui a cada elemento x no domínio X um grau de pertinência entre 0 e 1. Na imagem, vemos como a

função de pertinência varia gradualmente, refletindo a transição suave de não pertencimento para pertencimento total.

- **Tipos de funções de pertinência**: Existem diferentes tipos de funções de pertinência na Figura 2.16, são apresentados três formatos comuns de funções de pertinência. A função triangular é simples e fácil de interpretar, enquanto a trapezoidal é útil para representar intervalos mais amplos de pertinência. A função gaussiana, por sua vez, oferece uma transição mais suave entre os diferentes graus de pertinência.

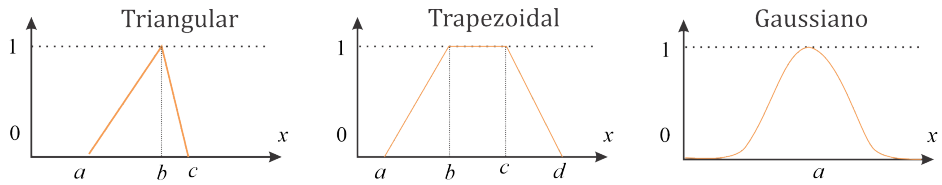


Figura 2.16: Funções de Pertinência: Triangular, Trapezoidal e Gaussiana.

Funções de Pertinência	Fórmulas
Triangular:	$\mu(x) = \begin{cases} \frac{x-a}{b-a}, & \text{se } x \in (a,b] \\ \frac{c-x}{c-b}, & \text{se } x \in (b,c] \\ 0, & \text{outro caso} \end{cases}$
Trapezoidal:	$\mu(x) = \begin{cases} \frac{x-a}{b-a}, & \text{se } x \in (a,b] \\ 1, & \text{se } x \in (b,c] \\ \frac{d-x}{d-c}, & \text{se } x \in (c,d] \\ 0, & \text{outro caso} \end{cases}$
Gaussiana:	$\mu(x) = \exp\left\{-\frac{(x-a)^2}{2\sigma^2}\right\}$

Tabela 2.6: Fórmulas de Funções de Pertinência

Na Tabela 2.6 são apresentadas as fórmulas matemáticas das funções de pertinência da Figura 2.16. Essas funções de pertinência podem ser escolhidas de acordo com o nível de precisão desejado, a simplicidade do modelo e as características do problema a ser modelado. São amplamente utilizadas para lidar com a incerteza em sistemas fuzzy, permitindo uma representação flexível de situações complexas [Ross 2004].

- **Variável Linguística e Termo Linguístico**
  Uma variável linguística x, definida no universo X, é uma variável cujos valores são subconjuntos fuzzy de X. Zadeh [Zadeh 1975] definiu uma variável linguística como uma variável cujos valores são expressos em termos de palavras

ou frases em uma linguagem natural ou artificial, permitindo a representação de conceitos subjetivos ou ambíguos de forma quantitativa.
A vantagem do uso de variáveis linguísticas é que elas permitem descrever fenômenos complexos de maneira intuitiva e flexível, utilizando termos linguísticos que refletem a forma como os humanos interpretam e compreendem a realidade [Ross 2004].

Exemplo: A variável *temperatura* pode assumir termos linguísticos como *baixa*, *média* e *alta* ver Figura 2.17, cada um desses valores linguísticos corresponde a um subconjunto fuzzy do universo de discurso, que, para essa variável, poderia ser o intervalo de temperaturas entre 0°C e 40°C.

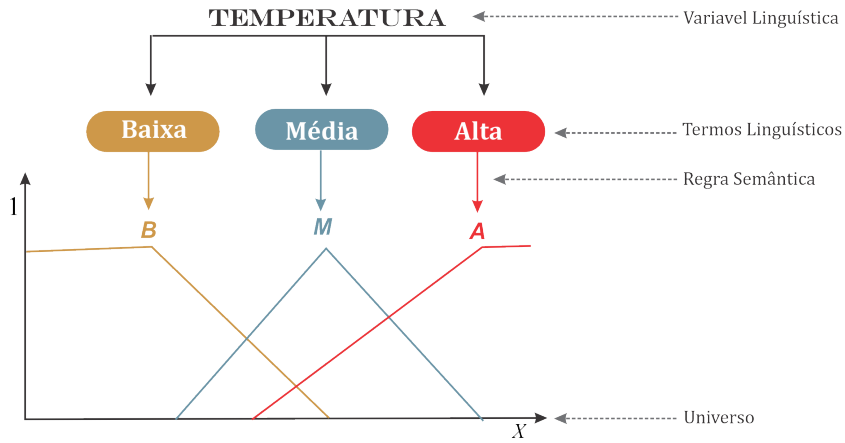


Figura 2.17: Variáveis Linguísticas e Termos Linguísticos.

A Figura 2.17, mostra como a *variável linguística Temperatura* é descrita por valores linguísticos: *Baixa*, *Média* e *Alta*. Cada *termo linguístico* tem uma função de pertinência que determina o grau de associação de qualquer valor de temperatura a essas categorias qualitativas, permitindo que o sistema trabalhe com descrições interpretáveis.

- **Lógica Fuzzy**

  A lógica fuzzy é uma extensão da lógica clássica desenvolvida para lidar com incertezas e imprecisões ao invés de valores binários estritos de verdadeiro e falso. Introduzida por [Zadeh 1975] , a lógica fuzzy permite graus intermediários de verdade, onde proposições podem ter valores contínuos entre 0 e 1. Esse modelo é particularmente útil em situações onde os conceitos não são estritamente definidos, permitindo uma representação mais flexível e adaptável de fenômenos complexos [Chen e Pham 2000].
  Com a lógica fuzzy, é possível representar condições e inferências utilizando variáveis linguísticas e conjuntos fuzzy, criando assim um sistema que pode

lidar com informações imprecisas. Para isso, são usados operadores como a conjunção (AND), a disjunção (OR) e o complemento (NOT) que permitem combinar condições de forma gradual atribuindo graus de verdade entre 0 e 1.

Operador	Fórmula Fuzzy	Descrição
Conjunção (AND)	$\mu_{A\wedge B}(x) = \min(\mu_A(x), \mu_B(x))$ ou $\mu_{A\wedge B}(x) = \mu_A(x) \cdot \mu_B(x)$	Combina condições usando o menor grau ou produto dos valores.
Disjunção (OR)	$\mu_{A\vee B}(x) = \max(\mu_A(x), \mu_B(x))$ ou $\mu_{A\vee B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$	Permite combinar alternativas tomando o maior grau ou soma.
Complemento (NOT)	$\mu_{\neg A}(x) = 1 - \mu_A(x)$	Calcula o grau de não-pertinência de uma condição fuzzy.

Tabela 2.7: Operadores Fuzzy e suas Fórmulas

Esses operadores são essenciais para construir inferências flexíveis: a conjunção permite combinar condições que devem ocorrer simultaneamente a disjunção considera alternativas e o complemento reflete a ausência ou negação de uma condição. A lógica fuzzy, forma a base para a criação de regras fuzzy e é fundamental na modelagem de sistemas que operam sob condições de incerteza e imprecisão [Zadeh 1975; Ross 2004].

- **Regras Fuzzy e o Raciocínio Se-Então**: Uma vez estabelecidos os conjuntos fuzzy e as variáveis linguísticas, o próximo passo é estruturar o conhecimento em **regras fuzzy** do tipo "se-então". As regras fuzzy são utilizadas para capturar o raciocínio humano de maneira formalizada, conectando condições a ações ou respostas específicas. Cada regra segue a estrutura básica:

  ***Se** (condição), **Então** (ação ou consequência).*

  Por exemplo, uma regra fuzzy poderia ser:

  ***Se** a temperatura é alta e a umidade é baixa, **então** aumentar o fluxo de ar.*

  Essa regra utiliza variáveis linguísticas e seus respectivos conjuntos fuzzy para descrever uma condição antecedente e associá-la a uma ação consequente.

  - O antecedente (parte ***se***) especifica as condições que devem ser avaliadas.
  - O consequente (parte ***então***) define a ação a ser executada caso as condições sejam atendidas.

  Cada regra fuzzy é ativada em graus proporcionais ao grau de pertencimento das variáveis de entrada aos conjuntos fuzzy especificados no antecedente da regra. Esse grau de ativação permite que o sistema fuzzy interprete situações

complexas e tome decisões de forma gradativa e adaptativa, o que é especialmente útil em sistemas de controle e decisão com informações imprecisas [Cox 1994; Kosko 1992].

- **Agregação dos Antecedentes e Consequentes nas Regras Fuzzy**

  A agregação dos antecedentes e consequentes em regras fuzzy é crucial para o processo de inferência fuzzy, pois permite combinar múltiplas condições e respostas de maneira coerente.

  - **Agregação dos Antecedentes**: A combinação dos antecedentes determina o grau de ativação da regra, dependendo da satisfação das condições de entrada. Operadores como o *mínimo* e o *produto* são amplamente usados para calcular essa ativação, com o mínimo exigindo que todas as condições sejam cumpridas ao menor grau, enquanto o produto proporciona uma ativação proporcional às condições [Uebele 1995; Bezdek 1999].
  - **Agregação dos Consequentes**: A combinação dos consequentes permite integrar as saídas de múltiplas regras para obter uma resposta única. Métodos como a média ponderada garantem que as regras mais fortemente ativadas tenham maior influência na saída final [Bouchon-Meunier 1998].

  A agregação dos antecedentes e consequentes ajuda os sistemas fuzzy a responderem de forma adaptativa e precisa em cenários com incertezas e interdependência.

Sistema de Inferência Fuzzy (SIF)

O sistema de inferência fuzzy é uma estrutura que transforma entradas numéricas em conjuntos fuzzy (fuzzificação), aplica regras do tipo SE-ENTÃO para relacionar essas entradas às saídas, combina os resultados das regras (agregação) e, por fim, converte o resultado em um valor concreto (defuzzificação). Esse processo é especialmente útil em problemas onde os dados são incertos ou vagos, como no controle de processos industriais e na tomada de decisão em sistemas complexos [Ross 2004].
Os modelos de inferência fuzzy de Mamdani (criado por Ebrahim Mamdani em 1975) e Takagi-Sugeno-Kang (TSK) (desenvolvido por Takagi e Sugeno em 1985) são os mais utilizados em sistemas fuzzy. A principal diferença entre os modelos de Mamdani e TSK está na forma dos consequentes das regras no processo de saída. O modelo Mamdani gera uma saída fuzzy que precisa ser defuzzificada, o que facilita a interpretabilidade, sendo ideal para aplicações que requerem ajustes manuais, como o controle industrial tradicional. Já o modelo TSK não precisa ser defuzzificada, pois produz uma saída numérica direta via funções consequentes, é mais eficiente

e preciso, tornando-se preferido para sistemas que exigem resposta rápida e alta precisão [Ruspini e Patrone 1998; Ross 2004]. A Figura 2.18 ilustra o fluxo completo do sistema de inferência fuzzy, dividindo-o em três operações principais. Também, podemos observar duas abordagens de saída para sistemas fuzzy: a saída de tipo Mamdani (parte superior) e a saída TSK (parte inferior). A seguir, descrevemos as etapas do processamento no modelo TSK, com base na Figura 2.18 apresentado.

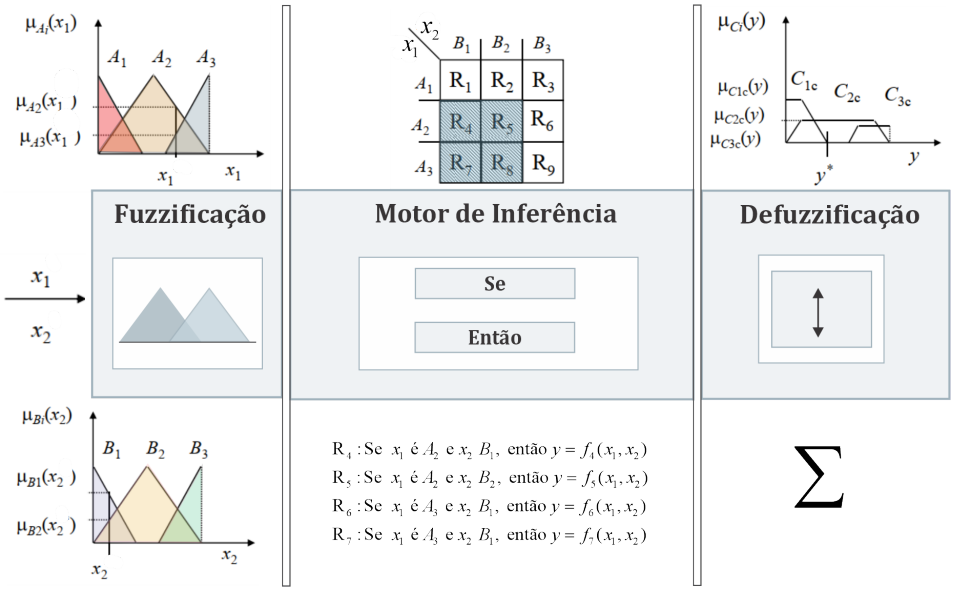


Figura 2.18: Sistema de Inferência Fuzzy.

No modelo TSK, os sistemas fuzzy produzem saídas **nítidas (crisp)** através de uma combinação ponderada dos consequentes das regras. Cada regra gera uma saída numérica diretamente, eliminando a necessidade de defuzzificação.

- **Fuzzificação**: Na etapa de fuzzificação, os valores de entrada x_1 e x_2 são convertidos em graus de pertinência para os conjuntos fuzzy correspondentes. Esses conjuntos fuzzy são representados por suas funções de pertinência, $\mu_{A_i}(x_1)$ para a variável x_1 e $\mu_{B_j}(x_2)$ para a variável x_2. Cada valor de entrada é associado a um grau de pertencimento para diferentes conjuntos fuzzy, por exemplo, A_1, A_2, A_3 para x_1; B_1, B_2, B_3 para x_2, conforme ilustrado na imagem.

$$\mu_{A_i}(x_1) \quad \text{e} \quad \mu_{B_j}(x_2)$$

- **Motor de Inferência**: No modelo TSK, o motor de inferência é composto por duas sub-etapas principais: a avaliação das regras e a agregação dos resultados.

  - **Avaliação das Regras**: Cada regra fuzzy é ativada de acordo com os graus de pertencimento dos antecedentes. No modelo TSK, cada regra R_k é expressa como ***Se*** x_1 é A_i e x_2 é B_j, ***então*** $y = f_k(x_1, x_2)$, onde

$f_k(x_1, x_2)$ é uma função linear ou constante das variáveis de entrada. Por exemplo, uma regra específica poderia ser:

$$R_4 : \text{Se } x_1 \text{ é } A_2 \text{ e } x_2 \text{ é } B_1, \text{ então } y = f_4(x_1, x_2) = p_4 x_1 + q_4 x_2 + r_4$$

O grau de ativação da regra R_k é determinado pelos operadores mínimo ou produto que são generalizações da interseção clássica, que combina os graus de pertinência dos antecedentes:

$$\text{Ativação de } R_k = \min(\mu_{A_i}(x_1), \mu_{B_j}(x_2)) \quad \text{ou} \quad \prod(\mu_{A_i}(x_1), \mu_{B_j}(x_2))$$

– **Agregação dos Resultados**: Diferente do modelo de Mamdani, onde a saída é um conjunto fuzzy, no modelo TSK cada regra gera uma saída nítida $f_k(x_1, x_2)$ que será agregada na saída final.

- **Saída Final**: A saída final do sistema é calculada por uma média ponderada das saídas das regras, ponderadas pelo grau de ativação de cada regra. A saída final y é dada por:

$$y = \frac{\sum_k \text{Ativação de } R_k \cdot f_k(x_1, x_2)}{\sum_k \text{Ativação de } R_k}$$

Esse método de ponderação combina as contribuições de cada regra para gerar uma resposta precisa e direta, eliminando a necessidade de defuzzificação. Além disso, é computacionalmente eficiente, já que evita o processo de defuzzificação, o que é especialmente útil em aplicações que demandam respostas rápidas e precisas, como em sistemas de classificação.

Embora o Sistema Takagi-Sugeno (TKS) ofereça alta precisão nas predições. Os sistemas fuzzy são conhecidos pela sua interpretabilidade, o que significa que eles são mais fáceis de entender e explicar em comparação com outros modelos mais complexos. Essa interpretabilidade se deve ao fato de que os sistemas fuzzy usam regras simples no formato ***se-então***. Essas regras linguísticas são intuitivas e próximas da linguagem humana, facilitando a compreensão do processo de decisão [Ross 2004; Gacto, Alcalá e Herrera 2011].

Além disso, a interpretabilidade é ainda mais trabalhada através de métodos que simplificam o sistema fuzzy. Alguns métodos reduzem o número de regras e variáveis, tornando o sistema menos complexo, ajudando a garantir que o sistema continue fácil de entender.

Modelagem da incerteza, imprecisão e ambiguidade dos SIF

A modelagem da incerteza, imprecisão e ambiguidade é essencial para a criação de modelos que lidam bem com dados complexos e interpretativos. Os sistemas fuzzy

oferecem uma abordagem flexível [Ruspini e Patrone 1998; Ross 2004; Cox 1994], A seguir uma explicação de cada um:

- **Incerteza**: Refere-se à falta de informações completas ou precisas sobre um dado ou fenômeno. A incerteza ocorre quando não temos certeza absoluta sobre a classificação de um dado porque as informações disponíveis são insuficientes. Em sistemas fuzzy, a incerteza é modelada permitindo que um elemento pertença parcialmente a diferentes categorias, cada uma com um grau de confiança específico.

  Exemplo 1. *Em um sistema de previsão meteorológica, a incerteza está presente quando há 70% de chance de chuva e 30% de chance de céu nublado, refletindo a falta de certeza total sobre as condições do tempo.*

- **Imprecisão**: A imprecisão refere-se à falta de exatidão inerente a uma descrição ou medição de um fenômeno. Em vez de exigir uma divisão exata entre os estados possíveis, a lógica fuzzy lida com a imprecisão ao permitir gradações contínuas de pertencimento dentro de um conjunto fuzzy. Assim, em sistemas fuzzy, conceitos vagos como *alto* ou *quente* são modelados por meio de **funções de pertinência**, que indicam o grau com que um elemento pertence a um determinado conjunto.

  Exemplo 2. *Para o conceito de **altura**, uma pessoa com 1,70 m pode pertencer parcialmente aos conjuntos **baixo** e **alto** com diferentes graus de pertinência, permitindo uma descrição mais detalhada.*

- **Ambiguidade**: Refere-se à possibilidade de múltiplas interpretações para um mesmo dado, onde cada interpretação pode ser igualmente válida. A ambiguidade ocorre quando um dado pode ser classificado de diferentes maneiras dependendo do contexto. Em sistemas fuzzy, isso é modelado permitindo que um dado tenha graus de pertinência em várias categorias ao mesmo tempo, refletindo a diversidade de interpretações.

  Exemplo 3. *Em um sistema de recomendação de filmes, um filme pode ser classificado como tanto **comédia** quanto **drama** com diferentes graus, pois o gênero pode ter uma interpretação ambígua dependendo do ponto de vista.*

Os sistemas fuzzy contribuem significativamente para os sistemas neuro-fuzzy, ajudando-os a lidar de maneira adaptativa com dados incertos, imprecisos e ambíguos. Ao incorporar a lógica fuzzy, os sistemas neuro-fuzzy podem classificar dados complexos, ajustando-se às variações e permitindo várias interpretações, o que é essencial em situações onde os dados não são totalmente claros [Ruspini e Patrone 1998]. Além disso, os sistemas fuzzy tornam o processo de decisão mais fácil de

entender, pois suas regras e graus de pertinência são mais interpretáveis. Isso significa que as decisões do modelo ficam mais transparentes, permitindo que usuários e especialistas compreendam melhor como as predições são feitas. Essa combinação de adaptabilidade e interpretabilidade faz dos sistemas neuro-fuzzy uma solução poderosa para problemas que envolvem dados complexos[Gacto, Alcalá e Herrera 2011; Czogała e Leski 2000].

2.2.3 Sistemas Neuro-Fuzzy (SNF)

Os Sistemas Neuro-Fuzzy combinam a habilidade de aprendizado das Redes Neurais Artificiais (RNA) com a interpretabilidade dos Sistemas de Inferência Fuzzy (SIF) [Ruspini e Patrone 1998; Czogała e Leski 2000]. Essa integração resulta em modelos capazes de lidar com dados incertos e vagos, adaptando-se a mudanças e aprendendo com novos dados. Ao unir essas duas abordagens, os sistemas neuro-fuzzy têm o potencial de capturar relações complexas entre variáveis e fornecer previsões precisas, mantendo uma estrutura interpretável e flexível.

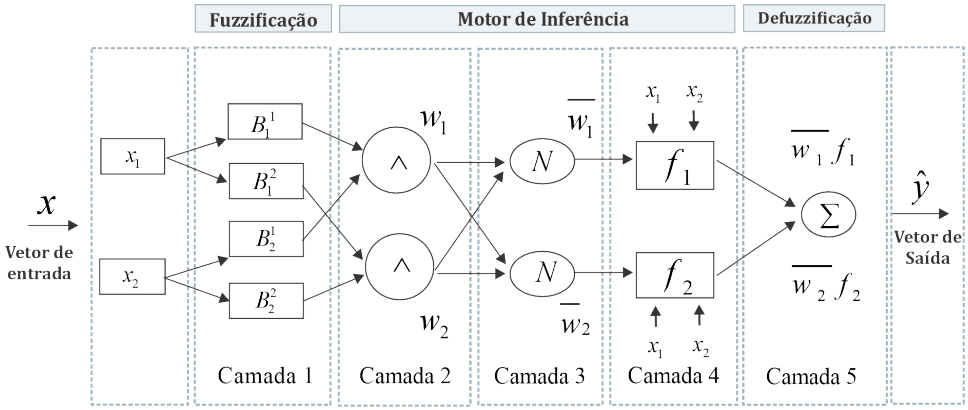


Figura 2.19: Estrutura de um Sistema Neuro-Fuzzy

A Figura 2.19 ilustra a estrutura básica de um sistema neuro-fuzzy, onde a camada fuzzy lida com a incerteza e a camada neural ajusta automaticamente os parâmetros com base nos dados.

Estrutura dos Sistemas Neuro-Fuzzy

Essa estrutura é inspirada no perceptron multicamada de redes neurais e adapta os pesos como conjuntos fuzzy, enquanto as funções de ativação, saída e propagação são configuradas para implementar uma trajetória de inferência fuzzy [Ruspini 1998].

A estrutura multicamada de um sistema neuro-fuzzy é geralmente composta por cinco camadas principais:

- **Camada de Entrada**: Essa camada é responsável por coletar as variáveis de entrada. Cada variável de entrada é representada por um nó que contem

informação vinda do ambiente, como dados de sensores ou outros tipos de sinais.

- **Camada de Fuzzificação**: Nessa etapa, os valores que entram no sistema são convertidos em valores *fuzzy*, que são mais flexíveis e podem lidar com incertezas. A fuzzificação utiliza funções de pertinência, que transformam os valores das variáveis de entrada em **graus** de pertinência.

- **Camada de Ativação das Regras**: Aqui, cada regra *fuzzy* é representada por um nó. As regras do tipo SE-ENTÃO são ativadas de acordo com as entradas fuzzificadas. Para isso, são utilizados operadores como o mínimo ou o produto para combinar essas entradas.

- **Camada de Normalização**: A normalização ajusta as forças de ativação, de modo que a soma de todas elas seja igual a 1. Esse processo é feito para evitar que alguma regra tenha um peso muito maior do que as outras, o que poderia desequilibrar o resultado.

- **Camada dos Consequentes**: Cada regra gera uma saída específica, usando cálculos simples nas entradas. Em muitos casos, são usados modelos lineares (funções matemáticas simples) que ajudam a criar respostas mais precisas e contínuas. Esse tipo de cálculo é comum em sistemas como o modelo TSK.

- **Camada de Saída (Defuzzificação)**: Esta é a última camada, onde os valores *fuzzy* gerados ao longo do processo são convertidos novamente em valores nítidos, ou seja, valores precisos e bem definidos no domínio numérico. Esse valor final é uma resposta numérica que pode ser usada para ações práticas, como controle de equipamentos ou tomadas de decisão.

A Figura 2.19 apresenta uma visão detalhada das cinco camadas principais, mostrando como os sinais são processados desde a entrada até a saída. Este modelo é projetado para facilitar algoritmos de aprendizado e permite a incorporação de conhecimentos prévios na forma de regras linguísticas, aumentando a interpretabilidade do sistema.

Modelos de Sistemas Neuro-Fuzzy

Ao longo dos anos, diversos modelos neuro-fuzzy foram propostos, cada um com características específicas que visam melhorar o desempenho em aplicações de controle, classificação, e modelagem de sistemas complexos. A Tabela 2.8 apresenta alguns dos principais modelos neuro-fuzzy desenvolvidos entre 1990 e 2022, destacando suas contribuições e arquiteturas. Os modelos listados na Tabela 2.8 mostram a evolução dos sistemas neuro-fuzzy desde abordagens mais tradicionais, como o ANFIS, até

modelos modernos e especializados, como o ML-TSK FS, que incorpora técnicas de classificação multi-rótulo com base em redes neurais e lógica fuzzy.

Cada um desses modelos apresenta características próprias e se destaca em diferentes aplicações:

- **FALCON e GARIC (Início dos anos 90)**: Ambos os modelos foram fundamentais para estabelecer as bases dos sistemas neuro-fuzzy, proporcionando os primeiros frameworks de controle adaptativo com inferência fuzzy e aprendizado neural.
- **ANFIS (1993)**: Este modelo é um dos mais amplamente utilizados em aplicações práticas, combinando lógica fuzzy de Takagi-Sugeno com ajuste de parâmetros via redes neurais. Sua popularidade se deve à simplicidade e eficiência para modelagem e controle.
- **Modelos Evolutivos e Dinâmicos (Década de 2000)**: Modelos como o DENFIS e o EFUNN introduziram capacidades de evolução e adaptação em tempo real, sendo essenciais em aplicações que requerem atualização contínua e adaptação a dados em fluxo.
- **Avanços Recentes em Classificação Multi-rótulo (2022)**: O ML-TSK FS é um exemplo de aplicação moderna, onde a lógica fuzzy é integrada com redes neurais para lidar com classificações complexas, considerando correlações entre múltiplos rótulos, o que é relevante em domínios como a bioinformática e a análise de sentimentos.

Esta visão geral dos modelos neuro-fuzzy destaca como as arquiteturas e técnicas de aprendizado foram se diversificando e especializando ao longo do tempo, cada qual contribuindo com características únicas para o avanço deste campo.

Ano	Modelo	Descrição
1991	**FALCON**	(Fuzzy Adaptive Learning Control Network), proposto por Lin e Lee, utiliza uma estrutura de cinco camadas de inferência fuzzy para controle adaptativo [Lin et al. 1991].
1992	**GARIC**	(Generalized Approximate Reasoning based Intelligence Control), um modelo de controle adaptativo desenvolvido por Berenji, com base em raciocínio aproximado [Berenji e Khedkar 1992].
1993	**ANFIS**	(Adaptive Neuro-Fuzzy Inference System), desenvolvido por Jang, combina redes neurais com lógica fuzzy de Takagi-Sugeno para ajuste de parâmetros fuzzy [Jang e Jyh-Shing 1993].
1993	**FUN**	Proposto por Sulzberger, integra redes neurais com lógica fuzzy de Takagi-Sugeno para ajuste de parâmetros, com foco em modelagem [Sulzberger, Tschichold-Gurman e Vestli 1993].
1996	**FINEST**	Proposto por Tano, combina redes neurais com um controlador fuzzy baseado em regras para controle em sistemas complexos [Tano, Oyama e Arnould 1996].
1998	**SONFIN**	Rede Neuro-Fuzzy de Inferência Auto-Construtiva, proposta por Juang e Lin, ajusta automaticamente sua estrutura de regras [Chia-Feng e Chin-Teng 1998].
1999	**NEFCON**	Proposto por Nauck e Kruse, combina redes neurais com controladores fuzzy para aplicações de controle adaptativo [Nauck e Kruse 1999].
1999	**EFUNN**	Desenvolvido por Kasabov, é um sistema neuro-fuzzy evolutivo que ajusta sua estrutura e parâmetros dinamicamente [Kasabov e S. Qun 1999].
1999	**NFN**	Proposto por Figueiredo e Gomide, foca na otimização de regras fuzzy para sistemas adaptativos [Figueiredo e Gomide 1999].
1999	**HYFIS**	(Hybrid Fuzzy Inference System), desenvolvido por Kim, utilizado em modelagem de séries temporais [Kim e Kasabov 1999].
2002	**DENFIS**	(Dynamic Evolving Neural-Fuzzy Inference System), desenvolvido por Kasabov, ideal para modelagem em tempo real [Kasabov, Song e Qun 2002].
2004	**SOFNN**	(Self-Organizing Fuzzy Neural Network), modelo que ajusta automaticamente sua estrutura e parâmetros durante o aprendizado [Leng, Prasad e McGinnity 2004].
2012	**mANFIS**	Versão modificada do ANFIS, usada para análise de emoções humanas complexas utilizando sinais visuais e EEG [Qing, Sungmoon e Minho 2012].
2019	**SOFIS**	(Local optimality of self-organising neuro-fuzzy inference systems), otimiza localmente a estrutura para desempenho aprimorado [Gu, Angelov e Rong 2019].
2021	**ML-TSK FS**	Modelo multi-rótulo que combina redes neurais e lógica fuzzy para classificação com foco na correlação de rótulos [Lou et al. 2021].

Tabela 2.8: Alguns Modelos Neuro-Fuzzy (1990-2022)

Motivação para o Uso de Sistemas Neuro-Fuzzy

A principal motivação para o uso de sistemas neuro-fuzzy está em sua capacidade de unir o raciocínio aproximado dos sistemas fuzzy com o aprendizado adaptativo das redes neurais, criando uma solução robusta para problemas complexos e dinâmicos.

- **Aprendizado e Adaptação das Redes Neurais**: As redes neurais fornecem ao sistema neuro-fuzzy a capacidade de aprender e adaptar-se continuamente a novos dados, essencial para contextos onde as informações estão em constante mudança.
- **Interpretabilidade dos Sistemas Fuzzy**: A lógica fuzzy utiliza regras linguísticas simples (do tipo ***se-então***), facilitando a transparência e a compreensão do modelo. Em áreas críticas como a medicina e finanças, essa interpretabilidade é essencial para que especialistas entendam e validem as decisões do sistema.
- **Tratamento de Ambiguidade, Imprecisão e Incerteza nos Dados**: Os sistemas fuzzy são projetados para lidar com dados vagos e ambíguos, permitindo que informações incertas sejam classificadas de maneira flexível. Isso torna os sistemas neuro-fuzzy ideais para problemas de classificação em cenários complexos, onde a variabilidade e a falta de precisão dos dados desafiam métodos tradicionais.

A Tabela 2.8 de evolução dos sistemas neuro-fuzzy, de 1991 a 2022, destaca o desenvolvimento contínuo desses modelos e a diversificação de suas aplicações. Essa trajetória reforça a relevância dos sistemas neuro-fuzzy para enfrentar desafios reais que demandam aprendizado adaptativo, transparência e manejo eficaz de incertezas.

Os sistemas neuro-fuzzy oferecem uma abordagem apropriada para problemas onde a incerteza e a necessidade de aprendizado se encontram. Ao combinar a interpretabilidade dos sistemas fuzzy com a capacidade adaptativa das redes neurais, esses sistemas fornecem uma solução flexível para uma ampla gama de aplicações, desde controle de processos industriais até reconhecimento de padrões.

2.3 A INTEGRAL DE CHOQUET DISCRETA

Para compreender plenamente a Integral de Choquet Discreta, é importante primeiro abordar a fundamentação teórica das funções de agregação e das medidas fuzzy. Também serão apresentados exemplos que facilitam a compreensão desses conceitos. Cada um desses elementos será explorado nas próximas subseções.

2.3.1 Fundamentos dos Operadores de Agregação

A agregação é o processo de combinar diferentes valores numéricos em um único valor representativo, conhecido como função de agregação [Grabisch, Marichal et al. 2009]. Em sistemas fuzzy, funções de agregação são amplamente aplicadas para combinar graus de pertinência, pesos de critérios, graus de preferência, entre outros. As funções de agregação têm uma importância fundamental em áreas como estatística, ciência da computação, matemática e economia, permitindo sintetizar informações para análises e tomadas de decisão.

Definição 4. *(Função de Agregação) Uma função de $n > 1$ argumentos que mapeia o hipercubo unitário no intervalo $[0,1]$, $\mathcal{A} : [0,1]^n \to [0,1]$, é chamada de função de agregação quando satisfaz:*

- *Condições de Fronteira:* $\mathcal{A}(0,0,\ldots,0) = 0$ *e* $\mathcal{A}(1,1,\ldots,1) = 1$.
- *Monotonicidade: Se* $\vec{x} \leq \vec{y}$ *então* $\mathcal{A}(\vec{x}) \leq \mathcal{A}(\vec{y})$, *para* $x_i \leq y_i$ *em todo* $i \in \{1,\ldots,n\}$.

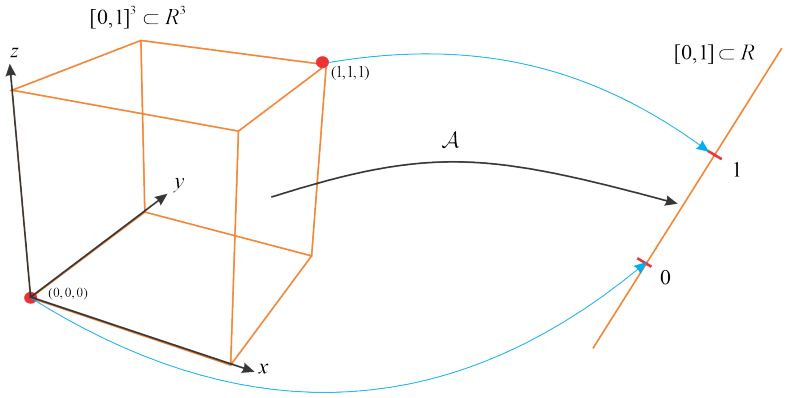


Figura 2.20: Representação geométrica de uma função de agregação

Exemplo 4. *Imagine um sistema de recomendação de filmes, onde um usuário avalia diferentes aspectos (enredo, direção, atuação) em uma escala de 0 a 1. As funções de agregação combinam essas notas para determinar uma avaliação final do filme.*

Classes de Operadores de Agregação

Os operadores de agregação podem ser classificados em quatro categorias principais: média, conjuntiva, disjuntiva e mista [Grabisch, Marichal et al. 2009]. Além disso, dependendo de suas propriedades específicas, essas funções são divididas em classes adicionais, incluindo T-normas, T-conormas, funções de sobreposição (overlaps) e funções de agrupamento (groupings) [Bustince, Fernandez et al. 2010; Bustince, Pagola et al. 2011].

- **T-Normas**: Uma função de agregação bivariada $T : [0,1]^2 \to [0,1]$ é dita ser uma t-norma se satisfizer as seguintes propriedades:

  - (T1) Comutatividade;
  - (T2) Associatividade;
  - (T3) Condição de fronteira: $\forall x \in [0,1] : T(x,1) = x$.

  As t-normas são amplamente utilizadas em operações de conjunção em lógica fuzzy.

- **T-Conormas**: Uma função de agregação bivariada $S : [0,1]^2 \to [0,1]$ é uma t-conorma se satisfizer as seguintes propriedades:

  - (S1) Comutatividade;
  - (S2) Associatividade;
  - (S3) Condição de fronteira: $\forall x \in [0,1] : S(x,0) = x$.

  As t-conormas são utilizadas em operações de disjunção e são o operador dual das t-normas.

- **Funções de Sobreposição (Overlap)**: Uma função bivariada $O : [0,1]^2 \to [0,1]$ é dita ser uma função de sobreposição se satisfizer as seguintes condições:

  - (O1) O é comutativa;
  - (O2) $O(x,y) = 0$ se, e somente se, $xy = 0$;
  - (O3) $O(x,y) = 1$ se, e somente se, $xy = 1$;
  - (O4) O é crescente;
  - (O5) O é contínua.

  Funções de sobreposição são úteis em casos onde se deseja medir o grau de sobreposição ou similaridade entre dois valores.

- **Funções de Agrupamento (Grouping)**: Uma função bivariada $G : [0,1]^2 \to [0,1]$ é dita ser uma função de agrupamento se satisfizer as seguintes condições:

  - (G1) G é simétrica;
  - (G2) $G(x,y) = 0$ se, e somente se, $x = y = 0$;
  - (G3) $G(x,y) = 1$ se, e somente se, $x = 1$ ou $y = 1$;
  - (G4) G é crescente;
  - (G5) G é contínua.

As funções de agrupamento são aplicadas em contextos onde se deseja combinar valores que representam diferentes agrupamentos ou categorias.

Na Tabela 2.9, apresentamos alguns exemplos.

Fórmula	**Notação**	**Classe**	**Tipo**
$\min\{x_1, \ldots, x_n\}$	min	Conjuntiva	t-norma, overlap
$\max\{x_1, \ldots, x_n\}$	max	Disjuntiva	t-conorma, grouping
$\prod_{i=1}^{n} x_i$	PR	Conjuntiva	t-norma, overlap
$\frac{1}{n}\sum_{i=1}^{n} x_i$	AM	Média	não tem tipo
$\left(\frac{1}{n}\sum_{i=1}^{n} x_i^2\right)^{1/2}$	QM	Média	não tem tipo
$\left(\prod_{i=1}^{n} x_i\right)^{1/n}$	GM	Média	não tem tipo
$\left(\frac{1}{n}\sum_{i=1}^{n} \frac{1}{x_i}\right)^{-1}$	HM	Média	não tem tipo

Tabela 2.9: Exemplos de Funções de Agregação

Onde $x_i \neq 0$ em HM.

Medidas Fuzzy

Antes de abordarmos a Integral de Choquet, é necessário entender o conceito de medidas fuzzy, que desempenham um papel essencial no cálculo da integral e na modelagem de interações complexas entre atributos. As medidas fuzzy permitem a agregação de valores, considerando incertezas e interdependências entre atributos, o que é fundamental para aplicações em sistemas onde a relação entre os dados não é estritamente aditiva.

As medidas fuzzy generalizam os conceitos habituais de medida, como comprimento, área e volume [Barros e Bassanezi 2010]. Ao contrário das medidas clássicas, que são σ-aditivas, as medidas fuzzy enfraquecem essa propriedade, preservando apenas a monotonicidade. Isso permite maior flexibilidade na modelagem de interdependências complexas, uma característica essencial para sistemas de classificação multi-rótulo, como o que abordamos em nossa pesquisa.

Para entender melhor a relevância das medidas fuzzy, é útil apresentar a definição formal de um espaço mensurável e de uma medida, antes de passarmos para o conceito específico de medida fuzzy e sua aplicação na agregação de atributos interdependentes.

Definição 5. *(Espaço Mensurável e Medida) [Pedrycz e Gomide 1998] Um* ***espaço mensurável*** *é descrito pelo par* $(\Omega, \mathbb{A})$*, onde* Ω *é o universo e* $\mathbb{A}$ *é uma* σ*-álgebra de subconjuntos de* Ω*. Uma* ***medida*** $\mathfrak{m}$ *é uma função definida em* $(\Omega, \mathbb{A})$*, com valores não negativos, que satisfaz:*

1. $\mathfrak{m}(\varnothing) = 0$.

2. $\mathfrak{m}$ *é* σ*-aditiva, ou seja, para uma família de conjuntos disjuntos* $\{A_i\}$*:*

$$\mathfrak{m}\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mathfrak{m}(A_i).$$

No entanto, em sistemas onde os elementos interagem, como na avaliação de produtividade de grupos de trabalho, essa abordagem pode ser restritiva. Considere, por exemplo, a produtividade de um grupo de trabalhadores: a combinação de duas subequipes pode não resultar em uma produtividade que seja simplesmente a soma das partes, devido à interdependência entre os trabalhadores.

Medidas Fuzzy e a Ausência de Aditividade: As medidas fuzzy relaxam a condição de σ-aditividade, exigindo apenas monotonicidade. Dessa forma, se $A \subset B$, então $m(A) \leq m(B)$, mas $m(A \cup B) \neq m(A) + m(B)$ necessariamente, mesmo que A e B sejam disjuntos. Isso permite que as medidas fuzzy capturem dependências mais complexas entre elementos. Essa flexibilidade torna as medidas fuzzy adequadas para representar interdependências e incertezas em sistemas como o modelo ML-TSKC FS, abordado em nosso estudo.

Definição 6. ***(Medida Fuzzy)****[Sugeno 1974] Seja* $N = \{1, \ldots, n\}$ *um conjunto finito e* 2^N *o conjunto das partes de* N*. Uma função* $m : 2^N \to [0, 1]$ *é uma medida fuzzy se:*

1. $m(\emptyset) = 0$ *e* $m(N) = 1$.

2. *Se* $X \subseteq Y$*, então* $m(X) \leq m(Y)$ *para qualquer* $X, Y \subseteq N$.

Interações Modeladas pelas Medidas Fuzzy: As medidas fuzzy possibilitam três tipos de interações entre conjuntos disjuntos A e B:

- **Independência**: $m(A \cup B) = m(A) + m(B)$.
- **Interação Positiva**: $m(A \cup B) > m(A) + m(B)$.
- **Interação Negativa**: $m(A \cup B) < m(A) + m(B)$.

Essas interações permitem que as medidas fuzzy capturem dinâmicas complexas entre atributos, o que é particularmente útil em sistemas de classificação multi-rótulo, onde a interação entre os atributos pode influenciar diretamente a performance do modelo.

A Figura 2.21 mostra a representação gráfica de uma estrutura de medida fuzzy para um conjunto com quatro elementos. Cada nó representa um subconjunto respeitando a ordem de inclusão.

Medidas Fuzzy Utilizadas no Estudo

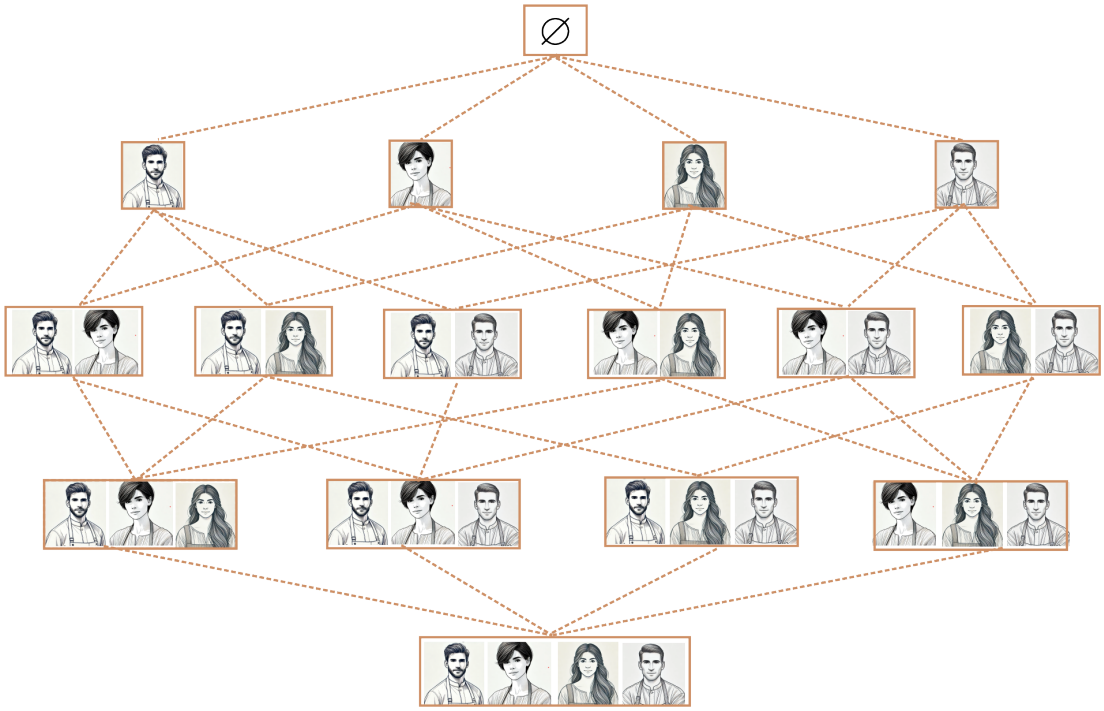


Figura 2.21: Estrutura de medidas fuzzy para um conjunto finito $N = \{1, 2, 3, 4\}$

Para o cálculo da Integral de Choquet em nosso estudo, selecionamos cinco tipos de medidas fuzzy, descritas na Tabela 2.10. Cada medida reflete uma abordagem diferente para representar incerteza e importância relativa entre atributos.

Medida	**Definição**
Uniforme	$\mathrm{m_U}(I) = \frac{\lvert I\rvert}{n}$
Relativa	$\mathrm{m_R}(I) = \frac{\sum_{j\in I} j}{\sum_{i\in N} i}$
Produto	$\mathrm{m_\Pi}(I) = \frac{\prod_{j\in I} j}{\prod_{i\in N} i}$
Potência	$\mathrm{m_p}(I) = \left(\frac{\lvert I\rvert}{n}\right)^q, \quad q > 0$
Ponderada	$\mathrm{m_w}(I) = \sum_{i\in I} p_i$

Tabela 2.10: Medidas fuzzy utilizadas no estudo.

Descrição das Medidas Utilizadas: Cada medida fuzzy possui características específicas que a tornam apropriada para contextos distintos:

- **Medida Uniforme**: Indica igual importância para todos os atributos, sendo útil em cenários onde a homogeneidade é desejada.

- **Medida Relativa**: Prioriza atributos com maior relevância relativa, refletindo situações onde alguns atributos têm impacto mais significativo.
- **Medida Produto**: Usa interdependência multiplicativa, adequada para análises de risco e situações de alta interação entre atributos.
- **Medida de Potência**: Ajusta a sensibilidade por meio de q, destacando subconjuntos específicos conforme o contexto.
- **Medida Ponderada**: Permite especificar pesos diretos para cada atributo, oferecendo controle total sobre a influência individual.

Assim, as medidas fuzzy são essenciais para modelar incertezas e interdependências complexas em sistemas onde a relação entre os atributos é crucial. Essa característica justifica a escolha da Integral de Choquet como técnica de agregação em nosso modelo, pois ela permite capturar essas interações de forma mais eficaz. Na próxima seção, discutiremos a Integral de Choquet e sua aplicação específica no modelo ML-TSKC FS, mostrando como ela contribui para melhorar a classificação multi-rótulo.

Integral de Choquet Discreta: Definição e propriedades.

A *integral de Choquet* foi introduzida por Gustave Choquet no contexto da teoria das medidas fuzzy, visando criar uma forma de integração que pudesse lidar com medidas não aditivas e, portanto, capturar interdependências entre variáveis [Choquet 1954]. Posteriormente, essa integral foi adaptada e explorada na teoria fuzzy, notavelmente pelos trabalhos de Murofushi e Sugeno, que utilizaram a integral de Choquet para modelar medidas fuzzy em contextos de decisão multicritério, onde as interações entre critérios são cruciais [Murofushi e Sugeno 1989].

No contexto discreto, a integral de Choquet atua como um **operador de agregação não aditivo**, permitindo que se capturem interações complexas entre as variáveis, ou critérios, sem as limitações dos operadores de agregação tradicionais, como a média ponderada ou o produto. Em particular, a integral de Choquet discreta é amplamente utilizada em sistemas de decisão multicritério e aprendizado de preferências, onde a importância de cada critério depende dos outros critérios considerados. Formalmente, a Integral de Choquet discreta é definida como:

Definição 7. *(Integral de Choquet discreta) Seja* $\mathfrak{m} : 2^N \to [0,1]$ *uma medida fuzzy. A integral de Choquet discreta de* $\vec{x} = (x_1, x_2, \ldots, x_n) \in [0,1]^n$ *com relação à medida fuzzy* $\mathfrak{m}$ *é uma função* $\mathcal{C}_\mathfrak{m} : [0,1]^n \to [0,1]$*, definida por*

$$\mathcal{C}_\mathfrak{m}(\vec{x}) = \sum_{i=1}^{n} \left(x_{(i)} - x_{(i-1)} \right) \cdot \mathfrak{m}\left(A_{(i)} \right), \tag{2.1}$$

onde $\vec{x}_{\nearrow} = \left(x_{(1)}, \ldots, x_{(n)}\right)$ *é uma permutação não-decrescente de* $\vec{x}$*, isto é,* $0 \leq x_{(1)} \leq \ldots \leq x_{(n)}$*, por convenção* $x_{(0)} = 0$*, e* $A_{(i)} = \{(i), \ldots, (n)\}$ *é o subconjunto dos índices das* $n - i + 1$ *maiores componentes de* $\vec{x}$*.*

Onde a medida $\mathfrak{m}$ fornece o peso relativo de cada subconjunto de N, permitindo que o operador de agregação considere a importância das variáveis em conjunto, refletindo a influência das interações entre elas [Grabisch e Labreuche 2010; Beliakov, James e Wu 2020].

Comparação com outros Operadores de Agregação

A integral de Choquet discreta oferece uma flexibilidade muito maior do que operadores de agregação mais simples, como o mínimo e o produto, principalmente quando precisamos lidar com situações de alta dependência entre as variáveis de entrada. Cada um desses operadores tradicionais possui limitações específicas que reduzem sua capacidade de capturar interações complexas entre variáveis.

O operador mínimo, denotado por $\min(x_1, x_2, \ldots, x_n)$, é um operador conservador que considera apenas o menor valor entre as variáveis de entrada $x_1, x_2, \ldots, x_n$. Isso significa que, mesmo que outras variáveis tenham valores mais altos, o mínimo só considera o valor mais baixo, o que o torna pouco adequado para situações onde a interação entre variáveis é importante.

O operador produto, por sua vez, combina as variáveis multiplicando seus valores:

$$\prod(x_1, x_2, \ldots, x_n) = x_1 \times x_2 \times \cdots \times x_n.$$

Embora esse operador leve em conta todas as variáveis, ele não consegue capturar interações específicas entre subconjuntos de variáveis. Ele pressupõe uma certa independência entre as variáveis, o que limita sua capacidade de modelar interações complexas entre critérios distintos [Beliakov, James e Wu 2020; Grabisch 2000].

Em contraste, a integral de Choquet se destaca porque permite:

- **Monotonicidade**: A integral de Choquet é **não-decrescente**, ou seja, se o valor de uma variável x_i aumenta, mantendo as outras constantes, o valor da integral de Choquet não diminui. Isso garante que o operador respeite aumentos nas variáveis individuais, capturando assim a natureza não-aditiva das interações. Em termos formais, para duas variáveis x_i e x_j em um conjunto $X = \{x_1, x_2, \ldots, x_n\}$, temos:

$$x_i \geq x_j \Rightarrow \mathcal{C}_{\mathfrak{m}}(x_1, \ldots, x_i, \ldots, x_n) \geq \mathcal{C}_{\mathfrak{m}}(x_1, \ldots, x_j, \ldots, x_n),$$

  onde $\mathcal{C}_{\mathfrak{m}}$ é a integral de Choquet discreta em relação à medida fuzzy $\mathfrak{m}$.

- **Sensibilidade às Interações**: A integral de Choquet consegue capturar a forma como as variáveis interagem entre si, graças à medida fuzzy $\mathfrak{m}$. Por exemplo, se duas variáveis x_i e x_j têm uma influência maior quando consideradas juntas do que individualmente, a medida fuzzy reflete essa sinergia, atribuindo um peso maior ao subconjunto $\{i, j\}$ do que à soma dos pesos individuais $\mathfrak{m}(\{i\})$ e $\mathfrak{m}(\{j\})$:

$$\mathfrak{m}(\{i, j\}) > \mathfrak{m}(\{i\}) + \mathfrak{m}(\{j\}).$$

  Se, por outro lado, duas variáveis são redundantes, a medida fuzzy pode refletir isso, atribuindo um peso menor ao subconjunto $\{i, j\}$:

$$\mathfrak{m}(\{i, j\}) < \mathfrak{m}(\{i\}) + \mathfrak{m}(\{j\}).$$

  Essa sensibilidade permite à integral de Choquet lidar com dependências complexas, o que é especialmente útil em sistemas de decisão e avaliação multicritério [Krishnan, Kasim e Bakar 2015; Grabisch, Roubens et al. 2000].

- **Flexibilidade**: A integral de Choquet se adapta ao contexto, ajustando o peso de cada variável conforme o grupo específico em que ela está inserida. Isso significa que, diferente de uma média ponderada comum, onde cada variável tem um peso fixo, na integral de Choquet o peso de uma variável x_i pode mudar dependendo dos outros critérios presentes. Essa flexibilidade é ideal em aplicações práticas, como recomendação de produtos e sistemas de ranking, isso ajuda a modelar as interações entre critérios de maneira precisa, refletindo relações de dependência que os operadores mínimo e produto não conseguem capturar [Tehrani, Cheng e Hullermeier 2012].

Essas características fazem da integral de Choquet uma excelente escolha para problemas que requerem uma visão detalhada das interações e dependências entre critérios. Em contextos como decisão multicritério e agregação de informações complexas, a integral de Choquet permite que o peso de cada critério seja ajustado conforme a presença de outros critérios, criando uma agregação adaptada ao contexto que reflete melhor as interações entre variáveis [Benvenuti, Vivona et al. 2002].

Exemplos Comparativos: Integral de Choquet vs. Operadores Mínimo e Produto

Vamos apresentar exemplos para cada uma das propriedades destacadas: **Monotonicidade**, **Sensibilidade às Interações** e **Flexibilidade**. Compararemos a

Integral de Choquet com o **operador mínimo** e o **operador produto** para demonstrar como, em determinadas situações, a integral de Choquet captura nuances que os operadores mais simples não conseguem.

- **Monotonicidade**: Queremos ver como cada operador reage a um aumento em uma das variáveis, mantendo a outra constante.
  - **Exemplo:** Suponha que temos duas variáveis representando critérios de avaliação: $x_1 = 0.7$ e $x_2 = 0.5$.
  - **Objetivo**: O operador deve aumentar seu valor final se x_1 aumentar, refletindo a importância crescente desse critério.
  - **Medida fuzzy:** para a integral de Choquet: $m(A_1) = 0.6$, $m(A_2) = 0.4$, e $m(A_{1,2}) = 0.9$.

  Operadores:

  - ***Operador Mínimo***:

    $$\min(x_1, x_2) = \min(0.7, 0.5) = 0.5$$

    Se aumentarmos x_1 para 0.9:

    $$\min(0.9, 0.5) = 0.5$$

    Observação: O operador mínimo não reflete o aumento em x_1, ele depende apenas do menor valor.
  - ***Operador Produto***:

    $$x_1 \times x_2 = 0.7 \times 0.5 = 0.35$$

    Se aumentarmos x_1 para 0.9:

    $$0.9 \times 0.5 = 0.45$$

    Observação: O produto reflete o aumento, mas penaliza o valor final, pois multiplicar por 0.5 ainda reduz significativamente o resultado.
  - ***Integral de Choquet***:

    $$C_m(x_1, x_2) = (x_{(1)} - x_{(0)}) \cdot m(A_{12}) + (x_{(2)} - x_{(1)}) \cdot m(A_2)$$

    Com $x_{(1)} = 0.5$, $x_{(2)} = 0.7$:

    $$C_m(x_1, x_2) = (0.5 - 0) \cdot 0.9 + (0.7 - 0.5) \cdot 0.4 = 0.5 \cdot 0.9 + 0.2 \cdot 0.4 = 0.53$$

Se aumentarmos x_1 para 0.9:

$$C_m(x_1, x_2) = (0.5 - 0) \cdot 0.9 + (0.9 - 0.5) \cdot 0.4 = 0.5 \cdot 0.9 + 0.4 \cdot 0.4 = 0.66$$

Observação: A integral de Choquet aumenta mais significativamente, capturando o impacto positivo do aumento de x_1.

- **Sensibilidade às Interações:** Vamos considerar um caso onde as variáveis representam dois critérios que, quando atuam juntos, têm uma influência maior do que quando isolados (sinergia).
  - **Exemplo:** Suponha que temos dois critérios de avaliação: $x_1 = 0.4$ e $x_2 = 0.8$.
  - **Objetivo**: O operador deve refletir que esses critérios são mais valiosos juntos do que separadamente.
  - **Medida fuzzy**: $m(A_1) = 0.4$, $m(A_2) = 0.5$, e $m(A_{1,2}) = 0.9$.

  Operadores:

  - ***Operador Mínimo***:

    $$\min(x_1, x_2) = \min(0.4, 0.8) = 0.4$$

    Observação: O operador mínimo ignora a interação entre x_1 e x_2 e usa apenas o menor valor.
  - ***Operador Produto***:

    $$x_1 \times x_2 = 0.4 \times 0.8 = 0.32$$

    Observação: O produto combina ambos os valores, mas sem capturar a sinergia entre eles.
  - ***Integral de Choquet***:

    $$C_m(x_1, x_2) = (x_{(1)} - x_{(0)}) \cdot m(A_{1,2}) + (x_{(2)} - x_{(1)}) \cdot m(A_2)$$

    Com $x_{(1)} = 0.4$, $x_{(2)} = 0.8$:

    $$C_m(x_1, x_2) = (0.4 - 0) \cdot 0.9 + (0.8 - 0.4) \cdot 0.5 = 0.4 \cdot 0.9 + 0.4 \cdot 0.5 = 0.56$$

    Observação: A integral de Choquet captura a sinergia entre x_1 e x_2, resultando em um valor agregado maior.

- **Flexibilidade:** Vamos considerar um caso onde o peso de um critério deve depender da presença de outro critério.

- **Exemplo:** Suponha que temos dois critérios de avaliação: $x_1 = 0.6$ e $x_2 = 0.8$.
- **Objetivo**: O operador deve dar mais peso a x_1 quando x_2 está presente, refletindo uma dependência contextual entre os critérios.
- **Medida fuzzy**: $m(A_1) = 0.4$, $m(A_2) = 0.5$, e $m(A_{1,2}) = 0.85$.

Operadores:

- ***Operador Mínimo***:

$$\min(x_1, x_2) = \min(0.6, 0.8) = 0.6$$

Observação: O operador mínimo ignora a influência adicional de x_2 sobre x_1.

- ***Operador Produto***:

$$x_1 \times x_2 = 0.6 \times 0.8 = 0.48$$

Observação: O produto considera ambos os valores, mas sem ajustar o peso de x_1 baseado na presença de x_2.

- ***Integral de Choquet***:

$$C_m(x_1, x_2) = (x_{(1)} - x_{(0)}) \cdot m(A_{1,2}) + (x_{(2)} - x_{(1)}) \cdot m(A_2)$$

ou

$$C_m(x_1, x_2) = x_{(1)} \cdot (m(A_{1,2}) - m(A_2)) + x_{(2)} \cdot m(A_2)$$

Observação: Notemos que, o peso de uma variável $x_{(i)}$ pode mudar dependendo dos outros critérios presentes, oferecendo uma agregação mais sensível ao contexto.

Para $x_{(1)} = 0.6$, $x_{(2)} = 0.8$ temos:

$$C_m(x_1, x_2) = (0.6 - 0) \cdot 0.85 + (0.8 - 0.6) \cdot 0.5 = 0.6 \cdot 0.85 + 0.2 \cdot 0.5 = 0.61$$

Esses exemplos mostram que a **Integral de Choquet** pode capturar nuances de interação e dependência entre variáveis de forma que os operadores de **mínimo** e **produto** não conseguem, tornando-a mais flexível e sensível ao contexto em situações de decisão multicritério e agregação de informações complexas.

A seguir, apresentamos alguns exemplos didáticos da Integral de Choquet.

Exemplo 5. ***Média Aritmética como Caso Particular da Integral de Choquet***

A média aritmética é um caso particular da Integral de Choquet quando usamos uma medida fuzzy uniforme, isto é, uma medida que atribui a mesma importância a todos os subconjuntos. Para ilustrar, consideremos três variáveis $x_1 = 1$, $x_2 = 3$, *e* $x_3 = 5$ *com pesos iguais* $w_1 = w_2 = w_3 = \frac{1}{3}$.

Calculando a média aritmética:

$$MA = \left(\frac{1}{3} \times 1\right) + \left(\frac{1}{3} \times 3\right) + \left(\frac{1}{3} \times 5\right) = 3.$$

Usando a Integral de Choquet com uma medida uniforme, obtemos o mesmo valor:

$$\mathfrak{m}(\{x_1\}) = \mathfrak{m}(\{x_2\}) = \mathfrak{m}(\{x_3\}) = \frac{1}{3},$$

$$\mathfrak{m}(\{x_1, x_2\}) = \mathfrak{m}(\{x_1, x_3\}) = \mathfrak{m}(\{x_2, x_3\}) = \frac{2}{3},$$

$$\mathfrak{m}(\{x_1, x_2, x_3\}) = 1.$$

Calculando a Integral de Choquet:

$$C_{\mathfrak{m}}(x) = (x_{(1)} - 0)\mathfrak{m}(\{x_1, x_2, x_3\}) + (x_{(2)} - x_{(1)})\mathfrak{m}(\{x_2, x_3\}) + (x_{(3)} - x_{(2)})\mathfrak{m}(\{x_3\}).$$

Substituindo os valores:

$$C_{\mathfrak{m}}(x) = (1 - 0) \times 1 + (3 - 1) \times \frac{2}{3} + (5 - 3) \times \frac{1}{3} = 3.$$

Assim, a Integral de Choquet recupera a média aritmética com uma medida uniforme.

Exemplo 6. ***Modelagem Prática com a Integral de Choquet***

Consideremos um cenário onde três trabalhadores, Leo, Kim e Eva, têm diferentes níveis de produtividade. Nosso objetivo é calcular a produção total do grupo usando a Integral de Choquet, que permite modelar a interação entre os diferentes trabalhadores (ou subconjuntos deles), capturando assim dependências e sinergias.

Os valores $\mathfrak{m}$ *são determinados com base no subconjunto dos trabalhadores que estão contribuindo para a produção total, como mostrado na Figura 2.22.*

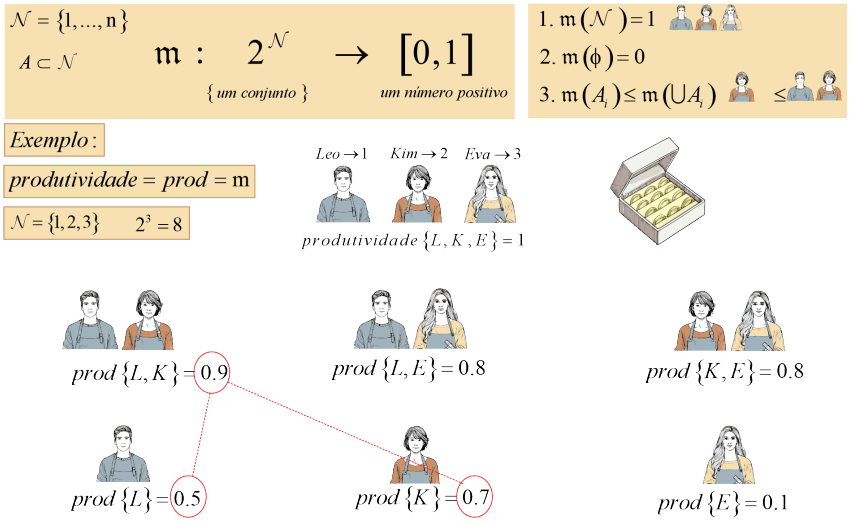


Figura 2.22: Exemplo de cálculo de produtividade com medida fuzzy

Dado o conjunto $\mathcal{N} = \{Leo, Kim, Eva\} = \{1, 2, 3\}$*, calculamos a produção total em um dia de trabalho com base nos subconjuntos formados pelos trabalhadores. O tempo de trabalho de cada trabalhador é representada na Figura 2.23 pelos valores* x_i*, com* $x_1 = 0.4$*,* $x_2 = 0.3$ *e* $x_3 = 0.6$*.*

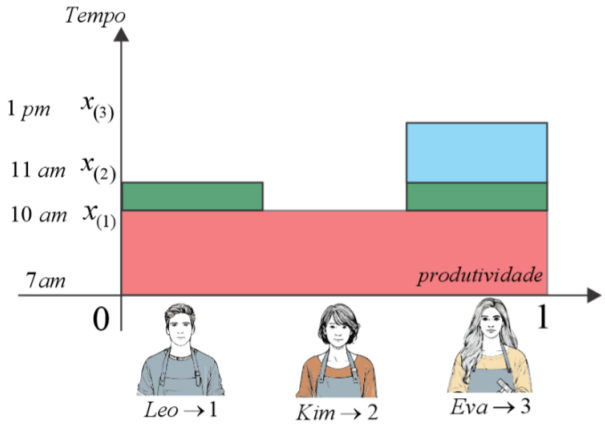


Figura 2.23: Exemplo de cálculo de produção com a Integral de Choquet

Dos dados de entrada temos:

$$x_{(1)} = 0.3, \quad x_{(2)} = 0.4, \quad x_{(3)} = 0.6$$

Os subconjuntos $A_{(i)}$ *associados são:*

$$A_{(1)} = \{(1), (2), (3)\} = \{1, 2, 3\}, \quad A_{(2)} = \{(2), (3)\} = \{1, 3\}, \quad A_{(3)} = \{3\}$$

A medida fuzzy m atribuída a cada um desses subconjuntos é:

$$m(\{3\}) = 0.1, \quad m(\{1,3\}) = 0.8, \quad m(\{1,2,3\}) = 1.0$$

Agora, aplicamos a fórmula da Integral de Choquet:

$$C_m(x) = (x_{(1)} - 0)m(A_{(1)}) + (x_{(2)} - x_{(1)})m(A_{(2)}) + (x_{(3)} - x_{(2)})m(A_{(3)})$$

Substituindo os valores:

$$C_m(x) = (0.3 - 0) \cdot 0.1 + (0.4 - 0.3) \cdot 0.8 + (0.6 - 0.4) \cdot 1.0$$

Realizando os cálculos:

$$C_m(x) = 0.3 \cdot 1.0 + 0.1 \cdot 0.8 + 0.2 \cdot 0.1$$

$$C_m(x) = 0.3 + 0.08 + 0.02 = 0.4$$

Portanto, a produção total avaliada usando a Integral de Choquet é 0.4*, o que reflete a interação entre os trabalhadores e suas contribuições individuais.*

Conclusão

A integral de Choquet discreta como operador de agregação é uma ferramenta poderosa para capturar relações de dependência e interação entre variáveis, oferecendo uma alternativa flexível e robusta aos operadores tradicionais. Sua aplicabilidade em contextos de alta complexidade a torna essencial para áreas que requerem uma análise detalhada das interações entre variáveis.

Capítulo 3

SISTEMA FUZZY MULTI-RÓTULO TAKAGI-SUGENO-KANG CHOQUET (ML TSKC-FS)

Neste capítulo, apresentamos o funcionamento do modelo ML-TSKC FS também explicaremos como cada componente do modelo contribui para o processo de inferência fuzzy e como a integral de Choquet é utilizada para capturar as interações complexas entre as variáveis. Além disso, descreveremos o fluxo de dados através das diferentes camadas do modelo, desde a fuzzificação das entradas até a ponderação dos resultados finais.

3.1 ARQUITETURA DO MODELO ML-TKSC FS

A arquitetura ML-TKSC FS mantém os princípios fundamentais dos sistemas de inferência fuzzy TSK (Takagi-Sugeno-Kang) (por exemplo, [Tomohiro e Sugeno 1985; Sugeno e Kang 1986; Sugeno e Kang 1988]), mas aprimora as capacidades de modelagem. A estrutura proposta incorpora a integral discreta de Choquet para a determinação do peso das regras, substituindo o operador de produto utilizado na arquitetura original ML-TKS FS (Sistema Fuzzy Multi-Rótulo Takagi-Sugeno-Kang)

introduzida em [Lou et al. 2021]. Esta generalização permite que o modelo considere de forma mais precisa as interações entre os atributos. Além disso, melhora a capacidade do modelo de captar interações complexas dos dados, resultando em um desempenho superior em sistemas multi-rótulo, como explicamos a seguir.

O ML-TKSC FS é construído a partir de K regras fuzzy, cada uma estabelecendo uma relação entre um conjunto de condições de entrada e uma correspondente função linear de saída. A k-ésima regra é representada como:

$$R^k : \text{SE } \boldsymbol{x} \text{ é } B^k, \text{ ENTÃO } \boldsymbol{y} = L^k(\boldsymbol{x}, \boldsymbol{p}^k), \quad k = 1, 2, \ldots, K. \tag{3.1}$$

Aqui, $\boldsymbol{x} = (x_1, \ldots, x_A)$ representa o vetor de entrada, $B^k = B_1^k \times \ldots \times B_A^k$ são os conjuntos fuzzy, e $L^k(\boldsymbol{x}, \boldsymbol{p}^k)$ define a função linear de saída, com parâmetros $\boldsymbol{p}^k$ ponderando linearmente as contribuições de cada variável de entrada x_j.

Na Figura 3.1, o modelo ML-TSKC FS é ilustrado, compreendendo cinco camadas distintas. A primeira camada lida com a fuzzificação, enquanto as camadas 2, 3 e 4 formam o núcleo com as regras base, abrangendo tanto os componentes antecedentes quanto consequentes. A quinta e última camada realiza uma agregação das saídas de cada regra.

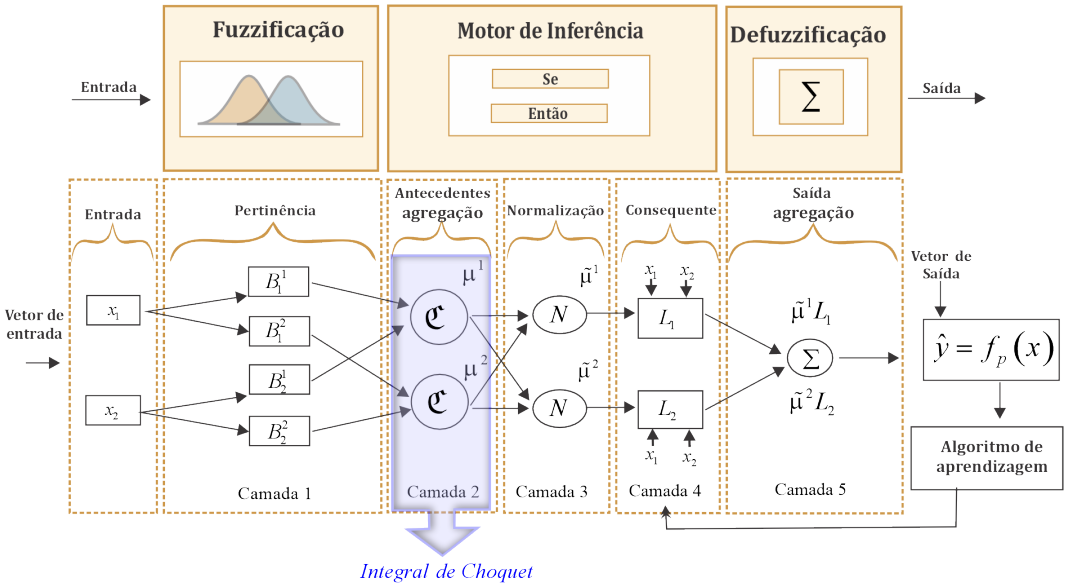


Figura 3.1: Uma visão geral do modelo ML TSKC-FS, destacando a camada modificada.

A seguir, explicamos e discutimos cada camada representada na Figura 3.1.

3.1.1 Camada 1: O processo de fuzzificação

O primeiro passo no processo de inferência fuzzy concentra-se na fuzzificação dos dados de entrada, onde as informações quantitativas são convertidas em formas

qualitativas. Essa conversão é essencial, pois permite que o sistema fuzzy lide com a incerteza e a imprecisão dos dados do mundo real, algo que não seria possível com valores exatos. No modelo ML TSKC-FS, essa conversão é realizada usando conjuntos fuzzy B_j^k.

Na estrutura do modelo ML-TSKC FS, cada B_j^k está associado a regras específicas e suas respectivas funções de pertinência $\mu_{B_j^k}$. Essas funções de pertinência desempenham um papel crucial ao medir o grau de relevância de um valor de entrada x_j dentro de um conjunto fuzzy B_j^k, determinando o quanto a entrada ativa a regra R^k. Para os subconjuntos fuzzy B_j^k de um universo U, suas funções de pertinência são definidas, para todos $x_j \in U$, por:

$$\mu_{B_j^k}(x_j) = \exp\left(-\frac{\left(x_j - v_j^k\right)^2}{2\left(\delta_j^k\right)^2}\right), \tag{3.2}$$

onde v_j^k e δ_j^k são, respectivamente, o centro e o desvio padrão da função Gaussiana. Esses parâmetros são calculados via o algoritmo Fuzzy C-Means (FCM) [Bezdek 1981], que ajusta a posição do centro v_j^k e a dispersão δ_j^k de acordo com os dados de entrada.

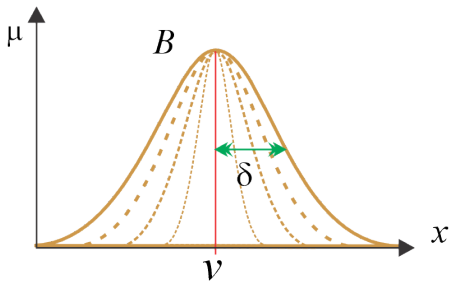


Figura 3.2: Gráfico da função Gaussiana, onde v é o centro e δ é o desvio padrão. O valor de pertinência μ decresce conforme x se afasta de v.

A função Gaussiana ilustrada na Figura 3.2 representa o comportamento de pertinência fuzzy: o ponto v é o centro do conjunto fuzzy, onde o grau de pertinência é máximo ($\mu = 1$). À medida que x se afasta de v, o valor de μ diminui exponencialmente, refletindo que o valor x_j está menos associado ao conjunto fuzzy. O desvio padrão δ determina a dispersão da curva, ou seja, quão rapidamente a pertinência decresce.

Exemplo 7 (Cálculo da Função de Pertinência)**.** *Considere os seguintes parâmetros para um conjunto fuzzy:*

$$v_j^k = 5, \quad \delta_j^k = 1$$

Para uma entrada $x_j = 5$, temos:

$$\mu_{B_j^k}(5) = \exp\left(-\frac{(5-5)^2}{2 \times 1^2}\right) = \exp(0) = 1$$

Nesse caso, $x_j = 5$ tem grau de pertinência máximo. Agora, para uma entrada $x_j = 4$, temos:

$$\mu_{B_j^k}(4) = \exp\left(-\frac{(4-5)^2}{2 \times 1^2}\right) = \exp\left(-\frac{1}{2}\right) \approx 0.606$$

Aqui, $x_j = 4$ está mais distante do centro $v_j^k = 5$, e, portanto, tem uma pertinência menor, mas ainda significativa.

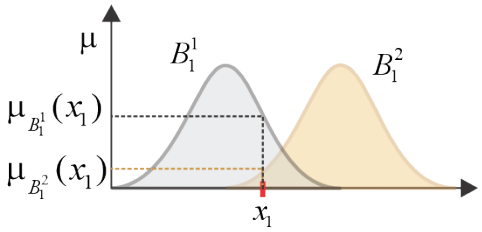


Figura 3.3: O processo de fuzzificação.

A Figura 3.3 mostra como uma entrada x_1 é processada no modelo ML-TSKC FS. O valor x_1 é fuzzificado pelas funções de pertinência $\mu_{B_1^1}(x_1)$ e $\mu_{B_2^1}(x_1)$, que atribuem diferentes graus de pertinência ao valor, dependendo da proximidade de x_1 em relação aos centros dos conjuntos fuzzy B_1^1 e B_2^1. Esse processo converte uma entrada precisa em uma representação fuzzy, permitindo que o sistema fuzzy manipule a incerteza associada à entrada.

Este processo é essencial para a operação do sistema fuzzy, pois permite que os dados reais sejam utilizados em inferências lógicas que capturam as nuances e incertezas do comportamento dos dados.

3.1.2 Camada 2: Determinação do peso da regra fuzzy pela sua força de ativação

O peso da regra desempenha um papel importante no sistema de inferência fuzzy TSK, sendo ativado quando todas as cláusulas antecedentes de uma regra fuzzy são satisfeitas. A força de ativação é definida como a quantificação da força da premissa de uma regra com base em um conjunto de valores de entrada, e é derivada das forças de pertinência dos valores de entrada correspondentes aos antecedentes da regra.

A força de ativação desempenha um papel crucial no sistema de inferência fuzzy, pois ela quantifica o grau de satisfação dos antecedentes de uma regra fuzzy. Quanto

maior a força de ativação, maior será a confiança de que a regra em questão deve ser ativada, o que afeta diretamente o valor da saída do sistema. Dessa forma, a força de ativação determina a relevância de cada regra fuzzy no cálculo da resposta final do sistema.

Em geral, a força de ativação de uma regra fuzzy R^k, com base em uma entrada x, é definida por meio de uma função de agregação $\mathcal{A} : [0,1]^A \to [0,1]$, que combina os graus de pertinência dos antecedentes para obter a força de ativação como:

$$\mu_{\mathcal{A}}^k(x) = \mathcal{A}\left(\mu_{B_1^k}(x_1), \ldots, \mu_{B_A^k}(x_A)\right). \tag{3.3}$$

Nesse contexto, $\mu_{B_j^k}(x_j)$ representa a força de pertinência do valor de entrada x_j para o antecedente B_j^k da regra R^k, sendo que a função de agregação $\mathcal{A}$ desempenha um papel crucial ao integrar essas forças para calcular a força de ativação final.

A função de agregação $\mathcal{A}$ é responsável por combinar os graus de pertinência presentes no antecedente de uma regra fuzzy, resultando em uma única força de ativação. Existem várias maneiras de definir essa função de agregação, como mínimo, produto, soma, média, ou até métodos mais complexos como a Integral de Choquet. A escolha da função de agregação pode impactar significativamente os resultados, já que cada método trata de forma diferente a combinação das pertinências.

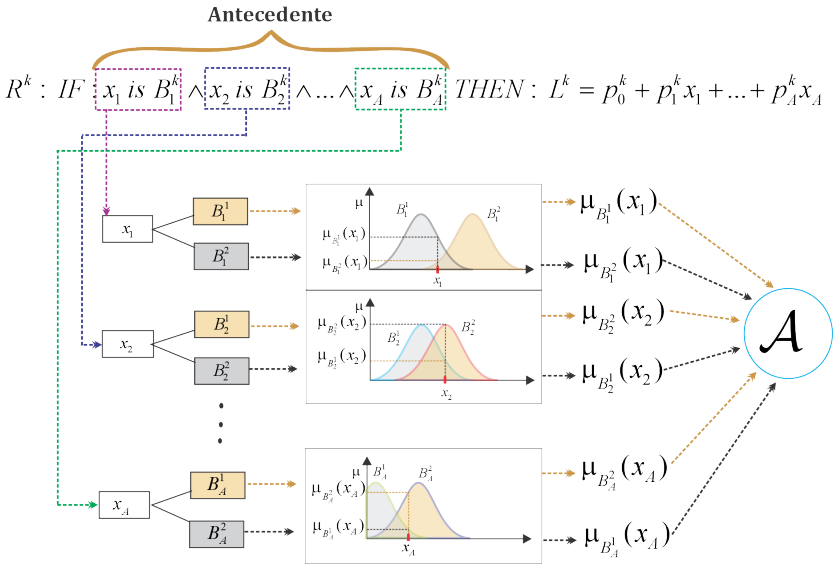


Figura 3.4: O processo da ativação das regras $\mu_{\mathcal{A}}^k(x)$.

A Figura 3.4 ilustra o processo de ativação de uma regra. No topo, temos os antecedentes da regra fuzzy, que são processados pelas funções de pertinência $\mu_{B_j^k}(x_j)$. Essas funções atribuem graus de pertinência a cada entrada x_j, dependendo da sua proximidade com os centros dos conjuntos fuzzy B_j^k. Em seguida, os valores resultantes dessas pertinências são agregados por uma função $\mathcal{A}$, resultando na força de ativação $\mu_{\mathcal{A}}^k(x)$, que é usada para determinar o impacto da regra fuzzy no sistema.

Trabalhos relacionados sobre a determinação das forças de ativação de regras fuzzy

Na literatura (por exemplo, [Jang e Jyh-Shing 1993; Kim e Kasabov 1999; Kasabov, Song e Qun 2002; Lou et al. 2021]), as funções de agregação $\mathcal{A}$ mais comuns na são as t-normas Mínimo e Produto (que também são funções de sobreposição [Bustince, Mesiar et al. 2021]). O Mínimo é usado quando a semântica da regra é "e", implicando que todas as cláusulas antecedentes devem ser verdadeiras para que a regra seja ativada. O Produto é utilizado quando a semântica da regra é "e também", indicando que as cláusulas antecedentes devem ser verdadeiras em conjunto, não individualmente.

Um dos trabalhos pioneiros que exploraram diferentes técnicas de agregação nos antecedentes, além das tradicionais, foi apresentado por [Uebele, Shigeo e Ming-Shong 1995], que introduziu um tratamento detalhado dos operadores de inferência fuzzy usados para calcular o grau de pertinência das regras fuzzy em sistemas de classificação. Esta abordagem é baseada em três principais funções de agregação, a saber, a t-norma Mínimo, a t-conorma Máximo e a Soma (de fato, a média aritmética), cada uma desempenhando um papel distinto no tratamento dos dados de entrada.

Utilizando o Mínimo, a força de ativação de uma regra fuzzy R^k, com base em uma entrada $\boldsymbol{x}$, é dada por:

$$\mu^k_{\min}(\boldsymbol{x}) = \min\left(\mu_{B^k_1}(x_1), \ldots, \mu_{B^k_A}(x_A)\right), \tag{3.4}$$

selecionando o menor grau de pertinência entre todos os fornecidos pelas funções de pertinência para uma regra específica. Embora seja eficaz para garantir a pertinência dentro dos limites da classe, essa abordagem pode desconsiderar informações valiosas sobre as relações dos dados de teste com outras regiões.

Agora, o Máximo identifica a regra fuzzy com o maior grau de pertinência para um dado vetor de entrada $\boldsymbol{x}$:

$$\mu^k_{\max}(\boldsymbol{x}) = \max\left(\mu_{B^k_1}(x_1), \ldots, \mu_{B^k_A}(x_A)\right). \tag{3.5}$$

Essa abordagem também pode desconsiderar informações valiosas sobre as relações dos dados de teste com outras fronteiras de regiões.

A Soma agrega os graus de pertinência, simulando a inferência de redes neurais e avaliando as distâncias médias dos dados de teste para os hiperplanos da região fuzzy, obtendo a força de ativação de uma regra fuzzy R^k com base em uma entrada $\boldsymbol{x}$ por:

$$\mu^k_{\sum}(\boldsymbol{x}) = \frac{1}{A}\sum_{j=1}^{A}\mu_{B^k_j}(x_j). \tag{3.6}$$

Essa soma fornece um grau de pertinência composto que pode refletir de maneira mais adequada a proximidade dos dados de teste à região de interesse.

Assim, essa abordagem de [Uebele, Shigeo e Ming-Shong 1995] ilustra a complexidade da classificação, destacando a necessidade de operadores de inferência que possam lidar adequadamente com incerteza e ambiguidade.

Na pesquisa realizada por [Bezdek, Keller et al. 1999] sobre sistemas fuzzy, é oferecida uma análise do processo de determinação do peso das regras em bases de regras fuzzy, com foco especial no lado esquerdo das regras, que abrange os componentes antecedentes. Nesta pesquisa, o cálculo da força de ativação de uma regra fuzzy R^k com base no vetor de entrada $\boldsymbol{x}$, é feito por

$$\mu_T^k(\boldsymbol{x}) = T\left(\mu_{B_1^k}(x_1), \ldots, \mu_{B_A^k}(x_A)\right) \tag{3.7}$$

onde $T\colon [0,1]^A \to [0,1]$ é uma t-norma. Observe que tanto o Mínimo quanto o Produto são t-normas. No entanto, outras t-normas diferentes podem ser usadas nesta abordagem.

A abordagem de [Chung et al. 2006] para o cálculo da força de ativação das regras no Sistema de Inferência Fuzzy Adaptativo Takagi-Sugeno-Kang (ATSFIS) destaca a flexibilidade dos sistemas fuzzy para modelar interações complexas entre variáveis de entrada. Ao utilizar uma função de pertinência do tipo sigmoide juntamente com o operador **ou interativo**, o sistema é capaz de capturar nuances nas relações entre as variáveis, o que é especialmente útil em aplicações onde as relações não são facilmente modeladas por operadores tradicionais de t-norma. Então, a força de ativação de uma regra fuzzy R^k com base no vetor de entrada $\boldsymbol{x}$ é calculada como:

$$\mu_*^k(\boldsymbol{x}) = \mu_{B_1^k}(x_1) * \mu_{B_2^k}(x_2) * \ldots * \mu_{B_A^k}(x_A), \tag{3.8}$$

onde $\mu_{B_i^k}(x_i)$ denota uma função de pertinência do tipo sigmoide para a i-ésima variável de entrada na k-ésima regra, e $*\colon [0,1]^2 \to [0,1]$ é um operador fuzzy específico definido, para todos $x, y \in [0,1]$, por:

$$x * y = \frac{xy}{(1-x)(1-y) + xy}. \tag{3.9}$$

Este operador, chamado *ou interativo*, conforme comentado por [Chung et al. 2006], fornece uma maneira inovadora de calcular a interação entre as funções de pertinência das variáveis de entrada. Ao contrário dos operadores tradicionais de t-norma, que geralmente assumem o mínimo ou o produto das funções de pertinência, o operador ou interativo permite uma representação mais rica da interação entre variáveis, levando em consideração sua coexistência e o grau de interação entre elas.

O estudo de [Han, Sun e Fan 2008] explora a estrutura e a funcionalidade de uma rede neural fuzzy aprimorada com base no modelo Takagi-Sugeno (T-S). Ele

enfatiza a metodologia para o cálculo da força de ativação e a importância das regras nos sistemas de inferência fuzzy. Primeiro, para cada regra R^k, o parâmetro α_k é calculado usando a t-norma do mínimo ou do produto, como:

$$\alpha_k(\boldsymbol{x}) = \min\left(\mu_{B_1^k}(x_1), \ldots, \mu_{B_A^k}(x_A)\right) \tag{3.10}$$

$$\alpha_k(\boldsymbol{x}) = \prod\left(\mu_{B_1^k}(x_1), \ldots, \mu_{B_A^k}(x_A)\right). \tag{3.11}$$

A importância da regra, denotada por $w_k \geq 1$, atua como um peso atribuído a cada regra R^k, representando sua relevância ou confiança dentro do sistema de regras. Essa metodologia permite diferenciar a influência de cada regra no resultado do sistema de inferência, refletindo a confiança ou prioridade de certas regras sobre outras. Essa diferenciação é crucial para ajustar o sistema de inferência a fim de capturar nuances específicas do domínio da aplicação. Assim, no modelo de [Han, Sun e Fan 2008], a força de ativação de uma regra fuzzy R^k com base no vetor de entrada $\boldsymbol{x}$ é calculada como:

$$\mu^k(\boldsymbol{x}) = \alpha_k(\boldsymbol{x}) w_k(\boldsymbol{x}). \tag{3.12}$$

Assim, os trabalhos citados acima não apenas esclarecem a importância de selecionar operadores de inferência apropriados para sistemas de regras fuzzy, mas também estabelecem uma base comparativa para a implementação de métodos avançados de agregação, como nossa proposta: a integral discreta de Choquet. Esse avanço abre caminho para sistemas de inferência fuzzy mais precisos e adaptativos, capazes de lidar com a complexidade e ambiguidade dos dados do mundo real de forma eficiente.

Nossa proposta de aplicação da integral de Choquet no ML-TSKC FS

Sistemas de regras fuzzy são ferramentas poderosas para modelar relações complexas. No entanto, os métodos tradicionais de agregação em regras fuzzy frequentemente assumem independência entre os atributos antecedentes (a parte **SE**). Essa limitação pode prejudicar a capacidade do sistema de capturar cenários do mundo real onde os atributos podem interagir e influenciar uns aos outros.

Nosso trabalho aborda essa limitação ao introduzir a integral de Choquet para o cálculo da força de ativação das regras fuzzy. Essa abordagem é inspirada pelos avanços recentes nos consequentes de regras fuzzy utilizando a integral discreta de Choquet (e algumas generalizações), conforme demonstrado em [Lucca, Sanz, G. Dimuro et al. 2018; Lucca, G. P. Dimuro et al. 2019; Marco-Detchart et al. 2021; Wieczynski, Fumanal-Idocin et al. 2022; Wieczynski, Lucca et al. 2023; Ferrero-Jaurrieta et al. 2023; Kim e Lee-Chae 2023; Riaz et al. 2023; Hongjuan Wang, Liu e Zhao 2023; Wang et al. 2024; Bozyiğit et al. 2024; Zhang, Mesiar e Pap 2024],

que exploraram sua aplicação em consequentes de regras fuzzy em vários contextos (por exemplo, processamento de imagens, Long Short-Term Memory, tomada de decisão multicritério e TOPSIS, classificação, interfaces cérebro-computador, reconhecimento de padrões, gestão de projetos e análise de riscos). Portanto, nosso trabalho dá um passo significativo ao aproveitar as capacidades da integral de Choquet nos antecedentes das regras (parte **SE**).

A integral de Choquet é uma ferramenta matemática poderosa que vai além dos operadores tradicionais de agregação (como mínimo, máximo ou soma) utilizados em sistemas de inferência fuzzy. Enquanto esses operadores se concentram nos graus de pertinência individuais das variáveis, a integral de Choquet, por ser definida em termos de uma medida fuzzy, captura a relação entre os atributos, considerando a importância relativa de cada combinação de variáveis [Lucca, Sanz, Dimuro et al. 2019; Wieczynski, Dimuro et al. 2020].

Então, a força de ativação de uma regra fuzzy R^k com base no vetor de entrada $\boldsymbol{x}$, utilizando a integral de Choquet $\mathfrak{C}_m\colon [0,1]^A \to [0,1]$ com respeito a uma medida fuzzy $\mathfrak{m}$ como a função de agregação $\mathcal{A}$ da Eq.(3.3), é dada por:

$$\mu^k_{\mathfrak{C}_\mathfrak{m}}(\boldsymbol{x}) = \mathfrak{C}_\mathfrak{m}\left(\mu_{B^k_1}(x_1), \ldots, \mu_{B^k_A}(x_A)\right), \tag{3.13}$$

onde $\mu_{B^k_i}$ denota a força de ativação da regra k para cada atributo x_i na regra k.

Neste estudo, exploramos várias medidas fuzzy integradas no modelo ML-TKSC FS. Como mencionado anteriormente, essas medidas são fundamentais para modelar as interações entre os atributos, influenciando diretamente o processo de classificação. Considerando um conjunto $N = 1, \ldots, n$ e um subconjunto $I \subseteq N$, juntamente com um vetor de pesos associado à medida ponderada, as medidas fuzzy selecionadas são definidas na Tabela 2.10.

A seleção dessas medidas é motivada por sua aplicabilidade e versatilidade na modelagem da influência dos atributos na classificação, conforme destacado por [H. Bustince et al. 2016]. Cada medida oferece uma abordagem distinta para quantificar e integrar características relevantes, enriquecendo o processo de agregação e aprimorando o desempenho do modelo ML-TKSC FS. Uma breve discussão de cada medida é dada a seguir.

- *Medida Uniforme* ($\mathfrak{m}_U$): distribui igual importância a todos os atributos, sendo útil na ausência de conhecimento prévio sobre a relevância dos atributos. Ela favorece uma agregação equitativa, o que pode ser limitado em casos onde alguns atributos são mais informativos.

- *Medida Relativa* ($\mathfrak{m}_R$): valoriza os atributos com base em sua ordem, assumindo que os atributos de maior índice são mais relevantes. Isso pode aumentar a precisão quando uma hierarquia de importância entre os atributos é conhecida.

- *Medida do Produto* ($\mathfrak{m}_{\Pi}$): enfatiza a combinação de atributos de alta ordem, potencialmente capturando interações complexas essenciais para a classificação.
- *Medida de Potência* ($\mathfrak{m}_p$): modula a influência do tamanho do conjunto de atributos, permitindo ajustes na sensibilidade do modelo ao número de atributos envolvidos.
- *Média Ponderada* ($\mathfrak{m}_w$): oferece flexibilidade ao permitir pesos específicos para cada atributo, refletindo conhecimento ou hipóteses sobre sua importância relativa.

Cada medida traz uma perspectiva única para interpretar e modelar a importância dos atributos em contextos de classificação multi-rótulo. A seleção dessas medidas é guiada tanto pelo conhecimento do domínio quanto pela experimentação, visando otimizar a combinação de informações no ML-TKSC FS para alcançar um desempenho ideal.

3.1.3 Camada 3: Normalização

Esse processo envolve dividir a força de ativação de cada regra pela soma das forças de ativação de todas as regras ativadas para a entrada atual. Essa operação resulta na forma normalizada da força de ativação da regra R^k, com base no vetor de entrada $\mathbf{x}$, dada por:

$$\tilde{\mu}^k_{\mathfrak{C}_{\mathfrak{m}}}(\boldsymbol{x}) = \frac{\mu^k_{\mathfrak{C}_{\mathfrak{m}}}(\boldsymbol{x})}{\sum_{i=1}^{K} \mu^i_{\mathfrak{C}_{\mathfrak{m}}}(\boldsymbol{x})}. \tag{3.14}$$

Aqui, $\tilde{\mu}^k_{\mathfrak{C}_{\mathfrak{m}}}(\boldsymbol{x})$ representa a proporção da força de ativação da regra R^k em relação à força de ativação total das regras ativadas para a entrada $\boldsymbol{x}$.

3.1.4 Camada 4: Contribuição da regra (consequente)

O consequente, também conhecido como a parte ***então*** de uma regra fuzzy, desempenha um papel crucial no modelo ML-TSKC FS ao determinar a contribuição da regra para a saída final para uma determinada entrada. A contribuição da k-ésima regra R^k, para o vetor de entrada $\boldsymbol{x} = (x_1, \ldots, x_A)$ e o vetor de parâmetros da regra $\boldsymbol{p}^k = (p^k_0, \ldots, p^k_A)$, no modelo ML-TSKC FS, é dada por:

$$L^k(\boldsymbol{x}, \boldsymbol{p}^k) = p^k_0 + p^k_1 x_1 + \cdots + p^k_A x_A, \quad k = 1, \ldots, K \tag{3.15}$$

onde:

- p^k_0 é o termo constante associado à k-ésima regra. Ele atua como um termo de viés, influenciando a saída mesmo quando todas as variáveis de entrada são zero.

- p_j^k denota os coeficientes lineares vinculados à j-ésima variável de entrada x_j dentro da k-ésima regra. Ele essencialmente determina o peso ou a influência de cada variável de entrada na saída da regra.

Essencialmente, a Eq.(3.15) representa uma combinação linear das variáveis de entrada, ponderada pelos seus coeficientes correspondentes, juntamente com um termo constante. O valor resultante indica a saída à qual a k-ésima regra se aplica para a entrada dada.

3.1.5 Camada 5: Ponderação

Na ponderação, o modelo ML-TSKC FS combina as contribuições das regras fuzzy, ponderadas pelas suas respectivas forças de ativação, para obter a saída final. Essa etapa permite agregar a informação proveniente de todas as regras de forma ponderada, refletindo a relevância de cada regra para a entrada dada.

A saída de uma instância $\boldsymbol{x}$ no modelo ML-TSKC FS pode ser expressa como uma combinação ponderada das saídas parciais de todas as regras R^k, dada por:

$$\hat{\boldsymbol{y}} = \sum_{k=1}^{K} \tilde{\mu}_{\mathfrak{C}_\mathfrak{m}}^k(\boldsymbol{x}) L^k(\boldsymbol{x}, \boldsymbol{p}^k),$$

onde:

- $\tilde{\mu}_{\mathfrak{C}_\mathfrak{m}}^k(\boldsymbol{x})$: representa a força de ativação normalizada da k-ésima regra fuzzy para a entrada $\boldsymbol{x}$. Esse valor atua como um peso, refletindo o grau de importância da regra R^k para a situação atual.

- $L^k(\boldsymbol{x}, \boldsymbol{p}^k)$: representa a contribuição linear da regra R^k para a saída final, onde $\boldsymbol{p}^k$ é o vetor de parâmetros específicos dessa regra.

Assim, cada regra contribui para a saída ponderada de acordo com sua relevância, determinada por $\tilde{\mu}_{\mathfrak{C}_\mathfrak{m}}^k(\boldsymbol{x})$.

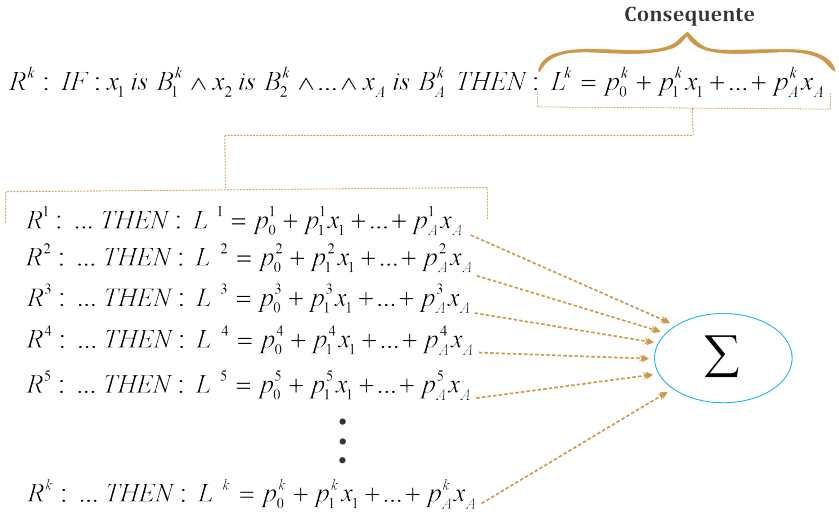


Figura 3.5: Ilustração da agregação das regras para obter a saída final.

A Figura 3.5 ilustra o processo de agregação das regras fuzzy. Cada linha representa uma regra R^k, que é ativada de acordo com a força de ativação $\tilde{\mu}^k_{\mathfrak{C}_m}(\boldsymbol{x})$. A contribuição de cada regra é dada pela função $L^k(\boldsymbol{x}, \boldsymbol{p}^k)$, e todas as contribuições são somadas (Σ) para formar a saída final $\hat{\boldsymbol{y}}$. Esse processo reflete a soma ponderada das saídas das regras, onde cada regra contribui de acordo com sua relevância para a entrada.

3.2 O MÉTODO DE APRENDIZAGEM

O processo de aprendizagem do modelo ML-TSKC FS é estruturado em duas fases, cada uma projetada para otimizar as capacidades do modelo. **Na primeira fase**, um algoritmo de agrupamento Fuzzy C-Means [Bezdek, Ehrlich e Full 1984] é utilizado para estabelecer funções de pertinência que refletem com precisão as características de distribuição dos dados de entrada. Esta etapa é fundamental para garantir que as funções de pertinência estejam bem ajustadas à estrutura natural dos dados.

Uma vez que as funções de pertinência são mapeadas de forma eficaz, o modelo passa para **a segunda fase**, que se concentra no aprendizado e refinamento dos parâmetros do modelo por meio de otimização. Esta fase é guiada por uma função objetivo que integra três componentes essenciais, cada um desempenhando um papel vital no processo de aprendizagem do modelo:

- *Termo de Perda por Regressão*: Este componente garante que o modelo capture com precisão as nuances e padrões presentes nos dados de treinamento, alinhando as previsões do modelo de forma próxima aos resultados observados.

- *Termo de Regularização*: Este termo penaliza modelos excessivamente complexos para evitar o sobreajuste e manter a generalização do modelo. Ele incentiva a simplicidade e reduz o risco de o modelo se tornar muito ajustado aos dados de treinamento.

- *Termo de Correlação*: Incorporando a correlação entre os rótulos, este termo extrai informações adicionais que aprimoram a capacidade do modelo de fazer previsões mais informadas, especialmente em contextos de múltiplos rótulos.

A otimização desta função objetivo é realizada utilizando o algoritmo de descida de gradiente proximal. O algoritmo ajusta iterativamente os parâmetros do modelo, refinando-os para minimizar a função objetivo. Através deste processo iterativo, o modelo aprende os melhores parâmetros possíveis, ao mesmo tempo que alcança um equilíbrio entre a precisão do ajuste, a simplicidade do modelo e a exploração das correlações entre os rótulos. Isso resulta em um desempenho robusto e confiável tanto nos dados de treinamento quanto nos dados não vistos.

3.2.1 Fase 1 - Encontrando as saídas desejadas usando Fuzzy C-Means (FCM)

O algoritmo Fuzzy C-Means é uma extensão do clássico algoritmo de clustering, projetado para lidar com a incerteza e a sobreposição entre grupos. Ao contrário de métodos como K-Means, que forçam cada ponto de dado a pertencer exclusivamente a um cluster, o FCM permite que cada ponto tenha um grau de pertinência associado a múltiplos clusters. Isso torna o FCM uma escolha ideal para sistemas fuzzy, onde a incerteza é uma parte inerente do modelo.

Para iniciar a primeira fase, é essencial definir o conjunto de dados de treinamento que servirá como base para o processo de agrupamento. Seja $\boldsymbol{D_{TR}}$ o conjunto de dados de treinamento, definido da seguinte forma:

$$\boldsymbol{D_{TR}} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}\,.$$

O conjunto de dados de treinamento D_{TR} consiste em N pares de vetores de entrada x_i, que representam os pontos de dado no espaço de entrada, e suas saídas associadas y_i, que correspondem às classes ou rótulos desejados. O objetivo do FCM é identificar os centros dos clusters com base nos pontos de entrada, ajustando-os iterativamente representar adequadamente a distribuição dos dados.

Processo de Inicialização e Iteração

O processo começa com a inicialização aleatória dos centros dos clusters para os pontos de dados $\{x_1, \ldots, x_N\}$, e refina iterativamente esses centros. A cada iteração, o FCM calcula o grau de pertinência u_{ij} para cada ponto de dado em relação a

cada cluster, utilizando esses graus para atualizar os centros dos clusters. A natureza iterativa do FCM garante que os centros finais dos clusters v_j e as respectivas dispersões σ_j reflitam com precisão a distribuição dos dados.

Inicialmente, os centros dos clusters v_i são atribuídos aleatoriamente. A cada iteração, os centros dos clusters são atualizados com base no grau de pertinência u_{ij}, que reflete a proximidade de cada ponto x_i a um centro v_j. Após várias iterações, os clusters convergem para uma posição final que reflete melhor a estrutura dos dados.

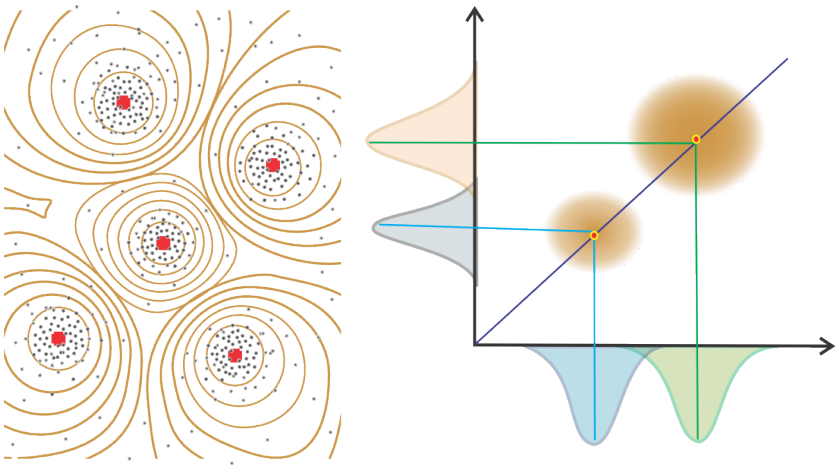


Figura 3.6: Representação gráfica do Fuzzy C-Means (FCM).

A Figura 3.6 ilustra o processo de agrupamento fuzzy. No gráfico à esquerda, as linhas concêntricas indicam os graus de pertinência de cada ponto de dado aos centros dos clusters. Quanto mais próximo um ponto está do centro, maior o grau de pertinência. O gráfico à direita mostra uma representação tridimensional dos clusters e suas funções de pertinência Gaussianas, com σ_j controlando a largura de cada Gaussiana.

Cálculo do grau de pertinência (u_{ij}), centros dos clusters (v_j) e desvio padrão(σ_j)

Em cada iteração, o grau de pertinência u_{ij} de cada ponto de dado x_i ao cluster j é calculado da seguinte forma:

$$u_{ij} = \frac{1}{\sum_{k=1}^{K} \left(\frac{\|x_i - v_j\|}{\|x_i - v_k\|} \right)^{\frac{2}{m-1}}}$$

onde v_j é o centro do cluster e m é o parâmetro de fuzzificação. O grau de pertinência de um ponto x_i ao cluster j indica o quão próximo o ponto está do centro v_j. Valores maiores de u_{ij} indicam maior pertencimento ao cluster.

Os centros dos clusters obtidos pelo FCM são calculados como:

$$v_j = \frac{\sum_{i=1}^{N} u_{ij}^m \cdot x_i}{\sum_{i=1}^{N} u_{ij}^m}$$

Os centros dos clusters v_j são calculados como a média ponderada dos pontos de dado x_i, onde os pesos são os graus de pertinência u_{ij}. O desvio padrão σ_j para cada cluster é determinado com base na distância média dos pontos de dados ao centro do cluster, ponderada pelos graus de pertinência, conforme a fórmula:

$$\sigma_j^2 = \frac{\sum_{i=1}^{N} u_{ij}^m \cdot \|x_i - v_j\|^2}{\sum_{i=1}^{N} u_{ij}^m}$$

Esse valor quantifica a dispersão dos pontos em torno do centro do cluster e define a largura das funções de pertinência Gaussianas. Quanto maior σ_j, maior a variabilidade nos dados, resultando em funções de pertinência mais amplas.

Esses resultados são fundamentais para definir as funções de pertinência Gaussianas que o modelo fuzzy utilizará para representar as relações entre os dados de entrada de maneira fuzzy. Isso permite que o modelo aproveite as informações estruturais capturadas durante o agrupamento para um aprendizado mais preciso e robusto. Os centros dos clusters v_j e os desvios σ_j serão usados na próxima etapa para determinar as funções de pertinência fuzzy que serão aplicadas nas regras do sistema.

Agora, prosseguimos para determinar as saídas desejadas utilizando as funções de pertinência contidas nos pesos das regras normalizadas $\tilde{\mu}_{\mathfrak{C}_\mathfrak{m}}^k$ definidas em Eq.(3.14). Essas funções são parametrizadas por v_j^k e $\delta_j^k = h\sigma_j^k$, onde h desempenha um papel crucial na escala do desvio padrão σ_j^k obtido a partir do algoritmo FCM para cada regra R^k. O valor de h não é fixo. Em vez disso, ele é tratado como um hiperparâmetro que é ajustado por meio de um processo de busca em grade, junto com outros hiperparâmetros importantes. Essa busca em grade permite a exploração sistemática de diferentes valores de h, otimizando o desempenho do modelo ao identificar a melhor combinação de hiperparâmetros.

Para uma instância arbitrária $(\boldsymbol{x}, \boldsymbol{y})$ do conjunto de treinamento $\boldsymbol{D_{TR}}$, quando a entrada $\boldsymbol{x}$ é inserida no modelo ML-TSKC-FS, a saída $\hat{\boldsymbol{y}}$ é determinada da seguinte forma:

$$\hat{\boldsymbol{y}} = \sum_{k=1}^{K} \tilde{\mu}_{\mathfrak{C}_\mathfrak{m}}^k (\boldsymbol{x})\, L^k \left(\boldsymbol{x}, \boldsymbol{p}^k\right),$$

onde $\tilde{\mu}_{\mathfrak{C}_\mathfrak{m}}^k (\boldsymbol{x})$ representa a força de ativação da k-ésima regra fuzzy e $L^k \left(\boldsymbol{x}, \boldsymbol{p}^k\right)$ é a função linear associada a essa regra fuzzy. A saída $\hat{\boldsymbol{y}}$ é calculada como uma combinação ponderada das regras fuzzy, onde $\tilde{\mu}_{\mathfrak{C}_\mathfrak{m}}^k (\boldsymbol{x})$ pondera a contribuição de cada regra fuzzy. A função linear $L^k \left(\boldsymbol{x}, \boldsymbol{p}^k\right)$ é expressa como:

$$L^k\left(\boldsymbol{x}, \boldsymbol{p}^k\right) = (\boldsymbol{p}^k)^T \begin{pmatrix} 1 \\ \boldsymbol{x} \end{pmatrix},$$

onde $\boldsymbol{p}^k$ é o vetor de coeficientes da função linear L^k e $\boldsymbol{x}$ é o vetor de entrada de dimensão A. Assim, a saída $\hat{\boldsymbol{y}}$ é uma soma ponderada das saídas lineares de cada regra fuzzy. Para ilustrar, o vetor $\boldsymbol{p}^k$ contém os coeficientes que definem a função linear associada a cada regra fuzzy, determinando a contribuição da regra para a saída final. Os componentes de $\boldsymbol{x}$ representam as variáveis de entrada do sistema.

Podemos reescrever a equação da saída $\hat{\boldsymbol{y}}$ usando a função $\boldsymbol{g}^k(x)$, que combina a força de ativação fuzzy $\tilde{\mu}^k_{\mathfrak{C}_\mathfrak{m}}(\boldsymbol{x})$ e a função linear da regra fuzzy R^k, como:

$$\hat{\boldsymbol{y}} = \sum_{k=1}^{K} (\boldsymbol{p}^k)^T \boldsymbol{g}^k(x),$$

onde $\boldsymbol{g}^k(x) = \left(\tilde{\mu}^k_{\mathfrak{C}_\mathfrak{m}}(\boldsymbol{x}) \begin{pmatrix} 1 \\ \boldsymbol{x} \end{pmatrix}\right)$. A função $\boldsymbol{g}^k(x)$ encapsula a contribuição de cada regra fuzzy para a saída final, permitindo que a saída $\hat{\boldsymbol{y}}$ seja expressa como uma soma ponderada das funções $\boldsymbol{g}^k(x)$.

Seja $\boldsymbol{g}(x) = (\boldsymbol{g}^1(x), \boldsymbol{g}^2(x), \ldots, \boldsymbol{g}^K(x))^T$, então a saída $\hat{\boldsymbol{y}}$ pode ser reescrita de forma compacta como:

$$\hat{\boldsymbol{y}} = \sum_{k=1}^{K} (\boldsymbol{p}^k)^T \boldsymbol{g}^k(x) = \boldsymbol{P}^T \boldsymbol{g}(x),$$

onde:

$$\boldsymbol{g}(x) = \begin{pmatrix} \boldsymbol{g}^1(x) \\ \boldsymbol{g}^2(x) \\ \vdots \\ \boldsymbol{g}^K(x) \end{pmatrix}, \quad \boldsymbol{P}^T = \begin{pmatrix} (\boldsymbol{p}^1)^T & (\boldsymbol{p}^2)^T & \ldots & (\boldsymbol{p}^K)^T \end{pmatrix}.$$

Portanto, encontramos que $\hat{\boldsymbol{y}} = \boldsymbol{P}^T \boldsymbol{g}(x)$, ou seja, para a entrada dada $\boldsymbol{x}$, a saída do modelo ML-TSKC-FS é uma função linear dos parâmetros $\boldsymbol{P}$. Agora, para todas as entradas $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$, as saídas correspondentes são calculadas como:

$$\hat{\boldsymbol{y}}_i = \boldsymbol{P}^T \boldsymbol{g}(\boldsymbol{x}_i), \quad i = 1, 2, \ldots, N.$$

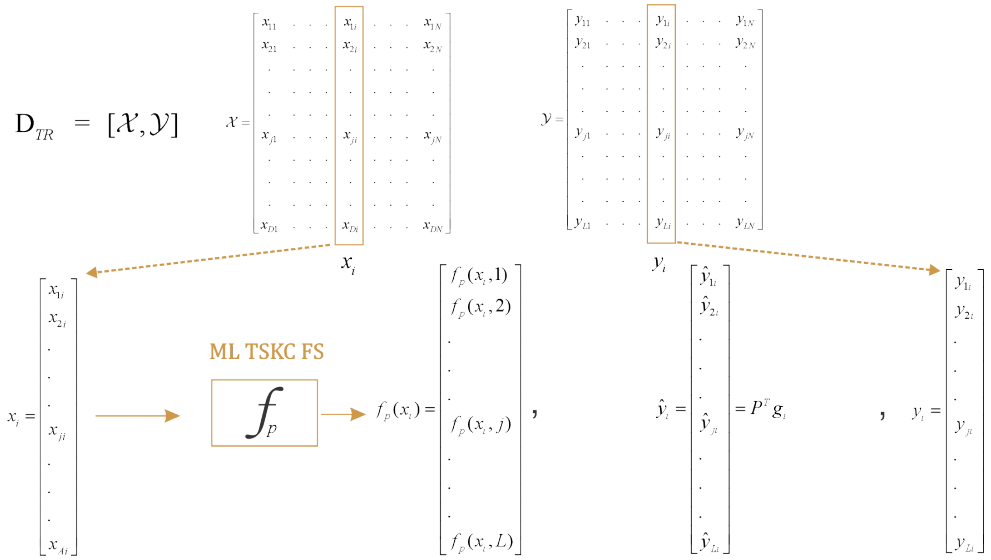


Figura 3.7: Representação matricial da saída do modelo ML-TSKC-FS.

A Figura 3.7 ilustra o processo de cálculo da saída de forma matricial. A matriz de entrada X contém os vetores de entrada $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$, e a matriz de saída Y contém os vetores de saída correspondentes $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N$. O modelo ML-TSKC-FS aplica uma função f parametrizada pelos vetores $\boldsymbol{P}$ e $\boldsymbol{g}(x)$, resultando nas saídas previstas $\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2, \ldots, \hat{\boldsymbol{y}}_N$. Essa representação visual destaca a relação entre as entradas e as saídas através do cálculo matricial.

Saída do modelo $\hat{\boldsymbol{Y}}$

Combinamos todas as saídas individuais $\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2, \ldots, \hat{\boldsymbol{y}}_N$ em uma única matriz $\hat{\boldsymbol{Y}}$, onde:

$$\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2, \ldots, \hat{\boldsymbol{y}}_N) = \boldsymbol{P}^T \left(\boldsymbol{g}(\boldsymbol{x}_1)\ \boldsymbol{g}(\boldsymbol{x}_2)\ \ \ldots\ \ \boldsymbol{g}(\boldsymbol{x}_N)\right)$$

ou simplesmente:

$$\hat{\boldsymbol{Y}} = \boldsymbol{P}^T \boldsymbol{G},$$

onde $\boldsymbol{G} = (\boldsymbol{g}(\boldsymbol{x}_1)\ \boldsymbol{g}(\boldsymbol{x}_2)\ \ \ldots\ \ \boldsymbol{g}(\boldsymbol{x}_N)$.

O objetivo do treinamento é minimizar a diferença entre $\hat{\boldsymbol{Y}}$ e a matriz de saídas verdadeiras $\boldsymbol{Y}$, ajustando os parâmetros $\boldsymbol{P}$ para garantir que as previsões estejam o mais próximo possível dos valores reais.

3.2.2 Fase 2 - Encontrando os parâmetros das partes consequentes para minimizar o erro da saída da rede

Com base na análise acima, consideramos a matriz de parâmetros consequentes $\boldsymbol{P}$ como a variável independente da função objetivo $O_f(\boldsymbol{P})$ para o ML-TSKC FS, que

tem a seguinte forma:

$$O_f(\boldsymbol{P}) = \frac{1}{2}\left\|\boldsymbol{P}^{\mathrm{T}}\boldsymbol{G} - \boldsymbol{Y}\right\|_F^2 + \beta\left\|\boldsymbol{P}\right\|_1 + \frac{\alpha}{2}\operatorname{Tr}\left(\boldsymbol{R}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P}\right), \tag{3.16}$$

onde $\left\|\boldsymbol{P}^{\mathrm{T}}\boldsymbol{G} - \boldsymbol{Y}\right\|_F^2$ é a perda por regressão e $||\cdot||_F$ é a norma de Frobenius; $\|\boldsymbol{P}\|_1$ é um termo de regularização $\operatorname{Tr}\left(\boldsymbol{R}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P}\right)$ é o termo que contém a correlação entre os rótulos. α e β são dois hiperparâmetros.

Agora, como a função objetivo do ML-TSKC-FS contém a norma L_1 de $\boldsymbol{P}$, que é não diferenciável em relação a $\boldsymbol{P}$, não podemos obter diretamente os gradientes em $\boldsymbol{P}$ para otimização. Técnicas eficientes de otimização foram desenvolvidas para resolver esse problema comum em métodos baseados na norma L_1. Neste trabalho, utilizamos o método de Descida de Gradiente Proximal [Combettes e Wajs 2005], que é empregado para resolver a matriz $\boldsymbol{P}$ do ML-TSKC-FS. O processo de otimização é descrito a seguir.

Os parâmetros ótimos de nosso modelo são obtidos minimizando a Eq. (3.16) e podem ser reescritos como:

$$\begin{aligned}\boldsymbol{P}^* &= \arg\min_{\boldsymbol{P}} \frac{1}{2}\left\|\boldsymbol{P}^{\mathrm{T}}\boldsymbol{G} - \boldsymbol{Y}\right\|_F^2 + \beta\left\|\boldsymbol{P}\right\|_1 + \frac{\alpha}{2}\operatorname{Tr}\left(\boldsymbol{R}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P}\right) \\ &= \arg\min_{\boldsymbol{P}} f(\boldsymbol{P}) + \beta\left\|\boldsymbol{P}\right\|_1,\end{aligned} \tag{3.17}$$

com

$$f(\boldsymbol{P}) = \frac{1}{2}\left\|\boldsymbol{P}^{\mathrm{T}}\boldsymbol{G} - \boldsymbol{Y}\right\|_F^2 + \frac{\alpha}{2}\operatorname{Tr}\left(\boldsymbol{R}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P}\right). \tag{3.18}$$

Como tanto $f(\boldsymbol{P})$ quanto a norma L_1 são convexos, e $\beta > 0$, o problema de otimização na Eq. (3.17) também é convexo. Além disso, a função na Eq. (3.18) é convexa e diferenciável,

$$\nabla f(\boldsymbol{P}) = GG^T P - GY^T + \alpha PR. \tag{3.19}$$

Além disso, $f(\boldsymbol{P})$ satisfaz

$$\left\|\nabla f\left(\boldsymbol{P_1}\right) - \nabla f\left(\boldsymbol{P_2}\right)\right\| \leq L_f\left\|\boldsymbol{P_1} - \boldsymbol{P_2}\right\|, \qquad \forall \boldsymbol{P_1}, \boldsymbol{P_2}, \tag{3.20}$$

onde

$$L_f = \sqrt{2\sigma_{\max}^2\left(GG^T\right) + 2\sigma_{\max}^2(\alpha R)}, \tag{3.21}$$

e $\sigma_{\max}(\cdot)$ indica o maior componente da matriz.

Assim, a Eq. (3.17) pode ser resolvida iterativamente. Para a t-ésima iteração, dado o ponto fixo $\boldsymbol{P}^{(t)}$, $f(\boldsymbol{P})$ pode ser aproximada usando uma expansão de Taylor

de segunda ordem:

$$\begin{aligned}\hat{f}(\boldsymbol{P}) &\simeq f\left(\boldsymbol{P}^{(t)}\right) \\ &+\left\langle \nabla f\left(\boldsymbol{P}^{(t)}\right), \boldsymbol{P}-\boldsymbol{P}^{(t)}\right\rangle + \frac{L_f}{2}\left\|\boldsymbol{P}-\boldsymbol{P}^{(t)}\right\|_F^2 \\ &= \frac{L_f}{2}\left\|\boldsymbol{P}-\left(\boldsymbol{P}^{(t)}-\frac{1}{L_f}\nabla f\left(\boldsymbol{P}^{(t)}\right)\right)\right\|_F^2 + C_0\end{aligned} \tag{3.22}$$

onde C_0 é uma constante independente de $\boldsymbol{P}$. Portanto, para a $(t+1)$-ésima iteração, a Eq. (3.17) pode ser aproximada por:

$$\begin{aligned}\boldsymbol{P}_{t+1} &= \arg\min_{\boldsymbol{P}} \hat{f}(\boldsymbol{P}) + \beta\|\boldsymbol{P}\|_1 \\ &= \arg\min_{\boldsymbol{P}} \frac{L_f}{2}\left\|\boldsymbol{P}-\boldsymbol{Z}^{(t)}\right\|_F^2 + \beta\left\|\boldsymbol{P}\right\|_1\end{aligned} \tag{3.23}$$

onde $\boldsymbol{Z}^{(t)} = \boldsymbol{P}^{(t)} - \nabla f\left(\boldsymbol{P}^{(t)}\right)/L_f$. Então, a Eq. (3.23) pode ser resolvida pela seguinte regra de atualização:

$$\boldsymbol{P}_{t+1} = S_{\beta/L_f}\left[\boldsymbol{Z}^{(t)}\right] \tag{3.24}$$

onde $S_{\beta/L_f}\left[\boldsymbol{Z}^{(t)}\right]$ é a função de limiar suave, definida para $\boldsymbol{Z}^{(t)} = [z_{ij}]$ e β/L_f como segue:

$$\left(S_{\beta/L_f}\left[\boldsymbol{Z}^{(t)}\right]\right)_{ij} = \begin{cases} z_{ij} - \beta/L_f, & \text{se } z_{ij} > \beta/L_f \\ z_{ij} + \beta/L_f, & \text{se } z_{ij} < -\beta/L_f \\ 0, & \text{caso contrário.} \end{cases} \tag{3.25}$$

Além disso, para obter a solução ótima da Eq. (3.17) de forma mais eficiente, primeiro obtemos a solução considerando apenas o primeiro termo da Eq. (3.16) e tomamos isso como o valor inicial (ou seja, o valor inicial de $\boldsymbol{P}$) para as iterações subsequentes no processo de aprendizado para resolver a Eq. (3.17). O processo ocorre da seguinte forma.

A derivada em relação a $\boldsymbol{P}$ é dada por

$$\nabla f(\boldsymbol{P}) = \boldsymbol{G}\boldsymbol{G}^T\boldsymbol{P} - \boldsymbol{G}\boldsymbol{Y}^T + \alpha\boldsymbol{P}\boldsymbol{R}. \tag{3.26}$$

A partir de $\nabla f(\boldsymbol{P}) = 0$, podemos derivar a seguinte aproximação:

$$\boldsymbol{P} = 2\left(\boldsymbol{G}\boldsymbol{G}^T\right)^{-1}\boldsymbol{G}\boldsymbol{Y}^T. \tag{3.27}$$

Portanto, definimos o valor inicial de $\boldsymbol{P}$ como

$$\boldsymbol{P}_0 = 2\left(\boldsymbol{G}\boldsymbol{G}^T + \gamma\boldsymbol{I}\right)^{-1}\boldsymbol{G}\boldsymbol{Y}^T, \tag{3.28}$$

onde γ é um hiperparâmetro de regularização, conforme definido para evitar problemas numéricos associados à inversão de matrizes próximas da singularidade. Assim, a introdução de $\gamma \boldsymbol{I}$ contribui para a estabilidade da solução, adicionando um termo diagonal que melhora as propriedades de condicionamento da matriz.
Para melhorar a velocidade de convergência do ML-TSKC-FS, redefinimos o ponto fixo $\boldsymbol{P}^{(t)}$ na Eq. (3.23) em cada iteração, atualizando-o para

$$\boldsymbol{P}^{(t)} = \boldsymbol{P}_t + \frac{(b_{t-1} - 1)}{b_t} (\boldsymbol{P}_t - \boldsymbol{P}_{t-1}), \tag{3.29}$$

onde a sequência (b_t) satisfaz $b_{t+1}^2 - b_{t+1} \leq b_t^2$, e $\boldsymbol{P}_t$ é o resultado da t-ésima iteração.

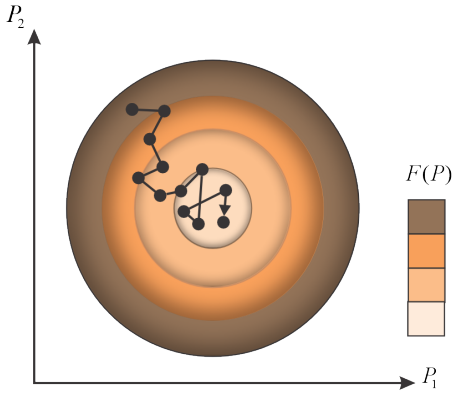


Figura 3.8: Ilustração da trajetória de P durante a Minimização

A Figura 3.8 ilustra o processo de ajuste dos parâmetros P durante o processo de minimização. Cada ponto no gráfico representa os valores de P após cada iteração, mostrando como os parâmetros convergem para o ponto de mínimo da função de custo.

Transformação da saída de valores reais a valores de rótulo

Após o término do processo de treinamento, foram obtidos os melhores parâmetros de aprendizagem, representados por P^*. A partir desses parâmetros, a predição dos valores do modelo ML-TSKC-FS é realizada conforme a expressão:

$$\boldsymbol{y} = \boldsymbol{P}^{*T} \boldsymbol{g}(x),$$

onde o vetor de saída $\boldsymbol{y}$ contém os valores reais previstos. No entanto, para realizar a classificação, é necessário converter esses valores reais em rótulos binários. Para isso, aplicamos a função limiar $\varphi(\cdot)$, que transforma o vetor de predição contínuo em um vetor de rótulos binários $\boldsymbol{y}' = (y'_1, y'_2, \ldots, y'_L)^{\mathrm{T}}$, conforme definido pela seguinte função:

$$
\begin{aligned}
y'_l &= \varphi_\tau\left(y'_l\right) \\
&= \begin{cases} 1, & \text{se } y'_l > \tau, \\ 0, & \text{caso contrário,} \end{cases} \qquad (1 \leq l \leq L),
\end{aligned} \tag{3.30}
$$

onde τ representa o limiar ajustável que define o ponto de corte para a classificação. Neste trabalho, o valor de τ foi fixado em 0,5, com base em um processo de validação cruzada para garantir a melhor performance.

3.2.3 Conclusão do capítulo

Neste capítulo, foi explicado o modelo de Sistema Fuzzy Multi-Rótulo Takagi-Sugeno-Kang Choquet (ML-TSKC-FS), destacando como ele funciona e quais são suas principais partes. Esse modelo foi escolhido porque ajuda a capturar as interações entre diferentes características, o que é muito importante em problemas onde há várias classes (ou rótulos) ao mesmo tempo, e esses rótulos podem ter alguma relação entre si.

Primeiro, foi discutida a estrutura do modelo ML-TSKC-FS, enfatizando sua capacidade de lidar com dados que têm alguma incerteza ou falta de precisão. Depois, exploramos o processo de aprendizagem do modelo e como ele ajusta seus parâmetros. Finalmente, foi feita a escolha da função limiar, que foi definido para ajudar o modelo a transformar os valores contínuos da saída em 0 ou 1, para representar a presença ou ausência de um rótulo.

Os exemplos e explicações dadas ao longo do capítulo mostram que o modelo ML-TSKC-FS é promissor para problemas de classificação multi-rótulo, especialmente onde é preciso lidar com relações complexas entre as variáveis. Assim, o conteúdo deste capítulo forneceu uma base teórica importante para entender o modelo. No próximo capítulo vamos explorar como o modelo (ML-TSKC-FS) se comporta em testes práticos e comparações com outros métodos.

Capítulo 4

METODOLOGIA EXPERIMENTAL

Neste capítulo, será detalhada a metodologia experimental utilizada para avaliar o desempenho do modelo ML-TSKC-FS em cenários de classificação multi-rótulo. O objetivo é mostrar, de forma clara, os passos que seguimos para configurar e testar o modelo, assim como as métricas que usamos para medir a performance do modelo.

Primeiro, apresentamos as bases para a análise experimental, explicando as técnicas e ferramentas que usamos, além das principais características dos conjuntos de dados. Em seguida, descrevemos como organizamos os experimentos, incluindo os parâmetros ajustados e o uso de validação cruzada para garantir que as conclusões fossem o mais precisas e confiáveis possível.

Um ponto importante deste capítulo é a comparação entre o modelo ML-TSKC-FS e sua versão anterior, o ML-TSK FS. Usamos diferentes tipos de medidas fuzzy para entender melhor o impacto da integração das medidas fuzzy com a integral de Choquet e ver se essa nova abordagem realmente melhora o desempenho.

Além disso, comparamos o desempenho do ML-TSKC-FS com outros métodos conhecidos da literatura. Essa comparação é essencial para ver como o nosso modelo se posiciona em relação aos modelos que já existem, o que nos ajuda a entender tanto suas vantagens quanto suas limitações.

Dessa forma, este capítulo não apenas descreve os procedimentos seguidos, mas também justifica as escolhas feitas e destaca a importância dos resultados obtidos,

mostrando como o ML-TSKC-FS pode contribuir para melhorar as técnicas de classificação multi-rótulo.

4.1 FERRAMENTAS DE ANÁLISE EXPERIMENTAL

Nesta seção, serão apresentados os principais elementos utilizados para avaliar o desempenho do modelo, justificando sua importância para a análise experimental. Conjuntos de dados, métricas, testes estatísticos e validação cruzada desempenham papéis essenciais ao garantir que o classificador seja robusto, preciso e confiável. Cada um desses elementos contribui para uma avaliação rigorosa, assegurando que o modelo generalize bem para novos dados.

4.1.1 Descrição dos Conjuntos de Dados Utilizados

Os experimentos foram feitos com dados de diferentes áreas, como som, texto, imagem e genética/biologia, obtidos do repositório MULAN[1].

A escolha dos conjuntos de dados buscou representar vários desafios que o modelo ML-TSKC FS pode encontrar em diferentes situações. A variedade de áreas ajuda a avaliar como o modelo lida com tipos variados de dados. Por exemplo, os dados de áudio, como o *Birds*, apresentam desafios por causa de ruídos e mudanças nas características do som, enquanto os dados de imagem, como o *Corel5k*, têm uma sobreposição de rótulos e muita informação visual. Já os dados de genética, como o *Yeast*, trazem relações complexas entre rótulos que são essenciais para uma classificação precisa. Essa diversidade permite uma análise completa da capacidade do modelo. A seguir, são apresentados os conjuntos de dados usados no trabalho, separados por categoria.

- **Som**
  - **Cal500** [Turnbull et al. 2008]: Esse conjunto traz informações sobre músicas, com 174 rótulos e 68 atributos. Aqui, o desafio é que o número de rótulos é muito maior que o de atributos, o que torna a tarefa de classificação mais complicada.
  - **Birds** [Briggs, Huang et al. 2013]: Usado para prever espécies de aves a partir de gravações de áudio. Nesse caso, várias espécies podem ser detectadas em uma mesma gravação, e o modelo precisa lidar com possíveis ruídos e confusões no som.
  - **Emotions** [Tsoumakas, Katakis e Vlahavas 2008]: Classifica músicas de acordo com as emoções que elas transmitem. Esse conjunto tem 72 atributos e 6 categorias emocionais, o que testa a capacidade do modelo de entender nuances de sentimento nas músicas.

[1] `https://mulan.sourceforge.net/datasets-mlc.html`

Os conjuntos de dados de **Cal500**, **Birds** e **Emotions** são úteis para entender como o modelo lida com dados de áudio, que podem ter ruídos e sobreposições de sinais. Esses testes ajudam a avaliar a eficiência do modelo em capturar detalhes sonoros.

- **Texto**
  - **Bibtex** [Katakis, Tsoumakas e Vlahavas 2008]: Dados usados para recomendar etiquetas em entradas bibliográficas, com 1836 atributos e 159 rótulos. O desafio aqui é a quantidade de atributos, típica em dados de linguagem.
  - **Rcv1s1 e Rcv1s2** [Lewis et al. 2004]: Conjuntos de dados com 6000 artigos sobre diferentes assuntos. Esses dados são úteis para ver como o modelo lida com textos complexos e múltiplas classificações ao mesmo tempo.

  Nos conjuntos de dados de **Bibtex**, **Rcv1s1** e **Rcv1s2**, o modelo precisa lidar com uma grande quantidade de atributos e rótulos, o que é comum em tarefas de processamento de textos.

- **Imagem**
  - **Corel16k1** [Nando 2003] e **Mirflickr**: Dados de classificação de imagens com muitos atributos, que exigem que o modelo interprete detalhes visuais complexos.
  - **Image** [Zhang e Zhou 2007]: Conjunto com 2000 imagens transformadas em vetores de 294 dimensões. As imagens foram convertidas para um espaço de cores específico, o que pode adicionar variações nos dados.
  - **Flags** [Gonçalves et al. 2013]: Dados sobre bandeiras, com 194 instâncias e 19 características, como cores e símbolos. Esse conjunto exige que o modelo relacione aspectos visuais com símbolos específicos.
  - **Scene** [Boutell et al. 2004]: Conjunto de imagens de cenas, com 2407 imagens e 6 classes. Aqui, o desafio é identificar corretamente várias categorias em uma única imagem.

  Os conjuntos de dados de **Corel16k1**, **Mirflickr**, **Image**, **Flags** e **Scene** são usados para classificação de imagens e apresentam o desafio de lidar com grandes quantidades de detalhes visuais e categorias múltiplas ao mesmo tempo.

- **Genética/Biologia**

– **Yeast** [Elisseeff e Weston 2001]: Conjunto de dados biológicos sobre genes de levedura, com até 14 categorias funcionais. Esse é um dos mais desafiadores, pois erros podem afetar diretamente a interpretação de dados biológicos.

A variedade dos conjuntos de dados permite uma avaliação completa do modelo ML-TSKC FS. Cada domínio tem desafios específicos, desde alta quantidade de atributos até ruídos nos dados. Isso ajuda a ver se o modelo consegue generalizar bem em situações diferentes e lidar com vários tipos de rótulos ao mesmo tempo.

A Tabela 4.1 apresenta um resumo das principais características dos conjuntos de dados utilizados no estudo, incluindo o número de instâncias, atributos e rótulos, além do domínio. No Apêndice B, você pode encontrar algumas das bases de dados da Tabela 4.1 apresentadas com mais detalhes.

Conjuntos de Dados	**Instâncias**	**Atributos**	**Rótulos**	**Domínio**
Bibtex	7395	1836	159	Texto
Birds	645	260	19	Áudio
Cal500	502	68	174	Áudio
Corel16k1	13766	500	153	Imagem
Emotions	593	72	6	Áudio
Flags	194	19	7	Imagem
Image	600	294	5	Imagem
Mirflickr	25000	1000	38	Imagem
Rcv1s1	6000	944	101	Texto
Rcv1s2	6000	944	101	Texto
Scene	2407	294	6	Imagem
Yeast	2417	103	14	Gen/Bio

Tabela 4.1: Resumo dos conjuntos de dados utilizados no estudo.

4.1.2 Métricas de Avaliação

Para avaliar o desempenho do modelo ML-TSKC FS, foram escolhidas métricas que medem diferentes aspectos importantes da classificação multi-rótulo. Essas métricas são reconhecidas na literatura como úteis para entender o comportamento do modelo em tarefas com múltiplos rótulos [Schapire e Singer 2000].

- **Average Precision (AP)**: Mede se o modelo acerta a ordem dos rótulos mais importantes nas primeiras posições. Em outras palavras, ela verifica se o modelo está dando prioridade aos rótulos corretos para cada instância. A AP é definida como:

$$AP = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|L_i|}\sum_{k=1}^{|L_i|}\frac{|\{f(x_i,r) \in L_i : rank(f(x_i,r)) \leq rank(f(x_i,k))\}|}{rank(f(x_i,k))} \quad (4.1)$$

Aqui, L_i é o conjunto de rótulos corretos para a instância i, e $rank(f(x_i,k))$ representa a posição (ou "ranking") do rótulo k na lista do modelo. Quanto maior o valor de AP, melhor o modelo está acertando a ordem dos rótulos, mostrando que ele está priorizando corretamente os mais importantes.

- **Hamming Loss (HL)**: Mede a proporção de rótulos que o modelo previu errado, ou seja, quantos rótulos ele classificou incorretamente. Ele calcula a taxa de erro em todas as instâncias e é definido como:

$$HL = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i \oplus \hat{y}_i|}{|L|} \quad (4.2)$$

Aqui, y_i é o conjunto de rótulos corretos para a instância i, $\hat{y}_i$ é o conjunto de rótulos previstos pelo modelo, e L é o número total de rótulos. Um valor de HL mais baixo indica que o modelo está errando menos rótulos e, portanto, é mais preciso.

- **Ranking Loss (RL)**: Mede se o modelo coloca rótulos irrelevantes acima dos rótulos corretos na lista de previsão. Em outras palavras, verifica se a ordenação dos rótulos pelo modelo está invertida. É definida como:

$$RL = \frac{1}{N}\sum_{i=1}^{N}\frac{|\{(r_j,r_k) \in L_i \times \overline{L_i} : rank(f(x_i,r_j)) > rank(f(x_i,r_k))\}|}{|L_i| \times |\overline{L_i}|} \quad (4.3)$$

Nesta fórmula, L_i são os rótulos corretos e $\overline{L_i}$ são os rótulos incorretos. Um RL baixo significa que o modelo está colocando corretamente os rótulos relevantes nas primeiras posições, o que é ideal para uma boa classificação.

- **Coverage (CV)**: Mede o quanto o modelo precisa percorrer a lista para encontrar todos os rótulos corretos. Em outras palavras, ela mede a profundidade da lista onde está o último rótulo correto. A CV é calculada como:

$$CV = \frac{1}{N}\sum_{i=1}^{N}\left(\max_{r \in L_i} rank(f(x_i,r)) - 1\right) \quad (4.4)$$

Um valor de CV mais baixo indica que o modelo acerta os rótulos corretos mais cedo na lista, o que é desejável. Isso mostra que o modelo está priorizando

os rótulos importantes logo no início, ao invés de colocá-los em posições mais baixas.

4.1.3 Testes de Significância Estatística

Para garantir que as diferenças nos resultados obtidos nos experimentos são confiáveis, foram aplicados testes estatísticos de significância. Esses testes nos ajudam a verificar se os modelos realmente têm desempenhos diferentes ou se as variações nos resultados podem ter ocorrido por acaso.

Teste de Friedman: É utilizado para analisar as diferenças de desempenho entre múltiplos algoritmos em diferentes conjuntos de dados. Esse teste é útil para situações em que queremos comparar mais de dois modelos ao mesmo tempo, verificando se há alguma diferença significativa entre eles.

As hipóteses do teste de Friedman são:

- H_0: Não há diferença significativa de desempenho entre os algoritmos.
- H_1: Existe uma diferença significativa de desempenho entre os algoritmos.

A estatística de Friedman é calculada como:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} \tag{4.5}$$

e

$$\chi_F^2 = \frac{12N}{K(K+1)}\left[\sum_k \text{Rank}_k^2 - \frac{K(K+1)^2}{4}\right] \tag{4.6}$$

onde N é o número de conjuntos de dados, K é o número de algoritmos, e Rank_k representa a média das posições de cada algoritmo. Se o valor de χ^2 for maior que o valor crítico, rejeitamos H_0, indicando que há uma diferença significativa entre os algoritmos.

Teste Post-Hoc Bonferroni-Dunn: Caso o teste de Friedman indique uma diferença significativa, o teste de Bonferroni-Dunn é usado para identificar quais pares de algoritmos apresentam essa diferença. Esse teste é útil para comparações específicas entre pares de algoritmos, ajudando a identificar qual modelo se destaca.

A diferença crítica para o teste de Bonferroni-Dunn é calculada como:

$$CD = q_\alpha \sqrt{\frac{K(K+1)}{6N}} \tag{4.7}$$

onde q_α é o valor crítico para o nível de significância escolhido (por exemplo, $\alpha = 0{,}05$). Se a diferença entre as médias de dois algoritmos for maior que o valor de CD, podemos concluir que essa diferença é significativa.

Comparação entre Dois Algoritmos usando o Teste de Wilcoxon
Para comparar o desempenho de apenas dois algoritmos, utilizamos o teste de Wilcoxon, um teste não paramétrico que verifica se há diferença significativa entre os dois modelos.

As hipóteses do teste de Wilcoxon são:

- H_0: Não há diferença significativa entre os desempenhos dos dois algoritmos.
- H_1: Há uma diferença significativa entre os desempenhos dos dois algoritmos.

Para aplicar o teste, calculamos a diferença de desempenho entre os dois algoritmos em cada conjunto de dados, atribuindo rangos às diferenças. A estatística do teste de Wilcoxon é baseada nos rangos positivos e negativos e é dada por:

$$W = \min(W^+, W^-) \tag{4.8}$$

onde W^+ e W^- são as somas dos rangos positivos e negativos, respectivamente. Se o valor de W for menor que o valor crítico, rejeitamos H_0, indicando que há uma diferença significativa entre os dois algoritmos.

Esses testes nos ajudam a avaliar se as diferenças observadas entre os modelos são estatisticamente confiáveis, dando mais segurança ao escolher o modelo que melhor se adapta ao problema.

4.1.4 Procedimentos de Avaliação

Para obter uma estimativa confiável do desempenho do modelo ML-TSKC FS, foi adotado o método de **validação cruzada** em cinco partes (5-fold). Esse método permite testar o modelo em várias divisões do conjunto de dados, ajudando a reduzir o viés que poderia ocorrer se apenas uma divisão fosse usada. A validação cruzada com cinco ou dez partes é amplamente recomendada, pois oferece um equilíbrio entre viés e variabilidade nos resultados, sendo muito utilizada para avaliar a precisão de modelos de aprendizado de máquina [Hastie, Tibshirani e Friedman 2009; Gareth et al. 2013].

Figura 4.1: Ilustração da validação cruzada de cinco partes (5-fold).

A Figura 4.1 mostra como funciona o processo de validação cruzada. Em cada uma das cinco etapas, uma parte do conjunto de dados é usada como teste, enquanto as outras partes são usadas para treinar o modelo. Esse processo é repetido cinco vezes, de forma que cada parte seja usada uma vez como teste. Esse método ajuda a avaliar o desempenho de forma mais precisa e confiável, diminuindo o efeito de possíveis variações nos dados [Hastie, Tibshirani e Friedman 2009].

Além disso, o modelo ML-TSKC FS foi comparado com outros algoritmos conhecidos para classificação multi-rótulo. Essa comparação é importante para entender as vantagens do modelo proposto e ver se ele realmente traz melhorias em termos de precisão, capacidade de generalização e robustez.

No Apêndice A, encontra-se a parte do código onde a Integral de Choquet é implementada no modelo ML-TSK FS, permitindo a obtenção dos resultados apresentados na próxima seção.

4.2 RESULTADOS E DISCUSSÕES

Nesta seção, são apresentados os parâmetros utilizados para obter os resultados do modelo ML-TSKC-FS em diferentes conjuntos de dados. Primeiramente, exploramos os efeitos das diferentes medidas fuzzy na Integral de Choquet, realizando um estudo comparativo entre o modelo proposto, ML-TSKC-FS, e o modelo original, ML-TSK FS. Essa análise tem como objetivo avaliar o impacto da integração da Integral de Choquet no desempenho do modelo. Em seguida, conduzimos uma comparação entre o modelo ML-TSKC-FS e os modelos tradicionais da literatura, destacando as principais vantagens e limitações de cada abordagem. Em ambas as etapas, os resultados obtidos foram submetidos a testes estatísticos, com o intuito

de proporcionar uma compreensão mais aprofundada e rigorosa das diferenças de desempenho observadas.

4.2.1 Configuração do Experimento

Para garantir uma avaliação completa e confiável do desempenho do ML-TSKC FS em tarefas de classificação multi-rótulo, adotamos protocolos de teste detalhados. Esses protocolos foram projetados para testar o modelo em diferentes condições, permitindo comparações com modelos de referência.

Parâmetros do Modelo:

Os parâmetros ajustáveis do ML-TSKC FS, como α, β, γ, o número de regras k, e o parâmetro fuzzy h, foram otimizados por meio de uma busca em grade, como mostrado na Tabela 4.2. Cada um desses parâmetros tem um papel importante:

- Os parâmetros α e β são usados, respectivamente, para ajustar o peso do aprendizado de correlação e a complexidade do modelo.
- γ: Define a estabilidade do modelo, garantindo que ele não oscile muito entre diferentes execuções.
- h: Ajusta a sensibilidade das funções fuzzy, permitindo que o modelo se adapte melhor às variações dos dados.
- k: Determina o número de regras fuzzy. Valores mais altos de k permitem capturar mais detalhes nos dados, mas aumentam a complexidade do modelo.

Esses parâmetros foram ajustados por meio de uma busca em grade, um processo que explora diferentes combinações de valores para identificar a configuração ideal. Esse processo é essencial para garantir que o modelo seja otimizado e tenha um desempenho consistente em cada conjunto de dados.

A Tabela 4.2 mostra os valores testados para cada parâmetro durante a busca em grade. Esses valores foram selecionados com base em literatura e testes iniciais, cobrindo uma gama ampla para encontrar a melhor configuração para o modelo ML-TSKC FS em cada conjunto de dados.

Parâmetro	**Valores Testados**
α	$\{0.01, 0.1, 1, 10, 100\}$
β	$\{0.01, 0.1, 1, 10, 100\}$
γ	$\{0.01, 0.1, 1, 10, 100\}$
K	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
h	$\{0.01, 0.1, 1, 10, 100\}$

Tabela 4.2: Configuração dos parâmetros.

Apresentação dos Resultados

Os resultados são apresentados em duas partes: primeiro, analisamos como as diferentes medidas fuzzy afetam o desempenho do modelo, buscando identificar aquelas que proporcionam melhores resultados. Em seguida, comparamos o desempenho do nosso melhor modelo com nove outros classificadores multi-rótulo que utilizam abordagens diferentes. Essa comparação é essencial para avaliar se nosso modelo realmente traz melhorias em relação a métodos já existentes para classificação multi-rótulo.

4.2.2 Estudo Comparativo entre o ML-TSKC FS (com diferentes medidas fuzzy) e o ML-TSK FS

Nesta seção, apresentamos um estudo comparativo entre o modelo original **ML-TSK FS** proposto por Lou [Lou et al. 2021], e o modelo **ML-TSKC FS** com diferentes medidas fuzzy. Essas variações incluem as configurações **Choq Uni**, **Choq Rel**, **Choq Pro**, **Choq Pot** e **Choq Pon**, onde cada uma representa um tipo específico de medida fuzzy aplicada: Uni (Uniforme), Rel (Relativa), Pro (Produto), Pot (Potência) e Pon (Ponderada).

Para avaliar o desempenho de cada modelo, foram utilizadas métricas específicas: **AP**, **HL**, **RL** e **CV**. Como explicado na Seção 4.1.2, essas métricas permitem mensurar a precisão do modelo, o erro de classificação, bem como a profundidade necessária para que todos os rótulos verdadeiros sejam corretamente cobertos. Esse estudo comparativo visa identificar a eficácia de cada variação do modelo ML-TSKC FS em relação ao ML-TSK FS original, proporcionando uma visão mais aprofundada sobre o impacto das diferentes medidas fuzzy no desempenho de classificação.

Nas tabelas, os valores destacados em **(azul)** indicam o melhor desempenho entre os modelos para cada métrica e conjunto de dados. Para facilitar a interpretação:

- A seta ↓ indica que, para essa métrica, valores menores representam melhor desempenho.
- A seta ↑ indica que, para essa métrica, valores maiores indicam melhor desempenho.

Dataset	ML-TSK FS	Choq Uni	Choq Rel	Choq Pro	Choq Pot	Choq Pon
Bibtex	0.61(0.00)	0.61(0.01)	0.61(0.01)	0.62(0.01)	0.61(0.02)	0.62(0.01)
Birds	0.34(0.03)	0.37(0.05)	0.37(0.03)	0.37(0.01)	0.37(0.04)	0.37(0.06)
Cal500	0.52(0.01)	0.52(0.01)	0.52(0.01)	0.52(0.01)	0.52(0.01)	0.52(0.02)
Corel16k1	0.35(0.01)	0.36(0.00)	0.36(0.01)	0.36(0.00)	0.36(0.00)	0.36(0.00)
Emotions	0.82(0.01)	0.82(0.03)	0.82(0.01)	0.82(0.01)	0.82(0.02)	0.82(0.03)
Flags	0.82(0.01)	0.84(0.02)	0.84(0.03)	0.83(0.02)	0.83(0.04)	0.83(0.03)
Image	0.79(0.03)	0.79(0.04)	0.79(0.02)	0.79(0.04)	0.79(0.04)	0.80(0.03)
Mirflickr	0.53(0.00)	0.53(0.00)	0.53(0.00)	0.53(0.00)	0.53(0.00)	0.53(0.00)
Rev1s1	0.61(0.00)	0.62(0.00)	0.62(0.01)	0.62(0.01)	0.62(0.01)	0.62(0.01)
Rev1s2	0.64(0.01)	0.64(0.01)	0.64(0.01)	0.64(0.01)	0.64(0.01)	0.64(0.01)
Scene	0.86(0.01)	0.86(0.01)	0.86(0.01)	0.86(0.01)	0.86(0.01)	0.86(0.00)
Yeast	0.76(0.01)	0.77(0.01)	0.77(0.00)	0.77(0.01)	0.77(0.01)	0.77(0.01)

Tabela 4.3: Resultados de (AP) ↑: ML TKS-FS vs Choquet

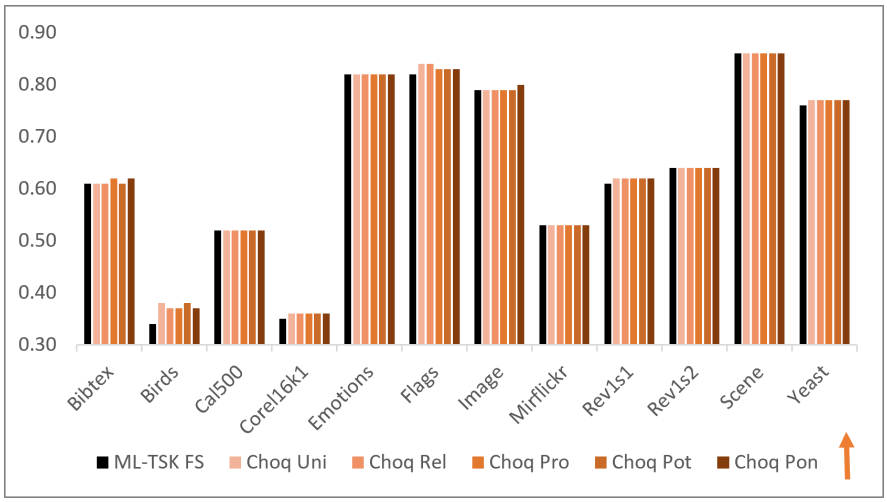


Figura 4.2: Diagrama de barras (AP) ↑ : Choquet vs ML TKS-FS

Average Precision (AP): A Tabela 4.3 e a Figura 4.2 mostram os resultados de AP, comparando o ML-TSKC FS com o modelo original ML-TSK FS. Notamos que o ML-TSKC FS tem melhor desempenho em vários conjuntos de dados. Esse ganho de precisão ocorre porque a Integral de Choquet consegue capturar interações complexas entre rótulos, especialmente quando eles são interdependentes.

Além disso, podemos observar que na maioria de conjuntos de dados a configuração **Choq Pon** apresentam um desempenho superior. Esses resultados sugerem que essas variante consegue lidar melhor com a interação entre rótulos.

Dataset	ML-TSK FS	Choq Uni	Choq Rel	Choq Pro	Choq Pot	Choq Pon
Bibtex	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.01(0.00)
Birds	0.05(0.01)	0.04(0.00)	0.04(0.00)	0.04(0.00)	0.04(0.00)	0.04(0.01)
Cal500	0.14(0.00)	0.13(0.00)	0.13(0.00)	0.13(0.00)	0.13(0.00)	0.13(0.00)
Corel16k1	0.02(0.00)	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.01(0.00)
Emotions	0.19(0.01)	0.19(0.01)	0.19(0.01)	0.19(0.01)	0.19(0.01)	0.19(0.01)
Flags	0.26(0.03)	0.24(0.03)	0.25(0.03)	0.25(0.03)	0.24(0.01)	0.25(0.01)
Image	0.18(0.01)	0.18(0.01)	0.17(0.01)	0.18(0.01)	0.18(0.01)	0.18(0.00)
Mirflickr	0.15(0.00)	0.15(0.00)	0.15(0.00)	0.15(0.00)	0.15(0.00)	0.15(0.00)
Rev1s1	0.03(0.00)	0.02(0.00)	0.02(0.00)	0.02(0.00)	0.02(0.00)	0.02(0.00)
Rev1s2	0.02(0.00)	0.02(0.00)	0.02(0.00)	0.02(0.00)	0.02(0.00)	0.02(0.00)
Scene	0.11(0.01)	0.10(0.01)	0.10(0.00)	0.10(0.01)	0.10(0.01)	0.10(0.01)
Yeast	0.20(0.01)	0.19(0.00)	0.19(0.01)	0.19(0.00)	0.19(0.01)	0.19(0.01)

Tabela 4.4: Resultados de (HL) ↓: ML TKS-FS vs Choquet

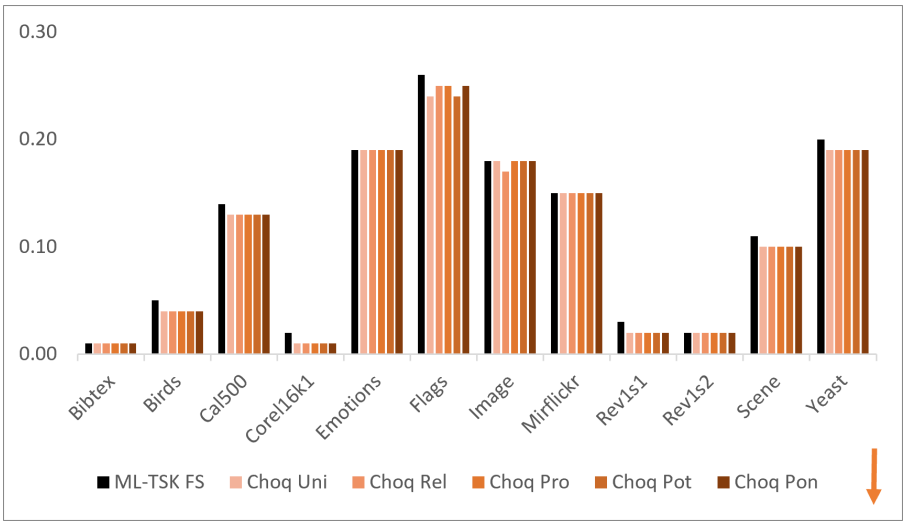


Figura 4.3: Diagrama de barras (HL) ↓ : Choquet vs ML TKS-FS

Hamming Loss (HL): Os resultados na Tabela 4.4 e a Figura 4.3 indicam uma redução no erro de classificação para o modelo ML-TSKC FS em comparação com o ML-TSK FS. A redução no HL significa que o ML-TSKC FS é mais preciso na classificação, minimizando os erros. As variantes **Choq Rel** e **Choq Pon** se destacam, sugerindo que essas variantes são especialmente úteis para minimizar o erro.

Dataset	ML-TSK FS	Choq Uni	Choq Rel	Choq Pro	Choq Pot	Choq Pon
Bibtex	0.07(0.00)	0.06(0.00)	0.06(0.00)	0.06(0.00)	0.06(0.00)	0.06(0.00)
Birds	0.09(0.02)	0.07(0.01)	0.07(0.01)	0.07(0.02)	0.07(0.01)	0.07(0.02)
Cal500	0.18(0.00)	0.17(0.01)	0.17(0.00)	0.17(0.00)	0.17(0.00)	0.17(0.00)
Corel16k1	0.14(0.00)	0.13(0.00)	0.13(0.00)	0.13(0.00)	0.13(0.00)	0.14(0.00)
Emotions	0.15(0.02)	0.15(0.03)	0.15(0.02)	0.14(0.01)	0.14(0.02)	0.14(0.01)
Flags	0.21(0.02)	0.19(0.03)	0.19(0.05)	0.20(0.03)	0.19(0.04)	0.20(0.03)
Image	0.17(0.03)	0.17(0.03)	0.17(0.01)	0.17(0.02)	0.17(0.03)	0.18(0.02)
Mirflickr	0.20(0.00)	0.19(0.00)	0.19(0.00)	0.19(0.00)	0.19(0.00)	0.19(0.00)
Rev1s1	0.05(0.00)	0.04(0.00)	0.04(0.00)	0.04(0.00)	0.04(0.00)	0.04(0.00)
Rev1s2	0.05(0.00)	0.04(0.00)	0.04(0.00)	0.04(0.00)	0.04(0.00)	0.04(0.00)
Scene	0.08(0.01)	0.08(0.01)	0.08(0.00)	0.08(0.00)	0.08(0.00)	0.08(0.01)
Yeast	0.17(0.01)	0.16(0.01)	0.16(0.01)	0.16(0.01)	0.16(0.01)	0.16(0.01)

Tabela 4.5: Resultados de (RL) ↓: ML TKS-FS vs Choquet

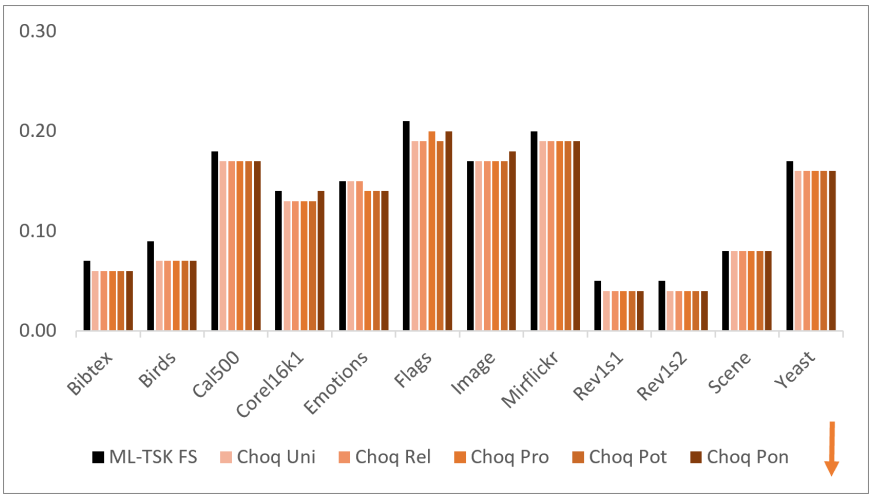


Figura 4.4: Diagrama de barras (RL) ↓ : Choquet vs ML TKS-FS

Ranking Loss (RL): A Tabela 4.5 e a Figura 4.4 mostram que as variantes **Choq Pro**, **Choq Pot** e **Choq Pon** frequentemente apresentam valores mais baixos de RL, indicando uma melhor ordenação dos rótulos relevantes em comparação com o modelo ML-TSK FS. Isso sugere que essas variantes ajudam a organizar corretamente a relevância dos rótulos, o que é essencial em cenários onde a hierarquia dos rótulos importa.

Dataset	ML-TSK FS	Choq Uni	Choq Rel	Choq Pro	Choq Pot	Choq Pon
Bibtex	0.12(0.01)	0.12(0.00)	0.12(0.00)	0.12(0.01)	0.12(0.01)	0.12(0.00)
Birds	0.11(0.03)	0.10(0.01)	0.09(0.02)	0.09(0.02)	0.09(0.01)	0.09(0.03)
Cal500	0.73(0.01)	0.71(0.01)	0.72(0.02)	0.72(0.02)	0.72(0.02)	0.72(0.02)
Corel16k1	0.29(0.00)	0.25(0.00)	0.25(0.00)	0.26(0.01)	0.26(0.00)	0.27(0.01)
Emotions	0.28(0.03)	0.28(0.02)	0.28(0.02)	0.28(0.02)	0.28(0.02)	0.28(0.01)
Flags	0.52(0.01)	0.51(0.02)	0.51(0.03)	0.53(0.03)	0.52(0.02)	0.53(0.03)
Image	0.18(0.02)	0.19(0.02)	0.18(0.01)	0.18(0.02)	0.18(0.02)	0.19(0.02)
Mirflickr	0.42(0.00)	0.42(0.00)	0.42(0.00)	0.42(0.00)	0.42(0.00)	0.42(0.01)
Rev1s1	0.11(0.00)	0.11(0.01)	0.11(0.01)	0.11(0.01)	0.11(0.00)	0.11(0.01)
Rev1s2	0.12(0.01)	0.11(0.01)	0.11(0.01)	0.11(0.00)	0.11(0.00)	0.10(0.01)
Scene	0.08(0.01)	0.08(0.01)	0.08(0.00)	0.08(0.00)	0.08(0.00)	0.08(0.01)
Yeast	0.46(0.01)	0.45(0.01)	0.45(0.01)	0.45(0.01)	0.45(0.01)	0.45(0.01)

Tabela 4.6: Resultados de (CV) ↓: ML TKS-FS vs Choquet

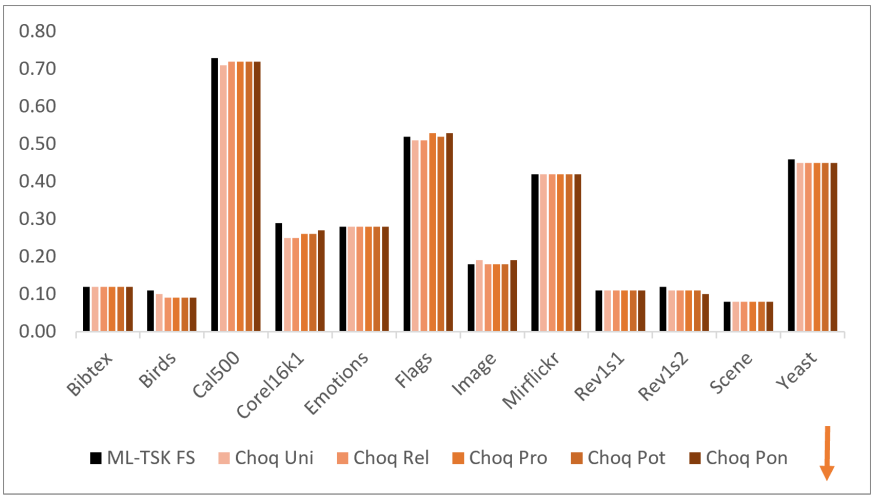


Figura 4.5: Diagrama de barras (CV) ↓ : Choquet vs ML TKS-FS

Coverage (CV): Apresentada na Tabela 4.6 e Figura 4.5, indica a profundidade necessária para cobrir todos os rótulos verdadeiros. O modelo ML-TSKC FS com as variantes **Choq Uni** e **Choq Rel** apresentou valores de CV mais baixos, sugerindo que essas configurações são eficazes na hierarquia de rótulos, o que reduz o número de rótulos a serem verificados.

Em resumo, os resultados indicam que o modelo ML-TSKC FS, com a inclusão da Integral de Choquet, apresenta desempenho superior em relação ao ML-TSK FS em quase todas as métricas e conjuntos de dados analisados. Essa melhoria é especialmente relevante, pois demonstra a eficácia da Integral de Choquet em capturar de forma mais precisa as interações entre os atributos, proporcionando classificações mais robustas em cenários complexos.

No entanto, ao comparar os resultados obtidos com diferentes medidas fuzzy dentro do modelo baseado em Choquet, observou-se uma semelhança nos desempenhos. Isso pode ser explicado pela natureza das funções de agregação, que tendem

a capturar relações similares entre os dados, independentemente da medida fuzzy selecionada. Além disso, é possível que os dados testados não apresentem variabilidade suficiente ou características suficientemente distintas para evidenciar diferenças significativas entre as medidas, resultando em desempenhos próximos.

A seguir, faremos um estudo estatístico para verificar se as melhorias obtidas nesta seção são estatisticamente significativas.

Estudo Comparativo Estatístico entre o ML-TSKC FS (com diferentes medidas fuzzy) e o ML-TSK FS

Para verificar se as diferenças de desempenho entre o modelo ML-TSKC FS (com medidas fuzzy baseadas na Integral de Choquet) e o modelo ML-TSK FS são estatisticamente significativas, realizamos um estudo utilizando o teste de Wilcoxon. Essa análise confirma a relevância das melhorias observadas.

Teste de Wilcoxon: Esse teste não paramétrico foi aplicado para comparações emparelhadas entre o modelo ML-TSK FS e as versões do ML-TSKC FS com diferentes medidas fuzzy (Choq Uni, Choq Rel, Choq Pro, Choq Pot e Choq Pon). As avaliações foram realizadas para as métricas AP (Average Precision), HL (Hamming Loss), RL (Ranking Loss) e CV (Cobertura).

Os resultados identificaram as configurações que apresentam ganhos significativos, destacando as medidas fuzzy mais adequadas para diferentes cenários.

- **Resultados do Teste de Wilcoxon para AP**: Na Tabela 4.7, vemos os resultados do teste de Wilcoxon para a métrica de AP. Os valores de p para todas as comparações são menores que 0,05, indicando que as versões do modelo ML-TSKC FS com medidas fuzzy superam o modelo ML-TSK FS em termos de AP . Isso significa que essas variações com Choquet são mais eficazes em identificar rótulos corretamente.

Comparação de Modelos	Valor p
ML-TSK FS vs Choq Uni	0,04
ML-TSK FS vs Choq Rel	0,04
ML-TSK FS vs Choq Pro	0,02
ML-TSK FS vs Choq Pot	0,03
ML-TSK FS vs Choq Pon	0,01

Tabela 4.7: Resultados do Teste de Wilcoxon para AP

- **Resultados do Teste de Wilcoxon para HL**: A Tabela 4.8 mostra os resultados para a HL. Aqui, o teste indica que todas as variantes de Choquet têm um desempenho significativamente melhor, reduzindo o erro de classificação

em comparação com o modelo ML-TSK FS. Isso significa que essas versões são mais precisas ao evitar classificações incorretas.

Comparação de Modelos	Valor p
ML-TSK FS vs Choq Uni	0,02
ML-TSK FS vs Choq Rel	0,01
ML-TSK FS vs Choq Pro	0,02
ML-TSK FS vs Choq Pot	0,02
ML-TSK FS vs Choq Pon	0,02

Tabela 4.8: Resultados do Teste de Wilcoxon para HL

- **Resultados do Teste de Wilcoxon para RL**: Na Tabela 4.9, vemos que os valores de p são pequenos para as comparações com todas as variantes de Choquet. Isso sugere que essas variantes ajudam o modelo a classificar melhor os rótulos, ordenando corretamente os mais importantes.

Comparação de Modelos	Valor p
ML-TSK FS vs Choq Uni	0,01
ML-TSK FS vs Choq Rel	0,01
ML-TSK FS vs Choq Pro	0,00
ML-TSK FS vs Choq Pot	0,00
ML-TSK FS vs Choq Pon	0,01

Tabela 4.9: Resultados do Teste de Wilcoxon para RL

- **Resultados do Teste de Wilcoxon para CV**: Por fim, a Tabela 4.10 apresenta os resultados para a métrica CV. Observa-se que as variantes Choq Rel e Choq Pot foram mais eficazes em identificar os rótulos mais relevantes de forma mais rápida, exigindo uma menor profundidade de busca. Isso indica que essas variantes tornam o modelo mais eficiente, permitindo localizar os rótulos corretos com maior agilidade. Para as demais variantes, onde p é maior que 0,05, as diferenças de desempenho em relação ao modelo ML-TSK FS não são estatisticamente significativas.

Comparação de Modelos	Valor p
ML-TSK FS vs Choq Uni	0,09
ML-TSK FS vs Choq Rel	0,02
ML-TSK FS vs Choq Pro	0,11
ML-TSK FS vs Choq Pot	0,04
ML-TSK FS vs Choq Pon	0,12

Tabela 4.10: Resultados do Teste de Wilcoxon para CV

Os resultados dos testes mostram que o modelo ML-TSKC FS, usando diferentes variantes da Integral de Choquet, como Choq Rel, Choq Pro e Choq Pon, supera o modelo tradicional ML-TSK FS em várias métricas. Esses valores de p pequenos indicam que essas melhorias são estatisticamente significativas. Assim, as medidas fuzzy adicionadas realmente aumentam a precisão e eficiência do modelo em identificar rótulos relevantes.

Em resumo, os resultados dos testes de Wilcoxon indicam que o modelo ML-TSKC FS, ao utilizar diferentes medidas fuzzy na Integral de Choquet, supera o modelo ML-TSK FS em diversas métricas, confirmando que as melhorias observadas são estatisticamente significativas.

Na próxima seção, analisaremos como o modelo ML-TSKC FS se compara a outros métodos consolidados na literatura. Para essa análise, utilizaremos o modelo com **Integral de Choquet e Medida Fuzzy Ponderada** (Choq Pon), pois estudos prévios demonstraram que essa configuração do ML-TSKC FS alcança os melhores resultados.

4.2.3 Estudo Comparativo entre o ML-TSKC FS e Modelos de Referência da Literatura

Para avaliar o desempenho do modelo ML-TSKC FS, realizamos uma análise comparativa com vários modelos amplamente utilizados na área de classificação multi-rótulo. Esses modelos representam diferentes abordagens para resolver os desafios dessa tarefa, como a interdependência entre rótulos e a complexidade dos dados. A Figura 4.6 mostra uma linha do tempo com a evolução desses modelos, oferecendo uma visão geral do desenvolvimento dessa área ao longo dos anos.

Os principais modelos de referência comparados estão descritos a seguir:

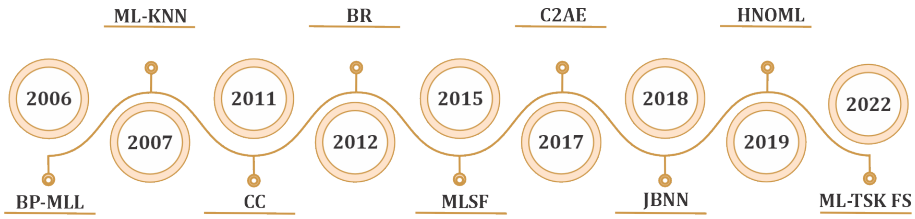


Figura 4.6: Evolução cronológica dos modelos de classificação multi-rótulo.

- **BP-MLL** [Min-Ling e Zhi-Hua 2006]: Este método foi um dos primeiros a usar redes neurais para capturar a correlação entre rótulos, buscando melhorar a precisão ao prever rótulos relacionados juntos.
- **ML-KNN** [Zhang e Zhi-Hua 2007]: Usa os k vizinhos mais próximos para prever rótulos, adaptando uma técnica clássica de aprendizado para o contexto multi-rótulo.

- **CC** [Read et al. 2011]: Converte a tarefa multi-rótulo em várias classificações binárias, possibilitando ajuste de parâmetros para cada rótulo.
- **BR** [Łęski 2002]: Trata cada rótulo como um problema separado de classificação binária (Binary Relevance), uma abordagem que melhora a capacidade de generalização da classificação fuzzy introduzindo o aprendizado *ε-insensitive learning*.
- **MLSF** [Sun, Kudo e Kimura 2016]: Combina aprendizado de meta-rótulos com seleção de características, levando em conta as correlações entre rótulos para melhorar a precisão.
- **C2AE** [Yeh et al. 2017]: Utiliza autoencoders para aprender representações eficazes dos rótulos, útil para problemas mais complexos.
- **JBNN** [He e Xia 2018]: Usa múltiplas funções de ativação para capturar correlações entre rótulos em uma rede neural.
- **HNOML** [Zhang, Yu et al. 2019]: Foca em reduzir ruídos nos rótulos e características, melhorando o desempenho em dados com ruídos.
- **ML-TSK FS** [Lou et al. 2021]: Aplica regras fuzzy (Takagi-Sugeno-Kang) para capturar relações entre características e rótulos, buscando previsões mais precisas e consistentes.

As métricas usadas para avaliar os modelos são as mesmas descritas na Seção 4.1.2, com destaque em azul para os melhores resultados em cada conjunto de dados. As Tabelas apresentam as médias dos resultados, obtidas a partir de validação cruzada em 5 partes, com desvios padrão para indicar a variação dos resultados para o ML-TSKC FS e os demais resultados da tabela são obtidos de Lou [Lou et al. 2021] .

Dataset	ML-kNN	HNOML	MLSF	CC	BR	C2AE	BP-MLL	JBNN	ML-TSK FS	ML-TSKC FS
Bibtex	0.35(0.01)	0.58(0.01)	0.37(0.02)	0.58(0.01)	0.60(0.01)	0.09(0.03)	0.54(0.01)	0.02(0.00)	0.61(0.00)	0.62(0.01)
Birds	0.22(0.02)	0.34(0.03)	0.26(0.03)	0.34(0.01)	0.33(0.03)	0.30(0.04)	0.34(0.02)	0.29(0.06)	0.34(0.03)	0.37(0.06)
Cal500	0.50(0.01)	0.43(0.18)	0.49(0.01)	0.46(0.01)	0.50(0.01)	0.33(0.02)	0.46(0.02)	0.45(0.01)	0.52(0.01)	0.52(0.02)
Corel16k1	0.28(0.00)	0.34(0.01)	0.28(0.01)	0.30(0.00)	0.34(0.00)	0.21(0.02)	0.26(0.01)	0.08(0.00)	0.35(0.01)	0.36(0.00)
Emotions	0.71(0.02)	0.80(0.03)	0.76(0.02)	0.78(0.00)	0.80(0.01)	0.57(0.03)	0.80(0.01)	0.76(0.02)	0.82(0.01)	0.82(0.03)
Flags	0.80(0.04)	0.81(0.01)	0.82(0.03)	0.80(0.04)	0.81(0.04)	0.74(0.06)	0.82(0.02)	0.80(0.04)	0.82(0.01)	0.83(0.03)
Image	0.74(0.02)	0.78(0.02)	0.72(0.02)	0.78(0.03)	0.79(0.03)	0.47(0.02)	0.79(0.02)	0.63(0.04)	0.79(0.03)	0.80(0.03)
Mirflickr	0.51(0.00)	0.51(0.00)	0.27(0.00)	0.48(0.00)	0.44(0.04)	0.45(0.02)	0.47(0.02)	0.42(0.03)	0.53(0.00)	0.53(0.00)
Rev1s1	0.49(0.01)	0.61(0.01)	0.52(0.02)	0.57(0.01)	0.60(0.01)	0.21(0.02)	0.53(0.05)	0.05(0.01)	0.61(0.00)	0.62(0.01)
Rev1s2	0.50(0.01)	0.63(0.00)	0.52(0.02)	0.58(0.01)	0.61(0.01)	0.18(0.04)	0.58(0.01)	0.05(0.01)	0.64(0.01)	0.64(0.01)
Scene	0.87(0.01)	0.85(0.01)	0.86(0.02)	0.84(0.01)	0.86(0.01)	0.42(0.01)	0.87(0.01)	0.59(0.03)	0.86(0.01)	0.86(0.00)
Yeast	0.76(0.02)	0.61(0.00)	0.75(0.02)	0.72(0.01)	0.76(0.01)	0.57(0.02)	0.74(0.01)	0.71(0.03)	0.76(0.01)	0.77(0.01)

Tabela 4.11: Resultados de (AP) : Classificadores vs. ML TKSC-FS

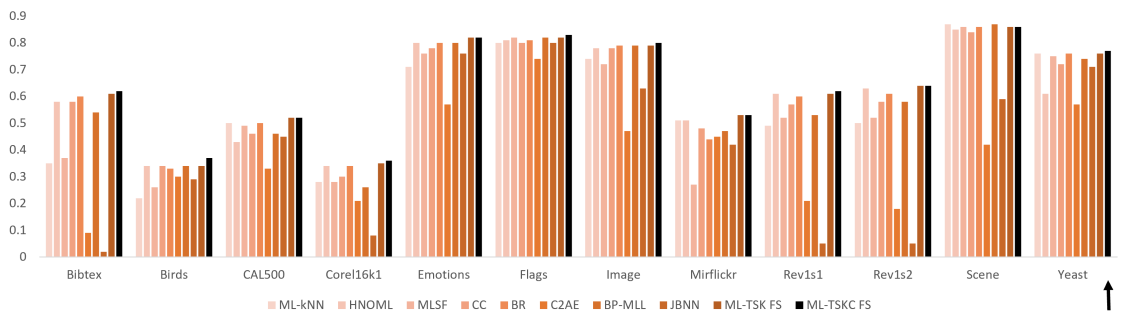


Figura 4.7: Diagrama de barras de (AP): Classificadores vs. ML TKSC-FS

Análise Comparativa de AP Com base na Tabela 4.11 e na Figura 4.7, que mostram os resultados de AP) comparando o ML-TSKC FS com os modelos da literatura, podemos chegar as seguintes conclusões:

- **Desempenho Superior em Vários Conjuntos de Dados**: O ML-TSKC FS teve uma precisão média superior em muitos conjuntos, como *Bibtex*, *Birds* e *Flags*. Isso indica que ele é eficiente para priorizar rótulos importantes.
- **Competição com Modelos de Rede Neural Profunda**: Apesar de modelos como o *C2AE* e o *JBNN* terem bom desempenho, o ML-TSKC FS consegue uma precisão comparável ou melhor em vários casos, oferecendo uma alternativa mais simples e fácil de interpretar.
- **Consistência**: Em cenários onde o numero de rótulos é bem menor que o número de atributos como os conjuntos *Image e Scene*, o ML-TSKC FS ainda

apresenta um desempenho semelhante aos melhores da Literatura, mostrando que ele pode ser eficaz em diferentes níveis de complexidade.

Os resultados mostram que o ML-TSKC FS não só é preciso, mas também consegue capturar bem a relação entre rótulos. Sua vantagem sobre métodos mais complexos sugere que ele é uma alternativa prática e interpretável para classificação multi-rótulo.

Dataset	ML-kNN	HNOML	MLSF	CC	BR	C2AE	BP-MLL	JBNN	ML-TSK FS	ML-TSKC FS
Bibtex	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.14(0.02)	0.02(0.00)	0.02(0.00)	0.01(0.00)	0.01(0.00)
Birds	0.05(0.00)	0.05(0.00)	0.05(0.01)	0.05(0.00)	0.06(0.00)	0.15(0.01)	0.18(0.02)	0.05(0.01)	0.05(0.00)	0.04(0.01)
CAL500	0.14(0.00)	0.14(0.01)	0.14(0.00)	0.14(0.00)	0.14(0.00)	0.19(0.01)	0.29(0.00)	0.14(0.00)	0.14(0.00)	0.13(0.00)
Corel16k1	0.02(0.00)	0.02(0.00)	0.02(0.00)	0.02(0.00)	0.02(0.00)	0.21(0.02)	0.12(0.00)	0.02(0.00)	0.02(0.00)	0.01(0.00)
Emotions	0.26(0.01)	0.21(0.01)	0.24(0.02)	0.21(0.02)	0.20(0.01)	0.41(0.03)	0.22(0.02)	0.20(0.00)	0.19(0.01)	0.19(0.01)
Flags	0.33(0.03)	0.27(0.01)	0.26(0.05)	0.27(0.03)	0.27(0.03)	0.42(0.03)	0.30(0.04)	0.30(0.01)	0.26(0.03)	0.25(0.01)
Image	0.20(0.01)	0.23(0.02)	0.21(0.01)	0.19(0.03)	0.18(0.01)	0.46(0.04)	0.21(0.01)	0.21(0.00)	0.18(0.01)	0.18(0.00)
Mirflickr	0.15(0.00)	0.15(0.00)	0.15(0.00)	0.16(0.00)	0.16(0.01)	0.30(0.03)	0.31(0.00)	0.15(0.00)	0.15(0.00)	0.15(0.00)
Rev1s1	0.03(0.00)	0.03(0.00)	0.03(0.00)	0.03(0.00)	0.03(0.00)	0.17(0.02)	0.04(0.00)	0.03(0.00)	0.03(0.00)	0.02(0.00)
Rev1s2	0.02(0.00)	0.02(0.00)	0.02(0.00)	0.02(0.00)	0.02(0.00)	0.17(0.03)	0.03(0.00)	0.03(0.00)	0.02(0.00)	0.02(0.00)
Scene	0.09(0.00)	0.12(0.00)	0.09(0.01)	0.10(0.01)	0.10(0.01)	0.41(0.03)	0.11(0.00)	0.14(0.00)	0.10(0.01)	0.10(0.01)
Yeast	0.19(0.01)	0.30(0.00)	0.19(0.00)	0.21(0.01)	0.19(0.00)	0.34(0.04)	0.22(0.01)	0.21(0.01)	0.20(0.01)	0.19(0.01)

Tabela 4.12: Resultados de (HL): Classificadores vs. ML TKSC-FS

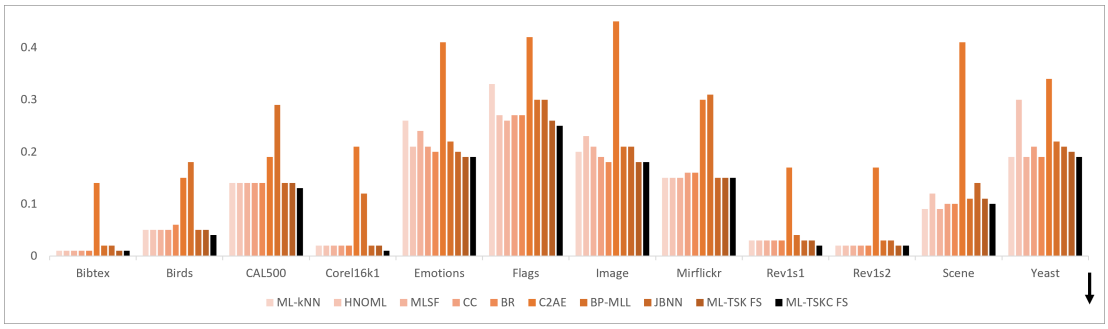


Figura 4.8: Diagrama de barras de (HL): Classificadores vs. ML TKSC-FS

Análise Comparativa de HL Baseando-se na Tabela 4.12 e na Figura 4.8, que apresentam os resultados de HL comparando o ML-TSKC FS com os modelos da literatura, podemos destacar as seguintes observações:

- **Redução Significativa de Erros em Múltiplos Conjuntos de Dados**: O modelo ML-TSKC FS apresentou valores de HL menores em vários conjuntos de dados, incluindo *Bibtex*, *Birds* e *Flags*. Isso demonstra que ele é eficaz em minimizar erros de classificação, mesmo em tarefas multi-rótulo complexas.

- **Consistência em Conjuntos de Dados com Estruturas Variadas**: Em conjuntos de dados onde a complexidade da estrutura de rótulos é mais baixa, como *Image* e *Scene*, o ML-TSKC FS também alcançou resultados sólidos. Essa consistência sugere que o modelo é robusto e pode ser eficaz em diferentes contextos de classificação multi-rótulo.

Os resultados indicam que o ML-TSKC FS é não só eficiente em reduzir erros, mas também robusto ao lidar com diferentes tipos de dados e configurações de rótulos. Sua performance consistente e interpretabilidade tornam-no uma escolha prática para aplicações de classificação multi-rótulo, especialmente em cenários onde a simplicidade e precisão são fundamentais.

Dataset	ML-kNN	HNOML	MLSF	CC	BR	C2AE	BP-MLL	JBNN	ML-TSK FS	ML-TSKC FS
Bibtex	0.21(0.01)	0.07(0.00)	0.14(0.01)	0.09(0.00)	0.08(0.00)	0.42(0.01)	0.08(0.01)	0.97(0.00)	0.07(0.00)	0.06(0.00)
Birds	0.16(0.01)	0.09(0.02)	0.08(0.02)	0.10(0.01)	0.10(0.02)	0.21(0.02)	0.10(0.01)	0.37(0.03)	0.09(0.02)	0.07(0.02)
CAL500	0.18(0.00)	0.14(0.08)	0.18(0.01)	0.23(0.01)	0.19(0.00)	0.16(0.19)	0.19(0.00)	0.28(0.02)	0.18(0.00)	0.17(0.00)
Core116k1	0.17(0.00)	0.15(0.00)	0.14(0.01)	0.16(0.00)	0.16(0.00)	0.30(0.02)	0.15(0.00)	0.75(0.02)	0.14(0.00)	0.14(0.00)
Emotions	0.26(0.02)	0.16(0.02)	0.11(0.02)	0.18(0.01)	0.17(0.02)	0.43(0.03)	0.16(0.01)	0.23(0.03)	0.15(0.02)	0.14(0.01)
Flags	0.24(0.04)	0.22(0.01)	0.12(0.02)	0.23(0.05)	0.22(0.04)	0.36(0.08)	0.21(0.03)	0.23(0.04)	0.21(0.02)	0.20(0.03)
Image	0.22(0.02)	0.18(0.02)	0.10(0.01)	0.18(0.03)	0.17(0.02)	0.52(0.03)	0.17(0.02)	0.37(0.05)	0.17(0.03)	0.18(0.02)
Mirflickr	0.21(0.00)	0.21(0.00)	0.26(0.00)	0.24(0.00)	0.32(0.04)	0.25(0.02)	0.21(0.01)	0.53(0.05)	0.20(0.00)	0.19(0.00)
Rcvlsl	0.09(0.00)	0.04(0.00)	0.08(0.01)	0.07(0.00)	0.06(0.00)	0.31(0.01)	0.07(0.01)	0.90(0.02)	0.05(0.00)	0.04(0.00)
Rcv1s2	0.09(0.00)	0.04(0.00)	0.06(0.00)	0.07(0.00)	0.06(0.00)	0.35(0.03)	0.07(0.01)	0.88(0.02)	0.05(0.00)	0.04(0.00)
Scene	0.08(0.01)	0.08(0.01)	0.04(0.01)	0.09(0.01)	0.08(0.01)	0.49(0.01)	0.07(0.00)	0.41(0.04)	0.08(0.01)	0.08(0.01)
Yeast	0.17(0.01)	0.34(0.00)	0.13(0.01)	0.21(0.00)	0.16(0.00)	0.30(0.02)	0.18(0.01)	0.23(0.03)	0.17(0.01)	0.16(0.01)

Tabela 4.13: Resultados de (RL): Classificadores vs. ML TKSC-FS

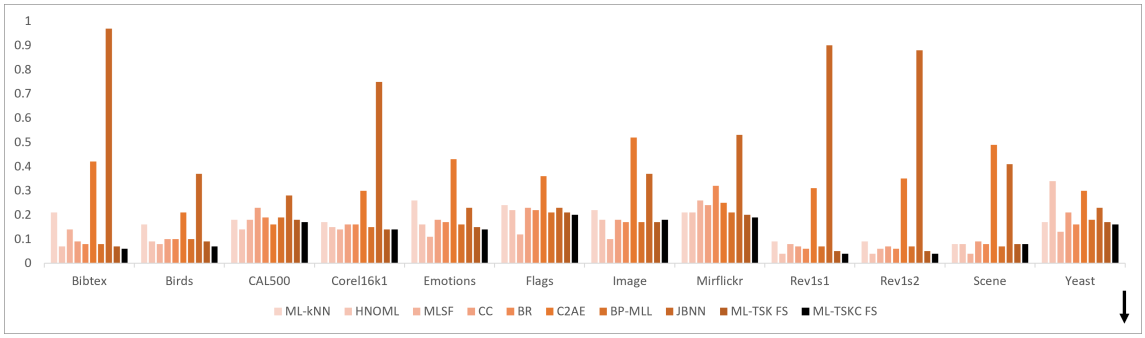


Figura 4.9: Diagrama de barras de (RL): Classificadores vs. ML TKSC-FS

Análise Comparativa de RL Com base na Tabela 4.13 e na Figura 4.9, que comparam os valores de RL entre o modelo ML-TSKC FS e outros modelos de referência da literatura, podemos chegar as seguintes conclusões:

- **Desempenho Superior e Consistente em Diversos Conjuntos de Dados**: O ML-TSKC FS apresentou valores de RL baixos em conjuntos como *Bibtex*, *Birds* e *Mirflickr*, e manteve resultados equivalentes em *Corel16k1*, *Rcv1s1* e *Rcv1s2*. Esse desempenho destaca a capacidade do modelo em ordenar corretamente os rótulos mais relevantes em tarefas de classificação multi-rótulo, evidenciando sua eficiência em cenários onde a priorização precisa dos rótulos é essencial.

- **Consistência em Conjuntos de Dados Mais Complexos**: Em conjuntos como *CAL500*, *Emotions*, *Flags*, *Image*, *Scene* e *Yeast*, os modelos HNOML e MLSF alcançaram desempenho superior na métrica de RL. Esse resultado é esperado, pois o HNOML foi projetado para lidar com ruídos tanto nas características quanto nos rótulos, enquanto o MLSF adota uma seleção de características que foca nas mais relevantes para cada conjunto de rótulos específicos. Essas capacidades permitem que esses modelos mantenham uma ordenação mais precisa dos rótulos, reduzindo a penalização na métrica RL. Ainda assim, o ML-TSKC FS apresentou resultados competitivos nesses conjuntos, demonstrando robustez mesmo em cenários desafiadores.

Os resultados indicam que o ML-TSKC FS é eficaz em priorizar e ordenar rótulos relevantes em cerca de metade dos conjuntos testados e mantém desempenho competitivo nos demais, quando comparado a métodos mais complexos da literatura. Sua performance consistente e robusta em diferentes conjuntos de dados reforça sua viabilidade como solução para tarefas de classificação multi-rótulo, sendo uma alternativa prática e interpretável para esses cenários.

Dataset	ML-kNN	HNOML	MLSF	CC	BR	C2AE	BP-MLL	JBNN	ML-TSK FS	ML-TSKC FS
Bibtex	0.34(0.01)	0.13(0.00)	0.35(0.02)	0.18(0.00)	0.16(0.00)	0.56(0.01)	0.14(0.02)	0.61(0.00)	0.12(0.01)	0.12(0.00)
Birds	0.19(0.01)	0.12(0.03)	0.17(0.05)	0.13(0.01)	0.13(0.01)	0.22(0.03)	0.13(0.02)	0.23(0.04)	0.11(0.03)	0.09(0.03)
Cal500	0.75(0.01)	0.77(0.06)	0.76(0.03)	0.89(0.02)	0.79(0.01)	0.79(0.08)	0.76(0.02)	0.91(0.01)	0.73(0.01)	0.72(0.02)
Core116k1	0.33(0.00)	0.31(0.00)	0.39(0.02)	0.33(0.01)	0.31(0.01)	0.53(0.03)	0.30(0.01)	0.71(0.00)	0.29(0.00)	0.27(0.01)
Emotions	0.38(0.02)	0.30(0.02)	0.33(0.03)	0.31(0.03)	0.30(0.03)	0.32(0.03)	0.37(0.01)	0.20(0.03)	0.28(0.03)	0.28(0.01)
Flags	0.56(0.02)	0.54(0.03)	0.54(0.04)	0.56(0.03)	0.55(0.02)	0.54(0.02)	0.48(0.04)	0.53(0.03)	0.52(0.01)	0.53(0.03)
Image	0.23(0.02)	0.20(0.02)	0.24(0.01)	0.20(0.03)	0.19(0.02)	0.24(0.03)	0.21(0.02)	0.17(0.04)	0.18(0.02)	0.19(0.02)
Mirflickr	0.44(0.00)	0.44(0.00)	0.45(0.00)	0.52(0.01)	0.62(0.04)	0.46(0.02)	0.39(0.00)	0.60(0.01)	0.42(0.00)	0.42(0.01)
Rcvlsl	0.20(0.00)	0.11(0.00)	0.24(0.04)	0.17(0.01)	0.14(0.01)	0.50(0.02)	0.15(0.01)	0.65(0.01)	0.11(0.00)	0.11(0.01)
Rcv1s2	0.19(0.01)	0.11(0.00)	0.19(0.01)	0.16(0.01)	0.14(0.01)	0.53(0.04)	0.14(0.01)	0.62(0.01)	0.12(0.01)	0.10(0.01)
Scene	0.08(0.01)	0.08(0.01)	0.08(0.01)	0.09(0.01)	0.08(0.01)	0.26(0.01)	0.09(0.00)	0.14(0.03)	0.08(0.01)	0.08(0.01)
Yeast	0.45(0.02)	0.62(0.01)	0.48(0.03)	0.51(0.02)	0.44(0.01)	0.47(0.03)	0.47(0.01)	0.48(0.05)	0.46(0.01)	0.45(0.01)

Tabela 4.14: Resultados de (CV): Classificadores vs. ML TKSC-FS

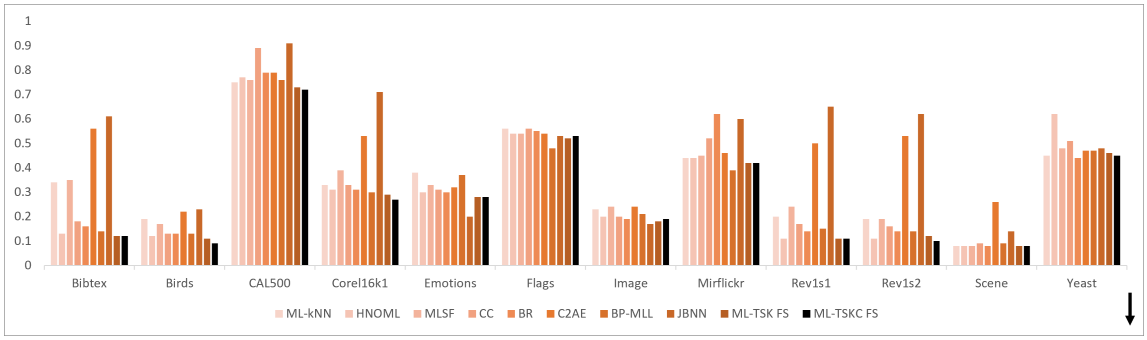


Figura 4.10: Diagrama de barras de (CV): Classificadores vs. ML TKSC-FS

Análise Comparativa de CV Com base na Tabela 4.14 e na Figura 4.10, que comparam os valores de CV entre o modelo ML-TSKC FS e outros modelos de referência da literatura, podemos chegar as seguintes conclusões:

- **Desempenho Superior e Consistente em Diversos Conjuntos de Dados**: O ML-TSKC FS apresentou valores de CV baixos em conjuntos como *Birds*, *Cal500*, *Corel16k1* e *Rcv1s2*, e manteve resultados equivalentes em *Bibtex*, *Rcv1s1* e *Scene*. Esse desempenho indica que o ML-TSKC FS é capaz de identificar os rótulos mais relevantes em profundidade mínima, o que é fundamental para aplicações em que a ordenação correta dos rótulos impacta diretamente a utilidade das previsões.

- **Competição com Modelos de Rede Neural**: Em vários casos, o ML-TSKC FS teve um desempenho comparável ou superior aos modelos baseados em redes neurais profundas, como o *BP-MLL* e o *JBNN*. Esses modelos são geralmente projetados para capturar a complexidade e interdependência entre rótulos, mas o ML-TSKC FS consegue resultados competitivos com uma abordagem mais simples e interpretável. Essa característica faz do ML-TSKC FS uma alternativa interessante para tarefas que requerem um balanceamento entre eficácia e interpretabilidade.

- **Consistência em Cenários de Menor Complexidade**: Nos conjuntos de dados com menor densidade de rótulos, como *Scene* e *Yeast*, o ML-TSKC FS ainda mantém desempenho competitivo, com valores de Cobertura semelhantes ao modelo *BR*. Isso demonstra que o modelo é adaptável a diferentes níveis de complexidade, mantendo a eficácia em contextos variados sem perda de performance significativa.

  Em resumo, os resultados para a métrica de CV mostram que o ML-TSKC FS é eficiente na ordenação de rótulos e na priorização dos rótulos mais relevantes em metade dos conjuntos testados, mantendo desempenho competitivo nos

demais. Sua consistência e robustez, aliadas à facilidade de interpretação, tornam o ML-TSKC FS uma alternativa prática e eficaz para problemas de classificação multi-rótulo em comparação com métodos mais complexos da literatura.

Concluindo, o ML-TSKC FS se mostrou uma alternativa eficiente e interpretável, com desempenho competitivo ou superior a muitos modelos da literatura, incluindo redes neurais profundas. Sua abordagem baseada na Integral de Choquet permite capturar bem as interações entre rótulos, resultando em uma classificação precisa e com menos erros, tornando-o uma escolha promissora para diversas aplicações de classificação multi-rótulo.

Estudo comparativo estatístico entre o ML-TSKC FS e Modelos de Referência da Literatura

Nesta seção, fizemos um estudo comparativo para avaliar a performance do **ML-TSKC FS** em relação aos principais modelos da literatura para classificação multi-rótulo. Usamos o **teste de Friedman** para ver se havia diferenças significativas entre os modelos, e depois aplicamos o **teste pós-hoc de Bonferroni-Dunn** para comparações detalhadas entre pares de modelos.

- **Aplicação do Teste de Friedman**: Para verificar se havia diferenças significativas entre o desempenho do ML-TSKC FS e dos modelos de referência, aplicamos o teste de Friedman em cada métrica de avaliação: Average Precision (AP), Hamming Loss (HL), Ranking Loss (RL) e Coverage (CV).

- **Resultados do Teste de Friedman**

  Os resultados do teste de Friedman para cada métrica estão na Tabela 4.15. Observamos que para todas as métricas o valor p foi menor que 0.05, o que significa que podemos rejeitar a hipótese nula e concluir que há diferenças significativas entre os modelos.

Métrica	Estatística de Friedman	Valor p
Average Precision (AP)	21.27	< 0.001
Hamming Loss (HL)	11.78	< 0.001
Ranking Loss (RL)	16.27	< 0.001
Coverage (CV)	9.86	< 0.001

Tabela 4.15: Resultados do Teste de Friedman para cada métrica de avaliação

- **Análise Post-hoc Bonferroni-Dunn**: Com a confirmação de que existem diferenças significativas, realizamos uma análise mais detalhada usando o teste

pós-hoc de Bonferroni-Dunn. Este teste ajuda a identificar quais modelos têm desempenho significativamente inferior ao ML-TSKC FS.

- **Diagrama de Diferenças Críticas (CD)**

  O **(CD)** é usado para representar graficamente a comparação do desempenho de diferentes modelos em relação a uma métrica específica, com base no teste estatístico pós-hoc Bonferroni-Dunn. Este diagrama oferece uma visão clara e intuitiva das relações entre os modelos, destacando aqueles que apresentam diferenças estatisticamente significativas.

  No diagrama, cada modelo é representado por uma linha horizontal, e suas posições relativas refletem as classificações médias (*rankings*) obtidas para a métrica analisada. As principais características do diagrama são descritas a seguir:

  - **Classificação Média (Ranking)**:
    - Os modelos são dispostos de acordo com suas classificações médias, calculadas com base nos resultados da métrica em análise.
  - **Barra de Diferenças Críticas (CD)**:
    - A barra horizontal vermelha conecta os modelos cujas diferenças de desempenho não são estatisticamente significativas, de acordo com o intervalo crítico calculado.
    - O valor da Diferença Crítica (CD), indicado no topo do diagrama, é obtido utilizando a Eq.(4.7). Ele define o limite acima do qual as diferenças nos rankings são consideradas significativas.
  - **Linhas de Conexão**:
    - Linhas conectam modelos cujas diferenças estão dentro do intervalo crítico de (CD), indicando desempenhos estatisticamente semelhantes.
    - A ausência de conexão entre dois modelos implica diferenças estatisticamente significativas entre eles.

  A seguir, apresentamos os diagramas de Diferenças Críticas para as diferentes métricas analisadas, acompanhados de suas respectivas conclusões. Essa abordagem gráfica complementa os testes estatísticos numéricos previamente realizados, proporcionando uma interpretação visual e acessível das diferenças entre os modelos.

- **Diagrama de Diferenças Críticas (CD) para a métrica AP**

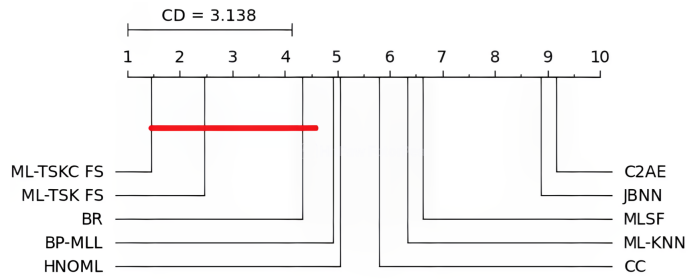


Figura 4.11: Diagrama de CD para a métrica AP: ML-TSKC FS vs Modelos da Literatura.

- **Conclusões para a Métrica AP**

  Com base no diagrama de diferenças críticas para a métrica AP, tiramos as seguintes conclusões sobre o desempenho do modelo ML-TSKC FS em comparação com os principais modelos da literatura.

  - **Desempenho do ML-TSKC FS**: O ML-TSKC FS teve a melhor classificação média, indicando que seu desempenho foi consistentemente superior aos outros modelos. Isso reforça que a inclusão da Integral de Choquet ajuda a capturar melhor as interações complexas entre os rótulos.
  - **Comparação com Modelos de Referência**: O ML-TSKC FS mostrou uma diferença significativa em relação a modelos como *ML-kNN* , *HNOML* , *MLSF* , *CC* , *C2AE* , *BP-MLL* e *JBNN* . Esse diferencial sugere que a metodologia neuro-fuzzy com a Integral de Choquet permite capturar melhor as interdependências entre rótulos.
  - **Comparação com o ML-TSK FS e Modelos Fuzzy**: A diferença entre o ML-TSKC FS , o ML-TSK FS e o BR foi menor, indicando que o uso da Integral de Choquet adiciona melhorias graduais. Essa vantagem é especialmente relevante para dados com interdependências complexas entre rótulos.

  Em resumo para a métrica AP o diagrama de diferenças críticas confirma que o ML-TSKC FS não só supera os modelos tradicionais, mas também apresenta uma melhoria incremental em relação ao ML-TSK FS. Esses resultados indicam que a aplicação da Integral de Choquet é uma solução eficaz para melhorar a precisão na classificação multi-rótulo.

- **Diagrama de Diferenças Críticas (CD) para a métrica HL**

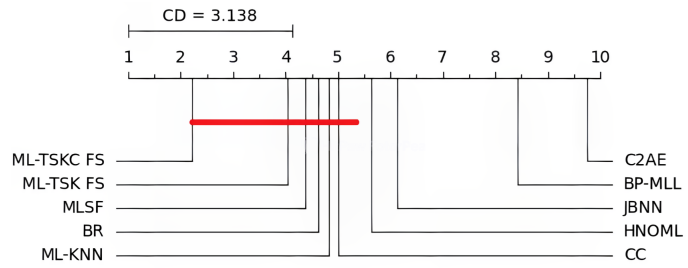


Figura 4.12: Diagrama de Diferenças Críticas (CD) para a métrica HL: ML-TSKC FS vs Modelos da Literatura.

- **Conclusões do Estudo Comparativo com Base na Métrica de HL**

  A análise do diagrama de diferenças críticas (CD) para a métrica HL mostra o desempenho do modelo ML-TSKC FS em comparação com outros modelos da literatura.

  - **Desempenho do ML-TSKC FS**: O ML-TSKC FS teve a melhor classificação média, indicando que seu desempenho foi consistentemente superior aos outros modelos. Isso reforça que a inclusão da Integral de Choquet ajuda a melhorar a precisão e a confiabilidade.
  - **Comparação com Modelos de Referência**: O ML-TSKC FS mostrou uma diferença significativa em relação a modelos como *HNOML* , *C2AE* , *BP-MLL* e *JBNN* . Esse diferencial sugere que a metodologia neuro-fuzzy com a Integral de Choquet permite capturar melhor as interdependências entre rótulos.
  - **Comparação com o ML-TSK FS e Modelos Fuzzy**: A diferença entre o ML-TSKC FS e os classificadores *ML-TSK FS* , *ML-kNN* , *MLSF* , *CC* , e *BR* foi menor, indicando que não a diferença significativa.

  A análise da métrica HL confirma que o ML-TSKC FS oferece um desempenho superior e consistente, especialmente em comparação com modelos tradicionais e redes neurais. Isso o torna uma escolha sólida para tarefas de classificação multi-rótulo, onde é essencial controlar erros e manter a precisão.

- **Diagrama de Diferenças Críticas (CD) para a métrica RL**

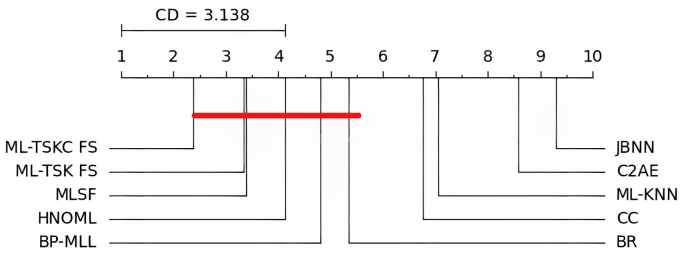


Figura 4.13: Diagrama de Diferenças Críticas (CD) para a métrica RL: ML-TSKC FS vs Modelos da Literatura.

- **Conclusões do Estudo Comparativo com Base na Métrica de RL**

  O diagrama de diferenças críticas (CD) para a métrica RL mostra o desempenho do modelo ML-TSKC FS em comparação com outros modelos da literatura.

  - **Desempenho do ML-TSKC FS**: O ML-TSKC FS teve a melhor classificação média, indicando que seu desempenho foi consistentemente superior aos outros modelos.
  - **Comparação com Modelos de Referência**: O ML-TSKC FS mostrou uma diferença significativa em relação a modelos como *CC* , *ML-kNN* , *C2AE* e *JBNN* . Esse diferencial sugere que a metodologia neuro-fuzzy com a Integral de Choquet permite ordenar corretamente os rótulos relevantes.
  - **Comparação com o ML-TSK FS e Modelos Fuzzy**: A diferença entre o ML-TSKC FS e os classificadores *ML-TSK FS* , *MLSF* , *HNOML* , *BP-MLL* , e *BR* foi menor, indicando que o ganho não foi suficiente para obter uma diferença significativa.

  O diagrama de diferenças críticas para a métrica RL reforça a posição do ML-TSKC FS como uma abordagem eficaz para a classificação multi-rótulo e reforça sua viabilidade como uma solução alternativa prática e interpretável para esses cenários.

- **Diagrama de Diferenças Críticas (CD) para a métrica CV**

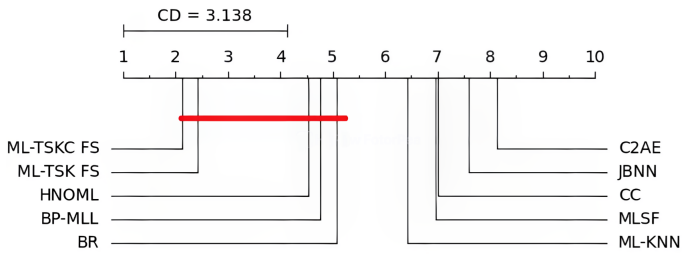


Figura 4.14: Diagrama de Diferenças Críticas (CD) para a métrica CV: ML-TSKC FS vs Modelos da Literatura.

- **Conclusões do Estudo Comparativo com Base na Métrica de CV**

  A análise do diagrama de diferenças críticas (CD) para a métrica de CV mostra o desempenho do modelo ML-TSKC FS em comparação com outros modelos da literatura.

  - **Desempenho do ML-TSKC FS**: O ML-TSKC FS teve a melhor classificação média, indicando que seu desempenho foi consistentemente superior aos outros modelos. Isso reforça que a inclusão da Integral de Choquet ajuda a minimizar o esforço para alcançar os rótulos corretos na lista.
  - **Comparação com Modelos de Referência**: O ML-TSKC FS mostrou uma diferença significativa em relação a modelos como *ML-kNN* , *MLSF* , *CC* , *JBNN* e *C2AE* . Esse diferencial sugere que a metodologia neuro-fuzzy com a Integral de Choquet permite capturar melhor as interdependências entre rótulos.
  - **Comparação com o ML-TSK FS e Modelos Fuzzy**: A diferença entre o ML-TSKC FS e os classificadores *ML-TSK FS* , *HNOML* , *BP-MLL* e *BR* foi menor, indicando que não a diferença significativa.

  O diagrama de diferenças críticas para a métrica CV reforça a posição do ML-TSKC FS como uma solução para classificação multi-rótulo em cenários onde é importante a otimização da profundidade de cobertura dos rótulos é relevante

4.2.4 Conclusão do capítulo

Neste capítulo, foi analisado o desempenho do modelo ML-TSKC-FS em conjuntos de dados de classificação multi-rótulo, destacando sua versatilidade e eficácia.

Na primeira parte do estudo, os resultados dos testes estatísticos de Wilcoxon indicaram que o modelo ML-TSKC-FS, ao utilizar a Integral de Choquet nos antecedentes das regras, superou o modelo ML-TSK FS em várias métricas, confirmando que as melhorias obtidas são estatisticamente significativas.

Na segunda parte, que comparou o ML-TSKC-FS com modelos consolidados da literatura, os testes de Friedman e as Diferenças Críticas de Bonferroni-Dunn mostraram que o ML-TSKC-FS obteve a melhor classificação média, indicando um desempenho consistentemente superior em relação a outros modelos. Embora essas melhorias não tenham sido suficientes para superar estatisticamente todos os métodos, o modelo proposto superou vários deles, incluindo redes neurais profundas e métodos especializados.

Esses resultados comprovam a robustez e a eficácia do modelo ML-TSKC-FS, posicionando-o como uma abordagem promissora na área de classificação multi-rótulo neuro-fuzzy. A inclusão da Integral de Choquet mostrou-se fundamental para aprimorar a capacidade do modelo em capturar as interações entre atributos e rótulos, resultando em um desempenho estatisticamente superior em relação a diversos modelos. Essa melhoria torna o ML-TSKC-FS uma alternativa vantajosa para aplicações práticas que exigem precisão e interpretabilidade.

Capítulo 5

CONCLUSÃO

A principal contribuição deste trabalho foi a introdução do modelo ML-TSKC FS, um sistema neuro-fuzzy desenvolvido para a classificação multi-rótulo, que se destaca por utilizar a integral de Choquet na agregação de informações para a ativação das regras no sistema de inferência. Essa abordagem oferece uma solução flexível e robusta para a tarefa de classificação multi-rótulo, especialmente ao lidar com a imprecisão, ambiguidade e incerteza presentes nos dados. Ao capturar as interações entre diferentes atributos, o modelo ML-TSKC FS melhora a precisão e a consistência dos resultados.

Resposta à pergunta de pesquisa

Com base nos resultados obtidos, podemos responder à nossa pergunta de pesquisa.

A inclusão da Integral Discreta de Choquet na agregação de atributos para calcular a ativação das regras em um modelo de classificação multi-rótulo neuro-fuzzy pode levar a uma melhoria no desempenho do modelo original?

Sim, os resultados apresentados neste trabalho confirmam que a aplicação da Integral Discreta de Choquet na agregação de atributos melhora o desempenho do modelo em sistemas de classificação multi-rótulo neuro-fuzzy. A integração dessa técnica aprimorou a capacidade do modelo proposto ML-TSKC-FS de capturar as interações complexas entre atributos, proporcionando resultados estatisticamente superiores em relação ao modelo original. Esse ganho de desempenho posiciona o

modelo como uma abordagem promissora e vantajosa para aplicações práticas que exigem precisão e interpretabilidade.

5.1 Contribuições e Resultados Principais

Os principais resultados e contribuições deste trabalho podem ser resumidos como:

- **Flexibilidade**: A inclusão da integral de Choquet nos antecedentes das regras oferece uma forma mais adaptável de agregar informações, captando nuances que métodos tradicionais não conseguem abordar.
- **Modelagem de Interações Complexas**: A inclusão de medidas fuzzy, permite uma análise detalhada da sinergia entre atributos, essencial para representar adequadamente a complexidade dos dados de entrada e melhorar a consistência e precisão dos resultados.
- **Desempenho Superior em Classificação Multi-Rótulo**: O modelo ML-TSKC FS demonstrou desempenho superior em termos de precisão e robustez, adaptando-se bem a diferentes bases de dados com múltiplos rótulos e exibindo uma menor taxa de erro quando comparado a modelos tradicionais.
- **Maior Adaptabilidade**: O modelo pode ajustar dinamicamente a influência de cada atributo com base em sua importância combinada com outros. Isso é particularmente benéfico em problemas de classificação multi-rótulo, onde as interações frequentemente exibem relações complexas.
- **Maior Precisão**: Ao considerar as relações intrínsecas entre os atributos, a integral de Choquet pode levar a previsões mais precisas em tarefas desafiadoras de classificação multi-rótulo, onde múltiplas saídas são igualmente importantes e as interações entre os atributos desempenham um papel crucial.

Essas vantagens se refletem em duas áreas principais de contribuição primeiro, no aumento da precisão estatística do modelo ao competir em diferentes métricas e, segundo, na capacidade de integração em sistemas complexos de inferência fuzzy.

5.2 Limitações do Trabalho

Apesar dos resultados promissores, este trabalho possui algumas limitações que devem ser consideradas:

- **Dependência da Complexidade Computacional**: A utilização da integral de Choquet, apesar de proporcionar ganhos em precisão, exige maior poder computacional em comparação com operadores tradicionais.

- **Necessidade de Ajuste dos Parâmetros Fuzzy**: A configuração dos parâmetros fuzzy, incluindo pesos da integral de Choquet, requer um ajuste cuidadoso.
- **Escalabilidade para Conjuntos de Dados Extensos**: Embora o modelo tenha se mostrado eficiente em conjuntos de dados de tamanho médio, a sua aplicação em dados extremamente volumosos pode ser limitada pois o custo de execução e o custo de equipamento necessário para executar o código aumentaria em grão medida.

5.3 Trabalhos Futuros

Com base nas limitações e descobertas deste trabalho, as pesquisas futuras se concentrarão em áreas-chave.

- O uso de outras medidas fuzzy no modelo poderia melhorar seu desempenho.
- Generalizações da Integral de Choquet podem oferecer um processo de agregação mais refinado, aprimorando ainda mais a precisão da classificação multirótulo.
- Estudaremos adaptações do modelo para o caso de conjuntos de dados semirotulados (como, por exemplo, em [Gull e Aguilar 2024]).
- Automatização da Configuração de Parâmetros Fuzzy, desenvolver algoritmos de aprendizado de máquina que ajustem automaticamente os parâmetros fuzzy, visando melhorar a adaptabilidade do modelo.
- Ampliar o escopo do modelo em diferentes bases de dados e métricas, investigando como a adaptação da integral de Choquet impacta novas áreas de classificação.
- Aplicações Práticas da Integral de Choquet, validar o modelo em cenários reais, como previsão de demanda em mercados dinâmicos ou diagnóstico médico, para testar a eficácia em ambientes práticos.

Bibliografia

Alpaydin, Ethem (2010). *Introduction to Machine Learning*. Second Edition. MIT press (ver pp. 7, 8, 10, 19).

Barros, Laécio Carvalho de e Rodney Carlos Bassanezi (2010). *Tópicos de lógica fuzzy e biomatemática*. Grupo de Biomatemática, Instituto de Matemática, Estatística e Computação ... (ver p. 44).

Beliakov, James e Wu (2020). *Discrete fuzzy measures*. Springer (ver p. 48).

Benvenuti, Pietro, Doretta Vivona et al. (2002). «Monotone set functions-based integral». Em: *Handbook Of Measure (Cap. 33) Theory*. Pap E., pp. 1329–1379 (ver p. 49).

Berenji, Hamid R e Pratap Khedkar (1992). *Learning and tuning fuzzy logic controllers through reinforcements*. Rel. téc., pp. 724–740 (ver p. 40).

Bezdek (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Springer, p. 256. DOI: 10.1007/978-1-4757-0450-1 (ver p. 59).

— (1999). «Activation and clustering functions in fuzzy logic and applications». Em: *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 1345–1352 (ver p. 33).

Bezdek, Robert Ehrlich e William Full (1984). «FCM: The fuzzy c-means clustering algorithm». Em: *Computers & Geosciences* 10.2-3, pp. 191–203. DOI: 10.1016/0098-3004(84)90020-7 (ver p. 68).

Bezdek, James Keller et al. (1999). *Fuzzy models and algorithms for pattern recognition and image processing*. Vol. 4. Springer Science & Business Media (ver p. 63).

Blockeel, Hendrik, Sašo Džeroski e Jasna Grbović (1999). «Simultaneous prediction of multiple chemical parameters of river water quality with TILDE». Em: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, pp. 32–40 (ver p. 17).

Bouchon-Meunier, B., ed. (1998). *Aggregation and Fusion of Imperfect Information*. New York: Physica-Verlag (ver p. 33).

Boutell, Matthew R et al. (2004). «Learning multi-label scene classification». Em: *Pattern recognition* 37.9, pp. 1757–1771 (ver pp. 17, 81).

Bozyiğit, Mahmut Can et al. (2024). «Parametric picture fuzzy cross-entropy measures based on d-Choquet integral for building material recognition». Em: *Applied Soft Computing* 166, p. 112167. DOI: 10.1016/j.asoc.2024.112167 (ver p. 64).

Briggs, Forrest, Yonghong Huang et al. (2013). «The 9th annual MLSP competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment». Em: *2013 IEEE international workshop on machine learning for signal processing (MLSP)*. IEEE, pp. 1–8 (ver p. 80).

Briggs, Forrest, Balaji Lakshminarayanan et al. (2012). «Acoustic classification of multiple simultaneous bird species: A multi-instance multi-label approach». Em: *The Journal of the Acoustical Society of America* 131.6, pp. 4640–4650 (ver p. 17).

Bustince, Fernandez et al. (2010). «Overlap functions». Em: *Nonlinear Analysis: Theory, Methods & Applications* 72.3-4, pp. 1488–1499 (ver p. 42).

Bustince, Humberto et al. (2016). «Pre-aggregation functions: definition, properties and construction methods». Em: *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, pp. 294–300 (ver p. 65).

Bustince, Radko Mesiar et al. (2021). «The Evolution of the Notion of Overlap Functions». Em: *Fuzzy Approaches for Soft Computing and Approximate Reasoning: Theories and Applications: Dedicated to Bernadette Bouchon-Meunier*. Ed. por Marie-Jeanne Lesot e Christophe Marsala. Cham: Springer International Publishing, pp. 21–29. ISBN: 978-3-030-54341-9. DOI: 10.1007/978-3-030-54341-9_3 (ver p. 62).

Bustince, Miguel Pagola et al. (2011). «Grouping, overlap, and generalized bientropic functions for fuzzy modeling of pairwise comparisons». Em: *IEEE Transactions on Fuzzy Systems* 20.3, pp. 405–415 (ver p. 42).

Chen, Guanrong e Trung Tat Pham (2000). *Introduction to fuzzy sets, fuzzy logic, and fuzzy control systems*. CRC press (ver pp. 19, 28, 31).

Chia-Feng, Juang e Lin Chin-Teng (1998). «An online self-constructing neural fuzzy inference network and its applications». Em: *IEEE transactions on Fuzzy Systems* 6.1, pp. 12–32 (ver p. 40).

Choquet, Gustave (1954). «Theory of capacities». Em: *Annales de l'institut Fourier*. Vol. 5, pp. 131–295 (ver p. 47).

Chung, Fu-Lai et al. (2006). «Catsmlp: Toward a robust and interpretable multilayer perceptron with sigmoid activation functions». Em: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 36.6, pp. 1319–1331 (ver p. 63).

Combettes, Patrick L e Valérie R Wajs (2005). «Signal recovery by proximal forward-backward splitting». Em: *Multiscale modeling & simulation* 4.4, pp. 1168–1200 (ver p. 74).

Cox, Earl (1994). *The Fuzzy Systems Handbook*. Boston: AP Professional (ver pp. 29, 33, 36).

Czogała, E. e J. M. Leski (2000). *Fuzzy and Neuro-Fuzzy Intelligent Systems*. Heidelberg: Physica-Verlag (ver p. 37).

Deniz, Emre, Hasan Erbay e Mustafa Coşar (2022). «Multi-label classification of e-commerce customer reviews via machine learning». Em: *Axioms* 11.9, p. 436 (ver p. 17).

Elisseeff, André e Jason Weston (2001). «A kernel method for multi-labelled classification». Em: *Advances in neural information processing systems* 14 (ver p. 82).

Ferrero-Jaurrieta, Mikel et al. (2023). «VCI-LSTM: Vector Choquet Integral-Based Long Short-Term Memory». Em: *IEEE Transactions on Fuzzy Systems* 31.7, pp. 2238–2250. DOI: 10.1109/TFUZZ.2022.3222035 (ver pp. 3, 64).

Figueiredo, Mauricio e Fernando Gomide (1999). «Design of fuzzy systems using neurofuzzy networks». Em: *IEEE transactions on Neural Networks* 10.4, pp. 815–827 (ver p. 40).

Fitch, Frederic B (1944). «McCulloch Warren S. and Pitts Walter. A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biophysics, vol. 5, pp. 115–133». Em: (ver p. 20).

Gacto, Maria Jose, Rafael Alcalá e Francisco Herrera (2011). «Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures». Em: *Information Sciences* 181.20, pp. 4340–4360 (ver pp. 35, 37).

Gareth et al. (2013). *An Introduction to Statistical Learning: With Applications in R*. Springer Texts in Statistics. New York, NY: Springer (ver p. 85).

Gibaja, Eva e Sebastián Ventura (2015). «A tutorial on multilabel learning». Em: *ACM Computing Surveys (CSUR)* 47.3, pp. 1–38 (ver p. 2).

Gonçalves et al. (2013). «A genetic algorithm for optimizing the label ordering in multi-label classifier chains». Em: *2013 IEEE 25th international conference on tools with artificial intelligence*. IEEE, pp. 469–476 (ver p. 81).

Grabisch, Michel (2000). «A graphical interpretation of the Choquet integral». Em: *IEEE Transactions on Fuzzy Systems* 8.5, pp. 627–631 (ver p. 48).

Grabisch, Michel e Christophe Labreuche (2010). «A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid». Em: *Annals of Operations Research* 175, pp. 247–286 (ver p. 48).

Grabisch, Michel, Jean-Luc Marichal et al. (2009). *Aggregation functions*. Vol. 127. Cambridge University Press (ver p. 42).

Grabisch, Michel, Marc Roubens et al. (2000). «Application of the Choquet integral in multicriteria decision making». Em: *Fuzzy Measures and Integrals-Theory and Applications*, pp. 348–374 (ver p. 49).

Grady, Leo e Gareth Funka-Lea (2004). «Multi-label image segmentation for medical applications based on graph-theoretic electrical potentials». Em: *International Workshop on Mathematical Methods in Medical and Biomedical Image Analysis*. Springer, pp. 230–245 (ver p. 17).

Gu, Xiaowei, Plamen Angelov e Hai-Jun Rong (2019). «Local optimality of self-organising neuro-fuzzy inference systems». Em: *Information Sciences* 503, pp. 351–380 (ver p. 40).

Gull, Carlos Quintero e Jose Aguilar (2024). «A semi-supervised learning algorithm for multi-label classification and multi-assignment clustering problems based on a Multivariate Data Analysis». Em: *Engineering Applications of Artificial Intelligence* 137, p. 109189. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2024.109189 (ver p. 111).

Han, Sun e Yingnan Fan (2008). «An improved fuzzy neural network based on T–S model». Em: *Expert Systems with Applications* 34.4, pp. 2905–2920 (ver pp. 63, 64).

Hastie, Trevor, Robert Tibshirani e Jerome Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Vol. 2. Springer Series in Statistics. New York, NY: Springer Science & Business Media (ver pp. 85, 86).

Haykin, Simon (2001). *Neural Networks and Learning Machines*. Prentice Hall (ver pp. 21, 27, 28).

He, Huihui e Rui Xia (2018). «Joint binary neural network for multi-label learning with applications to emotion classification». Em: *Natural Language Processing and Chinese Computing: 7th CCF International Conference, NLPCC 2018, Hohhot, China, August 26–30, 2018, Proceedings, Part I 7*. Springer, pp. 250–259 (ver pp. 16, 96).

Herrera, Francisco et al. (2016). *Multilabel classification*. Springer (ver pp. 1, 8–10, 12, 15, 18).

Jang e Jyh-Shing (1993). *ANFIS: Adaptive-Network-Based Fuzzy Inference System*. IEEE Transactions on Systems, Man, e Cybernetics (ver pp. 2, 28, 40, 62).

Jang, Jyh-Shing et al. (1997). «Neuro-fuzzy and soft computing-a computational approach to learning and machine intelligence [Book Review]». Em: *IEEE Transactions on automatic control* 42.10, pp. 1482–1484 (ver p. 19).

Kasabov (2002). *Evolving Connectionist Systems: The Knowledge Engineering Approach*. Springer (ver p. 28).

Kasabov e Song Qun (1999). *Dynamic Evolving Fuzzy Neural Networks with"m-out-of-n"Activation Nodes for On-line Adaptive Systems*. Department of Information Science, University of Otago (ver p. 40).

Kasabov, Song e Qun (2002). «DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction». Em: *IEEE transactions on Fuzzy Systems* 10.2, pp. 144–154 (ver pp. 2, 40, 62).

Katakis, Ioannis, Grigorios Tsoumakas e Ioannis Vlahavas (2008). «Multilabel text classification for automated tag suggestion». Em: *ECML PKDD discovery challenge* 75, p. 2008 (ver pp. 17, 81).

Keller, James et al. (2016). *Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation*. IEEE Press (ver pp. 19, 29).

Kim e Kasabov (1999). «HyFIS: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems». Em: *Neural networks* 12.9, pp. 1301–1319 (ver pp. 2, 40, 62).

Kim e Lee-Chae (2023). «Economic preference for semiconductor trade deals using similarity measures defined by Choquet integrals». Em: *Computational and Applied Mathematics* 42, p. 224. DOI: 10.1007/s40314-023-02365-z (ver p. 64).

Kosko, Bart (1992). *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Prentice-Hall (ver pp. 9, 19, 33).

Krishnan, Anath Rau, Maznah Mat Kasim e Engku Muhammad Nazri Engku Abu Bakar (2015). «A short survey on the usage of Choquet integral and its associated fuzzy measure in multiple attribute analysis». Em: *Procedia Computer Science* 59, pp. 427–434 (ver p. 49).

Lee e Lin (1991). *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Prentice Hall (ver p. 28).

Leng, Gang, Girijesh Prasad e Thomas Martin McGinnity (2004). «An on-line algorithm for creating self-organizing fuzzy neural networks». Em: *Neural Networks* 17.10, pp. 1477–1493 (ver p. 40).

Łęski, Jacek M (2002). «Improving the generalization ability of neuro-fuzzy systems by [epsilon]-insensitive learning». Em: *International Journal of Applied Mathematics and Computer Science* 12 (ver pp. 15, 96).

Lewis, David D et al. (2004). «Rcv1: A new benchmark collection for text categorization research». Em: *Journal of machine learning research* 5.Apr, pp. 361–397 (ver p. 81).

Lin et al. (1991). «Neural-network-based fuzzy logic control and decision system». Em: *IEEE Transactions on Computers* 40.12, pp. 1320–1336 (ver pp. 2, 40).

Lou, Qiongdan et al. (2021). «Multilabel Takagi-Sugeno-Kang Fuzzy System». Em: *IEEE Transactions on Fuzzy Systems* 30.9, pp. 3410–3425. DOI: 10.1109/TFUZZ.2021.3115967 (ver pp. 2, 16, 40, 58, 62, 88, 96, 123).

Lucca, Giancarlo, Graçaliz Pereira Dimuro et al. (2019). «Improving the Performance of Fuzzy Rule-Based Classification Systems Based on a Nonaveraging Generalization of CC-Integrals Named $C_{F_1F_2}$-Integrals». Em: *IEEE Transactions on Fuzzy Systems* 27.1, pp. 124–134. DOI: 10.1109/TFUZZ.2018.2871000 (ver p. 64).

Lucca, Giancarlo, Sanz, Graçaliz Dimuro et al. (2018). «CF-integrals: A new family of pre-aggregation functions with application to fuzzy rule-based classification systems». Em: *Information Sciences* 435, pp. 94–110. DOI: 10.1016/j.ins.2017.12.029 (ver p. 64).

Lucca, Giancarlo, Sanz, Dimuro et al. (2019). «Analyzing the performance of different fuzzy measures with generalizations of the Choquet integral in classification problems». Em: *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, pp. 1–6 (ver pp. 3, 65).

Marco-Detchart, Cedric et al. (2021). «Neuro-inspired edge feature fusion using Choquet integrals». Em: *Information Sciences* 581, pp. 740–754. DOI: 10.1016/j.ins.2021.10.016 (ver pp. 3, 64).

Min-Ling, Zhang e Zhou Zhi-Hua (2006). «Multilabel neural networks with applications to functional genomics and text categorization». Em: *IEEE transactions on Knowledge and Data Engineering* 18.10, pp. 1338–1351 (ver pp. 16, 95).

Mitchell, Tom (1997). *Machine learning*. Vol. 1. 9. McGraw-hill New York (ver pp. 7–10, 18, 19).

Murofushi, Toshiaki e Sugeno (1989). «An interpretation of fuzzy measures and the Choquet integral as an integral with respect to a fuzzy measure». Em: *Fuzzy sets and Systems* 29.2, pp. 201–227 (ver p. 47).

Nando, Freitas (2003). «Matching words and pictures». Em: *Journal of Machine Learning Research* 3 (ver p. 81).

Nauck, Detlef e Rudolf Kruse (1999). «Neuro-fuzzy systems for function approximation». Em: *Fuzzy sets and systems* 101.2, pp. 261–271 (ver p. 40).

Oliveira, JR (2007). «Redes Neurais e suas Aplicações». Em: *Revista Brasileira de Computação* (ver p. 28).

Pedrycz, Witold e Fernando Gomide (1998). «An introduction to fuzzy sets: analysis and design (complex adaptive systems)». Em: *NetLibrary, Incorporated* (ver p. 44).

Qing, Zhang, Jeong Sungmoon e Lee Minho (2012). «Autonomous emotion development using incremental modified adaptive neuro-fuzzy inference system». Em: *Neurocomputing* 86, pp. 33–44 (ver p. 40).

Ratnarajah, Nagulan e Anqi Qiu (2014). «Multi-label segmentation of white matter structures: application to neonatal brains». Em: *NeuroImage* 102, pp. 913–922 (ver p. 17).

Read, Jesse et al. (2011). «Classifier chains for multi-label classification». Em: *Machine learning* 85, pp. 333–359 (ver pp. 1, 15, 96).

Riaz, Muhammad et al. (2023). «Generalized linear diophantine fuzzy Choquet integral with application to the project management and risk analysis». Em: *Computational and Applied Mathematics* 42, p. 286. DOI: doi.org/10.1007/s40314-023-02421-8 (ver p. 64).

Ross, Timothy J. (2004). *Fuzzy Logic with Engineering Applications*. John Wiley Sons (ver pp. 29–36).

Ruspini (1998). «Applications of intelligent multiobjective fuzzy decision making». Em: *Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications*. Springer, pp. 514–520 (ver p. 37).

Ruspini e Piero Patrone, eds. (1998). *Handbook of Fuzzy Computation*. IOP Publishing Ltd (ver pp. 34, 36, 37).

Schapire, Robert E e Yoram Singer (2000). «BoosTexter: A boosting-based system for text categorization». Em: *Machine learning* 39, pp. 135–168 (ver p. 82).

Schulz, Axel, Eneldo Loza Mencía e Benedikt Schmidt (2016). «A rapid-prototyping framework for extracting small-scale incident-related information in microblogs: application of multi-label classification on tweets». Em: *Information Systems* 57, pp. 88–110 (ver p. 17).

Sugeno (1974). «Theory of Fuzzy Integrals and Its Applications». PhD Thesis. Tese de doutoramento. Tokyo, Japan: Tokyo Institute of Technology (ver p. 45).

Sugeno e GT Kang (1986). «Fuzzy modelling and control of multilayer incinerator». Em: *Fuzzy sets and systems* 18.3, pp. 329–345 (ver p. 57).

— (1988). «Structure identification of fuzzy model». Em: *Fuzzy sets and systems* 28.1, pp. 15–33 (ver p. 57).

Sulzberger, Sandra M, N Tschichold-Gurman e Sjur J Vestli (1993). «FUN: Optimization of fuzzy rule based systems using neural networks». Em: *IEEE international conference on neural networks*. IEEE, pp. 312–316 (ver p. 40).

Sun, Mineichi Kudo e Keigo Kimura (2016). «Multi-label classification with meta-label-specific features». Em: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, pp. 1612–1617 (ver pp. 15, 96).

Suthaharan, Shan (2016). *Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning*. Springer (ver pp. 9, 12, 19).

Tano, Shun'ichi, Takuya Oyama e Thierry Arnould (1996). «Deep combination of fuzzy inference and neural network in fuzzy inference software—FINEST». Em: *Fuzzy Sets and Systems* 82.2, pp. 151–160 (ver p. 40).

Tehrani, Ali Fallah, Weiwei Cheng e Eyke Hullermeier (2012). «Preference learning using the Choquet integral: The case of multipartite ranking». Em: *IEEE Transactions on Fuzzy Systems* 20.6, pp. 1102–1113 (ver p. 49).

Tomohiro, Takagi e Sugeno (1985). «Fuzzy identification of systems and its applications to modeling and control». Em: *IEEE transactions on systems, man, and cybernetics* 1, pp. 116–132 (ver p. 57).

Tsoumakas, Grigorios e Ioannis Katakis (2007). «Multi-label classification: An overview». Em: *International Journal of Data Warehousing and Mining (IJDWM)* 3.3, pp. 1–13 (ver p. 1).

Tsoumakas, Grigorios, Ioannis Katakis e Ioannis Vlahavas (2008). «Effective and efficient multilabel classification in domains with large number of labels». Em: *Proc.*

ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08). Vol. 21, pp. 53–59 (ver p. 80).

Turnbull, Douglas et al. (2008). «Semantic annotation and retrieval of music and sound effects». Em: *IEEE Transactions on Audio, Speech, and Language Processing* 16.2, pp. 467–476 (ver p. 80).

Uebele (1995). «Aggregation functions in fuzzy systems». Em: *Fuzzy Systems Handbook*, pp. 295–317 (ver p. 33).

Uebele, Shigeo e Lan Ming-Shong (1995). «A neural-network-based fuzzy classifier». Em: *IEEE Transactions on Systems, Man, and Cybernetics* 25.2, pp. 353–361 (ver pp. 62, 63).

Wang, Haobo et al. (2021). «Collaboration based multi-label propagation for fraud detection». Em: *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 2477–2483 (ver p. 17).

Wang, Hongjuan, Yi Liu e ChunYu Zhao (2023). «q-rung orthopair fuzzy bi-direction Choquet integral based on TOPSIS method for multiple attribute group decision making». Em: *Computational and Applied Mathematics* 42, p. 105. DOI: 10.1007/s40314-023-02222-z (ver p. 64).

Wang et al. (2024). «An integrated interval-valued spherical fuzzy Choquet integral based decision making model for prioritizing risk in Fine-Kinney». Em: *Engineering Applications of Artificial Intelligence* 127, p. 107437. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2023.107437 (ver p. 64).

Wei, Tong et al. (2022). «A survey on extreme multi-label learning». Em: *arXiv preprint arXiv:2210.03968* (ver p. 1).

Wieczynski, Dimuro et al. (2020). «Generalizing the GMC-RTOPSIS Method using CT-integral Pre-aggregation Functions». Em: *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–8. DOI: 10.1109/FUZZ48607.2020.9177859 (ver pp. 3, 65).

Wieczynski, Fumanal-Idocin et al. (2022). «d-XC Integrals: On the Generalization of the Expanded Form of the Choquet Integral by Restricted Dissimilarity Functions and Their Applications». Em: *IEEE Transactions on Fuzzy Systems* 30.12, pp. 5376–5389. DOI: 10.1109/TFUZZ.2022.3176916 (ver p. 64).

Wieczynski, Giancarlo Lucca et al. (2023). «dC_F-Integrals: Generalizing C_F-Integrals by Means of Restricted Dissimilarity Functions». Em: *IEEE Transactions on Fuzzy Systems* 31.1, pp. 160–173. DOI: 10.1109/TFUZZ.2022.3184054 (ver p. 64).

Yeh, Chih-Kuan et al. (2017). «Learning deep latent space for multi-label classification». Em: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 31. 1 (ver pp. 16, 96).

Zadeh (1965). «Fuzzy sets». Em: *Information and control* 8.3, pp. 338–353 (ver pp. 2, 28, 29).

— (1975). «The concept of a linguistic variable and its application to approximate reasoning-III». Em: *Information sciences* 9.1, pp. 43–80 (ver pp. 30–32).

Zhang, Min Ling et al. (2013). *A review on multi-label learning algorithms*. DOI: 10.1109/TKDE.2013.39 (ver p. 1).

Zhang, Radko Mesiar e Endre Pap (2024). «Double set-function Choquet integral with applications». Em: *Information Sciences* 677, p. 120948. DOI: 10.1016/j.ins.2024.120948 (ver p. 64).

Zhang, Ziwei Yu et al. (2019). «Hybrid noise-oriented multilabel learning». Em: *IEEE transactions on cybernetics* 50.6, pp. 2837–2850 (ver pp. 16, 96).

Zhang e Zhou Zhi-Hua (2007). «ML-KNN: A lazy learning approach to multi-label learning». Em: *Pattern recognition* 40.7, pp. 2038–2048 (ver pp. 15, 95).

Zhang e Zhou (2007). «Multi-label learning by instance differentiation». Em: *AAAI*. Vol. 7, pp. 669–674 (ver p. 81).

Zhang, Zhou e Grigorios Tsoumakas (2009). «Learning from multi-label data». Em: *ECML/PKDD*. Vol. 9 (ver p. 12).

Apêndice A: Código Utilizado no Estudo

Este apêndice apresenta o código utilizado no estudo, desenvolvido em Matlab como uma modificação do trabalho de [Lou et al. 2021]. A implementação incorpora a Integral de Choquet e explora diferentes medidas fuzzy, permitindo modelar de maneira mais robusta as interações complexas entre atributos em sistemas multi-rótulo.

Estrutura do Código

A seguir, detalhamos a estrutura do código, destacando as principais etapas e funcionalidades.

1. Configuração Inicial de Parâmetros

O código inicia configurando os parâmetros de busca (`alpha`, `beta`, `gamma`, `k`, `h`), que são fundamentais para o processo de otimização do modelo. Esses parâmetros definem o espaço de busca para os hiperparâmetros e influenciam diretamente os resultados obtidos.

```
function [ BestParameter, BestResult ] =
    ML_TSKFS_adaptive_validate(data, target, oldOptmParameter,
    TSKoptions)
optmParameter          = oldOptmParameter;
alpha_searchrange      = oldOptmParameter.alpha_searchrange;
beta_searchrange       = oldOptmParameter.beta_searchrange;
gamma_searchrange      = oldOptmParameter.gamma_searchrange;

k_searchrange          = TSKoptions.k_searchrange;
h_searchrange          = TSKoptions.h_searchrange;
q_s = 1;

total = length(alpha_searchrange) * length(beta_searchrange) * ...
        length(gamma_searchrange) * length(k_searchrange) * length
            (h_searchrange);
```

2. Busca Otimizada por Hiperparâmetros

Nesta etapa, o código realiza uma busca exaustiva pelos melhores hiperparâmetros. Essa busca cobre todas as combinações possíveis dos valores definidos, utilizando laços aninhados. A cada iteração, o modelo é avaliado, e os resultados são registrados para análise.

```
index = 1;
parameter_cell = zeros(total, 35);
ii = 1;
for p = 1:length(k_searchrange)
    for q = 1:length(h_searchrange)
        TSKoptions.k = k_searchrange(p);
        TSKoptions.h = h_searchrange(q);
        [v, b] = gene_ante_fcm(data, TSKoptions);
        [G_data] = calc_x_gf(data, v, b);

        train_data = G_data;
        num_train = size(train_data, 1);
        randorder = randperm(num_train);

        BestResult = zeros(15, 1);
        num_cv = 5;

        for i = 1:length(alpha_searchrange)
            for j = 1:length(beta_searchrange)
                for k = 1:length(gamma_searchrange)
                    fprintf('\n- %d/%d: TSK_k = %f, TSK_h = %f,
                        alpha = %f, beta = %f, gamma = %f', ...
                            index, total, k_searchrange(p),
                                h_searchrange(q), ...
                            alpha_searchrange(i), beta_searchrange
                                (j), gamma_searchrange(k));
                    index = index + 1;
```

3. Função da Integral de Choquet

Inclui-se a função de cálculo da Integral de Choquet, utilizada como parte do modelo.

```
 function [x_g] = calc_x_gf(x,v,b)
n_examples = size(x,1);
x_e = [x,ones(n_examples,1)];

[k,d] = size(v);

B = [];
```

```
8
9  p = 2;
10 q = 2;
11
12
13 for i=1:k
14
15     v1 = repmat(v(i,:),n_examples,1);
16     bb = repmat(b(i,:),n_examples,1);
17
18
19     v2 = exp(-(x-v1).^2./(2*bb.^2));
20
21
22     v3 = v2-v2;
23     [A,I]=sort(v2,2);
24
25
26     q_t = 4;
27     switch(q_t)
28         case 1
29             wt(:,i) = exp(-sum((x-v1).^2./(2*bb),2));
30         case 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
31             wt(:,i) = min(v2,[],2);
32         case 3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
33             wt(:,i) = max(v2,[],2);
34
35         case 4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% medida uniforme
36             for j=1:d
37                 if j == 1
38                     v3(:,j) = A(:,j);
39
40                 end
41                 if j > 1
42                     v3(:,j) = (A(:,j) - A(:,j-1)).*(d-j+1)/d;
43
44
45                 end
46             end
47
48             wt(:,i) = sum(v3,2);
49         case 5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% medida relativa
50             for j = 1:d
51                 if j == 1
52                     v3(:, j) = A(:, j);
53                     B(j,k) = 1;
54                 end
55
56                 if j > 1
```

```
57                      PP = I;
58                      QQ = I(:, j:d);
59
60                      B(j,k) = sum(QQ(j, :)) / sum(PP(j, :));
61                      v3(:, j) = (A(:, j) - A(:, j - 1)) * B(j,k);
62                  end
63              end
64              %display(B);
65              wt(:, i) = sum(v3, 2);
66          case 6 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% medida produto
67              for j = 1:d
68                  if j == 1
69                      v3(:, j) = A(:, j);
70
71                  end
72
73                  if j > 1
74
75                      QQ = I(:, 1:j-1);
76
77                      v3(:, j) = (A(:, j) - A(:, j - 1))./prod(QQ,2)
                            ;
78                  end
79              end
80
81              wt(:, i) = sum(v3, 2);
82          case 7 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% medida potencia
83              for j=1:d
84                  if j == 1
85                      v3(:,j) = A(:,j);
86                  end
87                  if j > 1
88                      v3(:,j) = (A(:,j) - A(:,j-1)).*((d-j+1)/d)^q;
89                  end
90              end
91              wt(:,i) = sum(v3,2);
92          case 8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% medida
                ponderada
93              %QQ = rand(size(A));
94              QQ = A;
95              SQQ=sum(QQ,2);
96              QQQ=QQ./SQQ;
97              for j=1:d
98                  if j == 1
99                      v3(:,j) = A(:,j);
100                 end
101                 if j > 1
102                     v3(:,j) = (A(:,j) - A(:,j-1)).*sum(QQQ(:,j:d)
                            ,2) + eps;
```

```
                end
            end
            wt(:,i) = sum(v3,2);


    end


end


wt2 = sum(wt,2);

% To avoid the situation that zeros are exist in the matrix wt2
ss = wt2==0;
wt2(ss,:) = eps;
wt = wt./repmat(wt2,1,k);




x_g = [];

for i=1:k
    wt1 = wt(:,i);
    wt2 = repmat(wt1,1,d+1);
    x_g = [x_g,x_e.*wt2];
end
end
```

4. Avaliação de Desempenho com Validação Cruzada

O desempenho do modelo é avaliado utilizando validação cruzada em k-dobras. O código calcula métricas específicas para cada combinação de parâmetros e seleciona os melhores com base nos critérios estabelecidos.

```
cv_index = 1;
TempResult = zeros(num_cv, 15);

for cv = 1:num_cv
    [cv_train_data, cv_train_target, cv_test_data, cv_test_target]
        = ...
        generateCVSet(train_data, target', randorder, cv, num_cv);
    [model_LLSF] = ML_TSKFS(cv_train_data, cv_train_target,
        optmParameter);
```

```
    Outputs = (cv_test_data * model_LLSF)';
    Pre_Labels = double(round(Outputs) >= 1);
    TempResult(cv_index, :) = EvaluationAll(Pre_Labels, Outputs,
        cv_test_target')';
    cv_index = cv_index + 1;
end

Result = mean(TempResult)';
STD = std(TempResult);
```

5. Resultados e Visualização

Por fim, o código apresenta os resultados em gráficos, permitindo uma análise visual das métricas de desempenho e da convergência dos parâmetros.

```
figure;
plot(rx(q, :), ry1(q, :), rx(q, :), ry2(q, :));
ylim([0.0, 1]);
xlabel(Itera  es);
ylabel(M tricas de Desempenho);
grid on;
```

Conclusão

A estrutura do código foi desenvolvida para explorar os benefícios da Integral de Choquet em sistemas fuzzy multi-rótulo. Utilizando loops aninhados e avaliações sistemáticas, o código garante um ajuste fino dos parâmetros, resultando em um modelo robusto e eficiente. As técnicas implementadas permitem modelar interações complexas entre atributos, com base nas medidas fuzzy abordadas neste trabalho.

Apêndice B: Conjunto de Dados Usados para Treinamento

Neste anexo, são apresentados exemplos dos atributos e rótulos dos primeiros quatro conjuntos de dados utilizados para o treinamento e para a análise comparativa entre o modelo proposto, *ML TSKC-FS*, e os modelos da literatura.

Conjunto de Dados BibTeX

O conjunto de dados **BibTeX** é amplamente utilizado em tarefas de aprendizado de máquina multi-rótulo. Neste apêndice, são apresentadas informações detalhadas sobre sua estrutura e formato.

Estrutura do Conjunto de Dados

O conjunto de dados BibTeX é composto por **7.395 instâncias**, **1.836 atributos** (termos extraídos dos documentos) e **159 rótulos** (categorias). A Tabela 5.1 apresenta uma pequena amostra do conjunto de dados.

ID do Documento	Termos (Atributos)	Rótulos (Categorias)
1	learning, neural, network, model	AI, Machine Learning
2	retrieval, information, database	Information Retrieval, Databases
3	classification, fuzzy, logic	Fuzzy Systems, AI
4	optimization, genetic, algorithm	Optimization, Genetic Algorithms
5	data, mining, patterns	Data Mining, Big Data

Tabela 5.1: Exemplo da Estrutura do Conjunto de Dados BibTeX

Formato Real do Conjunto de Dados

Os dados no BibTeX são armazenados no formato esparso, onde cada instância é representada por vetores com os seguintes elementos:

1. **Atributos**: Representados como uma matriz esparsa com valores **0** ou **1**, indicando a ausência ou presença de termos no documento. Por exemplo:

```
[0, 1, 0, ..., 1, 0, 1] // Vetor de tamanho 1836.
```

2. **Rótulos**: Uma matriz binária que indica se o documento pertence a cada uma das **159 categorias**. Exemplo:

```
[1, 0, 0, ..., 1, 0, 0] // Vetor de tamanho 159.
```

Formato Original (Texto Bruto)

Caso o arquivo seja visualizado diretamente, o conteúdo do arquivo pode ser apresentado no seguinte formato:

```
ID: 1
Features: {learning: 1, neural: 1, network: 1, model: 1, ...}
Labels: {AI: 1, Machine Learning: 1, ...}
```

Conjunto de Dados Birds

O conjunto de dados **Birds** é utilizado em tarefas de aprendizado de máquina multi-rótulo, especialmente no reconhecimento e categorização de cantos de aves em gravações de áudio. Este apêndice detalha a estrutura e o formato do conjunto de dados.

Estrutura do Conjunto de Dados

O conjunto de dados Birds contém **645 instâncias**, **260 atributos** e **19 rótulos** (categorias de aves). A Tabela 5.2 apresenta uma amostra dos dados.

ID do Som	Características (Atributos)	Rótulos (Categorias de Aves)
1	freq1, freq2, pitch1, pitch2	Sparrow, Thrush
2	freq2, freq3, pitch3, tempo	Robin, Crow
3	freq1, pitch1, tempo, volume	Finch, Sparrow
4	freq4, pitch2, tempo, volume	Warbler, Thrush
5	freq1, freq3, pitch1, pitch4	Crow, Finch

Tabela 5.2: Exemplo da Estrutura do Conjunto de Dados Birds

Formato Real do Conjunto de Dados

Os dados no conjunto Birds são armazenados no formato de matriz, em que cada linha representa uma instância com suas características e rótulos:

1. **Atributos**: Representam características extraídas dos áudios, como frequências, pitch (tom), tempo e volume. Exemplo de vetor de atributos:

```
[0.12, 0.35, 0.20, ..., 0.50]
```

O vetor tem dimensão **260**.

2. **Rótulos**: Representam as espécies de aves detectadas na gravação. Cada espécie é associada a um valor binário (0 ou 1). Exemplo de vetor de rótulos:

```
[1, 0, 0, ..., 1]
```

O vetor possui dimensão **19**.

Formato Original (Texto Bruto)

Se o arquivo for visualizado diretamente, o conteúdo pode aparecer da seguinte forma:

```
ID: 1
Features: {freq1: 0.12, freq2: 0.35, pitch1: 0.20, pitch2: 0.50, ...}
Labels: {Sparrow: 1, Thrush: 1, Robin: 0, Crow: 0, ...}
```

Conjunto de Dados CAL500

O conjunto de dados **CAL500** é utilizado para tarefas de aprendizado de máquina multi-rótulo, com foco na anotação automática de músicas. Este apêndice apresenta detalhes sobre a estrutura e o formato deste conjunto de dados.

Estrutura do Conjunto de Dados

O conjunto de dados CAL500 é composto por **502 instâncias** (músicas), **68 atributos** extraídos das características musicais e **174 rótulos** (anotações de palavras-chave relacionadas às músicas). A Tabela 5.3 apresenta uma pequena amostra.

ID da Música	Atributos (Características)	Rótulos (Anotações)
1	pitch, tempo, ritmo, harmonia	Happy, Upbeat, Instrumental
2	ritmo, volume, melodia	Sad, Slow, Acoustic
3	tempo, pitch, percussão	Energetic, Loud, Rock
4	harmonia, melodia, ritmo	Calm, Mellow, Jazz
5	volume, pitch, tempo	Instrumental, Classical, Soft

Tabela 5.3: Exemplo da Estrutura do Conjunto de Dados CAL500

Formato Real do Conjunto de Dados

Os dados do CAL500 são organizados da seguinte forma:

1. **Atributos**: Representam características extraídas das músicas, como *pitch* (tom), tempo, ritmo, harmonia, volume e percussão. Exemplo de vetor de atributos:

   ```
   [0.23, 0.45, 0.67, ..., 0.12]
   ```

   O vetor possui **68 dimensões**.

2. **Rótulos**: São palavras-chave anotadas manualmente que descrevem o conteúdo emocional, instrumental e de gênero das músicas. Exemplo de vetor de rótulos:

   ```
   [Happy, Upbeat, Instrumental]
   ```

   Cada música pode conter múltiplos rótulos, totalizando **174 categorias** possíveis.

Formato Original (Texto Bruto)

O conteúdo original do arquivo pode ser representado no seguinte formato:

```
ID: 1
Features: {pitch: 0.23, tempo: 0.45, ritmo: 0.67, harmonia: 0.12, ...}
Labels: {Happy, Upbeat, Instrumental}
```

Conjunto de Dados Corel16k1

O conjunto de dados **Corel16k1** é amplamente utilizado em tarefas de aprendizado de máquina multi-rótulo, com foco em recuperação e classificação de imagens. Este apêndice apresenta informações detalhadas sobre sua estrutura e características.

Estrutura do Conjunto de Dados

O conjunto de dados Corel16k1 contém **13.766 instâncias** (imagens), **500 atributos** extraídos das características visuais e **153 rótulos** (categorias). A Tabela 5.4 apresenta uma pequena amostra deste conjunto.

ID da Imagem	Atributos (Características Visuais)	Rótulos (Categorias)
1	cor, textura, borda	Montanha, Paisagem
2	cor, formato, textura	Animal, Savana
3	borda, cor, intensidade	Urbano, Arquitetura
4	textura, cor, brilho	Praia, Oceano
5	cor, formato, borda	Flores, Jardim

Tabela 5.4: Exemplo da Estrutura do Conjunto de Dados Corel16k1

Formato Real do Conjunto de Dados

Os dados no conjunto Corel16k1 são armazenados no formato matricial, onde cada instância representa as características visuais de uma imagem.

1. **Atributos**: Representam características extraídas das imagens, como cor, textura, borda, intensidade e brilho. Exemplo de vetor de atributos:

   ```
   [0.12, 0.35, 0.67, ..., 0.89]
   ```

   O vetor possui dimensão **500**.

2. **Rótulos**: Cada imagem pode estar associada a múltiplos rótulos que representam as categorias visuais da imagem. Exemplo de vetor de rótulos:

   ```
   [Montanha, Paisagem]
   ```

   No total, existem **153 rótulos** diferentes.

Formato Original (Texto Bruto)

O formato original dos dados pode ser apresentado da seguinte forma:

```
ID: 1
Features: {cor: 0.12, textura: 0.35, borda: 0.67, ...}
Labels: {Montanha, Paisagem}
```

Em conclusão, este apêndice apresenta uma descrição mais detalhada dos quatro primeiros conjuntos de dados, de acordo com a tabela apresentada no trabalho. Essa descrição complementa as informações fornecidas anteriormente, oferecendo maior clareza sobre as características dos dados utilizados no treinamento e na análise comparativa do modelo ML TSKC-FS em relação aos modelos da literatura.