UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TRATAMENTO DE IMPRECISÃO NA GERAÇÃO DE ÁRVORES DE DECISÃO

MARIANA VIEIRA RIBEIRO LOPES

ORIENTADORA: PROFª. DRª. HELOISA DE ARRUDA CAMARGO

São Carlos - SP
Março/2016

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TRATAMENTO DE IMPRECISÃO NA GERAÇÃO DE ÁRVORES DE DECISÃO

MARIANA VIEIRA RIBEIRO LOPES

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Inteligência Artificial

Orientadora: Drª. Heloisa de Arruda Camargo

São Carlos - SP
Março/2016

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de Dissertação de Mestrado da candidata Mariana Vieira Ribeiro Lopes, realizada em 03/03/2016.

Profa. Dra. Heloisa de Arruda Camargo
(UFSCar)

Profa. Dra. Maria do Carmo Nicoletti
(UFSCar)

****
Prof. Dr. Marcos Evandro Cintra
(UFERSA)

Certifico que a sessão de defesa foi realizada com a participação à distância do membro Marcos Evandro Cintra, depois das arguições e deliberações realizadas, o participante à distância está de acordo com o conteúdo do parecer da comissão examinadora redigido no relatório de defesa da aluna Mariana Vieira Ribeiro Lopes.

Profa. Dra Heloisa de Arruda Camargo
Presidente da Comissão Examinadora
(UFSCar)

Dedico este mestrado aos meus pais, Rosana e Jair, por serem os principais responsáveis por esta conquista, me apoiando, incentivando e encorajando a encarar todos os desafios. Sem o seu apoio, amor e dedicação, nada disso seria possível.

Parabéns, esta conquista é de vocês!

AGRADECIMENTO

A Deus, por me permitir realizar este trabalho.

À minha orientadora, Profª. Drª. Heloisa de Arruda Camargo, pela paciência e tempo dedicados ao ensinamento e orientação que tanto contribuíram para o meu desenvolvimento enquanto pesquisadora.

Aos membros da banca examinadora pelo tempo dedicado e contribuição.

Ao meu esposo Renato pelo apoio, incentivo, dedicação e paciência que foram essenciais para que esta etapa fosse concluída.

Aos meus pais, Rosana e Jair, por, desde sempre, me apoiarem e incentivarem em todos os projetos e decisões que me fizeram chegar até aqui.

Aos meus irmãos, Amanda e Rafael, pelo apoio, carinho e torcida.

À amiga Lianet pela ajuda e pelas dicas preciosas dadas em momentos oportunos.

Aos familiares e amigos, pelo apoio e incentivo.

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelo apoio financeiro.

"Por vezes sentimos que aquilo que fazemos não é senão uma gota de água no mar.

Mas o mar seria menor se lhe faltasse uma gota."

Madre Teresa de Calcutá

RESUMO

As Árvores de Decisão Indutivas (AD) são um mecanismo baseado no paradigma simbólico do Aprendizado de Máquina que tem como principais características a fácil interpretabilidade e baixo custo computacional. Ainda que sejam amplamente utilizadas, as ADs são limitadas à representação de problemas cujas variáveis são do tipo discreto ou contínuo. No entanto, para alguns tipos de problemas, pode haver variáveis que não são bem representadas por estes formatos. Diante deste contexto, foram criadas as Árvores de Decisão Fuzzy (ADF), que adicionam à interpretabilidade das Árvores de Decisão Indutivas, a capacidade de lidar com variáveis fuzzy, as quais representam adequadamente conhecimentos imprecisos. Neste texto, apresentamos o trabalho desenvolvido durante o mestrado, que tem como principal resultado um novo algoritmo para indução de Árvores de Decisão Fuzzy, cujo método de fuzificação dos atributos contínuos é realizado durante a indução da árvore e foi inspirado no método de particionamento de atributos contínuos adotado pelo C4.5. Para validação do algoritmo, foram realizados testes com 20 conjuntos de dados do repositório UCI (LICHMAN, 2013) *e o algoritmo foi comparado com outros três algoritmos que abordam o problema de classificação por meio de técnicas diferentes: o C4.5 que induz uma Árvore de Decisão Indutiva, o FURIA, que induz um Sistema Fuzzy Baseado em Regras, porém não segue a estrutura de árvore e o FuzzyDT que induz uma Árvore de Decisão fuzzy realizando a fuzificação dos atributos contínuos antes da indução da árvore. Os resultados dos experimentos realizados são apresentados e discutidos no Capítulo 4 deste texto.*

Palavras-chave: *Árvore de Decisão Indutiva, Árvore de Decisão Fuzzy, Problemas de Classificação, Sistemas Fuzzy, Sistemas Fuzzy Baseados em Regras, Sistemas Fuzzy de Classificação.*

ABSTRACT

Inductive Decision Trees (DT) are mechanisms based on the symbolic paradigm of machine learning which main characteristics are easy interpretability and low computational cost. Though they are widely used, the DTs can represent problems with just discrete or continuous variables. However, for some problems, the variables are not well represented in this way. In order to improve DTs, the Fuzzy Decision Trees (FDT) were developed, adding the ability to deal with fuzzy variables to the Inductive Decision Trees, making them capable to deal with imprecise knowledge. In this text, it is presented a new algorithm for fuzzy decision trees induction. Its fuzification method is applied during the induction and it is inspired by the C4.5's partitioning method for continuous attributes. The proposed algorithm was tested with 20 datasets from UCI repository (LICHMAN, 2013)*. It was compared with other three algorithms that implement different solutions to classification problem: C4.5, which induces an Inductive Decision Tree, FURIA, that induces a Rule-based Fuzzy System and FuzzyDT, which induces a Fuzzy Decision Tree where the fuzification is done before tree's induction is performed. The results are presented in Chapter 4.*

Keywords: *Inductive Decision Tree, Fuzzy Decision Tree, Classification Problems, Fuzzy Systems, Rule-based Fuzzy Systems, Fuzzy Classification Systems.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AD *- Árvore de Decisão Indutiva*

ADF *- Árvore de Decisão Fuzzy*

AG *- Algoritmo Genético*

AGMO *- Algoritmo Genético Multi-objetivo*

AM *- Aprendizado de Máquina*

BC *- Base de Conhecimento*

CV *- Cross-Validation*

HSM *- Heterogeneous Split Measure*

FHSM *- Fuzzy Heterogeneous Split Measure*

STIFI *– Simple Test Inspired Fuzzy Induction*

FID3 *- Fuzzy ID3*

G-FDT *- Gini Index Fuzzy Decision Tree*

GG-FSDT *- Gini Index-Gaussian Fuzzy Decision Tree*

GM3M *- Geometric Mean of Three Metrics*

IA *- Inteligência Artificial*

IFDTC *- Interpretability-based Fuzzy Decision Tree Classifier*

MI *- Mecanismo de Inferência*

RNA *- Rede Neural Artificial*

SCFBR *- Sistema de Classificação Fuzzy Baseado em Regras*

SDT *- Soft Decision Tree*

SF *- Sistema Fuzzy*

SFBR *- Sistema Fuzzy Baseado em Regras*

TCC *- Taxa de Classificação Correta*

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Contexto

Este trabalho está contextualizado em Aprendizado de Máquina (AM), uma subárea da Inteligência Artificial (IA). Uma rápida pesquisa em um dicionário nos traz algumas definições para o termo aprender ("*to learn*"), entre elas (COLLINS AMERICAN ENGLISH DICTIONARY, 2016):

i. Obter conhecimento (a respeito de algum assunto) ou habilidade (em alguma arte, negócio, etc) através de estudo, experiência, instrução, etc;
ii. Vir a conhecer;
iii. Gravar na memória, memorizar;
iv. Receber uma instrução, ficar informado (sobre algo).

De acordo com Witten, Frank e Hall (WITTEN; FRANK; HALL, 2011), quando estamos falando de máquinas, estas definições apresentam algumas deficiências. Segundo os autores, com relação às duas primeiras, é virtualmente impossível testar quando o aprendizado foi alcançado ou não. Na melhor das hipóteses, seria possível verificar a capacidade de uma máquina responder a determinadas perguntas, mas dificilmente saberíamos dizer quando ela, de fato, adquiriu conhecimento ou tornou-se consciente de algo. Já as duas últimas definições parecem ser muito aquém da capacidade de nossas máquinas. Tarefas como *"gravar na memória"* ou *"receber uma instrução"* são triviais para os computadores atuais.

Sendo assim, Witten, Frank e Hall (WITTEN; FRANK; HALL, 2011) definem o aprendizado de forma operacional, afirmando que:

"As máquinas aprendem quando mudam o seu comportamento de forma que as fazem ter um melhor desempenho no futuro."

Isso significa que, para os autores, a forma mais objetiva para testar o aprendizado de uma máquina é através da observação de seu comportamento, comparando-o com o seu comportamento passado.

Os estudos desenvolvidos no contexto do Aprendizado de Máquina têm como principal objetivo a criação de sistemas que são capazes de "aprender", adquirindo e generalizando conhecimento, automaticamente, a partir de conjuntos de dados que são denominados conjuntos de treinamento. Dentro do Aprendizado de Máquina, duas abordagens clássicas são adotadas: O Aprendizado Supervisionado e o Não Supervisionado. No Aprendizado Não Supervisionado o conjunto de treinamento é formado por dados não rotulados, ou seja, o conjunto não apresenta o atributo denominado classe ou rótulo, que classifica ou rotula uma instância. Já no Aprendizado Supervisionado estes conjuntos, a partir dos quais os modelos são induzidos, contém dados que foram previamente rotulados. Os modelos preditivos construídos a partir de algoritmos de Aprendizado Supervisionado são capazes de prever os rótulos de novos dados, anteriormente desconhecidos. Rótulos podem ser de natureza contínua ou discreta. Quando valores discretos definem os rótulos dos dados, caracteriza-se um problema de classificação. Já quando são definidos por valores contínuos, o problema é de regressão.

Um dos métodos de indução mais simples e bem-sucedidos conhecidos na literatura é o da indução de árvores de decisão (AD) (QUINLAN, 1986), que é comumente utilizada para tratar problemas de classificação.

Os diversos algoritmos propostos utilizam diferentes formas de induzir ADs para classificação (BREIMAN et al., 1984; FREUND; MASON, 1999; QUINLAN, 1986, 1992), sendo que um dos mais conhecidos para este fim é o C4.5, proposto por Quinlan em 1992 .

Depois de induzidas, as Árvores de Decisão são usadas como classificadores; recebem uma instância de dado, descrita por um conjunto de atributos e retornam a classe à qual esta instância pertence, de acordo com os valores dos atributos.

Para determinar a classe à qual uma instância pertence, a AD executa uma sequência de testes que são representados pelos nós internos da árvore. Cada teste verifica o valor de um dos atributos da instância (objeto ou exemplo) de entrada, percorrendo os ramos da árvore que correspondem ao valor do atributo testado, até chegar a um nó folha, que está associado a uma possível classe que será atribuída à instância em questão. O processo de construção das árvores de decisão indutivas será abordado na Seção 2.4 deste trabalho.

Embora os modelos induzidos pelos algoritmos deste tipo de método representem bem uma grande parte dos problemas, com eles, não é possível representar conhecimento impreciso. Para isso, dentre outras abordagens, algoritmos que induzem Árvores de Decisão *Fuzzy* (ADF) foram desenvolvidos, agregando a fácil interpretação e representação das ADs à capacidade de representar o conhecimento impreciso dos conjuntos *fuzzy*.

As Árvores de Decisão *Fuzzy* aplicadas a problemas de classificação são o tema da pesquisa desenvolvida neste trabalho e serão abordadas com mais detalhes no decorrer do texto.

1.2 Motivação e Objetivos

1.2.1 Motivação

Desde a proposta inicial das ADFs, diversos algoritmos foram disponibilizados na literatura (CHANDRA; VARGHESE, 2009; CINTRA; CAMARGO, 2010; JANIKOW; KAWA, 2005; JANIKOW, 1998; OLARU; WEHENKEL, 2003; PEDRYCZ; SOSNOWSKI, 2005; YUAN; KLIR; SWAN-

STONE, 1995). Muitos destes algoritmos, são extensões do clássico C4.5 (QUINLAN, 1992) ou até mesmo de outras ADs, como por exemplo a SDT (OLARU; WEHENKEL, 2003), que tem como base o algoritmo CART (BREIMAN et al., 1984).

Usualmente, as Árvores de Decisão *Fuzzy* são induzidas pelos mesmos algoritmos desenvolvidos para a indução de ADs. Para isso, fuzifica-se os atributos contínuos do conjunto de treinamento e, então, o algoritmo de indução é aplicado sobre este conjunto fuzificado, resultando em uma ADF. Outras abordagens realizam a fuzificação dos atributos durante a indução da ADF, por exemplo a G-FDT (CHANDRA; VARGHESE, 2009) e a SDT (OLARU; WEHENKEL, 2003).

Apesar de muitos desses algoritmos terem obtido bons resultados, ainda restam alguns problemas a serem tratados. Entre eles, pode-se citar i) a geração dos conjuntos *fuzzy* no processo de fuzificação dos atributos contínuos e ii) o elevado número de regras geradas pelos modelos induzidos.

É fato que o número de regras geradas pelos algoritmos de ADF está diretamente relacionado com o número de conjuntos *fuzzy* definido na fuzificação dos atributos contínuos, já que para cada possível valor do atributo selecionado pelo algoritmo em um determinado nó, um novo ramo será criado na árvore. Por isso, é possível que um método de geração de conjuntos *fuzzy* mais eficiente, com menor número possível de conjuntos que represente adequadamente as categorias de valores do domínio do atributo contínuo, resulte em uma árvore com menor número de regras.

Embora muitas formas de fuzificação já tenham sido propostas na literatura, não há um método consistente que defina o número de conjuntos *fuzzy*, bem como o formato e parâmetros destes conjuntos de forma que a solução apresente bom resultado para todos (ou pelo menos grande parte dos) problemas.

1.2.2 Objetivos

Com o intuito de explorar formas alternativas de gerar Árvores de Decisão Fuzzy, que ofereçam recursos mais adequados para o tratamento da imprecisão durante o processo de indução, o objetivo deste trabalho é:

O desenvolvimento de uma nova abordagem para a geração de ADFs, realizando a fuzificação dos atributos contínuos durante a indução da árvore.

Para isso, alguns objetivos específicos foram definidos:

- Identificar os métodos de fuzificação dos atributos contínuos disponíveis na literatura;
- Definir um método de fuzificação dos atributos contínuos para ser utilizado;
- Integrar o método de fuzificação dos atributos contínuos com o algoritmo de indução da ADF;
- Identificar os tipos de tratamento para diminuição do número de regras (poda) das ADFs disponíveis na literatura;
- Integrar um tratamento para diminuição do número de regras ao algoritmo de indução da ADF.

1.3 Metodologia de Desenvolvimento do Trabalho

A fim de atingir os objetivos citados na Seção 1.2, o desenvolvimento do trabalho seguiu os seguintes passos:

- Realização de Revisão Bibliográfica a fim de identificar diferentes abordagens para:
    - Indução de ADFs;
    - Fuzificação de atributos contínuos e;
    - Redução do número de regras induzidas propostos na literatura;
- Definição do algoritmo para indução da ADF;
- Definição do método de fuzificação dos atributos contínuos para ser embutido no algoritmo de indução da ADF;
- Definição de estratégia para a redução do número de regras induzidas pelo algoritmo de ADF;
- Implementação do algoritmo;
- Realização de experimentos e comparações com outras abordagens.

1.4 Organização do Trabalho

Este trabalho está organizado da seguinte forma: no Capítulo 2 são apresentados alguns conceitos preliminares relacionados ao tema de pesquisa do trabalho tais como Aprendizado de Máquina, Sistemas de Classificação, Árvores de Decisão Indutivas e Sistemas *Fuzzy*. O Capítulo 3 aborda o assunto Árvores de Decisão *Fuzzy*, inclusive apresentando alguns trabalhos da literatura desenvolvidos sobre o tema. No Capítulo 4 apresenta-se o algoritmo proposto neste trabalho, bem como os experimentos e resultados realizados com este e outros três algoritmos, a título de comparação. O Capítulo 5 finaliza o texto apresentando conclusões, contribuições e trabalhos futuros.

Capítulo 2

CONCEITOS PRELIMINARES

2.1 Considerações Iniciais

No decorrer deste trabalho muitos conceitos que embasaram o desenvolvimento da pesquisa são utilizados para. Por isso, neste capítulo serão apresentados alguns dos conceitos que foram considerados essenciais para o entendimento do texto.

A Seção 2.2 apresenta os conceitos mais gerais relacionados ao Aprendizado de Máquina. A Seção 2.3 aborda o tema Sistemas de Classificação. Na Seção 2.4 um tipo específico sistema de classificação é contemplado, as Árvores de Decisão Indutivas. O Algoritmo C4.5, um dos mais conhecidos para indução de ADs, é apresentado na Seção 2.5. A Seção 2.6 introduz os conceitos relacionados aos Sistemas *Fuzzy* e a Seção 2.7 apresenta as considerações finais deste capítulo.

2.2 Aprendizado de Máquina

O Aprendizado de Máquina, que é uma subárea da Inteligência Artificial, tem como um de seus objetivos o desenvolvimento de algoritmos capazes de extrair conhecimento, automaticamente, a partir de conjuntos de exemplos,

representando-o por meio de um modelo. No decorrer dos anos, muitos algoritmos de aprendizado foram desenvolvidos, com diferentes abordagens para arepresentação e aquisição de conhecimento.

Neste contexto, os exemplos, também chamados de instâncias, objetos ou dados, são descritos por um vetor de atributos, dentre os quais pode haver um atributo especial, denominado rótulo ou classe, que tem a finalidade de identificar a instância. Quando este atributo ocorre, diz-se que os dados são rotulados.

No Aprendizado Indutivo de Máquina, existem diversos paradigmas, dentre os quais, pode-se citar: Simbólico, Estatístico, Conexionista e Evolutivo (REZENDE, 2003). Na abordagem simbólica, um conceito é representado simbolicamente, no formato de expressões lógicas, Árvores de Decisão Indutivas ou regras. Como o próprio nome sugere, o paradigma estatístico utiliza modelos estatísticos para induzir um conceito. O paradigma conexionista, por sua vez, utiliza construções matemáticas inspiradas no sistema nervoso humano, conhecidas como Redes Neurais Artificiais (RNA). Já o paradigma evolutivo, tem como base a teoria de Darwin, que tem como um de seus princípios básicos a suposição de que somente os indivíduos mais aptos sobrevivem com o decorrer das gerações.

Tradicionalmente, o Aprendizado Indutivo de Máquina divide-se em Supervisionado e Não Supervisionado. No contexto de aprendizado Supervisionado, um conjunto de exemplos rotulados é apresentado ao algoritmo de aprendizado, que generaliza o conhecimento contido no conjunto, induzindo modelos preditivos. Estes modelos são capazes de determinar o valor dos rótulos de exemplos cujos rótulos são desconhecidos. Quando estes rótulos são discretos, o problema em questão é caracterizado como de classificação e quando são contínuos, trata-se de um problema de regressão.

No aprendizado Não Supervisionado, é realizada uma análise de um conjunto não rotulado a fim de identificar similaridades entre os dados apresentados (RUSSELL; NORVIG, 1995).

Como o foco deste trabalho é um algoritmo supervisionado que trata problemas de classificação, mais ênfase será dada a estes conceitos.

Uma forma de aquisição de conhecimento amplamente utilizada no desenvolvimento dos algoritmos de aprendizado supervisionado é a indução que, segundo Russel e Norvig, tem como objetivo definir um modelo preditivo (função ou, simplesmente, modelo) capaz de retornar o valor aproximado do rótulo de um exemplo, dados os valores de seus atributos (RUSSELL; NORVIG, 1995). Para a indução de modelos, os algoritmos de aprendizado utilizam um conjunto de exemplos que lhes é apresentado como entrada. Este conjunto é chamado conjunto de treinamento e contém dados que foram previamente rotulados.

O modelo induzido poderá ser utilizado posteriormente para, por meio do conhecimento adquirido, derivar novo conhecimento. A este processo dá-se o nome inferência.

Para medir a qualidade de um modelo preditivo, costuma-se verificar a sua capacidade de predizer corretamente rótulos de exemplos anteriormente conhecidos. Se o conjunto de treinamento usado na indução do modelo, for também utilizado para testá-lo, é possível que a Taxa de Classificação Correta (TCC) observada não reflita a real capacidade de predição do modelo, fazendo com que o mesmo apresente uma TCC pior que a esperada quando classificando exemplos desconhecidos. Por isso, a seguinte metodologia, descrita por Russell e Norvig (RUSSELL; NORVIG, 1995), costuma ser adotada:

1. Coleta-se um conjunto com número suficientemente grande de exemplos;
2. Divide-se este conjunto em dois conjuntos disjuntos: conjunto de treinamento e conjunto de teste;
3. Utiliza-se o conjunto de treinamento para induzir o modelo, através de um algoritmo de indução;
4. Mede-se a porcentagem de exemplos do conjunto de teste que o modelo é capaz de classificar corretamente;
5. Repetem-se os passos de 1 a 4 para diferentes tamanhos de conjuntos de teste e treinamento, aletoriamente selecionados.

Conforme foi citado anteriormente, este trabalho está inserido no contexto do aprendizado supervisionado aplicado a problemas de classificação. A Seção 2.3 trata dos sistemas de classificação, que é a solução desenvolvida para o tratamento deste tipo de problema.

2.3 Sistemas de Classificação

Segundo Quinlan (QUINLAN, 1992), classificar instâncias consiste, basicamente, em associá-las a categorias ou classes que são determinadas pelas propriedades da instância. Em um sistema de classificação, o elo entre as classes e as propriedades das instâncias pode ser definido por algo simples, como uma tabela, ou por um método bastante complexo e desestruturado, como um procedimento manual realizado por um especialista humano. Quando restringimos estas soluções a modelos executáveis, ou seja, modelos que podem ser computacionalmente representados, alguns tipos distintos de soluções são viáveis, entre elas:

i. Construir modelos com base no conhecimento de especialistas;
ii. Examinar um número grande de exemplos rotulados e utilizar a indução para construir um modelo, generalizando o conhecimento a partir da observação de exemplos.

Mesmo com todas as dificuldades conhecidas, entre elas, a subjetividade de definir a tomada de decisão com base no conhecimento de um ou mais especialistas, o primeiro tipo de solução foi adotado na construção de muitos sistemas baseados em conhecimento.

Com o intuito de aprimorar a construção dos modelos induzidos a partir da segunda solução, estudos foram realizados e algoritmos foram desenvolvidos para a indução automática de sistemas capazes de lidar com problemas de classificação.

Muitos destes sistemas utilizam regras de classificação para relacionar instâncias às classes, por meio dos seus valores de atributos. Nas Árvores de Decisão Indutivas, por exemplo, cada ramo do modelo é representado por uma regra, que é utilizada na classificação de novas instâncias. Geralmente, estas regras seguem o formato $\mathbf{SE} - \mathbf{ENT\tilde{A}O}$, ilustrado a seguir.

$$\mathbf{SE}\ antecedente\ \mathbf{ENT\tilde{A}O}\ consequente$$

Em que o *antecedente* da regra é, geralmente, composto por uma restrição ou uma conjunção de restrições que segue o seguinte formato:

$$(A_1\ \theta\ x_1)\ \boldsymbol{E}\ (A_2\ \theta\ x_2)\ \boldsymbol{E}\ \ldots\ (A_n\ \theta\ x_n)$$

Em que:

- A_i representa um atributo, que pode ser de natureza contínua ou discreta. Atributos contínuos representam valores numéricos, que podem ser medidos por números inteiros ou reais. Já os atributos discretos, nominais ou categóricos, representam valores de um conjunto de possibilidades finito e previamente definido (WITTEN; FRANK; HALL, 2011);
- x_i representa um valor dentro do domínio do atributo A_i;
- e θ pode ser um dos operadores relacionais: $>, <, \geq, \leq, =$ ou $\in$.

Já o *consequente*, representa a classe associada. Sendo assim, em geral, as regras seguem o seguinte formato:

$$\boldsymbol{SE}\ (A_1\ \theta\ x_1)\ \boldsymbol{E}\ (A_2\ \theta\ x_2)\ \boldsymbol{E}\ \ldots\ (A_n\ \theta\ x_n)\ \boldsymbol{ENT\tilde{A}O}\ Classe\ =\ C_i$$

Em que C_i representa uma das classes do conjunto de M classes $C = \{C_1, C_2, \ldots, C_M\}$.

Depois que as regras foram construídas via indução ou especialistas, um mecanismo de inferência é utilizado para classificar novas instâncias, ainda com classes desconhecidas.

Um método de aprendizado simbólico muito comumente utilizado para tratar problemas de classificação é a Árvore de Decisão, que será abordada na Seção 2.4.

2.4 Árvores de Decisão

Dentre tantos métodos de aprendizado de máquina, a Árvore de Decisãoconquistou muita popularidade, sendo amplamente adotada na literatura. Além de intuitividade, fácil interpretação e passível de dois tipos de representação (gráfica ou via regras) de ADs, algumas outras características atribuídas às ADs são (COPPIN, 2010; MITCHELL, 1997; QUINLAN, 1992):

- Competitividade com outros modelos computacionais mais complexos, quando comparada a acurácia do modelo induzido;
- Incorporação de um método de seleção de atributos ao algoritmo, o que agrega interpretabilidade ao modelo;
- Método considerado robusto e escalável;
- Capacidade de lidar com atributos contínuos e discretos;
- Processos de inferência e indução com baixo custo computacional.

Além disso, as Árvores de Decisão possuem as seguintes propriedades (COPPIN, 2010):

- Cada nó, com exceção do nó raiz, tem exatamente um *nó predecessor*, ou *pai*;
- Toda AD contém um *nó raiz*, que não tem predecessor e, normalmente, representa o seu início;

- As ADs também contêm alguns nós que não possuem nós sucessores. São os chamados *nós folha*;
- Com exceção dos nós folha, todos os outros nós contém um ou mais sucessores.

A Figura 1(a) mostra um exemplo bastante simplificado de AD.

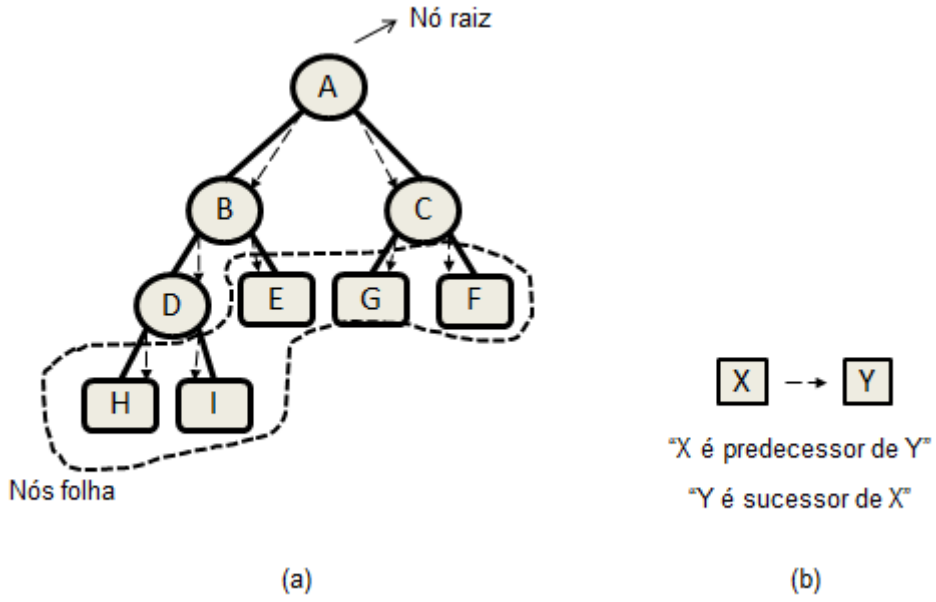

Figura 1: Um exemplo de Árvore de Decisão.

Conforme é indicado na árvore da Figura 1(a) o nó *A* é o nó raiz e os nós *H*, *I*, *E*, *G* e *F* são nós folha. A seta tracejada indica a relação *"é predecessor de"*, conforme é exibido na Figura 1(b). Exemplos dessa relação são *"o nó B é predecessor dos nós D e E"* ou *"o nó A é predecessor dos nós B e C"*. Já a seta contínua indica a relação *"é sucessor de"*, por exemplo, *"o nó I é sucessor do nó D"* ou *"o nó C é sucessor do nó A"*.

As ADs caracterizam-se por tratar problemas de aprendizado segundo uma abordagem de divisão e conquista, ou seja, um problema complexo é dividido em problemas mais simples, aos quais se aplica a mesma estratégia. Assim, as soluções encontradas pela resolução dos problemas menores

podem ser combinadas a fim de definir uma solução para o problema que se tinha inicialmente. A grande vantagem desta abordagem é a possibilidade de dividir o espaço de instâncias em subespaços que são utilizados para o ajuste de diferentes submodelos. Esta é a ideia que originou os principais algoritmos de AD conhecidos: ID3 (QUINLAN, 1986), CART (BREIMAN et al., 1984), C4.5 (QUINLAN, 1992).

Os *nós internos* das ADs podem, também, ser chamados *nós de divisão* ou *nós de teste* e representam *testes condicionais* que envolvem os valores de um atributo específico e irão compor o *antecedente* da regra, definindo as condições para que o *consequente* da mesma seja alcançado. Embora, normalmente, o teste de um nó compare o valor de um atributo com uma constante, há casos, menos comuns, nos quais dois atributos diferentes são comparados ou uma função que combina dois ou mais atributos é utilizada (FACELI, 2011; WITTEN; FRANK; HALL, 2011).

Supondo os atributos $Altura$, $Olhos$ e $Peso$, alguns exemplos de testes condicionais são:

- $Altura > 1{,}70$;
- $Olhos \in \{castanhos, verdes, azuis\}$;
- $\left(\frac{Peso}{Altura}\right)^2 < 25$.

Os nós folha são, usualmente, considerados como sendo o valor do rótulo dos exemplos que chegam até ele através do processo de indução. Neste caso, quando o rótulo é discreto, atribui-se à folha, a moda destes valores, ou seja, o valor do rótulo mais frequente entre os exemplos.

A Figura 2(b) exibe uma AD e a divisão correspondente no espaço definido pelos atributos contínuos A e B (Figura 2(a)). Cada região deste espaço é representada por um ou mais nós folha na AD e é definida de acordo com os testes que são realizados nos nós da árvore. Por exemplo, na árvore exibida na Figura 2(b) o primeiro teste $A < a_2$ define a região representada pela classe C_4 no espaço definido pelos atributos A e B. A união das regiões definidas pelas folhas cobre todo o espaço definido pelos atributos e são

mutuamente exclusivas, ou seja, a intersecção das regiões abrangidas pelas folhas, duas a duas, é vazia. Note que a árvore exibida na Figura 2(c) representa o mesmo espaço definido pelos atributos A e B.

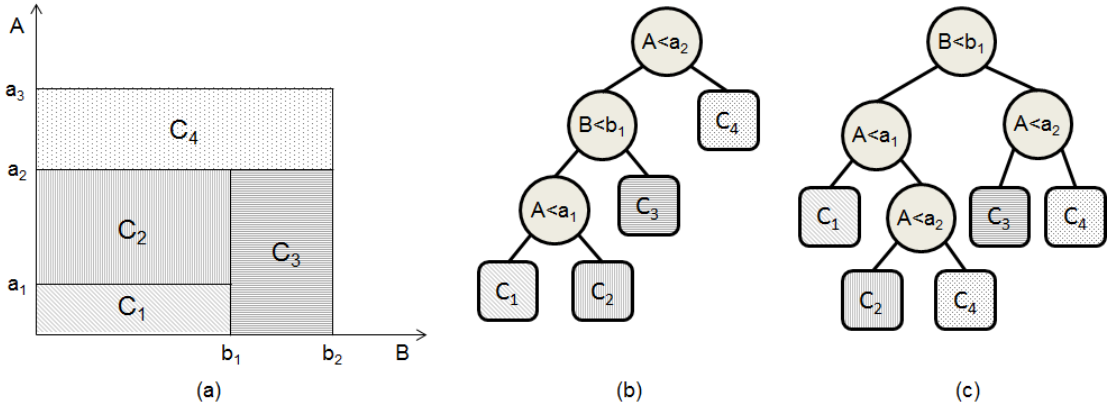

Figura 2: Espaço definido pelos atributos contínuos A e B e duas árvores (b) e (c) que representam este espaço.

Uma vez que a AD abrange todo o espaço de exemplos, ela pode ser utilizada para fazer predições para qualquer exemplo de entrada. Cada ramo da AD é identificado por um caminho que leva do nó raiz ao nó folha e é representado por uma regra. Cada regra segue o formato $\mathbf{SE-ENTÃO}$, abordado na Seção 2.3 e, portanto, tem o *antecedente* formado por uma conjunção das condições presentes nos nós internos do ramo e o *consequente* pela classe representada no nó folha. Por exemplo, as regras que representam a árvore da Figura 2(b) são:

1. $\mathbf{SE}\ A < a_2\ \mathbf{E}\ B < b_1\ \mathbf{E}\ A < a_1\ \mathbf{ENTÃO}\ classe = C_1$
2. $\mathbf{SE}\ A < a_2\ \mathbf{E}\ B < b_1\ \mathbf{E}\ A \geq a_1\ \mathbf{ENTÃO}\ classe = C_2$
3. $\mathbf{SE}\ A < a_2\ \mathbf{E}\ B \geq b_1\ \mathbf{ENTÃO}\ classe = C_3$
4. $\mathbf{SE}\ A \geq a_2\ \mathbf{ENTÃO}\ classe = C_4$

Usualmente, a indução de uma AD é realizada por um algoritmo relativamente simples, que tem como entrada um conjunto de dados $\boldsymbol{D} = (\boldsymbol{x_i}, y_i),\ i = 1, \ldots, n$, em que $\boldsymbol{x_i}$ é o vetor de atributos e y_i é o rótulo, e como saída a AD induzida. O Algoritmo 1 apresentado por (WITTEN; FRANK; HALL,

2011) descreve os passos que são seguidos para a indução de uma AD. Os passos consistem de, primeiramente, avaliar o critério de parada (linha 2), que pode ser, por exemplo, se a altura da árvore chegou a um determinado limite ou se a divisão do nó implicará em subconjuntos muito pequenos de acordo com um limite pré-definido. Se o critério de parada não foi atingido, é escolhido o atributo A_K que maximiza alguma medida de pureza (linha 5), por exemplo, o ganho de informação. As medidas de pureza medem a homogeneidade dos subconjuntos gerados caso a divisão ocorra. Conjuntos mais homogêneos apresentam maior grau de pureza. Uma vez que o atributo A_K foi escolhido, os subconjuntos $\boldsymbol{D_i}$, $i = 1, \ldots, m$ são gerados, onde m é o número de possíveis valores que A_K pode assumir, se for um atributo discreto, ou o número de divisões realizadas no domínio de A_K, caso o mesmo seja contínuo. Finalmente, o algoritmo é aplicado, recursivamente, a cada subconjunto $\boldsymbol{D_i}$ (linha 7).

Algoritmo 1: Algoritmo para construção de uma AD

Entrada: Um conjunto de treinamento $\boldsymbol{D} = (\boldsymbol{x_i}, y_i)$
Saída: Árvore de Decisão

1. **/* Função GeraArvore(*D*)*/**
2. **se** *critério de parada(**D**) = Verdadeiro* **então**
3. **retorne** *Um nó folha rotulado com a classe majoritária de **D**;*
4. **fim-se**
5. *Escolha o atributo que maximiza o critério de divisão em **D***;
6. **para cada** *partição $\boldsymbol{D_i}$ dos exemplos, definida com base nos valores do atributo*
7. *escolhido* **faça**
8. *Induz uma subárvore Árvore$_i$ = GeraArvore($\boldsymbol{D_i}$);*
9. **fim-para**
10. **retorna** *Árvore contendo um nó de decisão baseado no atributo escolhido e descendentes da Árvore$_i$;*

Os métodos de indução de modelos que utilizam exemplos para adquirir conhecimento, podem apresentar um problema bastante frequente denominado *overfitting*, ou superadaptação. Quando este problema ocorre, a AD induzida tende a classificar corretamente todos (ou a maior parte) os dados de treinamento, porém apresenta desempenho ruim na classificação de exemplos

desconhecidos. Normalmente isso acontece porque o modelo induzido está muito ajustado aos dados de treinamento, que podem conter ruídos ou até mesmo não representar bem o espaço completo de exemplos possíveis.

Uma solução frequentemente aplicada aos modelos para solucionar este tipo de problema é a poda. Sabe-se que a indução de uma AD mínima em termos de número de nós é um problema *NP completo*, isso significa que ainda não há algoritmo conhecido para resolver este problema em tempo menor que exponencial com relação ao tamanho do conjunto de treinamento. No entanto, na tentativa de alcançar uma AD com menor número de nós, sem afetar a acurácia do modelo e diminuindo o problema de *overfitting*, alguns métodos de poda foram desenvolvidos. Estes métodos dividem-se, basicamente, em duas vertentes:

- Métodos de pré-poda;
- Métodos de pós poda.

Os métodos de pré-poda consistem em interromper o crescimento da AD durante a indução da mesma, interferindo no momento em que deveria ocorrer a divisão de um nó. Portanto, grande parte das vezes, este tipo de método está diretamente relacionado ao critério de parada citado no Algoritmo 1.

Já os métodos de pós-poda deixam que a árvore completa seja induzida para, posteriormente, diminuir o seu tamanho através de um procedimento que transforma determinadas subárvores da AD induzida em nós folha.

Para a avaliação dos sistemas de classificação, inclusive das ADs, não basta que o algoritmo seja executado uma única vez, já que a indução via exemplos, pode gerar um modelo tendencioso, com *overfitting*. Por isso, foram criados alguns métodos de validação para que os resultados obtidos sejam mais confiáveis. Um método de validação bastante conhecido e que será mencionado diversas vezes neste trabalho á a Validação Cruzada ou *Cross-Validation* (CV). Neste método, o conjunto de dados é particionado em *n partições* e a técnica pode ser chamada de *n-fold CV.* O processo consiste em executar o algoritmo de indução do modelo *n* vezes, considerando, em cada execução, *n-1 folds* como conjunto de treinamento e o *fold* remanescente como

conjunto de teste. Ao final do procedimento, o erro é calculado fazendo a média dos erros obtidos em cada execução.

Um dos algoritmos de indução de AD mais populares é o C4.5 que foi desenvolvido por Quinlan em 1992 (QUINLAN, 1992) e será tratado na Seção 2.5.

2.5 Algoritmo C4.5

O algoritmo C4.5, proposto por Quinlan (QUINLAN, 1992), é um dos mais populares para a indução de ADs. Assim como todos os algoritmos de indução de Árvores de Decisão, implementa a abordagem de divisão e conquista. Ele utiliza um conjunto de treinamento $\boldsymbol{D} = (\boldsymbol{x_i}, y_i)$, que tem o mesmo formato descrito na Seção 2.4 e induz a árvore seguindo os passos do Algoritmo 1.

O critério de divisão, citado no passo da linha 5 do Algoritmo 1, para decidir sobre a qualidade dos atributos é a razão de ganho. Eles são selecionados recursivamente, construindo a estrutura da AD. O atributo com maior razão de ganho é selecionado para ser o nó raiz e define o teste a ser realizado nesse nó, bem como os subconjuntos gerados a partir da divisão após o teste.

Os testes padrão considerados pelo C4.5 são:

- $A = ?$, para um atributo discreto A, com uma saída para cada valor de A.
- $A \leq t$, para um atributo contínuo A, com duas saídas, *verdadeiro* e *falso*.

Para encontrar o *threshold* t que maximiza o critério de divisão dos atributos contínuos, o algoritmo C4.5 utiliza um método denominado *Teste Simples* ou *Pesquisa Exaustiva*, que será explicado na Seção 2.5.1.

O algoritmo é aplicado recursivamente, considerando cada um dos subconjuntos de instâncias resultantes da divisão feita pelo teste do nó raiz, descrito no passo da linha 6 do Algoritmo 1.

As principais características do algoritmo C4.5 são:

- Utilização de medidas de entropia e razão de ganho como critério para a seleção de atributos;
- Tratamento de atributos discretos e contínuos;
- Aplicação de um método de pós poda para o tratamento de *overfitting*.

A razão de ganho de um dado atributo informa quanta informação pode ser adquirida ao particionar um conjunto de exemplos segundo os valores deste atributo. Quanto mais homogêneos forem os subconjuntos resultantes, melhor. Este cálculo é definido pela normalização do ganho de informação, que é calculado pela redução em entropia (QUINLAN, 1992). A entropia de um conjunto $\boldsymbol{D}$ que contém M possíveis classes é definida por Shannon (SHANNON, 1948) como na Equação 1.

$$E(\boldsymbol{D}) = -\sum_{j=1}^{M} \frac{freq(C_j, \boldsymbol{D})}{|\boldsymbol{D}|} \cdot log_2\left(\frac{freq(C_j, \boldsymbol{D})}{|\boldsymbol{D}|}\right) \quad (1)$$

Em que, $freq(C_j, \boldsymbol{D})$ representa o número de exemplos de $\boldsymbol{D}$ que pertencem à classe C_j e $|\boldsymbol{D}|$ é o número de exemplos em $\boldsymbol{D}$. A entropia indica a quantidade média de informação necessária para classificar um exemplo em $\boldsymbol{D}$.

Quando $\boldsymbol{D}$ é particionado em subconjuntos $\boldsymbol{D_i}, i = 1,..,m$, pelo nó cujo atributo de teste X tem m possíveis valores, a entropia deste atributo é calculada pela média ponderada destes subconjuntos, como é exibido na Equação 2 (SHANNON, 1948).

$$E(X) = \sum_{i=1}^{m} \frac{|\boldsymbol{D_i}|}{|\boldsymbol{D}|} \cdot E(\boldsymbol{D_i}) \quad (2)$$

Esta medida indica a quantidade de informação que é requerida para classificar um exemplo após o particionamento dos dados do conjunto anterior, realizado por meio dos valores do atributo X.

Com os cálculos de entropia realizados, o ganho de informação do atributo X, que representa o ganho de informação quando o conjunto $\boldsymbol{D}$ é particionado por ele, é calculado da forma como é mostrado na Equação 3.

$$ganho(X) = E(\boldsymbol{D}) - E(X) \quad (3)$$

A razão de ganho do atributo X é definida como uma normalização do ganho de informação, e é definida conforme a Equação 4.

$$razaodeganho(X) = \frac{ganho(X)}{divisao(X)} \quad (4)$$

Onde $divisao(X)$ é representada na Equação 5.

$$divisao(X) = -\sum_{i=1}^{m} \frac{|\boldsymbol{D_i}|}{|\boldsymbol{D}|} \cdot log_2\left(\frac{|\boldsymbol{D_i}|}{|\boldsymbol{D}|}\right) \quad (5)$$

A razão de ganho define o atributo que será atribuído a cada nó durante o processo de indução da árvore.

A estratégia de particionamento recursivo adotada pelo C4.5 deve resultar em árvores consistentes com os dados de treinamento. Em aplicações práticas, no entanto, os dados podem apresentar ruídos quando, por exemplo, valores de atributos foram registrados incorretamente, levando à classificação incorreta destes exemplos. Este ruído leva à indução de árvores excessivamente complexas que tentam explicar essas anomalias. Conforme já foi explicado anteriormente, este problema é denominado *overfitting* e pode ser tratado através de métodos de poda.

O algoritmo C4.5 implementa um método de pós-poda com o intuito de generalizar o modelo final. Ele identifica subárvores que contribuem pouco para

a precisão da previsão, substituindo-as por nós folha. Mais detalhes deste método podem ser encontrados em (QUINLAN, 1992).

Um módulo do algoritmo C4.5 especialmente relevante para este trabalho é o tratamento atribuído aos atributos contínuos. O método apresentado, anteriormente, nesta seção faz sentido quando se considera atributos discretos, no entanto, quando trata-se de atributos contínuos, algumas questões podem surgir, por exemplo, qual é o valor de m para este tipo de atributo e como os conjuntos $\boldsymbol{D_i}$ são calculados? A Seção 2.5.1 apresenta o método implementado pelo algoritmo C4.5 para o tratamento dos atributos contínuos.

2.5.1 Tratamento de Atributos Contínuos

Conforme explicado na Seção 2.5, o teste padrão considerado pelo algoritmo C4.5, dado um atributo contínuo A é:

- $A \leq t$, com duas saídas, *verdadeiro* e *falso*.

Para determinar o *threshold* t, que maximiza a razão de ganho para o atributo A, o algoritmo C4.5 utiliza um método denominado *Teste Simples* ou *Pesquisa Exaustiva*. Neste método, a divisão que ocorre no domínio do atributo contínuo é sempre binária, ou seja, somente um ponto de divisão é definido para cada nó que tem como teste um atributo contínuo. O passo a passo deste método é descrito por Quinlan (QUINLAN, 1992) da seguinte maneira:

Considerando o conjunto de treinamento $\boldsymbol{D}$, apresentado como entrada para o algoritmo C4.5, os exemplos de $\boldsymbol{D}$ são ordenados de acordo com os valores do atributo A. Considere que estes valores ordenados são representados por $v = v_1, v_2, \dots, v_n$. Cada par de valores adjacentes sugere um *threshold* $t_i = \frac{(v_1 + v_2)}{2}, i = 1, \dots, n-1$ e um subconjunto correspondente de $\boldsymbol{D}$.

O *threshold* que maximiza o critério de divisão, neste caso a razão de ganho, é selecionado. Para ilustrar este procedimento, observe o exemplo abaixo:

Suponha $\boldsymbol{D}$, o conjunto de treinamento (extraído de (LICHMAN, 2013)), representado pela Tabela 1 e $A = Temperatura$, o atributo contínuo a ser particionado.

Tabela 1: Exemplo de conjunto de treinamento.

Aparência	Temperatura	Umidade	Vento	Classe
ensolarado	75	70	verdadeiro	jogar
ensolarado	80	90	verdadeiro	não jogar
ensolarado	85	85	falso	não jogar
ensolarado	72	95	falso	não jogar
ensolarado	69	70	falso	jogar
nublado	72	90	verdadeiro	jogar
nublado	83	78	falso	jogar
nublado	64	65	verdadeiro	jogar
nublado	81	75	falso	jogar
chuvoso	71	80	verdadeiro	não jogar
chuvoso	65	70	verdadeiro	não jogar
chuvoso	75	80	falso	jogar
chuvoso	68	80	falso	jogar
chuvoso	70	96	falso	jogar

Primeiramente, ordenam-se os exemplos do conjunto exibido na Tabela 1 segundo os valores do atributo a ser particionado, no caso, $Temperatura$. O resultado após a ordenação pode ser visto na Tabela 2.

Tabela 2: Exemplos da Tabela 1 ordenados de acordo com os valores do atributo Temperatura.

Aparência	Temperatura	Umidade	Vento	Classe
nublado	64	65	verdadeiro	jogar
chuvoso	65	70	verdadeiro	não jogar
chuvoso	68	80	falso	jogar
ensolarado	69	70	falso	jogar
chuvoso	70	96	falso	jogar
chuvoso	71	80	verdadeiro	não jogar
ensolarado	72	95	falso	não jogar
nublado	72	90	verdadeiro	jogar
ensolarado	75	70	verdadeiro	jogar
chuvoso	75	80	falso	jogar
ensolarado	80	90	verdadeiro	não jogar
nublado	81	75	falso	jogar
nublado	83	78	falso	jogar
ensolarado	85	85	falso	não jogar

Considera-se cada possível ponto de divisão do atributo $Temperatura$ $P = \{64, 65, 68, 69, 70, 71, 72, 75, 80, 81, 83, 85\}$. Para cada possível ponto de divisão calcula-se a razão de ganho, a fim de avaliar qual deles maximiza esta medida para o atributo em questão. Uma forma simples de entender como a razão de ganho é calculada para cada ponto é considerar como se os valores do atributo a ser avaliado fossem discretos. Por exemplo, para calcular a razão de ganho para o ponto de divisão 70, o conjunto seria considerado conforme é exibido na Tabela 3.

Tabela 3: Exemplo de como os valores do atributo Temperatura são divididos para o cálculo da razão de ganho.

Aparência	Temperatura	Umidade	Vento	Classe
nublado	≤ 70	65	verdadeiro	jogar
chuvoso	≤ 70	70	verdadeiro	não jogar
chuvoso	≤ 70	80	falso	jogar
ensolarado	≤ 70	70	falso	jogar
chuvoso	≤ 70	96	falso	jogar
chuvoso	> 70	80	verdadeiro	não jogar
ensolarado	> 70	95	falso	não jogar
nublado	> 70	90	verdadeiro	jogar
ensolarado	> 70	70	verdadeiro	jogar
chuvoso	> 70	80	falso	jogar
ensolarado	> 70	90	verdadeiro	não jogar
nublado	> 70	75	falso	jogar
nublado	> 70	78	falso	jogar
ensolarado	> 70	85	falso	não jogar

Desta forma, a razão de ganho é calculada da forma padrão, como se o atributo $Temperatura$ fosse discreto e seus subconjuntos fossem definidos pelos valores ≤ 70 e > 70.

Uma vez que este cálculo foi feito para todos os pontos de divisão em P, o ponto de divisão v_K que maximiza a razão de ganho para aquele atributo é selecionado. É comum que o *threshold* seja definido conforme mostra a Equação 6. O C4.5, no entanto, seleciona o maior valor do atributo (considerando todo o conjunto de treinamento) que não ultrapasse o valor de t, ao invés de selecionar o próprio t.

$$t = \frac{(v_k + v_{k+1})}{2} \quad (6)$$

2.6 Sistemas *Fuzzy*

Sistemas *Fuzzy* (SF) são sistemas que têm, pelo menos, uma de suas variáveis representadas por meio da teoria de conjuntos *fuzzy,* definida por Lofti A. Zadeh em 1965 (ZADEH, 1965).

Embora a teoria de conjuntos tradicional seja muito eficaz na representação e processamento de dados convencionais, o mesmo não se pode dizer quando se trata da representação e processamento de informações imprecisas, incertas ou com ruído. Como se sabe, grande parte dos dados do mundo real não é precisa. A teoria de conjuntos *fuzzy* se faz muito útil quando precisamos lidar com dados incertos, uma vez que ela permite a representação deste tipo de informação.

2.6.1 Conjuntos *Fuzzy*

De forma geral, um conjunto é uma coleção de elementos. Na teoria clássica de conjuntos, um elemento pertence ou não a um determinado conjunto. Neste contexto, nos referimos a conjuntos da teoria clássica como conjuntos convencionais ou clássicos. Muitas vezes o termo em inglês - conjunto *crisp* - também é empregado. Isto significa que, dado um conjunto *crisp* A, sobre o conjunto universo X, a função F_A definida pela Equação 7 denominada função característica, o representa bem, atribuindo valor 1 aos elementos do universo X que pertencem a A e 0 aos elementos que não pertencem.

$$F_A(x) = \begin{cases} 1, se\ x\ \in A \\ 0, se\ x\ \notin A \end{cases} \quad (7)$$

Ainda que os conjuntos clássicos representem adequadamente diversas situações reais, muitos problemas tem como elementos objetos que, não necessariamente, pertencem completamente a um conjunto, mas sim parcialmente. Este fato evidencia a necessidade da representação de uma transição gradual entre conjuntos, fazendo com que um elemento possa pertencer a ambos, com diferentes graus de pertinência. Nestes casos, a representação por meio de conjuntos clássicos torna-se inadequada. Por isso, uma generalização deste tipo de conjunto, denomina conjuntos *fuzzy*, é adotada para fazer este tipo de representação.

Os conjuntos *fuzzy* possibilitam a pertinência parcial de um elemento a um conjunto, permitindo uma transição gradual de significado. Muito úteis na representação de conceitos imprecisos, eles são representados por funções características que associam, a cada elemento, um grau de pertinência do mesmo ao conjunto.

Estas funções são denominadas funções de pertinência e o grau que retornam para a pertinência dos elementos nos conjuntos está, normalmente, no intervalo [0,1] (PEDRYCZ; GOMIDE, 1998).

O grau de pertinência de um elemento $x \in X$ a um conjunto *fuzzy* A, representa o grau de compatibilidade do elemento com o conceito representado pelo conjunto e é, normalmente, denotado pelos dois formatos a seguir (NICOLETTI; CAMARGO, 2004).

(1) $\mu_A: X \rightarrow [0,1]$;
(2) $A: X \rightarrow [0,1]$.

O conjunto X é o conjunto universo em que A está definido, também chamado de conjunto base. Note que, na notação (1), o símbolo que denota o conjunto *fuzzy*, isto é, A, é diferente do símbolo que identifica sua função de pertinência (μ_A), enquanto em (2), a função de pertinência é representada pelo mesmo símbolo que denota o conjunto (A).

Os conjuntos *fuzzy*, definidos por funções de pertinência, podem ser representados de diversas formas que serão abordadas na Seção 2.6.1.1.

2.6.1.1 Funções de Pertinência

As funções de pertinência definem conjuntos *fuzzy* que podem ser representados de diferentes formas. Algumas das representações, que serão descritas neste texto, são a representação gráfica, a tabular, via lista, e a analítica.

A representação gráfica define, graficamente, a função de pertinência sobre um domínio e é bastante conveniente quando este é definido sobre o espaço euclidiano unidimensional ou bidimensional. Considere a Figura 3, que exibe uma função de pertinência que representa o conceito temperatura baixa.

No exemplo, a temperatura varia no intervalo $T = [0{,}40]$ segundo o conjunto *fuzzy* TB. Conforme vemos na figura, os valores de 0 a 15 possuem grau de pertinência igual a 1 no conjunto, portanto, são considerados temperatura baixa. Já os valores de 30 a 40, possuem grau de pertinência zero ao conceito e não são, portanto, considerados temperatura baixa. Os valores intermediários de temperatura possuem graus definidos pela função de pertinência representada na Figura 3.

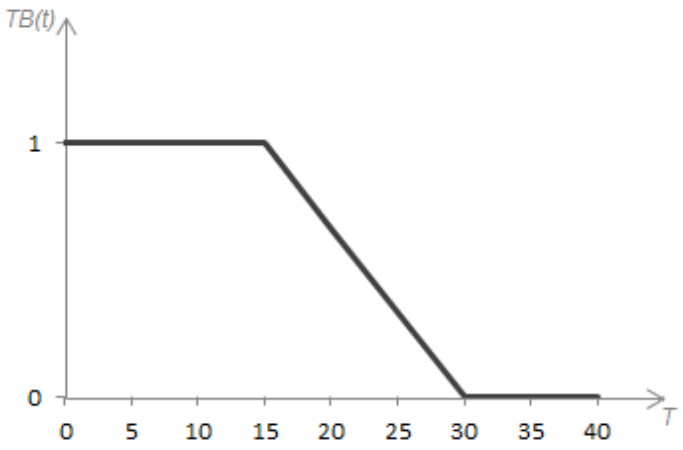

Figura 3: Conjunto *fuzzy* que representa o conceito temperatura baixa.

Quando os conjuntos universo são finitos, é possível representar as funções de pertinência por meio de tabelas ou listas. Na representação tabular, os elementos do conjunto universo e seus respectivos graus de pertinência são representados em uma tabela. Neste caso, é possível que somente os elementos cujos graus de pertinência são maiores que zero sejam representados, ficando subentendido que os elementos do conjunto universo

que não foram evidenciados na tabela possuem grau de pertinência zero no conjunto em questão. A Tabela 4 exibe a representação tabular da função de pertinência que define o conceito temperatura baixa. Neste caso, para que a função pudesse ser representada pela tabela, o universo infinito $T = [0{,}40]$ foi discretizado e reduzido ao conjunto finito $\boldsymbol{TD} = \{0, 5, 10, 15, 20, 25, 30, 35\}$.

Tabela 4: Representação tabular do conceito temperatura baixa ($\boldsymbol{TB}$) para o conjunto universo discretizado ($\boldsymbol{TD}$).

$\boldsymbol{x} \in \boldsymbol{TD}$	$\boldsymbol{TB}(\boldsymbol{x})$
0	1,00
5	1,00
10	1,00
15	1,00
20	0,67
25	0,33
30	0,00
35	0,00

Na notação de lista, utiliza-se a barra (/) para ligar os pares *grau-de-pertinência/elemento*. Cada par é unido pelo sinal de +, que indica que os pares definem o conjunto *fuzzy* conjuntamente. A representação tabular pode ser reescrita, segundo a notação de lista, da seguinte forma:

$$\boldsymbol{TB} = 1{,}00/0 + 1{,}00/5 + 1{,}00/10 + 1{,}00/15 + 0{,}67/20 + 0{,}33/25 + 0{,}00/30 + 0{,}00/35$$

Em se tratando de conjuntos *fuzzy* definidos sobre um domínio infinito, a notação mais utilizada é a analítica, ou seja, a representação é feita por meio de funções que podem ser agrupadas em famílias. Estas famílias são conhecidas por famílias de funções parametrizáveis e as mais conhecidas são:

- Funções Triangulares;
- Funções Trapezoidais;
- Funções tipo Sino;

- Funções S.

As funções de pertinência triangulares são, geralmente, definidas por três parâmetros: a, m e b, sendo $a \leq m \leq b$ e a definição da função dada pela Equação 8. A Figura 4 ilustra a função de pertinência triangular.

$$A(x) = \begin{cases} 0, & se\ x \leq a, \\ \frac{x-a}{m-a}, & se\ x \in (a, m), \\ 1, & se\ x = m, \\ \frac{b-x}{b-m}, & se\ x \in (m, b), \\ 0, & x \geq b \end{cases} \quad (8)$$

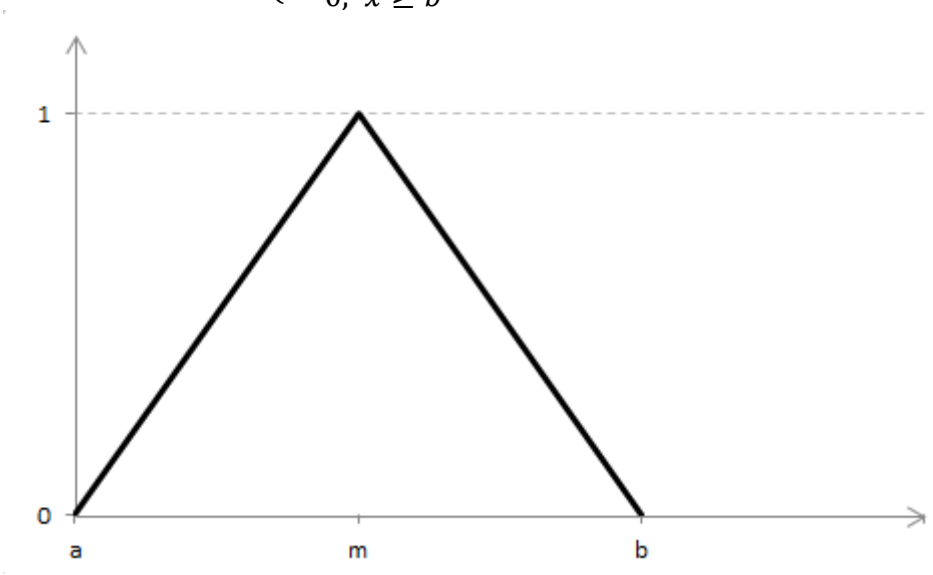

Figura 4: Função de pertinência triangular.

As funções de pertinência trapezoidais são definidas por quatro parâmetros: a, m, n e b, sendo $a \leq m \leq n \leq b$ e a definição da função dada pela Equação 9. A Figura 5 ilustra a função de pertinência trapezoidal.

$$A(x) = \begin{cases} 0, & se\ x \leq a, \\ \frac{x-a}{m-a}, & se\ x \in (a, m), \\ 1, & se\ x = [m, n], \\ \frac{b-x}{b-n}, & se\ x \in (n, b), \\ 0, & x \geq b \end{cases} \quad (9)$$

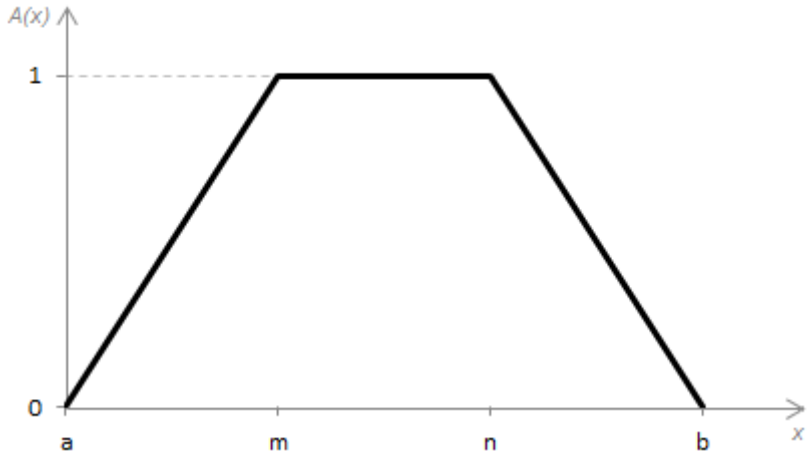

Figura 5: Função de pertinência trapezoidal.

As funções de pertinência do tipo S são definidas por três parâmetros: a, m e b, sendo que $m = \frac{(a+b)}{2}$. A definição da função de pertinência do tipo S é dada pela Equação 10. A Figura 6 ilustra a função de pertinência do tipo S.

$$A(x) = \begin{cases} 0, & se\ x \leq a, \\ 2 \cdot \left(\frac{x-a}{b-a}\right)^2, & se\ x \in (a, m], \\ 1 - 2 \cdot \left(\frac{x-b}{b-a}\right)^2, & se\ x = (m, b), \\ 1, & x \geq b \end{cases} \tag{10}$$

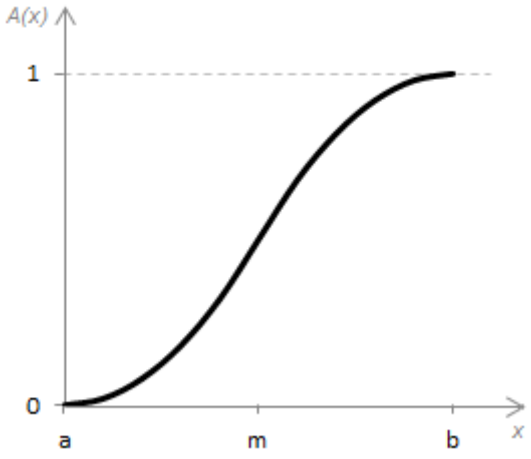

Figura 6: : Função de pertinência tipo S.

A função de pertinência do tipo sino é representada pela fórmula da Equação 11 e ilustrada pela Figura 7.

$$A(x) = e^{-b(x-a)^2} \quad (11)$$

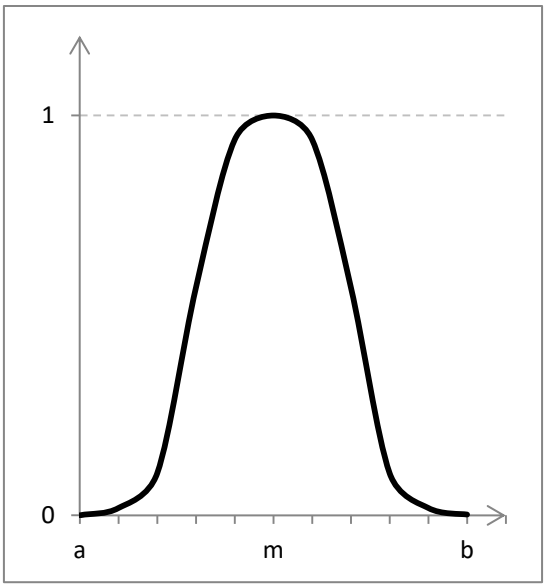

Figura 7: Função de pertinência tipo sino.

Uma vez que os conjuntos *fuzzy* foram devidamente definidos por suas funções de pertinência, algumas operações podem ser aplicadas sobre eles. Estas operações são abordadas na Seção 2.6.1.2.

2.6.1.2 Operações Sobre Conjuntos *Fuzzy*

Analisar operações que envolvam conjuntos *fuzzy* é essencial para a representação e processamento do conhecimento. As operações de união, intersecção e complemento da teoria de conjuntos clássica podem ser generalizadas para conjuntos *fuzzy*, porém para este tipo de conjunto, cada uma delas pode ser modelada por um conjunto de operadores (KLIR; YUAN, 1995). Estas operações podem ser divididas em operações padrão e operações generalizadas.

Sejam A e B dois conjuntos *fuzzy*, definidos por suas funções de pertinência $\mathrm{A(x)}$ e $\mathrm{B(x)}$, respectivamente. A operação de União Padrão entre A e B é definida pela Equação 12.

$$A \cup B = max[A(x), B(x)] = A(x) \vee B(x) \quad (12)$$

(O símbolo ∨ representa o operador máximo)

A Intersecção Padrão entre A e B é definida pela Equação 13.

$$A \cap B = min[A(x), B(x)] = A(x) \wedge B(x) \quad (13)$$

(O símbolo ∧ representa o operador mínimo)

O *Complemento Padrão* do conjunto *fuzzy* A é definido pela Equação 14.

$$A^C = 1 - A(x) \quad (14)$$

e indica o grau com que x não pertence a A.

As operações generalizadas são obtidas substituindo-se os operadores de máximo e mínimo das operações padrão por operadores das classes conhecidas por *t-normas*, que definem a operação de intersecção *fuzzy*, e *t-conormas* (ou *s-normas*), que definem a operação de união *fuzzy* (KLIR; YUAN, 1995).

As *t-normas* são operações binárias do tipo

$$i: [0,1]^2 \rightarrow [0,1]$$

Toda *t-norma* i satisfaz às seguintes propriedades (KLIR; YUAN, 1995):

1. Comutatividade: $i(x, y) = i(y, x)$;
2. Associatividade: $i(x, i(y, z)) = i(i(x, y), z)$;
3. Monotonicidade: se $x \leq y$ e $z \leq w$, então $i(x, z) \leq i(z, w)$;
4. Condição Limite: $i(0, x) = 0$ e $i(1, x) = x$.

Alguns exemplos de *t-normas* são as seguintes:

- Intersecção Padrão: $i(x, y) = min(x, y)$;
- Produto Algébrico: $i(x, y) = xy$;
- Diferença Limitada: $\mathrm{i(x, y)} = \max(0, \mathrm{x} + \mathrm{y} - 1)$;

- Intersecção Drástica: $\mathrm{i}_{\min}(\mathrm{x},\mathrm{y}) = \begin{cases} \mathrm{x}, \text{ quando } \mathrm{y} = 1 \\ \mathrm{y}, \text{quando } \mathrm{x} = 1 \\ 0, \text{caso contrário} \end{cases}$

Sendo assim, a intersecção generalizada de dois conjuntos *fuzzy A* e *B* é definida pela Equação 15.

$$(\mathrm{A} \cap \mathrm{B})(\mathrm{x}) = \mathrm{A}(\mathrm{x})\ \mathrm{t}\ \mathrm{B}(\mathrm{x}) \tag{15}$$

onde t denota uma *t-norma*.

As *s-normas*, por sua vez, são operações binárias do tipo (KLIR; YUAN, 1995):

$$u\colon [0,1]^2 \to [0,1]$$

e devem satisfazer os seguintes axiomas:

1. Comutatividade: $u(x,y) = u(y,x)$;
2. Associatividade: $u(x,u(y,z)) = u(u(x,y),z)$;
3. Monotonicidade: se $x \leq y$ e $z \leq w$, então $u(x,z) \leq u(z,w)$;
4. Condição Limite: $u(0,x) = x$ e $u(1,x) = 1$.

Alguns exemplos de *s-normas* utilizadas como união *fuzzy* são:

União Padrão: $u(x,y) = max(x,y)$;

- Soma algébrica: $u(x,y) = x + y - xy$;
- Soma Limitada: $\mathrm{u}(\mathrm{x},\mathrm{y}) = \min(1, \mathrm{x} + \mathrm{y})$;
- União drástica: $\mathrm{u}_{\max}(\mathrm{x},\mathrm{y}) = \begin{cases} \mathrm{x}, \text{ quando } \mathrm{y} = 0 \\ \mathrm{y}, \text{quando } \mathrm{x} = 0 \\ 1, \text{caso contrário} \end{cases}$

A operação de união generalizada é, então, definida pela Equação 16.

$$(\mathrm{A} \cup \mathrm{B})(\mathrm{x}) = \mathrm{A}(\mathrm{x})\ \mathrm{s}\ \mathrm{B}(\mathrm{x}) \tag{16}$$

onde *s* denota uma *s-norma*.

O complemento generalizado é obtido por meio de operadores definidos por:

$$\mathrm{c}\colon [0,1] \to [0,1]$$

e que devem satisfazer às seguintes propriedades:

1. Monotonicidade: $\forall x, y \in [0,1]$, se $x \leq y$, então $c(x) \leq c(y)$;
2. Condição Limite: $c(0) = 1$ e $c(1) = 0$;
3. Involução: $c(c(x)) = x, \forall x \in [0,1]$.

2.6.2 Variáveis Linguísticas

Variáveis Linguísticas são fundamentais para a representação do conhecimento e são o elemento básico na definição de regras *fuzzy*, que serão apresentadas na Seção 2.6.3.1.

Algumas vezes a representação do conhecimento seguindo o formalismo imposto pela lógica clássica, pode não ser conveniente. A representação do conhecimento impreciso ou ambíguo, por exemplo, pode ser mais facilmente realizada através da forma linguística, que possibilita a elaboração de sentenças imprecisas mais gerais. No entanto, este tipo de formalismo dificulta que o conhecimento descrito seja processado por computadores. Para isso, é importante que o conhecimento impreciso seja representado por variáveis bem definidas.

As variáveis linguísticas tem capacidade de formalizar a representação do conhecimento impreciso de forma que seja possível seu processamento por computadores. Elas utilizam palavras ou sentenças da linguagem natural como valores possíveis, diferentemente das variáveis contínuas e discretas que foram descritas anteriormente. Sua definição se dá por meio da granularização de um conjunto base, que representa uma variável contínua, em termos linguísticos que são definidos por conjuntos *fuzzy*.

Uma variável linguística pode ser caracterizada por uma quíntupla (v, T, X, q, m) onde v é o nome da variável, T é o conjunto de termos linguísticos de v, X é o conjunto universo da variável base sobre a qual cada termo $t \in T$ foi definido, q é a regra sintática (uma gramática) que define a geração dos termos

linguísticos e m é a regra semântica que atribui um significado a cada termo $t \in T$ (KLIR; YUAN, 1995).

A Figura 8 mostra um exemplo de variável linguística que foi definida sobre a variável base “temperatura”. Sendo assim, seu nome foi definido como sendo “Temperatura”, ela apresenta três termos linguísticos: *Baixa*, *Média* e *Alta.* Cada um dos termos linguísticos é associado a um conjunto *fuzzy* representado, graficamente, por suas funções de pertinência que, neste caso, têm formato trapezoidal e são definidas no intervalo [0, 40], o domínio da variável base.

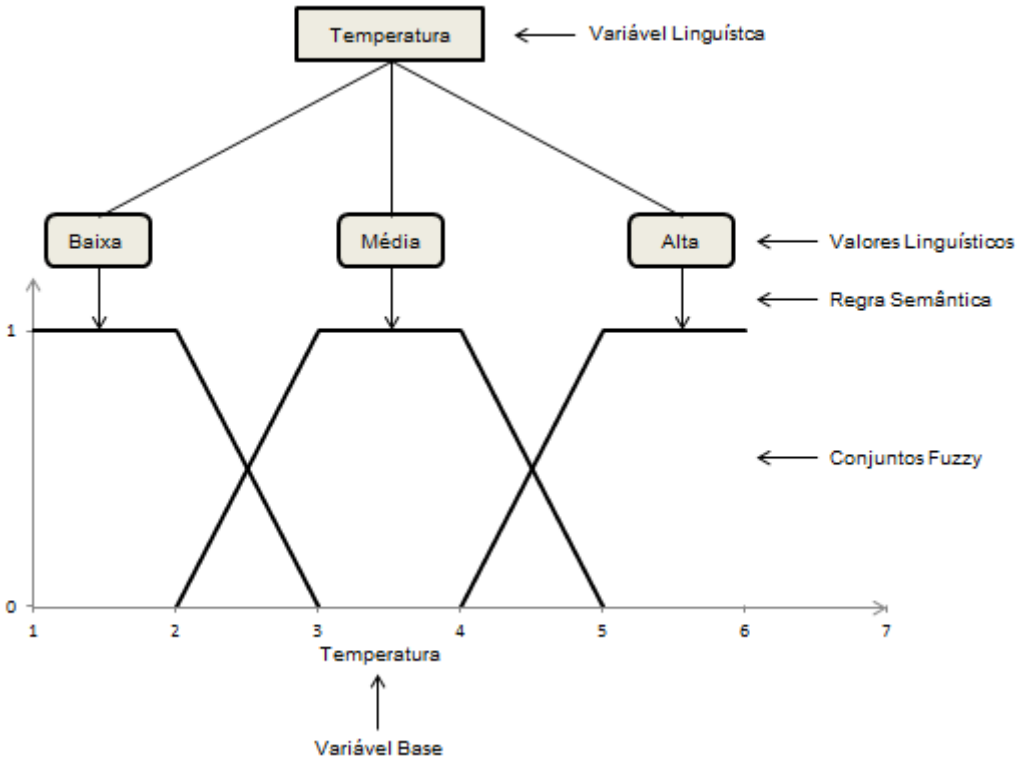

Figura 8: Exemplo de variável linguística definida sobre a variável base “Temperatura”

2.6.3 Sistemas *Fuzzy* Baseados em Regras

Segundo Klir e Yuan (KLIR; YUAN, 1995), quando um mecanismo derivado da lógica *fuzzy* é incorporado a um sistema qualquer e pelo menos uma de suas variáveis é representada por meio de valores linguísticos, então

este já passa a ser considerado um Sistema *Fuzzy*. No trabalho realizado, um tipo especial de sistema *fuzzy*, denominado Sistema *Fuzzy* Baseado em Regras (SFBR) ou Sistema de Regras *Fuzzy*, tem grande relevância e será abordado nesta seção.

Um Sistema de Regras *Fuzzy* possui dois principais componentes: a Base de Conhecimento (BC) e o Mecanismo de Inferência (MI).

A BC é formada por uma Base de Dados *Fuzzy* e uma Base de Regras *Fuzzy*. A Base de Dados *Fuzzy* define os conjuntos *fuzzy* relacionados aos termos linguísticos utilizados nas regras, que são armazenadas na Base de Regras *Fuzzy*.

O MI, por sua vez, realiza o processamento das regras através da aplicação de um método de inferência que infere conclusões a partir das regras e fatos armazenados.

2.6.3.1 Regras *Fuzzy*

Conhecidas por serem uma das técnicas mais antigas para a representação de conhecimento, as regras são capazes de expressar conhecimento com alto grau de interpretabilidade.

Para expressar o conhecimento impreciso com a mesma clareza, utilizam-se as regras *fuzzy* (KLIR; YUAN, 1995), que são, de forma geral, representadas da mesma forma que as regras de classificação, apresentadas na Seção 2.3.

$$\boldsymbol{SE}\ antecedente\ \boldsymbol{ENTÃO}\ consequente$$

Esta forma de representação também expressa uma relação condicional entre o antecedente e o consequente da regra. Isso significa que a condição verdadeira do antecedente implica o consequente.

Neste caso, diferentemente das regras de classificação da Seção 2.3, o *antecedente* é formado por uma proposição *fuzzy* que pode ser atômica ou composta. Uma proposição *fuzzy* atômica tem o formato:

$$X \text{ é } A$$

e especifica um valor linguístico A para uma variável linguística X. A proposição *fuzzy* composta, por sua vez, é formada por conjunções ou disjunções de proposições *fuzzy* atômicas, resultando, portanto, em proposições com um dos seguintes formatos:

1. X_1 é A $\boldsymbol{E}$ X_2 é B
2. X_3 é C $\boldsymbol{OU}$ X_4 é D

Se considerarmos o contexto de estudantes, estas regras poderiam representar, por exemplo, o seguinte conhecimento:

1. João estudou *muito* $\boldsymbol{E}$ João obteve nota *alta*.
2. Maria estudou *pouco* $\boldsymbol{OU}$ Maria obteve nota *alta*.

Neste caso, as variáveis linguísticas são “quantidade de estudo” e “nota obtida” e os termos *pouco*, *muito* e *alta* representam termos linguísticos definidos sobre o domínio destas variáveis.

O *consequente* da regra *fuzzy* também é representado por uma proposição (atômica ou composta). Entretanto, o formato de regras mais utilizado apresenta conjunções no *antecedente* e uma única proposição no *consequente*:

$$\boldsymbol{SE}\ X_1 \text{é } A_{1l_1}\ \boldsymbol{E}\ \ldots\ X_n \text{é } A_{nl_n}\ \boldsymbol{ENTÃO}\ Y \text{ é } B$$

Em que $X_i, i = 1, \ldots, n$ e Y são variáveis linguísticas e $\mathrm{A}_{il_\mathrm{i}}, i = 1, \ldots, n$ e B são termos linguísticos definidos por conjuntos *fuzzy* sobre os domínios das variáveis base de X_i e Y, respectivamente.

O segundo elemento que compõe o SFBR é o Mecanismo de Inferência (MI). Ele consiste de um método de raciocínio e é responsável pelo processamento das regras. Através da aplicação de um procedimento de inferência, derivam-se conclusões a partir das regras.

Um tipo de SFBR que é de interesse deste trabalho é o Sistema *Fuzzy* de Classificação. Este é o tema abordado na seção seguinte.

2.6.3.2 Sistemas *Fuzzy* de Classificação

Conforme foi apresentado na Seção 2.3, dada uma instância descrita por seus atributos, o objetivo da classificação é atribuir uma classe de um conjunto de classes conhecidas a esse exemplo. Para realizar esta tarefa, muitos métodos têm sido utilizados, dentre eles, os sistemas *fuzzy*.

Se um SFBR tiver suas regras modeladas para resolver um problema de classificação, define-se este SFBR como um Sistema de Classificação *Fuzzy* Baseado em Regras (SCFBR). As regras de um SCFBR têm, em seu *consequente*, a atribuição de uma classe ao exemplo, o que resulta em regras com o seguinte formato:

$$\boldsymbol{SE}\ X_1 \text{é}\ A_{1l_1}\ \boldsymbol{E}\ \ldots\ X_n \text{é}\ A_{nl_n}\ \boldsymbol{ENTÃO}\ Classe = C_i$$

Em que, $X_1, \ldots, X_n$ são os atributos dos objetos representados por variáveis linguísticas; $\mathrm{A}_{1\mathrm{l}_1}, \ldots\ \mathrm{A}_{\mathrm{nl}_\mathrm{n}}$ são os valores linguísticos usados para representar os valores dos atributos e $C_j, j = 1, \ldots, m$ a classe atribuída aos exemplos classificados pela regra.

Quando um conjunto de regras *fuzzy* é aplicado a um exemplo a ser classificado, utiliza-se um mecanismo de inferência o qual determina a classe à qual o objeto deve pertencer. Um mecanismo de inferência bastante utilizado nos SCFBR é o *Método de Raciocínio Fuzzy Clássico*, mais conhecido por *Método da Regra Vencedora*. Este método classifica um objeto usando a regra que possuir o maior grau de compatibilidade com o mesmo. O Algoritmo 2 descreve os passos deste método de inferência e está descrito conforme apresentado em (CINTRA, 2007).

Algoritmo 2: Algoritmo do Método de Raciocínio *Fuzzy* Clássico

Entrada: Um objeto $\boldsymbol{x_i} = (a_{i1}, \dots, a_{in})$ a ser classificado por um conjunto de s regras $\{R_1, \dots, R_s\}$ de um sistema de classificação, cada uma com *n* antecedentes.

Saída: Classe do objeto $\boldsymbol{x_i}$.

1. Calcular o grau de compatibilidade entre o padrão $\boldsymbol{x_i}$ e cada regra $R_k, k = 1, \dots, s$
2.
3. $$Compat(R_k, \boldsymbol{x_i}) = \boldsymbol{t}(A_{1l_1}(a_{i1}), \dots, A_{nl_n}(a_{in}))$$
4.
5. em que $\boldsymbol{t}$ denota uma *t-norma* e $A_{jl_j}(a_{ij}), j = 1, \dots, n$ é o grau de compatibilidade do
6. valor do atributo a_{ij} no $j - ésimo$ conjunto *fuzzy* da regra *fuzzy* R_k.
7. Encontrar a regra R_{kmax} com maior grau de compatibilidade com o padrão:
8.
9. $$max\{Compat(R_k, \boldsymbol{x_i})\}, k = 1, \dots, s$$
10.
11. Atribuir a classe C_i ao padrão $\boldsymbol{x_i}$, tal que C_i é classe predita pela regra R_{kmax} encontrada no passo anterior.

Há, também, o *Método de Raciocínio Fuzzy Geral*, que agrega os graus de compatibilidade do objeto com as regras que possuem a mesma classe no *consequente*. Este método é descrito pelo Algoritmo 3.

Algoritmo 3: Algoritmo do Método de Raciocínio *Fuzzy* Geral

Entrada: Um objeto $\boldsymbol{x_i} = (a_{i1}, \dots, a_{in})$ a ser classificado por um conjunto de s regras $\{R_1, \dots, R_s\}$ de um sistema de classificação, cada uma com n antecedentes.

Saída: Classe do objeto $\boldsymbol{x_i}$

1. Calcular o grau de compatibilidade entre o padrão $\boldsymbol{x_i}$ e cada regra $R_k, k = 1, \dots, s$
2.
3. $$Compat(R_k, \boldsymbol{x_i}) = \boldsymbol{t}(A_{1l_1}(a_{i1}), \dots, A_{nl_n}(a_{in}))$$
4.
5. em que $\boldsymbol{t}$ denota uma *t-norma* e $A_{jl_j}(a_{ij}), j = 1, \dots, n$ é o grau de compatibilidade do
6. valor do atributo a_{ij} no $j - ésimo$ conjunto *fuzzy* da regra *fuzzy* R_k.
7. Para cada classe C calcular o valor de $Classe_C$, ou, seja, o grau de classificação
8. do padrão na classe, agregando os graus de compatibilidade do passo anterior
9. de todas as regras cuja classe predita é C.
10.
11. $$Classe_C = f\{Compat(R_k, \boldsymbol{x_i}) | C \text{ é a } classe \text{ de } R_k\}$$
12.
13. sendo f um operador de agregação tal que $min \leq f \leq max$.
14. A classe C_j será atribuída ao padrão $\boldsymbol{x_i}$, sendo C_j a classe de $Classe_C$ que possui maior soma dos graus de pertinência.

2.7 Considerações Finais

Neste capítulo foram introduzidos os principais conceitos relevantes ao trabalho realizado. Alguns dos conceitos apresentados, como o Algoritmo C4.5 e os Sistemas *Fuzzy*, estão diretamente relacionados ao tema deste trabalho, cujo algoritmo proposto foi inspirado no algoritmo C4.5. Trata-se de uma Árvore de Decisão *Fuzzy*, tema do Capítulo 3.

Capítulo 3

ÁRVORES DE DECISÃO *FUZZY*

3.1 Considerações Iniciais

Nas últimas décadas as árvores de decisão foram expandidas para incluir a representação de conhecimento impreciso, utilizando a teoria de conjuntos *fuzzy* (KLIR; YUAN, 1995) e dando origem às chamadas Árvores de Decisão *Fuzzy* (CHANDRA; VARGHESE, 2009; JANIKOW; KAWA, 2005; JANIKOW, 1998; OLARU; WEHENKEL, 2003; PEDRYCZ; SOSNOWSKI, 2005; YUAN; KLIR; SWAN-STONE, 1995).

Uma das principais vantagens da ADF sobre a AD é que a ADF tem capacidade de lidar com dados imprecisos, incertos ou com ruídos. Com isso, geralmente, ela apresenta melhor desempenho com relação à taxa de classificação correta. No entanto, as ADFs também apresentam problemas que devem ser considerados. O principal deles é com relação à fuzificação dos atributos contínuos, que devem ser definidos em termos de conjuntos *fuzzy*. Porém, não há um consenso sobre qual é a melhor maneira de gerar estes conjuntos, nem mesmo sobre qual a quantidade de conjuntos *fuzzy* que cada variável fuzificada deve conter. Além disso, como podemos observar em alguns experimentos realizados (RIBEIRO; CAMARGO; CINTRA, 2013b), a ADF tende a gerar um número de regras significativamente maior que a AD. Este problema está diretamente relacionado à quantidade de conjuntos *fuzzy*

definidos para os atributos, uma vez que, quando um atributo for selecionado para um nó da árvore, a quantidade de ramos gerados por ele será igual ao número de conjuntos a ele associado.

Alguns algoritmos para construção de ADFs foram implementados, entre eles, o FuzzyDT proposto por Cintra e Camargo (CINTRA; CAMARGO, 2010) e o FID, proposto por Janikow (JANIKOW; FAJFER, 1999; JANIKOW; KAWA, 2005; JANIKOW, 1998). Na Seção 3.2 serão apresentados alguns trabalhos sobre ADFs.

3.2 Trabalhos Relacionados

Em se tratando de propostas de algoritmos para a indução de ADFs, foram identificados trabalhos que seguem duas vertentes:

i. Algoritmos que fuzificam os dados de entrada previamente, constroem a ADF e aplicam um mecanismo de inferência *fuzzy* para realizar a classificação de novos dados (Os algoritmos que seguem este padrão serão tratados por *Algoritmos com pré-fuzificação* de atributos contínuos).
ii. Algoritmos que realizam a fuzificação dos atributos durante a indução da ADF (Estes serão tratados por *Algoritmos sem pré-fuzificação* de atributos contínuos).

As seções 3.2.1 e 3.2.2 abordam alguns dos trabalhos relacionados que foram estudados, separando-os segundo as duas vertentes acima citadas, respectivamente.

3.2.1 Algoritmos Com Pré Fuzificação de Atributos

Em 2005, Janikow & colaboradores propuseram o FID3.4 (JANIKOW; KAWA, 2005). Este algoritmo conta com dois procedimentos distintos para o

que o autor chama de particionamento dos atributos contínuos. Um dos procedimentos segue uma bordagem "*top-down*" e o outro "*bottom-up*" e o algoritmo poderia ter sido incluído em ambas as seções, 3.2.1 e 3.2.2, já que um dos procedimentos é realizado durante a indução da árvore enquanto o outro é realizado antes.

A abordagem "*top-down*" inicia com uma partição única dos dados e, de acordo com o atributo a ser particionado, o domínio vai sendo dividido nos pontos em que aumentam a medida de informação no nó. Este tipo de particionamento é feito durante a indução da árvore e há um limite para o número de subconjuntos que é gerado para cada atributo.

Já a abordagem "bottom-up" inicia com o domínio do atributo particionado em conjuntos unários, associados a cada ponto do domínio e esses conjuntos vão sendo combinados até resultar na partição final. Os conjuntos gerados por ambos os métodos de particionamento são associados a funções de pertinência trapezoidais, as quais têm suas formas controladas por parâmetros.

Na fase de indução da árvore a seleção dos atributos é feita de maneira recursiva muito similar à descrita no Algoritmo 1, com base no atributo que maximiza a função de ganho de informação ou a razão de ganho, que é adotada para atributos que apresentam grande cardinalidade. O processo termina quando não há mais atributos a serem testados ou quando o nível de informação ou, então, o número de exemplos remanescentes atinge um limite previamente definido. Além disso, durante a seleção do melhor atributo, pode ocorrer a poda caso o mesmo falhe em um teste de relevância.

Em (UMANO et al., 1994) os autores propuseram um algoritmo capaz de lidar tanto com variáveis clássicas quanto *fuzzy*. Trata-se de uma extensão do algoritmo ID3 (QUINLAN, 1986) capaz de gerar uma árvore de decisão utilizando conjuntos clássicos e *fuzzy* definidos previamente pelo usuário. O método utiliza ganho de informação e entropia para selecionar os melhores atributos e, uma vez que o modelo foi induzido, um método de inferência é aplicado. Em (EVANS; LOHSE; SUMMERS, 2013) este algoritmo é utilizado para o auxílio na tomada de decisão a respeito da seleção de tecnologias de fabricação para classificar problemas de manufatura. A informação fornecida

pelo algoritmo é utilizada para fazer um ranqueamento dos possíveis métodos de manufatura. Esta lista ranqueada é apresentada a especialistas que tomam a decisão a respeito da solução que será aplicada.

Em (LEVASHENKO; ZAITSEVA; PUURONEN, 2007), os autores implementaram uma árvore de decisão *fuzzy* que utiliza a entropia acumulada dos atributos, para a seleção dos nós. Na proposta, a fuzificação dos dados iniciais é feita por um algoritmo proposto por Lee et. al. (LEE et al., 2001) que transforma os dados contínuos em conjuntos *fuzzy* associados a funções de pertinência triangulares. Este processo é realizado antes da indução da árvore. Os passos realizados para a fuzificação dos atributos são:

Para cada atributo contínuo:

1. Inicia-se com o número de intervalos $I = 2$.
2. Localizam-se os centros dos intervalos:

   O algoritmo de agrupamento *K-means* (CHI; YAN, 1995) é utilizado para localizar os centros de cada intervalo.
3. Associa-se uma função de pertinência triangular para cada intervalo.

   Para isso, associa-se o maior valor de pertinência "$1{,}0$" ao centro do intervalo definido no passo 2 e o menor valor "$0{,}0$" aos centros dos seus intervalos vizinhos. No caso dos intervalos das extremidades, que só apresentam um intervalo vizinho, ao menor valor e ao maior valor do domínio associa-se grau de pertinência "$0{,}5$". A Figura 9 exemplifica a definição das funções de pertinência.
4. Calcula-se a entropia *fuzzy* total para o atributo com I e $I - 1$ números de intervalos.
5. O valor da entropia *fuzzy* total diminuiu?
   a) Se o valor da entropia fuzzy total para I intervalos é menor que para $I - 1$ intervalos, então:
      - $I = I + 1$
      - Volta-se ao passo 2.

b) Se não, passa-se ao passo 6.

6. O número de intervalos é definido como sendo $I - 1$ e as funções de pertinência associadas a cada intervalo do atributo são definidas como sendo as calculadas na última iteração.

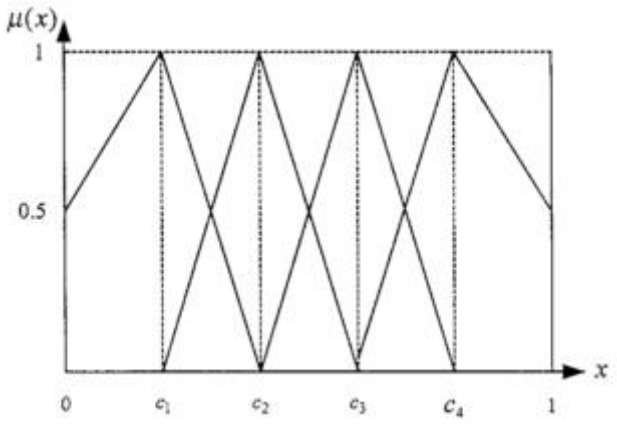

Figura 9: Exemplo de definição de funções de pertinência a partir dos centros dos intervalos.

Conforme informado no passo 2 do procedimento descrito anteriormente, para determinar os centros dos intervalos das funções triangulares, o algoritmo *K-Means* é utilizado.

Considere N vetores M-dimensionais $V_i = (v_{i1}, v_{i2}, \dots, v_{iM})^T, i = 1, 2, \dots, N$, descritos por N atributos. Para particionar os atributos em intervalos de dimensão j, primeiro são extraídos N valores dos atributos projetados na dimensão $x_i^{(j)} = v_{ij},\ i = 1, 2, \dots, N$. O algoritmo de agrupamento *K-Means* é utilizado para agrupar $x_i^{(j)},\ i = 1, 2, \dots, N$.

O algoritmo consiste nos sequintes passos:

1. Define-se o número de grupos inicial, I, seguindo o procedimento descrito acima.
2. Define-se os centros de gupos inciais.

   Os centros de grupos iniciais $c_1, c_2, \dots, c_I$são selecionados aleatoriamente de $x_i^{(j)},\ i = 1, 2, \dots, N$. No sistema proposto, o centro de grupo c_q de um grupo arbitrário q é definido como:

$$c_q = \frac{q-1}{I-1}, q = 1, 2, \dots, I.$$

3. Associa-se o rótulo do grupo a cada elemento.

   Após determinar os centros dos grupos, cada elemento é associado ao rótulo do grupo cujo centro de grupo está mais próximo, segundo a distância euclidiana. Sendo assim, o centro de grupo mais próximo satisfaz a seguinte medida de distância: $\left|x_i^{(j)} - c_q^*\right| = \min_{1 \le q \le I} \left|x_i^{(j)} - c_q\right|$ onde c_q^* é o centro de grupo mais próximo do elemento $x_i^{(j)}$.

4. Recalcula-se os centros de grupo.

   Uma vez que os centros de grupo inciais são selecionados aleatoriamente, cada centro precisa ser reavaliado, calculando-se a seguinte estimativa: $c_q = \frac{\sum_{q=1}^{N_q} x_i^{(j)}}{N_q}$, onde N_q é o número total de elementos no mesmo grupo q.

5. Algum centro de grupo mudou?

   Se nenhum centro de grupo mudou, os centros dos intervalos são definidos como os centros de grupos.

   Se não, volte ao passo 3.

No algoritmo proposto por Levashenko et. al. a seleção do atributo depende do resultado da seleção dos atributos anteriores. Para cada ramo da ADF, o atributo que fornece maior quantidade de informação com o menor custo para a classificação é selecionado. Para isso, calcula-se o custo somando o número de atributos que já foram selecionados e a informação é calculada através da entropia acumulada dos atributos. Para evitar uma busca exaustiva, o crescimento da árvore é limitado pela poda do ramo que provê menor quantidade de informação.

Em (AFSARI et al., 2013) é feita outra abordagem para a geração de um SFBR. No trabalho, Afsari et al. propõem o *Interpretability-based Fuzzy Decision Tree Classifier (IFDTC),* que explora as vantagens dos algoritmos de árvore de decisão, bem como as dos algoritmos de agrupamento sequencial (CHIU, 1994). A Figura 10 resume os 3 principais passos do algoritmo.

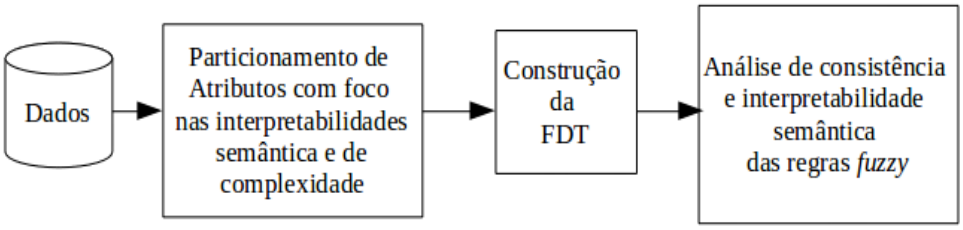

Figura 10: Esquema global do desenvolvimento da IFDTC (Traduzida de (Afsari et. al. 2013))

1. Particionamento dos atributos: Os conjuntos *fuzzy* são gerados automaticamente a partir dos dados de entrada. Para isso, utiliza-se um algoritmo de agrupamento sequencial e depois, os melhores conjuntos são selecionados através de um Algoritmo Genético Multi-objetivo (SPEA2 (ZITZLER; LAUMANNS; THIELE, 2001)). Neste algoritmo cada cromossomo é representado pela codificação real e tem *n = número de atributos* genes para determinar os valores dos raios utilizados no algoritmo de agrupamento e dois genes para determinar os parâmetros do algoritmo de indução da árvore que controlam quando as divisões cessam, interrompendo o crescimento da mesma. Para os diferentes conjuntos *fuzzy* produzidos pelo algoritmo de agrupamento, os seguintes objetivos devem ser otimizados pelo Algoritmo Genético Multi-objetivo (AGMO):

    i. Acurácia, definida pelo menor erro;
    ii. Interpretabilidade de complexidade: número de nós folha (número de regras);
    iii. Interpretabilidade semântica: medida pelo índice *GM3M* (*Geometric Mean of Three Metrics*) que consiste de uma média geométrica de: Entropia da partição, coeficiente da partição e *Chen Index*.

2. Construção da ADF: A ADF é construída utilizando os conjuntos obtidos no passo anterior. Neste processo a construção da ADF é feita calculando-se o ganho de informação *fuzzy* dos atributos para decidir sobre a ordem na qual os mesmos serão inseridos na árvore.

3. Análise de consistência e interpretabilidade semântica das regras *fuzzy*: Neste passo é realizado um ajuste da base de regras, aplicando-se um processo de refinamento denominado análise de consistência. O objetivo é identificar qualquer conflito potencial entre regras. Os conflitos são divididos em dois grupos: i) regras inconsistentes e ii) regras redundantes. O primeiro grupo abrange os conflitos que precisam ser solucionados para garantir a consistência da base de regras. O segundo afeta a interpretabilidade semântica da mesma. Existem cinco principais conflitos que podem ocorrer entre duas regras:

    i. Antecedentes iguais e consequentes diferentes (inconsistência);
    ii. Antecedente de uma regra está contido no antecedente de outra e seus consequentes são diferentes (especialização);
    iii. Antecedente de uma regra está contido no antecedente de outra e seus consequentes são iguais (redundância);
    iv. A intersecção dos antecedentes de duas regras não é vazia e seus consequentes são diferentes (inconsistência parcial);
    v. A intersecção dos antecedentes de duas regras não é vazia e seus consequentes são iguais (redundância parcial).

    O tratamento de conflitos adotado pelos autores consiste em diferentes ações para cada um deles:

    i. Inconsistência: Uma das regras é excluída;
    ii. Especialização: Há duas opções de tratamento: a) Mantém-se somente a regra que abrange a maior parte do domínio e define-se o melhor consequente; b) Mantém-se a regra mais específica e redefine-se a regra mais geral de forma que ela cubra o domínio que não é abrangido pela regra mais específica;
    iii. Redundância: Exclui-se a regra mais específica;

iv. Inconsistência parcial: Redefine-se as regras a fim de evitar intersecção no domínio abrangido por cada uma e define-se o melhor consequente para o a regra que abrange o domínio comum.

v. Redundância parcial: Exclui-se a parte redundante das regras.

Após a indução das regras, novas instâncias podem ser classificados através da aplicação de um mecanismo de inferência *fuzzy*.

O algoritmo FuzzyDT, proposto por Cintra e Camargo (CINTRA; CAMARGO, 2010), primeiramente fuzifica os atributos contínuos do conjunto de treinamento. Após a *fuzifi*cação, a ADF é construída através dos mesmos passos seguidos pelo algoritmo C4.5, considerando ganho de informação e entropia para determinar qual atributo definirá um determinado nó da árvore. Como resultado, obtém-se uma ADF em que cada ramo representa uma regra *fuzzy*. O Algoritmo 4 mostra os passos executados pelo FuzzyDT.

Algoritmo 4: Algoritmo FuzzyDT (CINTRA; CAMARGO, 2010)

Entrada: Um conjunto de treinamento $\boldsymbol{D} = (\boldsymbol{x_i}, y_i)$

Saída: Árvore de Decisão *Fuzzy*

1. Determinar o número de conjuntos *fuzzy* em que cada atributo contínuo será particionado;
2. Substituir os valores dos atributos contínuos pelos termos linguísticos dos conjuntos *fuzzy* com maior compatibilidade com os valores de entrada (fuzificação);
3. Calcular a entropia e o ganho de informação para cada atributo, para dividir o conjunto de treinamento e definir os testes que serão realizados nos nós, até que todos os atributos ou todo o conjunto de treinamento seja utilizado;
4. Aplicar o processo de poda usando 25% de confiança.

Para ilustrar, considere o conjunto de dados que a Tabela exibe, extraído do conjunto Iris do repositório *UCI* (LICHMAN, 2013).

Tabela 9: Exemplos do conjunto Iris.

Comp. de sépala	Larg. de sépala	Comp.de pétala	Larg. de pétala	Classe
4,4	2,9	1,4	0,2	Iris-setosa
4,9	2,4	3,3	1,0	Iris-versicolor
4,9	2,5	4,5	1,7	Iris-virgínica
4,8	3,4	1,9	0,2	Iris-setosa
5,0	2,0	3,5	1,0	Iris-versicolor
5,6	2,8	4,9	2,0	Iris-virgínica
5,7	3,8	1,7	0,3	Iris-setosa
5,7	2,6	3,5	1,0	Iris-versicolor
5,7	2,5	5,0	2,0	Iris-virgínica

O conjunto Iris apresenta exemplos de flores do tipo iris que podem ser classificadas como iris-setosa, iris-versicolor ou iris-virgÍnica, de acordo com os valores de seus quatro atributos: Comprimento de sépala, Largura de sépala, Comprimento de pétala e Largura de pétala.

Seguindo os passos do algoritmo FuzzyDT, primeiramente, é necessário definir a quantidade de conjuntos que representarão cada um dos atributos. No FuzzyDT, este procedimento pode ser feito manual ou automaticamente. Neste exemplo, consideram-se três conjuntos *fuzzy* associados a cada atributo. Estes conjuntos também podem ser gerados automaticamente pelo algoritmo FuzzyDT ou manualmente pelo usuário.

A geração automática dos conjuntos resulta em funções de pertinência triangulares, igualmente espaçadas, conforme a ilustram as Figuras 11, 12, 13 e 14.

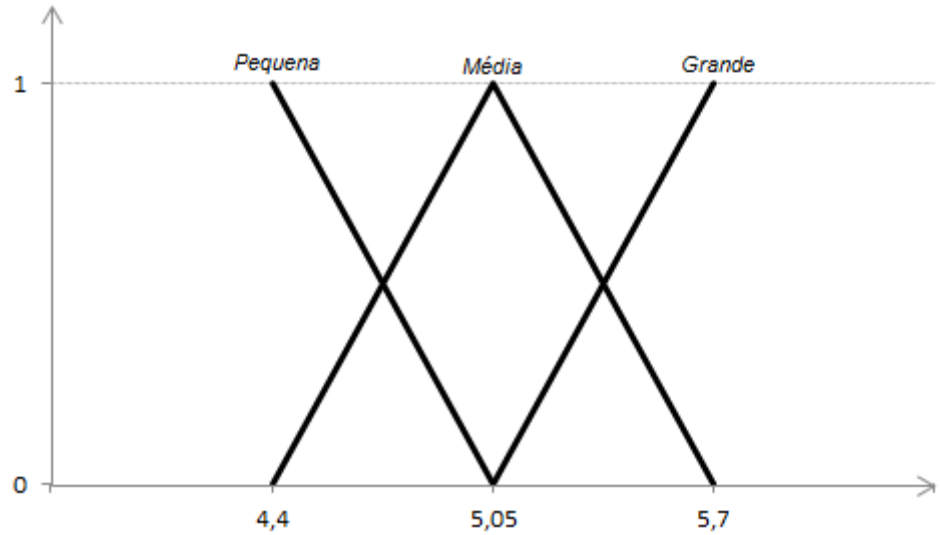

Figura 11: Conjuntos *fuzzy* que representam o atributo comprimento de sépala.

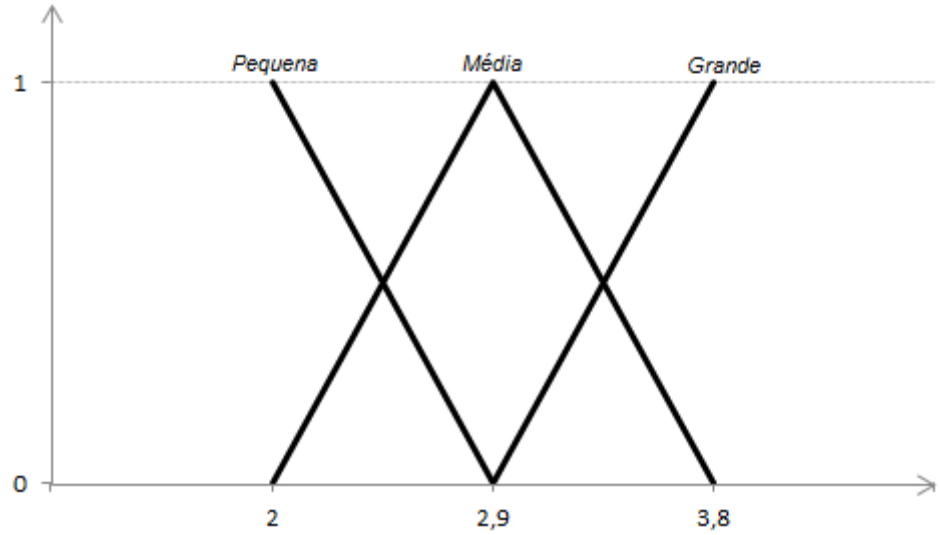

Figura 12: Conjuntos *fuzzy* que representam o atributo largura de sépala.

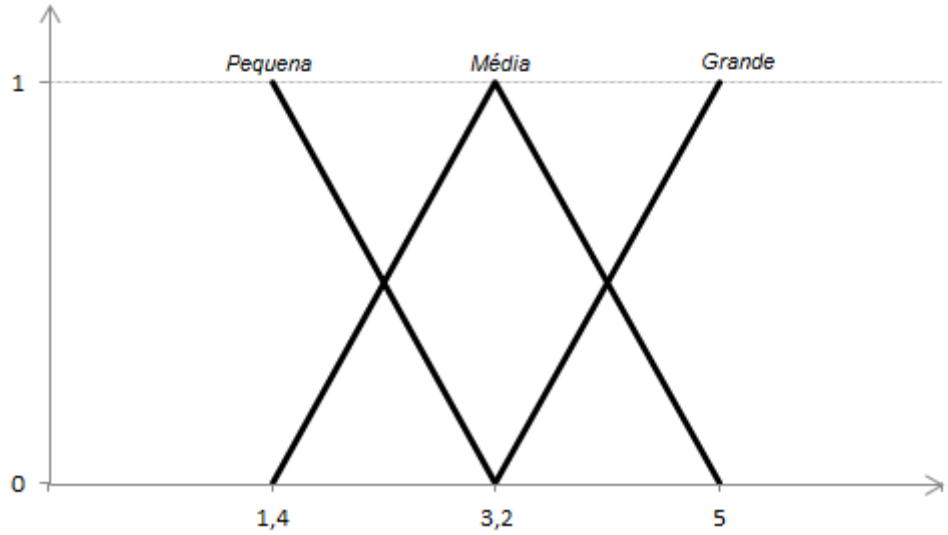

Figura 13: Conjuntos *fuzzy* que representam o atributo comprimento de pétala

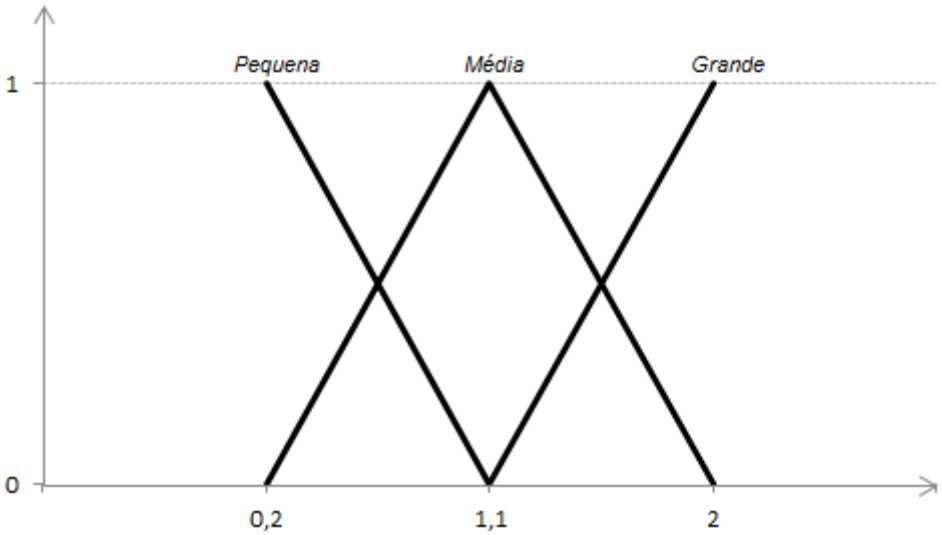

Figura 14:Conjuntos *fuzzy* que representam o atributo largura de pétala.

Depois que os conjuntos associados a cada um dos atributos foram definidos, realiza-se a fuzificação dos dados de treinamento, calculando, para cada valor de cada atributo, o grau seu de pertinência a cada conjunto *fuzzy* que define o domínio do atributo. Para os dados Tabela , substitui-se o valor do atributo pelo termo linguístico que representa o conjunto *fuzzy* no qual ele obteve maior grau de pertinência, resultando nos dados da Tabela 5.

Tabela 5: Exemplos do conjunto Iris fuzificados.

Comp. de sépala	Larg. de sépala	Comp.de pétala	Larg. de pétala	Classe
Pequena	Média	Pequena	Pequena	Iris-setosa
Média	Pequena	Média	Média	Iris-versicolor
Média	Pequena	Grande	Grande	Iris-virgínica
Média	Grande	Pequena	Pequena	Iris-setosa
Média	Pequena	Média	Média	Iris-versicolor
Grande	Média	Grande	Grande	Iris-virgínica
Grande	Grande	Pequena	Pequena	Iris-setosa
Grande	Pequena	Média	Média	Iris-versicolor
Grande	Pequena	Grande	Grande	Iris-virgínica

O conjunto representado pela Tabela 5 será a entrada do algoritmo C4.5. Note que após a fuzificação, os atributos passam a serem considerados atributos nominais e não é necessário fazer qualquer alteração para que a indução da ADF seja feita pelo algoritmo C4.5.

Depois da árvore induzida, aplica-se um método de inferência *fuzzy* usando as regras geradas para classificar novos objetos.

Em 2014, Feng, Liu e Wang (FENG; LIU; WANG, 2014) aplicaram a medida euclidiana *fuzzy* como critério de seleção de atributos para uma árvore de decisão *fuzzy*, o classificador EFDT. Esta medida é utilizada para medir a imprecisão obtida após um nó ser dividido.

Seja A um conjunto *fuzzy* definido sobre o domínio do atributo $X = \{x_1, x_2, \ldots, x_N\}$, onde N é o número de valores em X, a medida euclidiana *fuzzy* do conjunto A é definida pela Equação 17.

$$Euclid(A) = \frac{2}{N^{\frac{1}{2}}} \left(\sum_{i=1}^{N} |\mu_A(x_i) - A_{0,5}(x_i)|^2 \right)^{\frac{1}{2}} \quad (17)$$

$$A_{0,5}(x_i) = \begin{cases} 1, & \mu_A(x_i) \geq 0{,}5; \\ 0, & \mu_A(x_i) < 0{,}5. \end{cases}$$

O algoritmo EFDT consiste de três passos principais:

1. Construir a árvore EFDT;
2. Definir o threshold δ;
3. Poda da base de regras extraída da EFDT induzida.

Dado um nó N que contém um conjunto de exemplos pertencentes a c diferentes classes e o atributo A (definido por k conjuntos *fuzzy* $T(A) = \{m1, m2, \ldots, mk\}$), candidato para fazer a divisão de N, a incerteza do nó N pode ser mensurada pela medida euclidiana *fuzzy*, definidapela Equação 18.

$$Euclid(N) = \frac{2}{c^{\frac{1}{2}}}\left(\sum_{i=1}^{c}\left(\frac{1}{2} - \left|\frac{1}{2} - \frac{|N_{v_i}|}{|N|}\right|\right)^2\right)^{\frac{1}{2}} \quad (18)$$

onde N_{v_i} representa os exemplos pertencentes à classe v_i. A redução da incerteza obtida pela divisão do nó N pelo atributo A é calculada conforme a Equação 19.

$$Redução(N, A) = Euclid(N) - \sum_{j=1}^{k}\frac{|N \cap m_j|}{|N|} Euclid(N \cap m_j) \quad (19)$$

onde $N \cap m_j$ representa o $j - ésimo$ filho do nó N. O algoritmo EFDT seleciona o atributo que maximiza a redução da incerteza ao dividir o nó. As funções de pertinência m_j utilizadas no cálculo da redução da incerteza são previamente definidas por funções de pertinência triangulares que tem seus valores a, b e m definidos por:

a: valor mínimo do domínio do atributo
b: valor máximo do domínio do atributo
m: $\frac{a+b}{2}$

A construção da árvore é feita pela divisão dos nós, que ocorre recursivamente, até que um dos seguintes critérios de parada seja atingido:

i. O número de instâncias cujo grau de pertinência é maior que um dado *threshold* δ no nó é menor que 2;

ii. Não há mais atributos a serem particionados

A classe dos nós folha é definida pela classe majoritária dos exemplos no nó.

A definição de δ é feita através da execução de um algoritmo genético que tem a função de *fitness* definida conforme a Equação 20.

$$F(\delta) = |X| \cdot acurácia\ de\ treinamento - \ \delta \cdot no\_nós \quad (20)$$

Onde $|X|$ é número de exemplos de treinamento e $no\_nós$ é o número de nós da árvore.

A base de regras, extraída do modelo EFDT induzido, é podada de acordo com os passos:

1. Remove-se cada regra da base de regras e classifica-se os dados de treinamento utilizando as regras remanescentes.
2. Exclui-se a regra cujo conjunto de regras remanescentes apresenta maior aumento da acurácia nos dados de treinamento
3. Repete-se os passos 1-2 e para-se o processo de poda caso a base de regras podada resultante tenha resultados piores que a base de regras original, quando aplicada aos dados de treinamento.

3.2.2 Algoritmos Sem Pré Fuzificação de Atributos

Em (BOYEN; WEHENKEL, 1999), Xavier Boyen e Louis Wehenkel propuseram uma ADF restrita a problemas de classificação binários que foi aplicada a um domínio de avaliação de segurança de um sistema de energia. Neste contexto, o objetivo era representar o grau de pertinência de um objeto do domínio à classe, sendo que grau zero correspondia a estados completamente instáveis e grau um aos estados completamente estáveis.

Neste método, o universo de discurso *U* é considerado como sendo todos os possíveis objetos do contexto do problema. *U* também corresponde ao

nó raiz da árvore, que contém todas as instâncias, enquanto os outros nós correspondem a subconjuntos de *U* que são, recursivamente construídos. A construção se dá através da combinação de funções de pertinência *fuzzy* nos nós teste que são induzidas pelos valores dos atributos. Os nós folha, por sua vez, correspondem a subconjuntos *fuzzy* que foram definidos de acordo com as funções de pertinência de seus nós pai. Cada nó folha é rotulado com um valor sintético que representa o grau de pertinência médio encontrado entre os objetos deste subconjunto.

O processo de construção da árvore consiste em buscar a árvore mais simples, cujo modelo seja suficientemente próximo da situação real. O algoritmo de busca é definido através de um espaço de hipóteses (um conjunto de árvores candidatas) e sua estratégia de busca.

O espaço de hipóteses de árvores *fuzzy* é definido indiretamente com a definição dos conjuntos das funções de pertinência *fuzzy* atribuídas aos nós teste.

A busca pela melhor árvore utiliza a estratégia de busca gulosa. As árvores são induzidas por uma abordagem recursiva do tipo *top-down*. Sendo assim, a indução começa pelo nó raiz, que tem todos os objetos do domínio. Para lidar com atributos de domínio contínuo, realiza-se uma busca pela melhor função de pertinência local, utilizando-se uma função de avaliação *ad hoc,* que seleciona a função de pertinência que melhor define o atributo. Uma vez que a função de pertinência foi definida, o conjunto é particionado em dois subconjuntos, definidos pela função de pertinência, gerando dois nós filhos que serão expandidos recursivamente.

As funções de pertinência que definem os atributos contínuos podem ter um dos três formatos mostrados na Figura 15.

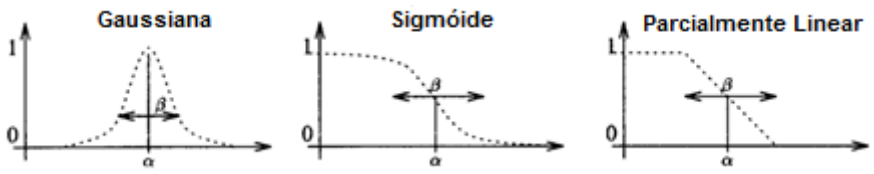

Figura 15: Possibilidades de formatos a serem utilizados na definição das funções de pertinência dos atributos contínuos (Extraída de (BOYEN; WEHENKEL, 1999)**).**

Os valores α e β, que representam a posição e o formato da função, são definidos pela função a*d hoc* citada acima. Nos testes realizados pelos autores, somente as funções monotônicas (sigmóide e parcialmente linear) foram utilizadas.

Antes da expansão, algumas condições de parada são testadas e caso sejam atingidas, o nó é rotulado e transformado em folha. Uma vez que as árvores foram construídas, a poda é realizada e a melhor árvore é selecionada. A árvore selecionada é escrita no formato de função e um ajuste de parâmetros é feito a fim de minimizar o erro médio quadrático da mesma.

Em (CHANDRA; VARGHESE, 2009) uma abordagem diferente é utilizada. Neste trabalho, os autores propõem uma árvore de decisão *fuzzy* a qual chamaram de Gini Index *Fuzzy* Decision Tree (*G-FDT*). Nela, a fuzificação dos limites dos atributos é realizada em cada nó da AD, ou seja, durante a sua construção. A árvore é construída a partir da raiz, e uma medida chamada *Gini Index é* utilizada como método de seleção de atributos.

O algoritmo consiste em, primeiramente, calcular os *Split Points* de cada atributo, que são os pontos nos quais o mesmo será particionado. Uma vez feito isso, é calculado o *Gini Index* para todos os *Split Points* identificados. O *Gini Index*, assim como o ganho de informação utilizado por outros algoritmos, é uma medida de pureza utilizada na seleção dos atributos. O atributo que contiver o *Split Point* cujo *Gini Index* foi o de melhor valor encontrado, será o atributo selecionado para o nó em questão. Este *Split Point* será o ponto no qual ocorrerá a *fuzifi*cação dos limites do atributo selecionado, por meio de uma função de pertinência do tipo “S”.

Neste trabalho, o método proposto pelos autores foi comparado à sua versão *crisp*, o algoritmo *SLIQ* (MEHTA; AGRAWAL; RISSANEN, 1996). A validação foi feita utilizando-se *10-fold Cross Validation* e foram testados 14 conjuntos do repositório *UCI* (LICHMAN, 2013). Os resultados foram analisados comparando-se a Taxa de Classificação Correta (TCC), e a interpretabilidade, medida pelo tamanho da AD gerada pelos algoritmos.

Após as comparações, os autores concluíram que a *G-FDT* teve melhor desempenho que seu concorrente, *SLIQ*, uma vez que sua TCC é

significativamente melhor e o tamanho das ADs geradas por este algoritmo é significativamente menor em comparação com o *SLIQ*.

Varma et al., propuseram uma modificação da *G-FDT* em (VARMA et al., 2014). Com o intuito de desenvolver um modelo capaz de diagnosticar um paciente com diabetes o mais cedo possível, os autores propuseram seu método em busca de uma solução eficiente e computacionalmente eficaz. Assim, criaram a *Gini Index-Gaussian Fuzzy Decision Tree* (GG-FSDT).

Enquanto em (CHANDRA; VARGHESE, 2009) Chandra e Varghese propuseram a fuzificação por meio de funções de pertinência do tipo “S”, Varma et. al. propuseram que a GG-FSDT fosse construída pelo mesmo método da G-FDT, porém que sua fuzificação fosse feita por meio de funções de pertinência do tipo “gaussianas”.

Olaru e Wehenkel (OLARU; WEHENKEL, 2003) propõem, em seu trabalho, uma árvore de decisão *fuzzy* chamada *Soft Decision Tree* (*SDT*), cujas classes são fuzzy, ou seja, a saída do algoritmo é o grau de pertinência do objeto em uma determinada classe. Sua indução é feita em, basicamente, três passos.

O primeiro consiste em construir uma árvore “suficientemente grande” utilizando um conjunto de objetos denominado *Growing Set*. Nesta fase, os nós são adicionados à árvore em uma abordagem *top-down* até que seja atingido um critério de parada.

Analogamente ao algoritmo CART (BREIMAN et al., 1984), a construção da *SDT* se dá através de três itens básicos: um método automático para selecionar um ponto de divisão para cada novo nó da árvore, uma regra para determinar quando um nó deve ser considerado final, ou folha, e uma regra para determinar que rótulo este nó folha receberá.

A seleção de atributos, bem como de seus melhores pontos de divisão são calculados através da minimização de uma função de erro quadrático. Uma vez que o atributo e o ponto de divisão foram definidos, duas funções de pertinência *fuzzy,* parcialmente lineares, são associadas ao domínio do atributo.

As divisões são cessadas assim que for identificado que a expansão de um nó será inútil. Uma vez construída, a árvore é podada segundo uma

abordagem *bottom-up* com o intuito de eliminar partes irrelevantes do modelo. Uma vez podada, inicia-se o terceiro passo que pode ser o *refitting* ou *backfitting*. Ambos consistem de fazer o ajuste dos parâmetros da árvore podada, a fim de aumentar sua capacidade de aproximação. Ao final de cada passo (construção, poda e *refitting* ou *backfitting*), as árvores obtidas (completa, podada e "reconstruída") são testadas com o intuito de quantificar sua capacidade de generalização. Para isso, um conjunto de testes é utilizado para avaliar suas acuidades.

Em trabalho de 2014 (BAJAJ; KUBBA, 2014) propõem o algoritmo *Fuzzy Heterogeneous Split Measure* (*FHSM*) para a construção de árvores de decisão que utilizam funções de pertinência trapezoidais para fuzificar o *Heterogeneous Split Measure* (*HSM*). Os autores enfatizam o problema da dimensionalidade da árvore e o tratam fixando o valor da variável que controla o tamanho da árvore induzida.

O *Heterogeneous Split Measure* (*HSM*) (CHANDRA; BABU KUPPILI, 2011) trabalha com a média quase-linear da função exponencial, uma medida heterogênea que considera a informação local e global do conjunto de dados e visa a redução tanto da altura quanto do número de nós da árvore induzida. Neste método, o atributo de divisão é selecionado através da avaliação do ganho de informação parcial dos subconjuntos. Para calcular o ganho de informação parcial, é considerado o raio da média quase-linear ponderada do ganho de informação do subconjunto com relação ao conjunto antes de ser particionado. As divisões ocorrem sucessivamente até que todos os exemplos do subconjunto pertençam à mesma classe.

Sejam A, B e C as três classes de um conjunto $\boldsymbol{D} = (\boldsymbol{x}_i, y_i)$. L e U são os subconjuntos de X definidos a partir do ponto de divisão T. $L1$ e $U1$ representam o número de exemplos nos conjuntos L e U que pertencem à classe A. $L2$ e $U2$ representam o número de exemplos nos conjuntos L e U que pertencem à classe B. $L3$ e $U3$ representam o número de exemplos nos conjuntos L e U que pertencem à classe C, conforme é mostrado na Tabela 6.

Tabela 6: Histograma das Classes de D

$Y(x_i) < T$	A	B	C
L	$L1$	$L2$	$L3$
U	$U1$	$U2$	$U3$

N é número de exemplos de X.

O valor de *HSM*, para o atributo Y, é calculado pela Equação 21 (Extraída de (BAJAJ; KUBBA, 2014)).

$$HSM = \frac{L1+L2+L3}{N}\log\left[\left(1-\frac{L1}{L1+L2+L3}\right)*e^{-\frac{L1}{N}}+\left(1-\frac{L2}{L1+L2+L3}\right)*e^{-\frac{L2}{N}}+\left(1-\frac{L3}{L1+L2+L3}\right)*e^{-\frac{L3}{N}}\right]+\frac{U1+U2+U3}{N}\log\left[\left(1-\frac{U1}{U1+U2+U3}\right)*e^{-\frac{U1}{N}}+\left(1-\frac{U2}{U1+U2+U3}\right)*e^{-\frac{U2}{N}}+\left(1-\frac{U3}{U1+U2+U3}\right)*e^{-\frac{U3}{N}}\right] \quad (21)$$

A proposta de Bajaj e Kubba visa possibilitar que o *HSM* lide com os dados de forma mais flexível, incluindo conceitos da lógica *fuzzy* ao algoritmo originalmente proposto por Chandra e permitindo que a saída do algoritmo seja dada por mais de uma classe, com diferentes graus de certeza. Eles propõem o *FHSM*, que utiliza funções de pertinência trapezoidais para fuzificar o *HSM*. Os pontos de divisão são determinados pelo ponto médio dos valores dos atributos onde ocorre variação do valor da classe. A cada nó, é calculado o grau de pertinência *fuzzy* dos valores de atributos às classes, utilizando estas funções e substituindo os valores $L1, U1, L2, U2, L3\ e\ U3$, que anteriormente eram definidos pela proporção de exemplos em cada classe, pela soma dos graus de pertinência dos valores em cada classe.

Sendo assim, os valores $L1$ e $U1$ representam a soma dos graus de pertinência das instâncias que pertencem à classe A. $L2$ e $U2$ representam a soma dos graus de pertinência das instâncias que pertencem à classe B. $L3$ e $U3$ representam a soma dos graus de pertinência das instâncias que pertencem à classe C.

Para um problema com 3 classes, a função de pertinência trapezoidal, utilizada para o cálculo dos valores $Li\ e\ Ui, i = 1, 2, 3$, é definida conforme ilustrado na Figura 16.

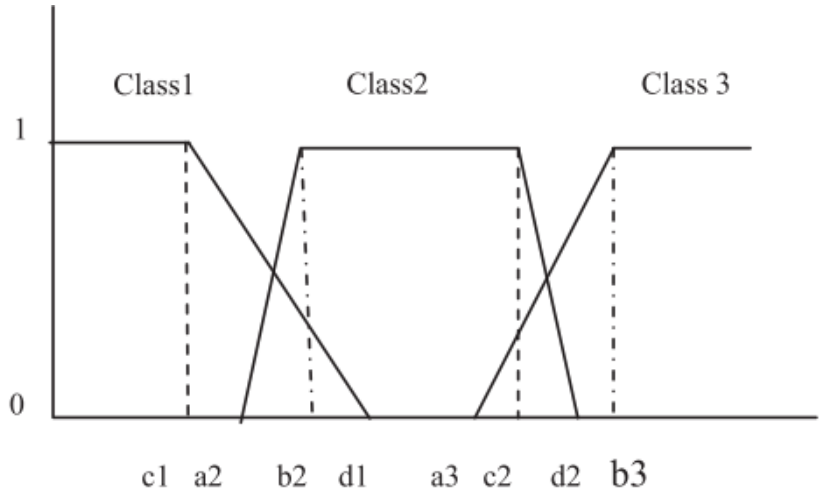

Figura 16: Função de Pertinência (Extraída de (BAJAJ; KUBBA, 2014)**).**

Os passos do algoritmo *FHSM* são apresentados no Algoritmo 5.

Algoritmo 5: Algoritmo *FHSM*

Entrada: Um conjunto de treinamento $\boldsymbol{D} = (\boldsymbol{x_i}, y_i)$
Saída: Atributo que será utilizado para divisão

Se todos os exemplos pertencem à mesma classe:
    Há somente um nó rotulado com o nome da classe.
Senão:
    Calcule o grau de pertinência *fuzzy* para cada exemplo a cada classe como
    sendo 1/número de classes.
    Atualize o grau de pertinência dos exemplos às classes multiplicando-os
    pelo grau de pertinência obtido através da função de pertinência trapezoidal.
    Para cada atributo **faça**:
        Ordene os exemplos com base nos valores do atributo.
        Encontre os pontos de divisão (pontos onde ocorre mudança no valor da
        classe)
        Para cada ponto de divisão **faça**:
            Calcule o valor *FHSM*
        Fim-para
        Selecione o ponto de divisão de maior valor *FHSM* para o atributo.
    Fim-para
Fim-se
Retorne o atributo que tem maior valor de *FHSM*.

Observa-se que neste algoritmo, embora o processo de seleção do nó seja feito com base em um procedimento que utiliza funções de pertinência fuzzy para calcular o valor de *FHSM*, o domínio do atributo não é fuzificado. Sendo assim, os nós de divisão da árvore apresentam as mesmas características de uma árvore de decisão *crisp*.

A diferença no modelo induzido é observada somente nos nós folha que, ao invés serem rotulados com a classe majoritária dos exemplos no nó, considera todas as classes presentes neste subconjunto, atribuindo a cada rótulo um grau de certeza, definido pela proporção de exemplos em cada classe.

Chen and Ludwig (2013) implementam uma ADF que realiza a fuzificação com base na identificação do melhor “ponto de corte”. Os autores aplicam um método de seleção de atributos baseado nas ideias de informação mútua (*mutual information*) e algoritmos genéricos (AG) como parte do preprocessamento. Na Figura 17 são exibidos os passos do processo de fuzificação dos atributos proposto pelos autores.

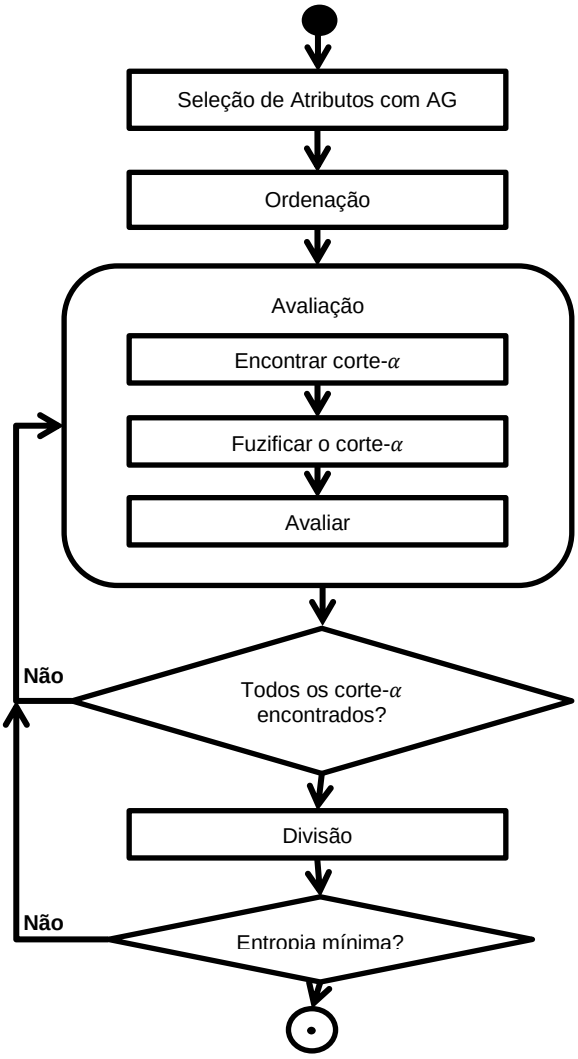

Figura 17: Processo de Fuzificação.

A fase de preprocessamento, que consiste na seleção de atributos para redução de dimensionalidade, abrange os passos descritos no Algoritmo 6.

Algoritmo 6: Método de seleção de tributos baseado em AG.

Entrada: O número de atributos, conjunto $\boldsymbol{D} = (\boldsymbol{x_i}, y_i)$
Saída: Atributo que será utilizado para divisão
Definições: gen_{max}: Número máximo de gerações do AG.
N_{pop}: Tamanho da população do AG.

1. Calcule a entropia de cada atributo.
2. Calcule a entropia de $\boldsymbol{D}$.
3. Calcule a informação mútua entre o atributo e $\boldsymbol{D}$.
4. Calcule a informação mútua entre os atributos.
5. Inicialize a população aleatoriamente.
6. **para** gen de 1 até gen_{max} **faça**:
7. **para** $index$ de 1 até N_{pop} **faça**:
8. Calcule a máxima relevância.
9. Calcule a mínima redundância.
10. Calcule
11. $função\ de\ avaliação = máx.relevância - mín.redundância$
12. **fim-para**
13. Ordene a população de acordo o valor da $função\ de\ avaliação$.
14. Faça o cruzamento
15. Remova atributos repetidos ou com $entropia = 0$
16. **fim-para**

Uma vez que a seleção de atributos foi feita, ordena-se os valores dos atributos contínuos.

O objetivo da fase de avaliação é encontrar o melhor “ponto de corte”, que irá subdividir o domínio do atributo contínuo em duas partes. Uma lista de pontos de corte candidatos $T = \frac{(a_i + a_{i+1})}{2}$ é gerada usando os pontos onde ocorrem as mudanças de classe nos dados previamente ordenados. Usando um par de conjuntos *fuzzy* A_1 e A_2, tais que $A_1(c_i) + A_2(c_i) = 1$, os pontos de corte c_i são fuzificados.

Na fase de divisão, os intervalos são divididos seguindo uma estratégia *top-down*. Para escolher o melhor “ponto de corte” que irá dividir o domínio do atributo em dois, o algoritmo é executado recursivamente para cada subconjunto, até que um critério de parada definido pelos autores seja satisfeito.

Em trabalho publicado em 2014, Jin, Li e Li (JIN; LI; LI, 2014) propuseram um algoritmo de indução de ADF baseado no clássico ID3 (QUINLAN, 1986), o *Fuzzy* ID3 (FID3). Os autores utilizaram a entropia *fuzzy* como métrica para seleção dos atributos durante a indução da árvore. Os

passos do algoritmo proposto são descritos a seguir. **Passo 1:** Para o nó corrente D, com n exemplos:

a) Dado δ um valor no intervalo $(0,1]$ previamente definido, se existe um número $k_0 \in \{1,2,\ldots,n\}$, onde $f_{k_0}(D) \geq \delta$, então D é um nó folha.

b) Se não há mais atributos serem selecionados, então D é um nó folha.

Sendo: $f_k(D) = \frac{M(D \cap c_k)}{M(D)}$, onde, dado um conjunto *fuzzy* $B, M(B) = \sum_{i=1}^{N} B(i)$.

Passo 2: Se D não é folha, então calcula-se a entropia das partições *fuzzy* do atributo A_i e seleciona-se o atributo que apresenta entropia *fuzzy* mínima. A entropia *fuzzy* do atributo A_i é calculada pela Equação 22.

$$FE(D, A_i) = \sum_{j=1}^{k_i} \frac{m_{ij}}{mi} E(D \cap A_{ji}) \quad (22)$$

Onde $E(D \cap A_{ji})$ é definido pela Equação 23.

$$E\left(D \cap A_{ji}\right) = -\sum_{k=1}^{n} \frac{m_{ijk}}{\bar{m}_{ij}} log_2 \left(\frac{m_{ijk}}{\bar{m}_{ij}}\right) \quad (23)$$

Sendo: $m_{ij} = M\left(D \cap A_{ij}\right),\ m_i = \sum_{j=1}^{k_i} m_{ij},\ m_{ijk} = M\left(D \cap A_{ij} \cap c_k\right)$ *e* $\bar{m}_{ij} = \sum_{k=1}^{n} m_{ijk}$.

Passo 3: Repete-se os passos 1 e 2 até que um critério de parada seja atingido.

Passo 4: Converte-se a árvore induzida para um conjunto de regras.

3.3 Considerações Finais

Inicialmente, neste capítulo foram apresentados os principais conceitos e caraterísticas relacionados ao tema da pesquisa desenvolvida, Árvores de

Decisão *Fuzzy*. Ao revisar a literatura, observou-se que os trabalhos desenvolvidos podem ser separados em dois grandes grupos: os que apresentam algoritmos que realizam a fuzificação dos atributos durante a indução da ADF e os que fazem este processo antes da indução. Alguns dos trabalhos presentes na literatura foram apresentados, sendo separados nos dois grupos citados acima, nas seções 3.2.1 e 3.2.2, respectivamente. Dentre estes trabalhos, pode-se também observar algumas outras características que diferem entre os algoritmos. Por exemplo, alguns algoritmos utilizam entropia fuzzy ao invés de entropia clássica como métrica de seleção de atributos. Além disso, alguns algoritmos implementam a saída *fuzzy*, permitindo que o exemplo classificado pertença a mais de uma classe, com diferentes graus de certeza.

O algoritmo proposto neste trabalho se caracteriza como sem pré fuzificação de atributos (seção 3.2.2), ou seja, realiza a fuzificação durante a indução da árvore. Este algoritmo é apresentado no Capítulo 4.

Capítulo 4

ALGORITMO PROPOSTO

4.1 Considerações Iniciais

Como citado anteriormente, uma das contribuições deste trabalho consiste em um algoritmo para a geração de ADFs.

A revisão bibliográfica apresentada no Capítulo 3 - identificou que, os algoritmos propostos na literatura seguem, basicamente, duas abordagens:

i. Algoritmos com pré-fuzificação de atributos e;
ii. Algoritmos sem pré-fuzificação de atributos.

O algoritmo *Simple Test Inspired Fuzzy Induction – STIFI,* proposto neste trabalho, segue a primeira abordagem, embora seu método de fuzificação de atributos contínuos tenha sido inspirado pelo método de particionamento dos atributos contínuos do algoritmo C4.5 (QUINLAN, 1992), descrito na Seção 2.5. A Seção 4.2 apresenta o algoritmo *STIFI* evidenciando os passos seguidos para a sua definição, bem como discutindo os processos envolvidos na indução da ADF.

4.2 O Algoritmo *STIFI*

Inicialmente, o principal objeto de estudo deste trabalho foi o algoritmo FuzzyDT (CINTRA; CAMARGO, 2010), que realiza a fuzificação dos atributos antes de fazer a indução da árvore. Muitos estudos foram desenvolvidos sobre este algoritmo (RIBEIRO; CAMARGO; CINTRA, 2013a, 2014; RIBEIRO et al., 2014), porém a ideia de criar um novo algoritmo que fizesse a fuzificação dos atributos durante a indução da árvore surgiu com a hipótese que este tipo de abordagem poderia apresentar melhores resultados, uma vez que, quando a fuzificação está embutida no processo de indução da árvore, definições como o número de conjuntos *fuzzy* e até mesmo o formato dos mesmos podem ser realizadas de acordo com o subconjunto associado ao nó que está sendo subdividido.

Sendo assim, definiu-se um novo algoritmo de indução de Árvores de Decisão *Fuzzy* cuja fuzificação dos atributos contínuos é realizada durante o processo de indução e inspirada pelo método de particionamento de atributos contínuos, do algoritmo C4.5 (QUINLAN, 1992), apresentado na Seção 2.5.1. Por este motivo, foi denominado *Simple Test Inspired Fuzzy Induction – STIFI.* Também, durante a indução, o *STIFI* é capaz de determinar o melhor número de conjuntos *fuzzy* para representar o domínio do atributo contínuo, sendo que cada atributo é representado por diferentes números de conjuntos, que podem variar de 2 a 7, já que $7(\pm 2)$ é o maior número de entidades conceituais que o ser humano consegue lidar, sem prejuízo à interpretação (MILLER, 1956).

Os principais passos do *STIFI* são similares aos do C4.5, apresentados na Seção 2.5, e são descritos no Algoritmo 7.

Algoritmo 7: Algoritmo *STIFI.*

Entrada: Um conjunto de treinamento $\boldsymbol{D} = (\boldsymbol{x_i}, y_i)$
Saída: Árvore de Decisão *Fuzzy*

1. **se** $Critério\ de\ parada(\boldsymbol{D}) = Verdadeiro:$
2. ***retorna*** *um nó folha rotulado com a classe mais frequente em* $\boldsymbol{D}$*;*
3. **senão**
4. *Seleciona o atributo* A_k *que maximiza o ganho de informação em* $\boldsymbol{D}$;
5. **para cada** *subconjunto* $\boldsymbol{D}_i$ *definido com base nos valores do atributo* A_k
6. **faça**:
7. $\boldsymbol{D} = \boldsymbol{D}_i$;
8. *Constrói a subárvore induzida a partir de* $\boldsymbol{D}$*, voltando ao passo 1.*
9. **fim-para**
10. **fim-se**

Como descrito no Algoritmo 7, para o *STIFI* o critério de seleção dos atributos é baseado no ganho de informação. Para controlar o crescimento da árvore, um dos $critérios\ de\ parada$ (Linha 1 do Algoritmo 7) implementados considera o valor do parâmetro *numMinToSplit,* que pode ser definido manual ou automaticamente, pelo algoritmo. Ele é responsável por finalizar o crescimento da árvore, quando o número de instâncias no nó é menor que o valor definido para *numMinToSplit*. Quando o modo automático é escolhido para definição, o valor do parâmetro é estabelecido como sendo 5% do tamanho do conjunto de treinamento. Para a definição deste valor (5%) alguns testes foram realizados, incluindo *numMinToSplit* igual a 10 e 2% do tamanho do conjunto. A exemplo do valor padrão adotado pelo C4.5, valores inteiros também foram testados: dois e dez. Os resultados destes testes podem ser vistos no Apêndice A. O outro $critério\ de\ parada$ ocorre quando todos os exemplos no nó corrente pertencem à mesma classe.

A principal contribuição do *STIFI* consiste em seu método de fuzificação dos atributos contínuos durante a seleção do atributo para o nó que está sendo dividido. Este procedimento foi inspirado no método de particionamento dos atributos contínuos do C4.5 e é descrito na Seção 4.2.1.

4.2.1 Seleção de atributos do *STIFI*

Os passos realizados pelo algoritmo *STIFI* para selecionar o atributo que maximiza o valor de ganho de informação em um nó são descritos no Algoritmo 8.

Algoritmo 8: Algoritmo de seleção de atributos do *STIFI.*

Entrada: Um conjunto de treinamento $\boldsymbol{D} = (\boldsymbol{x_i}, y_i)$
Saída: Atributo de divisão

1. **/*Função Escolhe Atributo($\boldsymbol{D}$)*/**
2. **para cada** atributo A_k de $\boldsymbol{D}$ **faça**:
3.
4. **se** A_ké discreto:
5. *Calcula o ganho de informação para o atributo* A_k;
6.
7. **senão**:
8. *Ordena o conjunto* $\boldsymbol{D}$ *de acordo com o valor do atributo* A_k;
9. **para** *número de divisões de 1 a 6* **faça**:
10. *Define os pontos de divisão candidatos (cada possível*
11. *valor do atributo numérico);*
12.
13. *Define o intervalo entre os pontos de divisão;*
14.
15. *Define os pontos de divisão;*
16.
17. *Define as funções de pertinência fuzzy:*
18. *Triangulares para conjuntos internos;*
19. *Trapezoidais do tipo RS ou LS nas extremidades do domínio;*
20.
21. *Fuzifica os valores do atributo de acordo com as funções*
22. *definidas na linha 13 do algoritmo;*
23.
24. *Calcula o ganho de informação do atributo para*
25. *fuzificação realizada na linha 16;*
26. **fim-para**
27. *Seleciona o número de divisões que maximiza o ganho de*
28. *informação e define estes valores para o atributo* A_k*;*
29. **fim-se**
30. **retorna** *o atributo que obteve maior ganho de informação.*

O método apresentado no Algoritmo 8 tem como objetivo selecionar o atributo que será utilizado na divisão do nó. Para isso, calcula-se o ganho de informação de cada atributo e seleciona-se aquele que maximiza este valor. No caso de atributos discretos, o ganho de informação é calculado normalmente, conforme apresentado na Seção 2.5. No entanto, quando o atributo é contínuo,

é necessário que o método faça a fuzificação dos valores antes de calcular o ganho de informação. Para isso, primeiro deve-se determinar o número d de conjuntos *fuzzy* que será utilizado para representar o domínio do atributo contínuo. Na linha 9 do Algoritmo 8 observa-se que são testados números de divisões de um até seis, ou seja, o domínio do atributo contínuo poderá ser particionado em n conjuntos *fuzzy*, onde $n \in \{2, 3, 4, 5, 6, 7\}$. O valor de d que maximiza o ganho de informação é selecionado como o número de divisões que será utilizado para a fuzificação do atributo testado. Para d divisões, $n = d + 1$ conjuntos *fuzzy* são construídos a fim de representar o domínio do atributo contínuo. As funções de pertinência que representam este conjuntos *fuzzy* são definidas no formato trapezoidal (*Right Shoulder* ou *Left Shoulder*) para os conjuntos que representam as extremidades do domínio do atributo e no formato triangular para as funções que representam os valores intermediários. Os parâmetros da funções de pertinência são calculados pelos pontos médios dos valores de divisão definidos ou pelos valores das extremidades do domínio, conforme mostra a Figura 20.

Para exemplificar como são gerados os conjuntos *fuzzy* associados aos atributos contínuos, considere o conjunto apresentado na Tabela 1 e o atributo Temperatura a ser fuzificado. Os pontos de divisão candidatos são considerados cada valor possível do atributo. Neste caso, para o atributo Temperatura, os pontos de divisão candidatos são exibidos na Figura 18.

75	80	85	72	69	72	83	64	81	71	65	75	68	70

Figura 18: Pontos de divisão do atributo Temperatura.

O próximo passo do algoritmo consiste em criar um vetor com estes valores ordenados, sem repetição, conforme o vetor V exibido na Figura 19.

$V =$

64	65	68	69	70	71	72	75	80	81	83	85

Figura 19: Valores da Figura 18 ordenados e sem repetição.

Supondo que o número de divisões a ser testado é $d = 4$, define-se o intervalo I entre os pontos de divisão conforme a Equação 24.

$$I = \frac{|V|}{d} \tag{24}$$

Onde $|V|$ denota o tamanho do vetor V, ou seja $|V| = 12$. Portanto, para este exemplo, $I = \frac{|V|}{d} = \frac{12}{4} = 3$.

Uma vez que o intervalo $I = 3$ entre os pontos de divisão foi definido, calcula-se os pontos de divisão $p_i \in P, i = 0, \dots, I$ da seguinte forma:

1. $p_0 =$ valor mínimo do domínio do atributo
2. para i de 1 até $d - 1$:
   $$p_i = V[i * I]$$
3. $p_I =$ valor máximo do domínio do atributo

Portanto, para este exemplo:

- $p_0 = 64$
- $p_1 = 69$
- $p_2 = 72$
- $p_3 = 81$
- $p_4 = 85$

Os conjuntos *fuzzy* são definidos conforme indicado na Figura 20, sendo que os conjuntos das extremidades do domínio são representados por funções de pertinência trapezoidais (especificamente, funções do tipo "*Left Shoulder*" e "*Right Shoulde*r"), enquanto os conjuntos do interior do intervalo do domínio são definidos por funções triangulares.

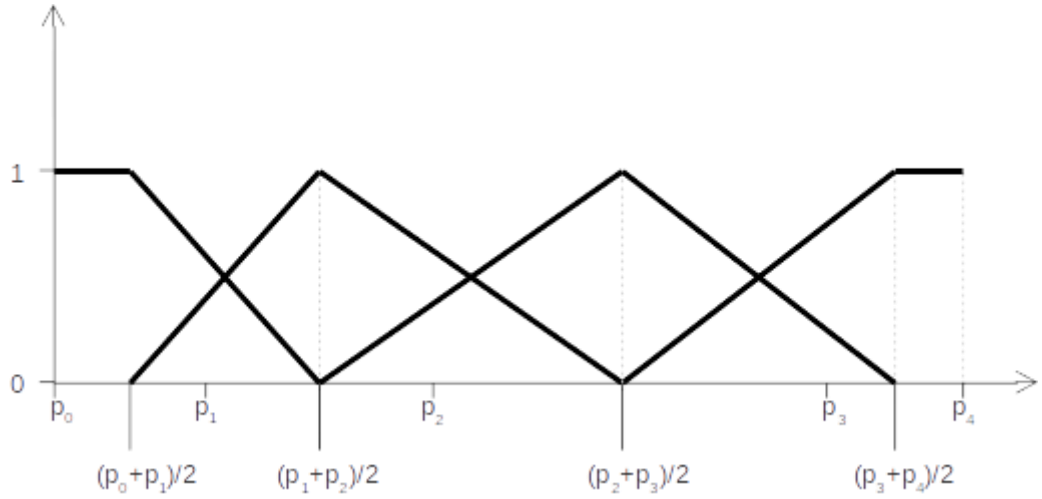

Figura 20: Definição dos conjuntos *fuzzy* para atributos contínuos.

Uma das características dos conjuntos *fuzzy* criados a partir deste método é que, se houver uma região mais densa no domínio do atributo, eles tendem a se concentrar nessas regiões. Isto ocorre devido ao fato do intervalo I ser definido com base no tamanho do vetor V e não no tamanho do domínio do atributo, sendo que os pontos p_i são definidos de acordo com a posição em que aparecem no vetor V. Esta característica pode ser especialmente interessante em casos no qual o domínio do atributo tem grande dimensão, porém a maior parte dos valores das instâncias estão concentradas em um subintervalo do domínio.

É importante observar que os conjuntos *fuzzy* são definidos a cada vez que um atributo é selecionado. Sendo assim, se o mesmo atributo for selecionado em ramos diferentes da árvore, a fuzificação será também diferente, já que subconjuntos distintos estarão sendo considerados em cada ramo. Esta característica resultará em regras com fuzificações diferentes para a mesma variável.

Uma vez que os conjuntos *fuzzy* foram definidos, os valores do atributo são fuzificados de acordo com estas funções e, posteriormente, calcula-se o ganho de informação para o atributo. Este valor será considerado para a seleção do atributo a ser associado ao nó da árvore que será dividido.

4.3 Experimentos

Os experimentos realizados com o algortimo *STIFI* incluíram 21 conjuntos de dados disponíveis no repositório UCI (LICHMAN, 2013) e são descritos na Tabela 7, onde são exibidas as características dos conjuntos utilizados para os experimentos, que incluem: Nome, Tamanho (Número de instâncias), Número de Atributos, Número de Classes e Porcentagem dos dados que pertencem à Classe Majoritária.

Tabela 7: Características dos conjuntos de dados utilizados para realização dos exeprimentos.

Nome do Conjunto	Num. Instâncias	Núm. Atributos	Núm. Classes	% Classe Majoritária
au8_500	500	1001	10	17,2
balance-scale	625	5	3	46,0
car	1728	7	4	70,0
credit	1000	21	2	70,0
dermatology	366	35	6	30,6
ecoli	336	8	8	42,6
glass	214	10	6	35,5
haberman	306	4	2	73,5
ionosphere	351	35	2	64,1
iris	150	5	3	33,3
page_blocks	5473	11	5	89,8
pima-indians-diabetes	768	9	2	65,1
segment	2310	20	7	14,3
transfusion	748	5	2	76,2
vehicle	846	19	4	25,8
waveform	5000	22	3	33,9
weatherN	14	5	2	64,3
weatherNum	14	5	2	64,3

wine	178	14	3	39,8
yeast	1484	9	10	31,2
zoo	101	17	7	40,6

Os experimentos com o algoritmo *STIFI* foram realizados considerando o parâmetro *numMinToSplit* calculado automaticamente, ou seja, este valor foi definido como sendo igual a 5% do tamanho do conjunto de treinamento apresentado ao algoritmo. Para a classificação das instâncias, foi utilizado o método de raciocínio *fuzzy* clássico, apresentado no Algoritmo 2, com a *t-norma mínimo*.

Para comparação dos resultados, foram executados algoritmos que induzem três tipos de modelos de classificação: FuzzyDT (CINTRA; CAMARGO, 2010) que, conforme foi visto na Seção 3.2.1, induz ADFs a partir de um conjunto de treinamento previamente fuzificado, FURIA (HÜHN; HÜLLERMEIER, 2009), que induz SCFBRs e C4.5 (QUINLAN, 1992), que induz ADs.

Para execução do C4.5 e FURIA foram utilizadas as implementações destes algoritmos disponibilizadas no WEKA (HALL et al., 2009) e para a FuzzyDT, uma implementação em Java disponibilizada em (CINTRA, 2012). O algoritmo *STIFI* foi implementado em Python.

Para todos os experimentos, *10-fold CV* foi utilizado como forma de validação.

Os resultados obtidos são exibidos na Tabela 8, onde, para cada um dos algoritmos pode-se observar a Taxa de Classificação Correta (TCC) e o número de regras, bem como o desvio padrão destes valores, apresentado por cada um dos modelos induzidos.

Tabela 8: Resultados dos Experimentos.

	STIFI				FURIA				C4.5				FuzzyDT			
	TCC	**DP**	**Regras**	**DP**	**TCC**	**DP**	**Regras**	**DP**	**TCC**	**DP**	**Regras**	**DP**	**TCC**	**DP**	**Regras**	**DP**
au8_500	12,96	4,6	46,60	8,2	15,00	3,0	55,30	5,9	10,80	4,9	124,60	5,5	12,60	4,22	318,50	13,6
balance-scale	45,96	12,4	25,00	0,0	79,54	3,7	37,60	5,4	64,48	4,3	34,60	3,9	64,48	4,33	34,60	3,9
car	78,33	6,8	18,50	0,5	93,81	1,4	79,30	4,7	92,36	2,1	120,40	4,4	92,36	2,10	120,40	4,4
credit	71,70	3,5	66,00	4,4	72,50	4,1	9,20	1,8	70,50	3,6	83,10	23,3	71,00	5,64	120,10	15,5
dermatology	91,14	5,6	36,50	2,8	95,35	3,7	21,40	1,7	94,00	4,2	26,80	4,1	93,58	2,95	25,60	2,8
ecoli	77,00	5,2	45,40	3,1	83,63	5,3	16,00	2,0	84,23	7,5	18,30	1,9	62,56	5,36	54,20	5,7
glass	58,36	11,2	58,70	8,9	71,93	9,0	14,90	2,1	65,87	8,9	24,00	1,9	61,17	7,11	28,20	4,5
haberman	67,66	2,9	58,10	6,2	72,19	5,1	3,30	0,5	71,86	4,1	2,60	1,3	72,88	2,05	1,80	2,5
ionosphere	84,74	7,6	44,20	4,9	91,17	2,8	11,60	2,8	91,46	3,3	14,20	1,9	89,18	5,82	23,80	6,0
iris	94,00	4,7	17,80	4,0	94,67	5,3	4,50	0,7	96,00	5,6	4,70	0,5	92,67	7,98	5,00	0,0
page_blocks	93,11	1,6	56,60	10,3	97,39	0,5	25,50	3,0	96,88	0,4	45,90	5,9	92,73	0,95	42,60	6,6
pima-indians-diabetes	72,13	5,5	62,80	5,0	75,66	4,3	8,70	3,4	73,83	5,7	19,20	6,5	75,52	4,28	29,00	11,9
segment	87,62	1,8	40,60	2,5	97,10	1,1	31,20	2,0	96,93	1,1	41,10	2,6	92,12	0,88	137,00	11,3
transfusion	73,27	10,0	64,50	7,8	78,48	2,7	5,00	1,1	77,81	3,8	6,00	1,7	75,94	1,01	1,40	1,3
vehicle	63,38	6,1	65,10	7,5	70,57	2,5	24,40	3,1	72,47	5,9	68,50	10,7	68,68	4,53	187,00	20,5
waveform	74,32	1,3	67,30	3,9	82,94	1,8	80,70	5,1	75,94	1,4	271,40	17,0	76,96	1,28	722,60	38,8
weatherN	80,00	33,2	5,50	0,9	70,00	35,0	4,10	0,7	55,00	43,8	4,10	1,0	55,00	43,78	4,10	1,0
weatherNum	45,00	41,5	10,90	2,8	45,00	43,8	3,50	0,7	70,00	42,2	4,60	0,7	55,00	43,78	3,60	3,2
yeast	49,11	3,5	55,60	4,7	56,87	3,7	25,30	3,4	55,99	4,8	155,60	12,6	50,47	3,25	114,60	14,3
zoo	96,92	4,8	13,50	0,5	94,18	6,6	7,50	1,0	92,18	8,9	10,70	3,0	92,18	8,94	10,70	3,0
Média	70,84	8,7	42,96	4,4	76,90	7,3	23,45	2,5	75,43	8,3	54,02	5,5	72,35	8,01	99,24	8,5

A Tabela 9 mostra o número de antecedentes médio gerado pelas regras do *STIFI*.

Tabela 9: Número de antecedentes médio.

	Antecedentes
au8_500	2,37
balance-scale	2,00
car	2,78
credit	2,87
dermatology	3,68
ecoli	2,56
glass	2,68
haberman	2,94
ionosphere	2,39
iris	1,87
page_blocks	2,84
pima-indians-diabetes	2,98
segment	2,57
transfusion	2,92
vehicle	2,71
waveform	2,68
weatherN	1,92
weatherNum	1,74
yeast	2,96
zoo	2,40
Média	2,59

4.4 Análise dos Resultados

Observou-se que, quando comparado com o FURIA, em média, o *STIFI* obteve pior desempenho tanto com relação ao número de regras quanto com relação à TCC, apresentando melhores resultados somente para dois conjuntos com relação à TCC e para quatro com relação ao número de regras. Assim como o *STIFI*, as regras geradas pelo FURIA apresentam problemas com relação à interpretabilidade, gerando uma fuzificação diferente para o mesmo atributo quando o mesmo ocorre em diferentes regras. Com relação ao C4.5, em média, o *STIFI* obteve resultado inferior com relação à TCC. Sobre o número de regras, no entanto, a média apresentada pelo *STIFI* foi melhor. O

algoritmo proposto obteve melhores resultados para três dos vinte conjuntos com relação à TCC e menor número de regras para oito. Com relação ao FuzzyDT, em média, o *STIFI* obteve melhor resultado quanto ao número de regras, enquanto a TCC média do algoritmo ficou 1,51 pontos percentuais abaixo da média apresentada pelo FuzzyDT. O *STIFI* apresentou menor número de regras para nove dos vinte conjuntos testados enquanto a TCC foi melhor para seis dos testes realizados.

Sobre a interpretabilidade das regras geradas pelo *STIFI*, embora o algoritmo resulte em regras com fuzificações diferentes para a mesma variável, o que prejudica a interpretabilidade semântica das regras, conforme descrito em (MILLER, 1956), 7($\pm$2) é o número de entidades conceituais que o ser humano consegue interpretar. Conforme pode ser observado na Tabela 9, este número de antecedentes não é ultrapassado pelas regras geradas pelo STIFI, preservando a interpretabilidade com relação à complexidade das regras.

Algumas ideias para a melhoria do algoritmo já estão sendo investigadas para serem implementadas em uma próxima versão, como discutido a seguir.

Com relação aos resultados obtidos sobre a TCC, acredita-se que algumas modificações podem fazer com que estes valores sejam mais expressivos. Algumas ideias para atingir melhores valores de TCC com o algoritmo *STIFI* são:

- Implementação da Razão de Ganho *Fuzzy* no método de seleção de atributos;
- Desenvolvimento de um critério de parada baseado no ganho obtido com a divisão do nó;
- Ajuste dos parâmetros das funções de pertinência por meio de um algoritmo genético.

Já com relação aos números de regras gerados pelo *STIFI*, a implementação do método de pós-poda do algoritmo C4.5 é uma das melhorias que podem ser consideradas, uma vez que experimentos em trabalho realizado em 2013 mostraram que este método traz bons resultados (RIBEIRO;

CAMARGO; CINTRA, 2013b). Outra questão que deverá ser investigada é sobre como a fuzificação dos atributos contínuos pode ser modificada ou tratada para que a interpretabilidade não seja prejudicada, uma vez o algoritmo gera fuzificações diferentes para a mesma variável quando ela ocorre em diferentes ramos da árvore, refletindo na interpretabilidade das regras.

4.5 Considerações Finais

Neste capítulo foi apresentado o algoritmo para indução de árvores de decisão *fuzzy STIFI*, proposto neste trabalho. Um novo método de seleção de atributos, que realiza a fuzificação dos atributos contínuos durante a indução da ADF foi implementado. Tanto o número de conjuntos *fuzzy* quanto os parâmetros das funções de pertinência são definidos no momento da indução, de forma automática, levando em consideração o subconjunto do domínio do atributo que está sendo representado no nó.

Os experimentos com o novo algoritmo resultaram em valores promissores que podem ser melhorados, tanto com relação à TCC quanto ao número de regras com a implementação de algumas modificações, apresentadas na Seção 4.4.

O Capítulo 5 apresenta as conclusões e considerações finais deste trabalho.

Capítulo 5

CONCLUSÃO

Este trabalho propôs o algoritmo *STIFI*, que induz uma árvore de decisão *fuzzy* a partir de um conjunto de treinamento que pode conter atributos contínuos ou discretos. Ele seleciona, recursivamente, o atributo que maximiza a medida do ganho de informação no nó, subdividindo-o em subconjuntos de acordo com os valores do atributo selecionado.

No método de seleção de atributos proposto, a fuzificação dos atributos contínuos é realizada durante a indução da ADF, bem como a definição do número e parâmetros dos conjuntos *fuzzy* utilizados para a fuzificação.

Uma vez induzida a ADF, o método de raciocínio *fuzzy* clássico foi aplicado sobre exemplos de conjuntos de teste, utilizando-se a *t-norma mínimo*, a fim de verificar a acurácia do modelo induzido.

Os experimentos realizados incluíram 20 conjuntos do repositório de dados UCI (LICHMAN, 2013) e três sistemas de classificação, cada um seguindo uma abordagem diferente: O FuzzyDT, que induz árvores de decisão fuzzy realizando a fuzificação dos atributos contínuos antes da indução da ADF, o FURIA, que induz Sistemas de Classificação Fuzzy Baseados em Regras e o C4.5, que induz Árvores de Decisão Indutivas. Todos os experimentos foram validados com a utilização do *10-fold CV*.

Os resultados obtidos evidenciaram que, embora os resultados apresentados pelo *STIFI* sejam satisfatórios, algumas melhorias devem ser realizadas a fim de induzir um modelo mais competitivo com as outras abordagens em termos de Taxa de Classificação Correta e número de regras induzidas.

Observou-se a necessidade de aprofundar as investigações e, possivelmente, implementar uma nova versão do algoritmo com alguns pontos a serem considerados, entre eles:

1. Implementação do método de seleção de tributos com base em uma métrica *fuzzy*, por exemplo, a Razão de Ganho *Fuzzy* ;

2. Desenvolvimento de um critério de parada baseado no ganho obtido com a divisão do nó;

3. Ajuste dos parâmetros das funções de pertinência por meio de um algoritmo genético;

4. Implementação de um método de pós poda para redução do número de regras produzidas pelo algoritmo;

5. Modificação ou tratamento do método de fuzificação dos atributos contínuos para que a interpretabilidade não seja prejudicada.

Acredita-se que, com algumas destas modificações, melhores resultados em termos de TCC e número de regras podem ser atingidos pelo *STIFI*

Dos estudos realizados durante este trabalho, os resultados intermediários obtidos foram apresentados em três artigos.

O primeiro, dedicado à análise de diferentes estratégias de poda de árvores de decisão (RIBEIRO; CAMARGO; CINTRA, 2013b), evidenciou que a pós poda adotada pelo algoritmo C4.5 apresentou melhores resultados quando aplicada ao algoritmo FuzzyDT, enquanto as estratégias de pré poda testadas fizeram com que o algoritmo não fosse capaz de aprender para 6 dos 15 casos analisados, apresentando Taxa de Classificação Correta menor que a porcentagem da classe majoritária presente nos exemplos.

O segundo trabalho analisou o comportamento da FuzzyDT em um problema real de classificação de solos e foi publicado em 2014 (RIBEIRO et al., 2014). Sua performance foi comparada aos algoritmos FURIA e C4.5 e,

embora tenha obtido melhor Taxa de Classificação Correta, o número de regras geradas pelo algoritmo foi significativamente maior que o gerado pelos outros modelos, confirmando a necessidade de investigação de métodos de poda capazes de reduzir o número de regras sem prejudicar a TCC.

O terceiro trabalho visou o estudo de métodos de seleção de atributos para a utilização em sistemas fuzzy genéticos (RIBEIRO; CAMARGO; CINTRA, 2014). Nele, o método de seleção de atributos *fuzzy*, da FuzzyDT foi comparado a três métodos *crisp*, os filtros CFS e Consistency e o método embutido do C4.5. Os resultados obtidos pelo CFS foram considerados os mais promissores.

Apêndice A

EXPERIMENTOS COM O ALGORITMO *STIFI*

Neste apêndice, são apresentados os resultados obtidos com o algoritmo *STIFI* para diferentes valores para o parâmetro *numMinToSplit*. Após estes testes, o valor padrão definido pelo algoritmo foi *numMinToSplit = 5%* do tamanho do conjunto de dados. Estes resultados foram apresentados na Seção 4.3. As execuções foram feitas utilizando-se *10-fold CV.* Nas Tabela 10 e 12 são apresentados a Taxa de Classificação Correta, o número de regras geradas, bem como o desvio padrão destes valores, além do número de antecedentes médio que compõe as regras. Os resultados da Tabela 11 consideram *numMinToSplit* como sendo valores fixos, dois e dez, enquanto os resultados da Tabela 12 são quando diferentes porcentagens do tamanho do conjunto de dados definem o valor de *numMinToSplit,* no caso, 2% e 10%.

Tabela 10: Resultados para *numMinToSplit* igual a dois e dez.

	STIFI - numMinToSplit = 2					**STIFI - numMinToSplit = 10**				
	TCC	**DP**	**Regras**	**DP**	**Antecedentes**	**TCC**	**DP**	**Regras**	**DP**	**Antecedentes**
au8_500	**11,9**	**3,45**	**407,9**	**10,15**	**3,67**	**11,9**	**4,42**	**170,9**	**9,05**	**3,01**
balance-scale	43,37	6,72	381,40	15,44	3,85	38,27	10,36	121,00	4,38	2,99
car	81,21	5,20	270,50	5,90	5,47	80,70	5,29	141,20	6,01	4,72
credit	67,90	4,06	543,10	14,82	4,40	68,20	3,82	267,40	16,72	3,79
dermatology	88,32	6,50	63,40	8,26	4,49	89,45	6,76	43,80	5,23	3,82
ecoli	72,53	4,11	139,00	6,60	3,38	73,67	5,92	78,20	9,40	2,87
glass	64,82	14,11	110,30	6,10	3,16	58,36	11,15	58,70	8,92	2,69
haberman	68,33	3,64	73,30	7,35	3,12	68,33	3,64	73,30	7,35	3,12
ionosphere	82,70	6,36	74,80	6,49	2,92	83,89	7,75	52,30	6,65	2,57
iris	95,33	5,21	22,60	4,00	2,22	94,00	4,67	17,20	3,76	1,84
page_blocks	94,95	1,98	326,50	22,09	3,84	94,95	1,98	326,50	22,09	3,84

pima-indians-diabetes	67,19	3,83	357,10	16,83	4,11	67,31	3,98	197,20	15,00	3,58
segment	94,81	1,61	221,70	11,18	4,07	94,94	1,33	169,30	10,42	3,71
transfusion	73,27	9,95	64,50	7,76	2,92	73,27	9,95	64,50	7,76	2,92
vehicle	64,57	4,31	362,90	10,40	3,93	64,81	3,81	207,10	20,10	3,43
waveform	70,64	1,74	1767,70	48,43	5,08	71,82	1,23	1002,40	15,72	4,61
weatherN	80,00	33,17	5,50	0,92	1,91	50,00	38,73	2,80	0,40	1,00
weatherNum	45,00	41,53	10,90	2,77	1,74	50,00	38,73	5,70	2,24	1,00
yeast	42,02	3,17	1034,70	20,75	4,91	44,86	3,58	486,40	23,44	4,24
zoo	96,92	4,80	13,90	0,30	2,48	88,36	4,99	10,21	0,40	1,92
Média	70,29		312,59		3,58	68,35		174,81		3,08

Tabela 11: Resultados para *numMinToSplit* igual a 2% e 10% do tamanho do conjunto.

	STIFI - numMinToSplit = 2%					**STIFI - numMinToSplit = 10%**				
	TCC	**DP**	**Regras**	**DP**	**Antecedentes**	**TCC**	**DP**	**Regras**	**DP**	**Antecedentes**
au8_500	11,6	3,65	141,6	9,93	2,92	14,8	4,89	25,8	2,68	1,98
balance-scale	39,75	11,62	119,00	4,82	2,97	45,96	12,41	25,00	0,00	2,00
car	77,51	5,86	51,60	2,73	3,78	76,44	9,29	15,10	4,96	2,53
credit	70,80	3,51	115,40	9,23	3,28	71,40	3,14	31,10	3,62	2,39
dermatology	88,34	7,09	45,70	4,34	3,90	91,41	5,12	28,20	3,92	3,52
ecoli	73,59	5,93	94,30	6,07	3,00	77,57	4,63	26,40	2,29	2,12
glass	58,62	10,11	91,90	7,43	2,97	61,71	13,90	26,40	4,32	2,22
haberman	68,33	5,96	104,20	7,55	3,34	69,02	4,28	31,00	6,00	2,51
ionosphere	83,00	6,88	58,80	5,33	2,68	85,88	7,26	27,80	4,68	2,09
iris	95,33	5,20	22,10	4,08	2,16	94,00	4,67	15,80	2,64	1,73
page_blocks	93,81	2,37	81,30	11,65	3,05	91,15	2,97	36,20	5,08	2,62
pima-indians-diabetes	69,00	5,99	107,40	8,65	3,25	73,31	6,24	29,10	3,36	2,48
segment	91,13	2,35	71,00	4,40	2,90	81,17	2,35	19,50	1,36	1,86
transfusion	73,27	9,95	64,50	7,76	2,92	74,08	5,36	33,60	4,90	2,38
vehicle	64,46	4,41	122,60	10,00	3,06	59,92	5,39	35,70	2,93	2,19
waveform	74,50	1,20	96,80	10,90	3,01	68,06	2,07	22,30	4,71	1,99
weatherN	80,00	33,17	5,50	0,92	1,92	80,00	33,16	5,50	0,92	1,90
weatherNum	45,00	41,53	10,90	2,77	1,74	45,00	41,53	10,90	2,77	1,74
yeast	49,08	3,23	126,80	6,10	3,49	49,12	2,75	35,90	3,24	2,50
zoo	96,92	4,80	13,90	0,30	2,48	88,36	4,99	10,20	0,40	1,92
Média	70,20		77,27		2,94	69,92		24,58		2,23

REFERÊNCIAS

AFSARI, F. et al. Interpretability-based Fuzzy Decision Tree Classifier a Hybrid of the Subtractive Clustering and the Multi-objective Evolutionary Algorithm. Soft Comput, v. 17, n. 9, p. 1673–1686, 2013.

BAJAJ, S. B.; KUBBA, A. FHSM: Fuzzy Heterogeneous Split Measure Algorithm for Decision Trees, IEEE International Advance Computing Conference (IACC) , p. 574-578, 2014.

BOYEN, X.; WEHENKEL, L. Automatic Induction of Fuzzy Decision Trees and its Application to Power System Security Assessment. Fuzzy Sets and Systems, v. 102, n. 1, p. 3–19, 1999.

BREIMAN, L. et al. Classification and Regression Trees. Belmont, California: Wadsworth International Group, 1984.

CHANDRA, B.; BABU KUPPILI, V. Heterogeneous Node Split Measure for Decision Tree Construction, SMC, p. 872-877, 2011.

CHANDRA, B.; VARGHESE, P. P. Fuzzifying Gini Index Based Decision Trees. Expert Syst. Appl, v. 36, n. 4, p. 8549–8559, 2009.

CHI, Z.; YAN, H. Feature Evaluation and Selection Based on an Entropy Measure with Data Clustering. Optical Engineering, v. 34, n. 12, p. 3514–3519, 1995.

CHIU, S. L. Fuzzy Model Identification Based on Cluster Estimation. Journal of Intelligent and Fuzzy Systems, v. 2, n. 3, p. 267–278, 1994.

CINTRA, M. E. Geração Genética de Regras Fuzzy com Pré-seleção de Regras Candidatas. Encontro Nacional de Inteligência Artificial--ENIA. Rio de Janeiro, p. 1341–1350, 2007.

CINTRA, M. E. FuzzyDT. Disponível em: <http://dl.dropbox.com/u/16102646/FuzzyDT.zip>. Acesso em: 7 fev. 2016.

CINTRA, M. E.; CAMARGO, H. A. Feature Subset Selection for Fuzzy Classification Methods. Information Processing and Management of Uncertainty in Knowledge-Based Systems - IPMU, v. 80, p. 318–327, 2010.

COLLINS AMERICAN ENGLISH DICTIONARY. Definition of “learn” | Collins American English Dictionary. Disponível em: <http://www.collinsdictionary.com/dictionary/american/cluster?showCooki

ePolicy=true>. Acesso em: 3 fev. 2016.

COPPIN, B. Inteligência Artificial, LTC, 2010.

EVANS, L.; LOHSE, N.; SUMMERS, M. A fuzzy-decision-tree Approach for Manufacturing Technology Selection Exploiting Experience-based Information. Expert Systems with Applications, v. 40, n. 16, p. 6412–6426, nov. 2013.

FACELI, K. Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina. Grupo Gen-LTC, 2011.

FENG, X.; LIU, X.; WANG, X. A New Fuzzy Decision Tree Based on Euclidean Fuzzy Measure. Journal of Computational Information Systems, v. 10, n. 3, p. 1037–1044, 2014.

FREUND, Y.; MASON, L. The Alternating Decision Tree Learning Algorithm. Proceedings of the Sixteenth International Conference on Machine Learning - ICML, p. 124-133, 1999.

HALL, M. A. et al. The WEKA Data Mining software: an update. SIGKDD Explorations, v. 11, n. 1, p. 10–18, 2009.

HÜHN, J. C.; HÜLLERMEIER, E. FURIA: an Algorithm for Unordered Fuzzy Rule Induction. Data Min. Knowl. Discov, v. 19, n. 3, p. 293–319, 2009.

JANIKOW, C. Z. Fuzzy Decision Trees: Issues and Methods. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), v. 28, n. 1, p. 1–14, 1998.

JANIKOW, C. Z.; FAJFER, M. Fuzzy partitioning with FID3.1. 18th International Conference of the North American Fuzzy Information Processing Society - NAFIPS, p. 467–471, 1999.

JANIKOW, C. Z.; KAWA, K. Fuzzy Decision Tree FID. NAFIPS 2005 - 2005 Annual Meeting of the North American Fuzzy Information Processing Society, p. 379–384, 2005.

JIN, C.; LI, F.; LI, Y. A Generalized Fuzzy ID3 Algorithm Using Generalized Information Entropy. Knowledge-Based Systems, v. 64, p. 13–21, jul. 2014.

KLIR, G. J.; YUAN, B. Fuzzy Sets and Fuzzy Logic - Theory and Applications. Prentice Hall, 1995.

LEE, H. M. et al. An Efficient Fuzzy Classifier with Feature Selection Based on Fuzzy Entropy. IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society, v. 31, n. 3, p. 426–432, 2001.

LEVASHENKO, V.; ZAITSEVA, E.; PUURONEN, S. Fuzzy Classifier Based on Fuzzy Decision Tree. EUROCON 2007 - The International Conference on Computer as a Tool, p. 823–827, 2007.

LICHMAN, M. UCI Machine Learning Repository, 2013.

MEHTA, M.; AGRAWAL, R.; RISSANEN, J. SLIQ: A Fast Scalable Classifier for Data Mining. Lecture Notes in Computer Science, v. 1057, 1996.

MILLER, G. A. The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information. Psychological review, v. 101, n. 2, p. 343–352, 1956.

MITCHELL, T. M. Machine Learning. McGraw-Hill, 1997.

NICOLETTI, M. DO C.; CAMARGO, H. DE A. Fundamentos da Teoria de Conjuntos Fuzzy. São Carlos: EdUFSCar, 2004.

OLARU, C.; WEHENKEL, L. A Complete Fuzzy Decision Tree Technique. Fuzzy Sets and Systems, v. 138, n. 2, p. 221–254, 2003.

PEDRYCZ, W.; GOMIDE, F. An Introduction to Fuzzy Sets : Analysis and Design. Cambridge, EUA : Massachusetts Institute of Technology, , 1998.

PEDRYCZ, W.; SOSNOWSKI, Z. A. C-fuzzy Decision Trees. IEEE Trans. Systems, Man and Cybernetics, v. 35, n. 4, p. 498–511, 2005.

QUINLAN, J. R. Induction of Decision Trees. Machine Learning, v. 1, n. 1, p. 81–106, 1986.

QUINLAN, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1992.

REZENDE, S. O. Sistemas Inteligentes: Fundamentos e Aplicações. Manole, 2003.

RIBEIRO, M. V et al. Applying a Fuzzy Decision Tree Approach to Soil Classification. Information Processing and Management of Uncertainty in Knowledge-Based Systems - IPMU, 2014.

RIBEIRO, M. V; CAMARGO, H. A.; CINTRA, M. E. A Comparative Analysis of Pruning Strategies for Fuzzy Decision Trees. IFSA/NAFIPS, 2013

RIBEIRO, M. V; CAMARGO, H. A.; CINTRA, M. E. On Feature Subset Selection for Fuzzy and Classic Machine Learning Classification Methods. KDMile - PROCEEDINGS OF THE II SYMPOSIUM ON KNOWLEDGE DISCOVERY, MINING AND LEARNING, v. 1, p. 1–6, 2014.

RUSSELL, S. J.; NORVIG, P. Artificial Intelligence: A Modern Approach. Prentice Hall, 1995.

SHANNON, C. E. A Mathematical Theory of Communication. The Bell System Technical Journal, v. 27, p. 379–423, 1948.

THEODORIDIS, S.; KOUTROUMBAS, K. Pattern Recognition. Academic Press, 1999.

UMANO, M. et al. Fuzzy Decision Trees by Fuzzy ID3 Algorithm and its Application to Diagnosis Systems. Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference, n. 1, p. 2113–2118, 1994.

VARMA, K. V. S. R. P. et al. A Computational Intelligence Approach for a Better Diagnosis of Diabetic Patients. Computers & Electrical Engineering, v. 40, n. 5, p. 1758–1765, jul. 2014.

WITTEN, I. H.; FRANK, E.; HALL, M. A. Data Mining: Practical Machine Learning Tools and Techniques, Third Edition. 1999.

YUAN, B.; KLIR, G. J.; SWAN-STONE, J. F. Evolutionary Fuzzy C-Means Clustering Algorithm. Proc. 4th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'95), p. 2221–2226, 1995.

ZADEH, L. Fuzzy Sets. Information and Control, v. 8, p. 338–353, 1965.

ZITZLER, E.; LAUMANNS, M.; THIELE, L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, p. 95–100, 2001.