Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Faculdade de Engenharia

Patricia Macedo da Costa Jorge

Síntese de Comitê de Árvores de Padrões *Fuzzy* através da Programação Genética Cartesiana em Ambientes Não Estacionários

Rio de Janeiro
2018

Patricia Macedo da Costa Jorge

Síntese de Comitê de Árvores de Padrões *Fuzzy* através da Programação Genética Cartesiana em Ambientes Não Estacionários

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao programa de Pós-Graduação em Engenharia Eletrônica da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Orientadores: Prof. Dr. Jorge Luís Machado do Amaral

Prof. Dr. Douglas Dias Mota

Rio de Janeiro

2018

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

J82 Jorge, Patricia Macedo da Costa.
Síntese de comitê de árvores padrão fuzzy através de programação genética cartesiana em ambientes não estacionários / Patrícia Macedo da Costa Jorge. – 2018.
89f.

Orientadores: Jorge Luís Machado do Amaral, Douglas Dias Mota.
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia eletrônica - Teses. 2. Sistemas difusos - Teses. 3. Programação estruturada - Teses. 4. Fluxo de dados (Computadores) - Teses. I. Amaral, Jorge Luís Machado do. II. Mota, Douglas Dias. III. Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia. IV. Título.

CDU 681.5

Bibliotecária: Júlia Vieira – CRB7/6022

Assinatura

Data

Patricia Macedo da Costa Jorge

Síntese de Comitê de Árvores de Padrões *Fuzzy* através da Programação Genética Cartesiana em ambientes não estacionários

Dissertação apresentada como requisito parcial para obtenção do título de Mestre, ao programa de Pós-Graduação em Engenharia Eletrônica da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Aprovado em 13 de dezembro de 2018.

Banca Examinadora:

Prof. Dr. Jorge Luís Machado do Amaral (Orientador)

Faculdade de Engenharia – UERJ

Prof. Dr. Douglas Mota Dias (Orientador)

Faculdade de Engenharia – UERJ

Profa. Dra. Luciana Faletti Almeida

Centro Federal de Educação Tecnológica Celso Suckow da Fonseca – CEFET

Prof. Dr. Harold Dias de Mello Junior

Faculdade de Engenharia – UERJ

Rio de Janeiro

2018

DEDICATÓRIA

Ao Laboratório de Redes Industriais e Sistemas de Automação (LARISA), e às pessoas com quem convivi neste espaço. A experiência de uma produção compartilhada na comunhão com amigos neste laboratório foi a melhor experiência da minha formação acadêmica.

AGRADECIMENTOS

Agradeço a Deus por ser essencial em minha vida, autor do meu destino, meu guia, socorro presente na hora da angústia. Ao meu pai Luiz Antônio e minha mãe Sonia por me proporcionarem a melhor educação que eu poderia receber, suas palavras de ânimo nunca me deixaram desistir. Ao meu irmão Estéfano e cunhada Flávia pela disponibilidade e carinho. Ao meu filho Lucas que tantas vezes abriu mão do passeio no parque porque a mãe esteve ocupada, e com seu sorriso inocente iluminou meus pensamentos. Ao Marcelo grande incentivador, obrigada por todo o apoio. Aos amigos que o mestrado me trouxe em especial Noemi e Sávio, vocês tornaram a caminhada mais leve, meu muito obrigada por todo o apoio cognitivo, pelo carinho e força extraordinários. Ao Adriano pelo apoio por meio das explicações e pelas risadas extras. Aos colegas do laboratório Hugo, Éverton, George e Anderson por ajudarem a manter este espaço tão necessário para a comunidade. Ao meu amigo mente brilhante, Igor. À amiga Tatiana minha inspiração. Por fim, ao professor Jorge Amaral pela orientação, por disponibilizar seu conhecimento, tempo e paciência ao longo deste projeto. Ao professor Douglas Mota pela orientação, conhecimento e dicas. Ao meu eterno mestre professor José Helayel Abdala Neto. E ao meu cachorrinho Thor por estar sempre por perto.

Nele estava a vida, e esta era a luz dos homens.
A luz brilha nas trevas, e as trevas não a derrotaram.
João 1: 4 e 5

RESUMO

JORGE, Patricia M. C. **Síntese de comitê de árvores de padrões f*uzzy* através da programação genética cartesiana em ambientes não estacionários**. 2018. 89f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2018.

A extração de dados em ambientes não estacionários requer técnicas diferentes das tradicionalmente utilizadas no aprendizado supervisionado. Aplicações como medição de temperatura por sensores, detecção de spam, previsão climática e preferência de compras de um consumidor são alguns dos exemplos de situações para as quais não é possível se supor que a distribuição dos dados não mudará no decorrer do tempo. Ambientes onde este cenário ocorre é caracterizado por mudanças denominadas *concept drift.* Neste trabalho é proposto um método baseado em um comitê de classificadores de árvores de padrões *fuzzy*, induzidos pela programação genética cartesiana, o qual deve ser capaz de se auto adaptar e manter o desempenho acurado, além de fornecer conhecimento de forma a auxiliar na tarefa de análise e interpretação dos dados. Para a execução dos experimentos foram analisadas bases de dados reais e artificiais com *drifts* distintos, alta dimensionalidade e diferentes balanceamentos entre as classes. Os resultados mostraram que o método é robusto e competitivo em comparação com outros da literatura.

Palavras-chave: *Concept Drift*, Árvores de Padrões *Fuzzy;* Programação Genética Cartesiana; Não Estacionário; Interpretabilidade; Classificação.

ABSTRACT

JORGE, Patricia M. C. **Synthesis of fuzzy pattern trees ensemble through cartesian genetic programming in non-stationary environments**. 2018. 89f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2018.

Extraction of data in non-stationary environments requires techniques different from traditional approaches used in supervised learning. Applications such as temperature measurement by sensors, spam detection, climate forecast and customer buying preference are some examples of situations in which information on the data distribution may vary over time. In environments where this scenario occurs is called "concept drift" and it is a challenge for the classifier to learn from the data. In this work is presented a method based on an ensemble of Fuzzy Patterns Trees induced by Cartesian Genetic Program. This model must be able to self-adapt and maintain accurate performance as well as providing knowledge to help in analysis and interpretability of the data. For the execution of experiments, artificial and real database with different drifts, high dimensionality and different balance between classes were used. The results show that the method is robust and competitive with others in the literature.

Keywords: Concept Drift; Fuzzy Pattern Trees; Cartesian Genetic Programming; Non-Stationary; Interpretability; Classification.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

APF	Árvore de Padrões *Fuzzy*
AUC	*Area Under Roc Curve*
BBh	Bloco *Bag* histórico
BB	*Bag* Bloco
CART	*Classification and Regression Trees*
KNN	*K-Nearest Neighbors*
MLP	*Multilayer Perceptron*
OOBh	*Out-of-Bag* da base histórico
NB	*Naive Bayes*
PG	Programação Genética
PGC	Programação Genética Cartesiana
SVM	*Support Vector Machines*

SUMÁRIO

INTRODUÇÃO

A evolução da ciência da informação e tecnologia tem produzido uma quantidade de dados gigantesca e, portanto, tornou-se inviável para o homem realizar a análise desse volume por si próprio. Dessa maneira, a extração de conhecimento de dados é um tema que tem sido abordado em diversos campos de pesquisa, a exemplo de mineração de dados, aprendizagem de máquinas e reconhecimento de padrões (BIFET et al., 2018), (LU et al., 2018), (SINGH et al., 2013).

Tradicionalmente, essa extração ou aprendizagem é realizada de modo *offline* em ambientes estáticos, onde o conjunto de dados estará disponível para o classificador/regressor treinar quantas vezes for necessário. Durante a fase de treinamento nenhuma atividade pode ser executada, e após esta fase nenhum conhecimento novo é aperfeiçoado ou alterado. Ou seja, a relação entradas e saídas não muda ao longo do tempo. Vários métodos foram propostos para a realização destas tarefas, tais quais por exemplo: diagnóstico de doenças pulmonares, onde em (PINTO, 2018) foram empregadas Redes Bayesianas para apoio na classificação do indivíduo como saudável ou doente; para a classificação de texto e hipertexto o algoritmo, Support Vector Machines (SVM) foi utilizado como sucesso em (JOACHIM, 1998); de maneira similar, também foi usado na classificação de proteínas em ciências biológicas em (GUYON et.al, 2002); para a classificação de imagens, métodos como redes de aprendizado profundo em (CIRESAN et. al, 2012) foi possível obter ótimos resultados no reconhecimento de dígitos manuscrito; existem ainda métodos para lidar com dados heterogêneos, como Árvores de Decisão ou métodos que empegam uma função de distância como o algoritmo K-Nearest Neighbors (KNN). Desta maneira, atualmente, muitos dos problemas relacionados a generalização a partir dos dados de treinamento para instâncias não vistas são resolvidos de maneira satisfatória (GAMA et al., 2014).

Entretanto, em muitos problemas práticos, os dados que descrevem o problema podem passar por modificações ao longo do tempo (GAMA et al., 2014). Estas mudanças ocorrem em ambientes com constante flutuação, instabilidade e irregularidade no decorrer do tempo, e desta forma são chamados de ambientes não estacionários. Exemplos de aplicações incluem o monitoramento do histórico de compras de cliente: as preferências dos clientes podem mudar de acordo com a época do ano, inflação ou disponibilização de novos produtos. Desta maneira, ao mudar as preferências do usuário, um modelo que aprendeu como prevê-lo no passado falhará se

não puder se atualizar devido às alterações no problema (SCHOLZ & KLIMBERG, 2007). Outros exemplos incluem a extração do conhecimento em ambientes com fluxo contínuo de dados, tais quais detecção de presença por meio de sensores e o monitoramento da temperatura da água (KADWE & SURYAWANSHI, 2015), sistemas para segurança de computadores (GAO et. al, 2007), detecção de *spam* (NISHIDA, 2008), entre outros. Para estes casos, os dados chegam em tempo real e necessitam ser analisados de modo *online* e, da mesma maneira como ocorre com os hábitos dos clientes, também é esperado que a distribuição dos dados se modifique com o tempo (ZLIOBAITE, 2012). Como consequência, o modelo construído com base em dados históricos se torna inconsistente com os novos dados (KRAWCZYK et al., 2017). Esses cenários típicos de ambientes não estacionários exigem que os algoritmos de aprendizado possam lidar com um fenômeno conhecido por *concept drift*. Em linhas gerais, em tarefas de classificação, *concept drift* ocorre quando os rótulos das classes mudam com o tempo, de modo que os atributos utilizados na discriminação das classes no tempo t podem não ser mais úteis no tempo $t + k$, o que constitui um desafio para o modelo na tarefa de aprender a partir dos dados. Devido a esse fato são necessárias estratégias específicas, diferentes das técnicas tradicionalmente utilizadas (TSYMBAL, 2004). O aprendizado efetivo em ambientes com a presença de *concept drift* requer um algoritmo de aprendizado que possa: (i) detectar mudanças de contexto sem ser explicitamente informado sobre elas; (ii) se recuperar rapidamente de uma mudança de contexto e ajustar suas hipóteses a um novo contexto; e (iii) fazer uso da experiência anterior em situações de recorrência. Desta maneira, o algoritmo deve ser capaz de se adaptar à mudança de conceito, caso contrário haverá perda de desempenho ao longo do tempo.

Existem diversas abordagens e métodos propostos para a detecção de *concept drift*, por exemplo a utilização de janelas deslizantes (HULTEN et al., 2001), detecção de desvios (CARVALHO & COHEN, 2006) e técnicas de aprendizado não-incremental (ZLIOBAITE, 2009), (GARBER et al., 2005). No entanto, uma forma bem-sucedida e comum de lidar com mudanças é a utilização de comitês de classificadores (*Ensemble Classifiers*) (KRAWCZYK et al., 2017). Neste tipo de abordagem, vários classificadores são treinados com um algoritmo base e suas predições individuais são combinadas para formar a predição global. Os mecanismos para atualização do comitê (criação, troca e exclusão de classificadores) e sua forma de integrar as predições dos membros individuais para uma predição global variam de acordo com cada método.

Embora, em princípio, qualquer classificador possa ser usado como participante de um comitê, quando se deseja compreender "como" o modelo induzido consegue fazer a distinção entre diferentes classes, isto é, caso se deseje extrair o conhecimento de como esta distinção é feita, os modelos gerados por abordagens mais simbólicas tornam-se mais atrativos.

Portanto, o classificador base utilizado neste estudo é baseado na teoria *fuzzy*. Isso garante características como a gradualidade, granularidade e a interpretabilidade. Os sistemas *fuzzy* mais comuns são baseados em regras (SFBR); este é um fator que restringe a interpretabilidade do sistema quando a aplicação requer uma grande quantidade de regras para se obter a acurácia necessária. Como alternativa ao SFBR, optou-se pela utilização de árvores de padrões *fuzzy* (APF), pois se deseja obter um tipo de descrição lógica de uma classe (SENGE & HUELLERMEIR, 2011), porém sem o uso de regras.

Vários modelos diferentes de comitê de classificadores foram propostos na literatura: (KUNCHEVA, 2008), (ELWELL & POLIKAR, 2011), (KUNCHEVA, 2004), (GAMA et al., 2014) são alguns desses exemplos. Eles possuem aprendizagem com boa acurácia em cenários com *concept drift*, segundo (WANG et al., 2003). Porém, dentre vários desses métodos e técnicas, optou-se por explorar as vantagens do mecanismo de busca global dos algoritmos evolutivos na indução das APF, especificamente a Programação Genética Cartesiana (PGC) (MILLER, 2003).

Particularmente, neste trabalho é considerado o problema de *concept drift* no aprendizado supervisionado em tarefas de classificação, nas quais o rótulo verdadeiro para cada instância que é apresentado ao modelo é conhecido somente algum momento após ser classificada (LUDMILA et al., 2008). Podem ser citados como exemplos de tarefas em que existem atrasos nos rótulos das classes: previsão de vendas, previsão de falências, resultado de tratamento de pacientes, detecção de intrusão ou fraude e tarefas de categorização de spam.

O modelo apresentado neste estudo não conta com mecanismo de detecção de *drift* explícito. Contudo, com intuito de mantê-lo com bom desempenho foram empregadas estratégias de aprendizado adaptativas que atuam na evolução do comitê. Essas estratégias irão atuar a partir dos resultados do modelo, após a realização da comparação do rótulo previsto pelo modelo com o rótulo verdadeiro recebido. Dessa forma é feita uma abordagem reativa ao *drift*, visto que o *drift* já ocorreu efetivamente.

A acurácia é uma das medidas de desempenho mais comumente usadas em

algoritmos de aprendizagem de máquinas. Neste trabalho também foi aplicada para a avaliação do método proposto. Deseja-se que o modelo possa atingir acurácia competitiva, mostrando-se autoadaptável e adequado para ambientes não estacionários. Além disso, é de particular interesse que este modelo possa ser interpretável de maneira a validar o modelo induzido e gerar leituras possíveis de serem compreendidas pelo homem.

O objetivo deste trabalho é explorar um método alternativo ao sistema *fuzzy* baseado em regras (SFBR), que seja flexível e capaz de aprender em ambientes com a presença de *concept drift* e possua interpretabilidade.

A contribuição deste trabalho é um arcabouço projetado para sintetizar os comitês e permitir a análise da metodologia de síntese e das características dos comitês formados ao longo do tempo.

Portanto, o modelo *fuzzy*-genético sintetizado é composto por árvores de padrões *fuzzy* (APF), organizadas em comitês de classificadores, para ambientes não estacionários utilizando-se da Programação Genética Cartesiana como método de busca. E, levando-se em consideração o *trade-off* acurácia-interpretabilidade.

Este documento está estruturado da seguinte forma: o Capítulo 1 e o Capítulo 2 apresentam os fundamentos de Árvores de Padrões *Fuzzy* e Programação Genética Cartesiana respectivamente; o Capítulo 3 aborda a terminologia, fundamentos, causas e tipos de *concept drift*; o Capítulo 4 apresenta a solução proposta nesta dissertação para aprendizado em ambientes não estacionários; o Capítulo 5 apresenta os estudos de caso com resultados experimentais; por fim, há a conclusão do trabalho e a apresentação da possibilidades de trabalhos futuros.

1. ÁRVORES DE PADRÕES *FUZZY*

As Árvores de Padrões *Fuzzy* (APF) foram criadas por (HUANG et al., 2008). Trata-se de modelos hierárquicos, com uma estrutura de árvore, na qual os nós internos são operadores lógicos e aritméticos utilizados nos sistemas *fuzzy* e as folhas destas árvores são compostas por termos *fuzzy* associados a um determinado atributo de entrada. O mecanismo de aprendizado é chamado de *bottom-up induction*. Nele, a indução das árvores de padrões procura criar uma "representação lógica" para cada classe de uma forma iterativa. O processo ocorre do fundo para o topo onde, dado um conjunto de valores de atributos como entrada, cada nó folha produz um grau de pertinência como saída, que é então enviada como entrada para seu sucessor. Cada nó interno recebe os valores de seus antecessores e os combina utilizando um operador de agregação e, novamente, envia a saída para o nível superior, de forma recursiva.

Desta maneira um classificador baseado em árvores de padrões é construído criando-se uma árvore de padrões *fuzzy* para cada classe. São inseridos os valores dos atributos que se deseja classificar nas entradas das árvores de cada classe e a predição da classe é feita em favor da classe cuja árvore produz o maior resultado, ou seja, valor de saída, sendo este valor compreendido em um intervalo unitário.

Procedendo da configuração padrão para aprendizado supervisionado, e assumindo a representação das instâncias como sendo atributo-valor e cada instância sendo um vetor $\boldsymbol{x}$, temos que:

$$\boldsymbol{x} = (x_1, x_2, \ldots x_m) \in X = X_1 \times X_2 \times \ldots \times X_m, \quad (1)$$

onde X_i é o domínio do i-ésimo atributo e a quantidade de atributos é dada por *m.* Cada domínio X_i é discretizado por meio de partições $\mathrm{F}_{ij} = \{F_{i,1}, \ldots, \mathrm{F}_{i,d}\}$, isto é, uma coleção de subconjuntos *fuzzy*, tal que:

$$F_{i,j} : X_i \rightarrow [0,1] \quad (j = 1, \ldots, d_i), \quad (2)$$

onde d é a quantidade de partições que os domínios foram divididos, sendo que $\sum_{j=1}^{\mathrm{d}_i} F_{i,j}\ (x_i) > 0$ para todo $x \in X_i$. O termo $F_{i,j}$ define premissas individuais sobre os valores dos atributos. Estas premissas estão associadas com rótulos linguísticos tais como "pequeno" ou "alto", os quais são chamados de *termos fuzzy*. Além disso, existe um espaço de saída Y, que, dependendo do problema, pode ser binário ($Y = \{0,1\}$), categórico ou numérico ($Y \subset \Re$). Em resumo, dada como entrada para o nó folha uma

instância representada por $\boldsymbol{x} = (x_1, x_2, \ldots x_m)$, o nó folha irá produzir como saída $F_{i,j}(x_i)$, que é o grau de pertinência de x_i em $F_{i,j}$. Este grau de pertinência será propagado em direção aos níveis superiores até a raiz.

As árvores de padrões *fuzzy* se apoiam em três conceitos: os atributos de entrada, o estruturamento hierárquico de uma dependência funcional, através do particionamento recursivo de um critério em sub-critérios, e a agregação flexível de sub-critérios utilizando operadores *fuzzy* parametrizados, segundo (SENGE & HULLERMEIER, 2011). São utilizados os operadores *fuzzy* t-norms (conjuntivos) e t-conorm (disjuntivos). Além destes, para preencher a lacuna entre agregações conjuntivas e disjuntivas, uma APF permite o uso de operadores de média OWA – *Ordered Weighted Average* – e o WA – *Weighted Average*. Detalhes do funcionamento do operador OWA podem ser observados em (EMROUZNEJAD & MARRA, 2014). O operador WA é similar ao OWA, porém sem ordenação.

Os operadores *fuzzy* t-norm e t-conorm utilizados estão listados na Tabela 1 e na Tabela 2.

Mínimo	min(a,b)
Algébrico	a * b
Lukasiewicz	max(a - 1+ b, 0)
Einstein	$\frac{a*b}{2-(a+b-a*b)}$

Tabela 1: Operadores *Fuzzy* T-Norm.

Máximo	max(a,b)
Algébrico	a + b – a * b
Lukasiewicz	min (a + b, 1)
Einstein	$\frac{a+b}{1+a*b}$

Tabela 2: Operadores *Fuzzy* T- Conorm.

A Figura 1 exemplifica o funcionamento do processo *bottom-up* no qual características mais básicas criadas pelos atributos originais são usadas de forma iterativa para a construção de características complexas. Este processo ocorre do fundo

para o topo, repetidamente combinando as duas melhores árvores candidatas com um novo nó interno.

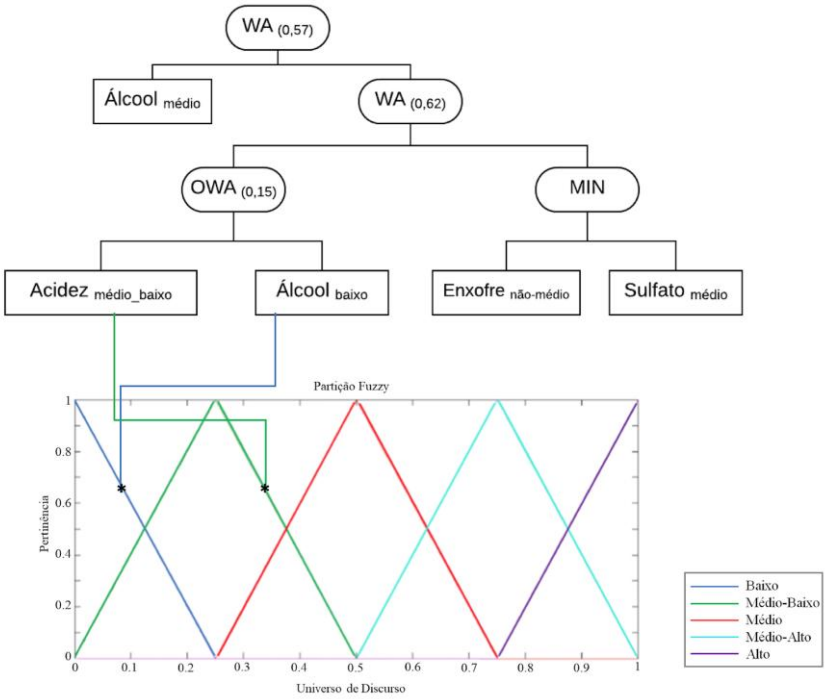


Figura 1: Árvore de Padrões *Fuzzy* da qualidade de um vinho tinto – *botton-up.*

No exemplo da Figura 1, foi gerada uma árvore que determina a qualidade de um vinho tinto. Os atributos de entrada são: acidez, teor alcóolico, enxofre e sulfato. Associa-se a cada atributo um termo *fuzzy* que representa um intervalo do universo de discurso deste atributo. Na Figura 1, o termo *fuzzy* álcool_baixo representa o conjunto *fuzzy* teor alcóolico baixo.

Segundo (SENGE & HULLERMEIER, 2011), este método simplifica a avaliação global de uma propriedade pois estas avaliações são agregadas de modo a produzir apenas uma variável de saída. Na Figura 1, duas subárvores estão relacionadas à qualidade do vinho, uma delas relacionando a acidez média e baixo teor alcóolico e a outra, sulfato médio e enxofre com teor não-médio. As informações desta subárvores são propagadas para o nível superior e agrupadas em resultados parciais [0,1], e após combinada com teor alcóolico médio, resulta em uma saída que determina uma classe que representa a qualidade do vinho. Quanto mais próximo de 1 melhor a árvore representa a classe "vinho de boa qualidade".

Dois aspectos podem ser analisados quanto à APF: a possibilidade de identificar que diferentes conhecimentos são combinados nas subárvores e o relacionamento entre

as variáveis. Por exemplo, a variável teor alcóolico médio (álcool médio) exerce grande influência no resultado final, devido à sua proximidade com a raiz da árvore.

Embora os primeiros estudos experimentais sobre APF conduzidos por (HUANGE et al., 2008) sugiram que os classificadores de árvores de padrão funcionem razoavelmente bem em termos de precisão preditiva, o algoritmo de aprendizagem oferece margem para melhorias, segundo (SENGE & HULLERMEIR, 2011). Um novo método de aprendizagem foi proposto por estes autores, principalmente no que tange a construção da árvore, que mudou a direção de *"bottom-up"* para *"top-down"*.

No método *bottom-up*, duas subárvores são combinadas para gerar uma terceira nova árvore, sendo esta maior e com uma estrutura bem diferente. Este processo pode ser visto como um operador de recombinação. A indução *bottom-up* tende a produzir grandes saltos no espaço de busca; por exemplo, o operador combinação pode dobrar o número de nós de árvores candidatas a cada interação.

Na indução *top-down* a ideia é modificar a árvore atual apenas um pouco de forma mais granular, em pequenos passos no espaço de busca. Isso pode ser feito expandindo um dos seus nós de folha, ou seja, substituindo uma característica básica $F_{i,j}$ por uma característica composta (duas características básicas, uma delas sendo $F_{i,j}$ que é combinada por um operador). Assim novos operadores são introduzidos na parte inferior da árvore e não no topo, o que normalmente significa que os operadores posteriores têm uma influência menor no comportamento de entrada-saída do que aqueles que foram escolhidos anteriormente (nós pais), além de proporcionar uma exploração maior do espaço de busca. Este algoritmo é chamado de *best search*, o qual explora a árvore expandindo o nó mais promissor, onde somente um número predeterminado de melhores soluções parciais é mantido como candidatas.

O procedimento de gerar em cada nível da árvore todos os sucessores (largura do feixe) e ordená-los é custoso computacionalmente, principalmente para conjuntos de dados com muitos exemplos ou muito atributos, no qual o tempo de execução pode se tornar inaceitavelmente alto. De forma contrária, se a largura do feixe for pequena, apenas uma região do espaço será explorada. Diante desta questão, um novo aperfeiçoamento no método *top-down* foi proposto em (SENGE & HULLERMEIR, 2015).

A Figura 2 exemplifica uma sequência de criação de uma árvore de padrão *fuzzy top-down.* O processo é iniciando com uma árvore de padrão primitiva. Uma árvore primitiva é uma árvore que é composta por somente um nó rotulado por um termo *fuzzy*.

Em seguida ocorre então a expansão das árvores candidatas de forma iterativa, selecionando-se a melhor árvore baseando-se em um critério de desempenho, até que um critério de término seja atingido.

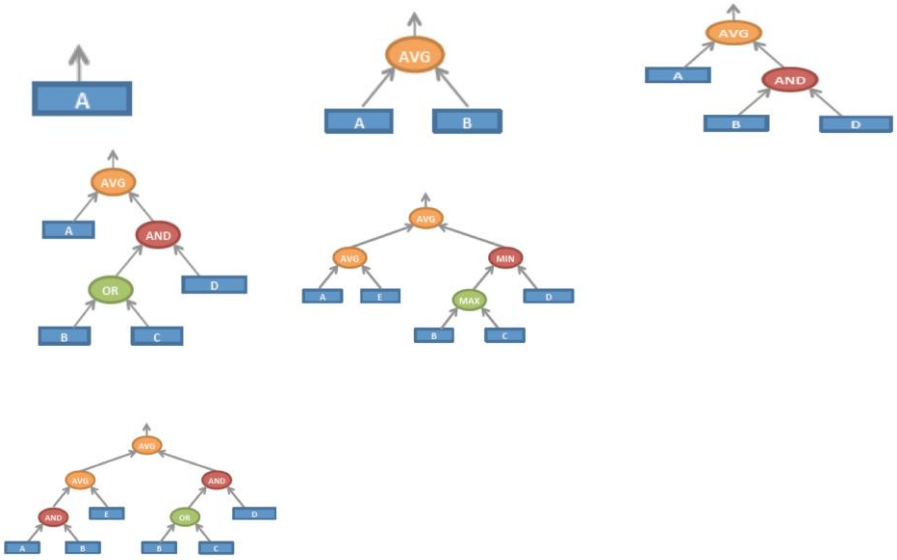


Figura 2: Árvore de Padrão *Fuzzy* - *Top Down*.

Neste trabalho o método de aprendizado das árvores de padrões *fuzzy* (APF) foi substituído pelo uso da Programação Genética Cartesiana (PGC). A PGC é um método de busca global capaz de explorar espaços de busca bastante grandes de forma eficiente e a representação dos programas na forma de grafos pode ser facilmente utilizada para representar APFs.

2. PROGRAMAÇÃO GENÉTICA CARTESIANA

2.1 Introdução

A Computação Evolucionária compreende um conjunto de técnicas de busca e otimização inspiradas na evolução natural das espécies. Desta forma, cria-se uma população de indivíduos que vão se reproduzir e competir pela sobrevivência. Os melhores sobrevivem e transferem suas características a novas gerações. As técnicas principais incluem: Programação Evolucionária, Estratégias Evolucionárias, Algoritmos Genéticos, Programação Genética, EDAs e Sistemas Classificadores.

A Programação Genética (PG) é a evolução de um conjunto de programas com o objetivo de aprendizagem por indução (BANZHAF,1998). Esta técnica permite que computadores apresentem soluções sem a necessidade de serem programados explicitamente (KOZA,1992). Basicamente, a Programação Genética mantém uma população de programas de computador, usa métodos de seleção baseados na capacidade de adaptação (mérito) "*fitness"* de cada programa (escolha dos "melhores"), aplica operadores genéticos para modificá-los e convergir para uma solução. O objetivo é encontrar uma solução no espaço de todos os programas possíveis (candidatos) usando apenas um valor de *fitness* como auxilio no processo de busca.

A Programação Genética (PG) se mostra promissora, pois esta técnica gera programas (soluções mais flexíveis) que solucionam diversos problemas práticos de diversas áreas do conhecimento (KOZA,1992). A representação dos indivíduos na Programação Genética é feita tipicamente por estruturas conhecidas como árvores.

2.2 Funcionamento

Na PG os nós das árvores são campos de memória que podem armazenar valores usados como entrada do programa ou também funções. Cada nó da árvore representa um gene do cromossomo (CELES et al., 2004).

A Função de Aptidão na Programação Genética calcula a diferença entre a saída obtida por aquele cromossomo e a saída desejada. Para retornar um valor numérico, a Função de Avaliação precisa entrar com o valor, executar o programa e verificar sua saída (KOZA, 1992). Entende-se por executar o programa como sendo a tarefa de decodificar o cromossomo.

A mutação não altera apenas um nó da árvore, mas sim uma sub-árvore inteira (MICHALEWICKZ,1992), (MITCHELL,1999); (POLI et al., 2008). Dessa forma, o

operador genético gera estruturas válidas para serem repostas onde se deseja realizar a mutação do gene. Isso faz com que se altere a estrutura da árvore, porém garante que os indivíduos ainda continuem dentro dos parâmetros permitidos. Uma representação do operador pode ser observada na Figura 3.

O operador genético de cruzamento utiliza a mesma ideia de substituição de sub-árvores: seleciona aleatoriamente uma sub-árvore do primeiro e do segundo pai, e as comuta entre eles para gerar novos filhos (EIBEN & SMITH, 2003).

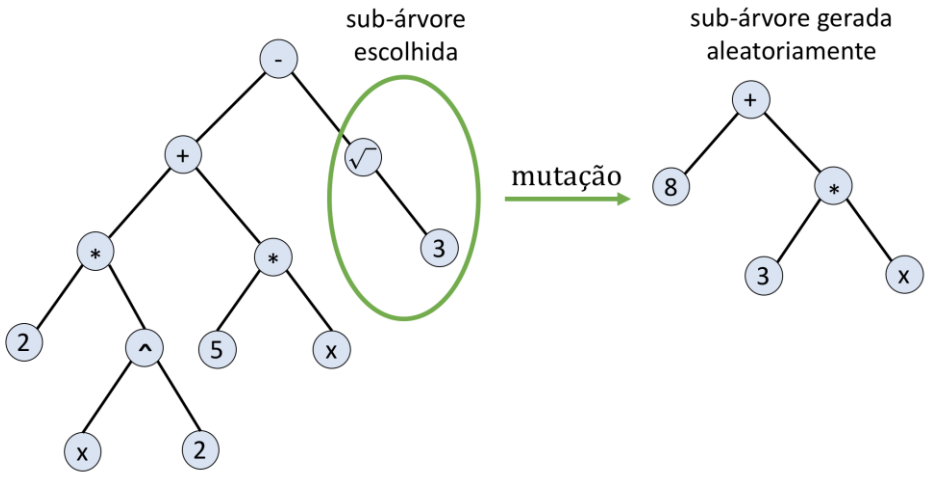


Figura 3: Processo de Mutação.

2.3 Programação Genética Cartesiana

A Programação Genética Cartesiana (PGC) é uma derivação da Programação Genética porque também trabalha com a geração de programas solucionadores de problemas, porém, a representação utilizada para manipular os indivíduos não são estruturas de árvores. Os indivíduos nesse caso são representados por grafos (MILLER, 2011).

Os grafos são estruturas de dados muito utilizadas em computação e possuem dois conjuntos: um conjunto de vértices ou nós, e um conjunto de arestas. Os grafos utilizados são indexados, ou seja, cada nó possui um índice. Os nós representam os genes dos cromossomos. Cada nó guarda o índice de uma função que segue uma tabela de funções predefinida e os índices dos outros genes que se interligam a ele. Organizam-se em linhas e colunas, e os nós se conectam apenas entre colunas, como se cada uma fosse um nível do grafo. Os nós da primeira coluna se conectam com as entradas do programa, enquanto que as saídas dos genes da última coluna se conectam

com as saídas do sistema. O número máximo de níveis que cada gene pode se conectar é chamado de "*level-back*" e é representado pelo parâmetro *l*. Se *l = 2*, por exemplo, os genes podem se conectar com no máximo duas colunas anteriores. Devido ao esquema linha/coluna, o nome "Cartesiana" foi proposto pois os nós são endereçados pelo sistema de coordenadas cartesianas.

O número de linhas n_r, o número de colunas n_c, o número de entradas n e o número de saídas m são parâmetros definidos pelo usuário antes de se iniciar a evolução, sendo que n_r e n_c são parâmetros que determinam quantos genes cada indivíduo possuirá e consequentemente o tamanho de cada cromossomo, onde o número máximo de nós alocados por indivíduo será: $\mathcal{L}_n = n_r$ x n_c. Além disso, é necessário conhecer o número de funções que serão utilizadas na codificação genética.

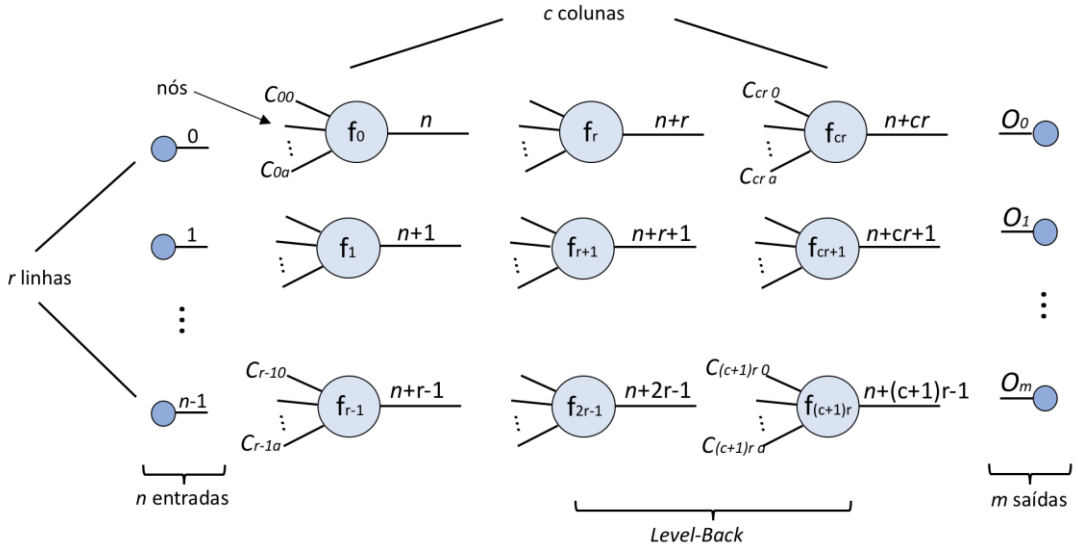


Figura 4: Grade de nós computacionais da PGC adaptado de (MILLER, 2011).

A Figura 4 representa o esquema geral proposto por (MILLER, 2011) de um mapeamento genético, a organização dos grafos e as regras para se determinar os índices na codificação. O número de linhas e colunas diz o formato da grade. É possível diversos tipos de representação de nós alterando-se estes dois parâmetros.

Os nós que vão do índice *0* até n-1 são as entradas do programa e os nós O_0, O_1,..., O_m são as suas saídas. Os demais são nós que representam os genes do cromossomo, com suas entradas *C* e saídas que vão dos índices até n + (c + 1) r - 1, de acordo com a Figura 4.

Na Figura 4, C_{ia} representa as entradas de cada gene, onde *i* representa o índice

do nó referido e a indica qual entrada do gene é referenciada, também chamado de aridade. Por exemplo, C_{31} representa a segunda entrada (pois a contagem se inicia do 0) do gene de índice 3. O termo r é uma constante que representa o número de linhas da grade computacional e c representa o índice da coluna a qual o gene está localizado, começando ambos em 0.

O vetor de índices que representa a grade computacional exibida na Figura 4 é construído então com sucessivas *n-uplas*, onde cada uma representa um nó da grade e contém sempre o endereço da função utilizada na tabela de funções; os endereços dos nós que servem de entrada para a função e as saídas do programa podem ser vistos na Figura 5.

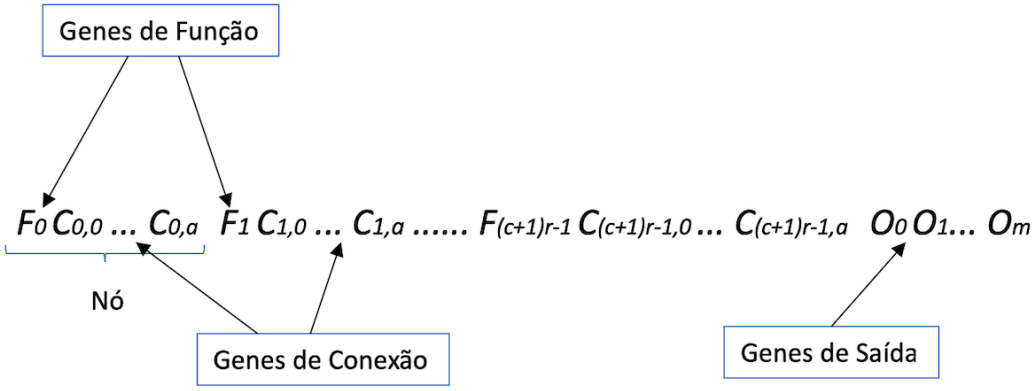


Figura 5: Genótipo ou Cromossomo (MILLER, 2011).

A Figura 6 exemplifica a utilização dessa organização de nós para a geração de funções matemáticas e a codificação correspondente no vetor de endereços, considerando $n = 2$, $m = 2$, $l = 2$, $n_r = 2$ e $n_c = 2$, e a Tabela 3 de funções aritméticas.

Índice	**Operação**
0	Adição
1	Subtração
2	Multiplicação
3	Divisão

Tabela 3: Funções aritméticas.

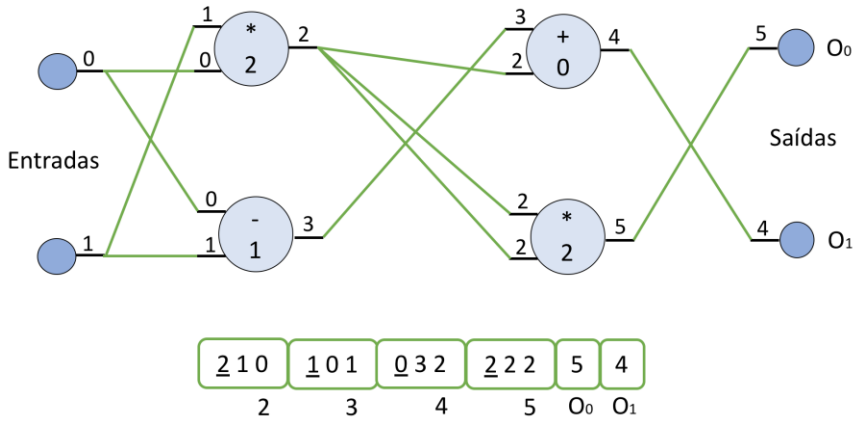


Figura 6: Exemplo de esquema de gráficos indexados e sua respectiva codificação no cromossomo – adaptado de (MILLER, 2011).

O processo evolutivo ocorre nos genótipos, sendo necessária uma decodificação deste genótipo em fenótipos cuja formatação está no domínio da solução do problema. O programa gerado pela decodificação do genótipo é chamado de fenótipo. Enquanto os genótipos têm um tamanho fixo, os fenótipos podem variar de tamanho.

O operador mutação é o operador mais importante no PGC, diferentemente do cruzamento que é pouco utilizado. A estratégia evolutiva utilizada no PGC é a 1+λ. Escolhem-se indivíduos denominados pais para gerarem filhos, sendo a população formada por 1+ λ indivíduos competindo entre si e novamente são escolhidos indivíduos, geralmente os mais adaptados, para sobreviverem até a próxima geração (LINDEN, 2012). Normalmente faz-se λ=4 (MILLER, 2011), logo são criados cinco indivíduos aleatoriamente e aquele que foi considerado mais apto através da aplicação da função de fitness (aptidão) é selecionado para a geração seguinte. O operador mutação é aplicado quatro vezes no individuo mais apto da geração anterior, totalizando cinco indivíduos na população. Novamente, o operador mutação é aplicado quatro vezes no mais apto da geração anterior. Este processo se estende até atingir o critério de parada determinado pelo usuário. Neste trabalho se adotou o total de gerações. A parada é acionada quando a quantidade de gerações atinge o valor pré-estabelecido.

Uma das vantagens da PGC é a utilização de grafos ao invés da árvore, utilizada na programação genética convencional, pois grafos são mais gerais, flexíveis e compactos (DHARWADKER & PIRZADA, 2011). Outras vantagens da PGC são as características de neutralidade, importante para a variabilidade genética, a redundância e a minimização do *bloat* (BNZAF, 1994); (MILLER ,2006); (MILLER 2001). *Bloat* ocorre quando há ocorrência de redundância funcional, tendência usual dos programas gerados pela PG. A redundância funcional gera programas com uma quantidade maior de nós do que o necessário para o funcionamento, sendo estes nós sem contribuição para o desempenho do programa. Além disso, a ocorrência de *bloat* prejudica a criação e avaliação de aptidão do programa, tornando-a mais demorada. A PGC minimiza o *bloat* de três maneiras: o tamanho do genótipo é fixo freando o crescimento desregular do programa; a PGC pode utilizar funções de aptidão que favoreçam soluções com uma menor quantidade de nós conectados. E a última é a possibilidade de ativação e desativação de nós, porque durante a evolução os nós que não influenciam o desempenho da solução podem ser simplesmente desconectados, não fazendo assim parte da solução na geração seguinte (MILLER 2001).

Por esse motivo e devido à sua capacidade de representar as APF, com habilidade para explorar grandes espaços de busca sem a limitação do método *beam search*, foi escolhida a PGC como método de indução das APF em (SANTOS, 2014) e também adotada neste estudo, porém, com o objetivo de geração de árvores que irão compor um comitê.

3. CONCEPT DRIFT

Neste capítulo é dada uma visão geral das abordagens relacionadas ao tema principal desta dissertação: definições, tipos de *concept drift*, classificação dos modelos, funcionamento geral dos modelos de aprendizado e breve revisão dos modelos existentes.

Embora a área de pesquisa de *concept drift* tenha recebido atenção significativa nos últimos anos (a maioria dos trabalhos foi publicada nos últimos dez anos), o campo sofre de uma falta de terminologia padrão (Moreno-Torres et al., 2012). As pesquisas neste ramo diferem uma das outras de acordo com a perspectiva do autor para o problema (KOH & LIN, 2009); (VORBURGER & BERNSTEIN, 2006); (BAENA et al., 2006); (GAMA et al., 2014). Isso cria novos tópicos, ideias e desafios, por outro lado, o problema recebe novos aspectos e nomes que dificultam o entendimento real da questão. Apesar da dificuldade em consenso, optou-se neste trabalho por utilizar o termo *"concept drift",* uma vez que é o termo mais comumente utilizado, para quaisquer problemas de variação nos dados, sejam eles nas variáveis de entrada *x*, nas classes *y* ou na relação entre *x* e *y*.

3.1 Definição

Concept drift refere-se a um cenário de aprendizagem onde a relação entre os dados de entrada e a variável alvo muda ao longo do tempo (GAMA et al., 2014). Em diversas aplicações do mundo real, os dados que descrevem o problema podem passam por mudanças ao longo do tempo devido à sua natureza não estacionária (WIDMER & KUBAT, 1996). Para fins de exemplificação, suponha-se que o histórico de um cliente de um determinado cartão de crédito seja armazenado. Esse cliente, durante vários anos, na maior parte do tempo, utilizou esse cartão de crédito para comprar alimentos, sempre em uma mesma cidade. A partir das informações extraídas, um perfil do comportamento desse cliente é criado. Entretanto, não é razoável considerar que esse perfil jamais será modificado (TSYMBAL, 2004).

Concept drift também pode ser definido como um obstáculo causado por informações insuficientes, desconhecidas ou não observadas em um conjunto de dados, caracterizando um cenário dito como de contexto oculto. Um exemplo típico de um problema real são as regras de predição climática, que podem variar radicalmente de acordo com a estação do ano. Em geral assume-se que *concept drift* ocorre de forma inesperada e não-previsível. Apesar de algumas situações particulares do mundo real,

mudanças podem ser previstas à frente do seu tempo em correlação com a ocorrência de eventos ambientais próprios.

Em aprendizado de máquinas, o aprendizado supervisionado é definido pela previsão da classe $y \in \mathcal{R}$ em tarefas de regressão (ou *y* categórico em tarefas de classificação), dado cum conjunto de entrada $X \in \mathcal{R}^p$.

Mais especificamente em tarefas de classificação, de uma maneira mais formal, de acordo com a Teoria de Decisão Bayesiana (Duda et al., 2001), um classificador pode ser descrito pela probabilidade *a priori* das classes p(y) e as funções de densidade de probabilidade da classe $p(X|y)$ para toda classe $y = 1,...,c$, onde *c* é o número de classes. A decisão de classificação é feita de acordo com a probabilidade *a posteriori* das classes, a qual para a classe *y* pode ser representada por

$$p(y|X) = \frac{p(y)p(X|y)}{p(X)}, \tag{3}$$

onde $p(X) = \sum_{y=1}^{c} p(y).p(X|y)$. É assumido custo igual para o erro de classificação, sendo que:

- $p(X)$ descreve a probabilidade das características dos dados;
- $p(X|y)$ descreve a probabilidade de uma determinada amostra *x* pertencer a uma classe em particular.
- $p(y)$ define as probabilidades prévias das classes e relaciona o balanceamento de cada classe à distribuição geral.

De acordo com (ELWELL & POLIKAR, 2011), considerando a probabilidade *a posteriori* de uma amostra *x* pertencer a uma classe *y*, *concept drift* é qualquer cenário em que esta probabilidade é alterada ao longo do tempo, isto é: $p_{t+1}(y|X) \neq p_t(y|X)$. O termo "conceito" relaciona-se à distribuição dos dados em um determinado período de tempo. Dá-se o nome de "mudança de conceito", ou *concept drift*, às alterações nestas probabilidades. Desta maneira, os atributos utilizados na discriminação das classes no tempo *t* podem não ser mais úteis no tempo $t+1$, caracterizando alterações na descrição do problema.

3.2 Tipos de Concept Drift

Diversos autores, dentre os quais (GAMA et al., 2014), (TSYMBAL, 2004) e (ZLIOBAITE, 2009), categorizam as mudanças de conceito de acordo com o padrão de

configuração das fontes de distribuição de dados ao longo do tempo. Supondo duas fontes de dados A e B, temos os quatro tipos de *drift* descritos abaixo, e sua representação gráfica na Figura 7.

- Repentino ou abrupto: no tempo *t* a fonte A é substituída repentinamente por B no tempo *t*+1. Ocorre a troca brusca de um conceito Y por YY; por exemplo: um sensor é substituído por outro com diferente calibração em uma planta química.

- Incremental: inclui a presença de pequenas e não muito severas mudanças de conceito, caminhando mais gradativamente de um conceito Y para YY, podendo estas mudanças serem observadas com um longo período de tempo. Por exemplo: um sensor gradualmente se desgasta e se torna menos acurado. Este tipo de *drift* inclui várias fontes intermediárias entre A e B.

- Gradual: a mudança não é abrupta, aos poucos conceitos vão sendo alterados de Y por YY. As fontes de dados A e B estão ativas, porém, ao logo do tempo, a probabilidade de amostragem de uma fonte aumenta, enquanto a outra diminui. Como exemplo, temos uma mulher grávida que muda seu interesse de leitura por notícias e artigos relacionados a bebês, porém, não de forma abrupta, às vezes retorna com leitura sobre o tema de viagens.

- Recorrentes: conceitos que nunca foram vistos antes ou que já apareceram anteriormente reaparecem após algum tempo. Difere da noção de sazonalidade comum pois não é periódico e não há clareza de quando uma fonte B irá aparece para substituir a fonte ativa A, neste caso, o conceito Y muda para YY. Um exemplo comum do dia-a-dia é a moda. Peças de roupas que fizeram sucesso em uma determinada época *t* voltam a fazer sucesso no tempo *t*+*x*.

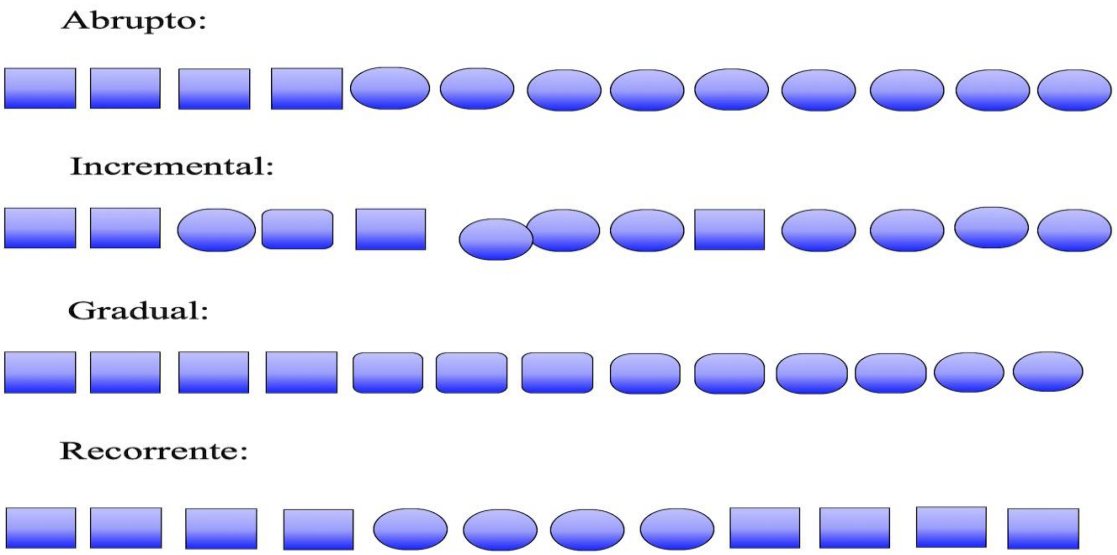


Figura7: Tipos de *Drift.*

Dependendo do tipo de mudanças, estratégias diferentes podem ser apropriadas para a construção do classificador.

É importante destacar que os algoritmos utilizados para a manipulação de *concept drift* não devem confundir o *drift* verdadeiro com um *outlier* ou ruído, o qual se refere a um desvio aleatório ou uma anomalia. Alguns algoritmos podem ter uma reação exagerada ao ruído, e desta forma erroneamente interpretá-lo como um *concept drift*, enquanto outros podem ser muito robustos a ruídos, adaptando-se às mudanças de maneira muito vagarosa. Um classificador ideal deve segundo (WIDMER & KUBAT, 1996), combinar robustez a ruídos e sensibilidade ao *concept drift.*

3.3 Estratégias adaptativas

A aprendizagem efetiva em ambientes com contextos ocultos e com a presença de *concept drift* requer um algoritmo de aprendizagem que possa detectar mudanças de contexto sem ser explicitamente informado sobre elas, se recuperar rapidamente de uma mudança de contexto, ajustar-se ao novo contexto e poder fazer uso da experiência anterior em situações onde velhos contextos e conceitos correspondentes reaparecem (NISHIDA & YAMAUCHI, 2009). Ou seja, o modelo deverá apresentar aprendizado adaptativo. Caso contrário, o desempenho do modelo treinado com base em dados históricos irá se degradar ao longo do tempo devido às alterações nos conceitos. Em função disso diferentes estratégias para tratar modelos de aprendizado em ambientes não estacionários com a presença de *concetp drift* foram desenvolvidas e tem recebido atenção crescente, muito em parte devido às aplicações práticas e cotidianas, como

exemplo, filtro de detecção de *spam*, intrusão ou fraude, diagnóstico médico, previsão climática etc.

Podem-se distinguir duas principais estratégias para lidar com aprendizado adaptativo: modelos de aprendizado que evoluem continuamente como em FLORA1 (WIDMER & KUBAT, 1996), onde o modelo é retreinado periodicamente usando-se uma janela deslizante de tamanho fixo sobre os dados antigos; e de forma alternativa, os modelos podem usar mecanismos de alerta (*trigger*) ou detector de mudança ativo para iniciar a atualização do modelo.

De acordo com (ELWELL & POLIKAR, 2011), (MINKU & XAO, 2012) e (GAMA et al., 2004), abordagens com *trigger* determinam como os modelos ou a amostragem devem ser alterados em um determinado momento, quando da determinação do *drift.* Em contrapartida, (Stanley, 2003), (KOLTER& MALOOF, 2005), (KOLTER& MALOOF, 2007) apresentam abordagens nas quais o algoritmo deve encontrar maneiras para se manter atualizado com cada mudança que ocorre.

De uma maneira geral, para enfrentar os desafios impostos pela presença de *concept drift*, foram criados algoritmos tais como VFDT (DOMINGOS & HULTEN, 2000), métodos de janelas deslizantes, detectores de *drift*, métodos de seleção de amostras e comitês de classificadores adaptativos. Para este trabalho o arcabouço desenvolvido procurou explorar especificamente duas das quatro áreas que tornam o modelo adaptativo (ZLIOBAITE, 2010). A primeira está relacionada à formação do conjunto de treinamento e a segunda diz respeito ao uso de comitê de classificadores e suas regras de fusão.

Responder a diversos tipos de *concept drift* é uma tarefa difícil para um classificador simples. Por este motivo, diversos sistemas baseados em comitês de classificadores foram propostos para lidar com aprendizagem em *concept drift* tais como em (STANLEY, 2003), (STREET & KIM, 2001), (WANG et al., 2003), (ESCOVEDO, 2015). Nestas abordagens o conhecimento adquirido pelos membros do comitê é agregado de alguma forma para chegar a uma solução global que é supostamente superior àquela obtida por qualquer um dos componentes isolados.

Existem alguns meios para um comitê se manter adaptativo quando este não utiliza método de detecção. São métodos relacionados à diversidade na construção dos classificadores base, estratégias de treinamento, e avaliação das regras de combinação de seus membros e mudanças na estrutura do comitê. Neste trabalho, optou-se pela

escolha da última opção. Nestes algoritmos, em caso de mudança no ambiente, os classificadores individuais são reavaliados e segundo um critério específico, estratégias evolutivas são aplicadas na manutenção do comitê, tais quais: exclusão, adição, substituição de um classificador.

A Seção 3.4 fornece mais detalhes e exemplos do método escolhido para o desenvolvimento do modelo e o Capítulo 4, o funcionamento das estratégias evolutivas adaptativas aplicados no comitê de classificadores.

3.4 Comitê de classificadores

Um comitê de classificadores é baseado na combinação das saídas de um conjunto de classificadores, podendo estes serem de tipos diferentes ou não, chamados, nesse contexto, de classificadores base. A combinação de resultados de classificadores ou regras de seleção geralmente são chamadas de regras de fusão. O objetivo desta combinação é conseguir uma classificação mais acurada. O aumento de desempenho tende a ocorrer se forem respeitados critérios de diversidade entre os classificadores componentes do sistema, neste caso, a diversidade é percebida se os erros dos classificadores base não coincidem, isto é, se os classificadores base não cometerem os mesmos erros quando classificam os mesmos objetos (MINKU et. al, 2010), (ZLIOBAITE, 2010). Neste caso, diz-se que os classificadores generalizam de forma diferente.

Comitês de classificadores podem utilizar mecanismos de adaptação via *trigger* ou de adaptação contínua. Na adaptação contínua a regra para combinação do modelo ocorre de forma dinâmica baseada no desempenho. Na adaptação através de *trigger*, pró-ativamente é atribuído o modelo mais relevante para a decisão baseado no contexto. Além disso, o processo base de funcionamento deste método inclui a leitura dos dados, a construção dos classificadores, a sua avaliação e descarte dos membros.

O algoritmo SEA (STREET & KIM, 2001) é um dos primeiros algoritmos utilizados para lidar com *concept drift* através do uso de um comitê de classificadores. Um classificador é treinado separadamente em cada lote sequencial do conjunto de treinamento. Um classificador treinado é adicionado ao conjunto, enquanto o classificador de pior desempenho é descartado. A previsão final para a classe é feita usando votação majoritária simples.

O algoritmo DWM (KOLTER& MALOOF, 2007) utiliza um comitê de

classificadores incremental baseado em votação majoritária ponderada, e todos os classificadores pertencentes ao comitê possuem um peso. Esse peso é controlado por uma constante multiplicativa β, onde os classificadores que erram são diminuídos em β, e não há limite de tamanho no comitê.

O DWM elimina dinamicamente os especialistas em resposta a mudanças no desempenho. Para evitar a criação de um número excessivo de classificadores, o DWM remove aqueles cujo desempenho é ruim historicamente e com peso abaixo de um limite θ.

(WANG et.al, 2013) propuseram o algoritmo *Accuracy Weighted Ensemble* (AWE) que de forma similar ao SEA, utiliza o mesmo esquema de divisão do conjunto de treinamento. Adicionalmente, as respostas dos classificadores são combinadas usando votação majoritária ponderada. Os pesos dos classificadores dependem da acurácia obtida por eles usando as amostras do bloco de treinamento atual. De forma similar ao SEA, o algoritmo se adapta bem a *drift* do tipo gradual, porém, apresenta problemas ao tipo abrupto; uma das razões é devido à ineficiência do AWE de ter que esperar pelo próximo bloco para ter que atualizar os pesos dos classificadores. Infelizmente, apenas reduzir o tamanho do bloco não resolve o problema pois compromete a acurácia do sistema de forma global.

Desta maneira há diversos métodos que podem ser usados em comitês para a adaptação ao *concept drift*. Existe um grande interesse da comunidade em melhorar os métodos propostos bem como apresentar novas soluções frente aos desafios que as novas tecnologias impõem. Muitos outros algoritmos estão disponíveis para estudo e pesquisa, tais como (ELWELL & POLIKAR, 2011), o qual utiliza uma estratégia para dinamicamente atribuir pesos aos classificadores com objetivo de encontrar o equilíbrio entre estabilidade-plasticidade; Em (FAN, 2004b), o autor critica a noção de que conceitos antigos auxiliam na geração da uma nova hipótese mais acurada do que usar somente os conceitos mais recentes. O autor propõe um algoritmo baseado em comitê de árvores de decisão com validação cruzada para selecionar e comparar todas as sensíveis escolhas de treinamento de um classificador; (KOLTER & MALOOF, 2003) criaram um método o qual dinamicamente cria e remove classificadores do comitê, sendo estes classificadores ponderados; Em (MINKU & YAO, 2012) é apresentado um estudo sobre a análise da diversidade sob diferentes tipos de *drifts* e como esta variável atua no comportamento de um comitê; (SCHOLZ & KLINKENBERG, 2007) propõem um método do tipo "*boosting*" para treinar os classificadores no comitê, no qual o

modelo induzido é reponderado continuamente considerando somente o último bloco de dados.

Neste estudo, optou-se por um comitê formado por APF sem métodos para ponderar ou ajustar o peso dos classificadores por questões de implementação e para deixar o arcabouço mais simples, porém governado por estratégias que visam atuar em sua manutenção adaptativa buscando mantê-lo acurado com o passar do tempo.

4. MÉTODO PROPOSTO

Este capítulo apresenta a solução proposta para aprendizagem em ambientes não estacionários com a presença de *concept drift*. O capítulo está organizado em cinco seções: a Seção 4.1 descreve brevemente alguns fundamentos utilizados no modelo, a Seção 4.2, a solução proposta e visão geral do modelo, a Seção 4.3 aborda alguns detalhes sobre *fuzzy*, a Seção 4.4 fornece informações sobre a PGC, e por fim, as árvores na Seção 4.5.

4.1 Fundamentos

Antes de entrar nos detalhes da abordagem proposta são apresentados na Seção 4.1.1 e Seção 4.1.2 alguns fundamentos importantes que foram usados nesta pesquisa.

4.1.1 Método Bagging

O método *Bagging* é um método de agregação utilizado para gerar múltiplas hipóteses de uma preditor (classificador/regressor) gerando um preditor agregado com melhor desempenho (BREIMAN, 1996). Este método usa amostras *bootstrap* do conjunto de treinamento (EFRON, 1979). *Bootstrap* é uma técnica de amostragem com reposição: a partir do conjunto de treinamento inicial, são selecionados aleatoriamente exemplos para um novo subconjunto de treinamento. Assim, no método *Bagging* diferentes subconjuntos são aleatoriamente construídos, com reposição, a partir do conjunto original. Devido à reposição, alguns exemplos podem ser escolhidos mais de uma vez e outros podem não ser escolhidos, conforme exemplo demonstrado na Figura 8.

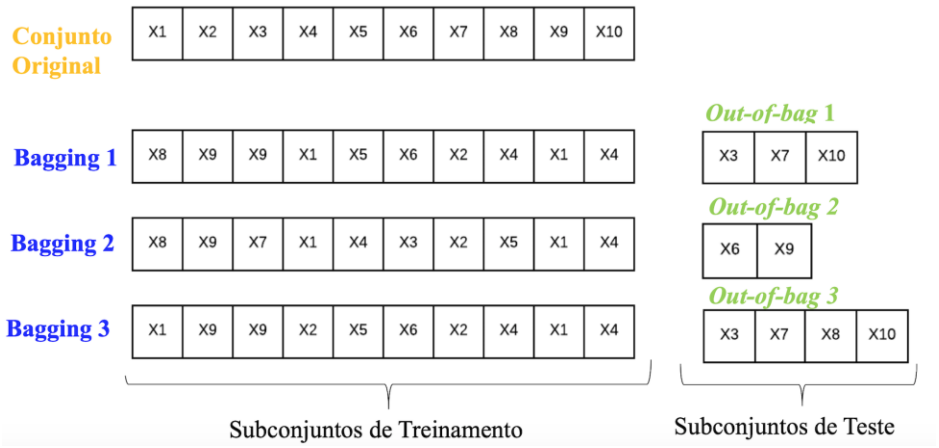


Figura 8: Método Bagging (a).

Na amostragem *bootstrap* cada subconjunto gerado possui o mesmo tamanho (número de exemplos) do conjunto original. Considerando um conjunto de treinamento *T* com *n* exemplos, T_k é uma amostra *bootstrap* do conjunto de treinamento a partir de *T* com reposição, contendo *n* exemplos. Cada subconjunto T_k é usado para treinar um classificador diferente $\{h_k(x)\}$. Em seguida, deve-se agregar as respostas do classificador através de um método de combinação apropriado, tal como o voto majoritário (BREIMAN, 1996), representado na Figura 9.

Para um valor grande de *n*, isto é aproximadamente $1 - 1/e \approx 63{,}2\%$, o que significa que cada amostra *bootstrap* contém, em média, somente cerca de 63,2% dos exemplos do conjunto de treinamento original *T*, com muitos exemplos de treinamento aparecendo múltiplas vezes (BAUER & KOHAVI, 1999), (DIETTERICH, 2000). Os exemplos que não foram selecionados na amostra T_k formam o conjunto *out-of-bag* demonstrado na Figura 8, ou seja, são os exemplos que não foram utilizados para o treino na construção do classificador h_k .

Assim, a avaliação *out-of-bag* funciona da seguinte maneira: dado um conjunto de treinamento específico *T*, geram-se conjuntos de treinamento *bootstrap* T_k, constroem-se classificadores $\{h_k(x)\}$ e os deixam votarem para criar o classificador *bagged*. Para cada (*x*,*y*) no conjunto de treinamento, agregam-se os votos somente sobre os classificadores para os quais T_k não contém (*x*, *y*). Este é o meio pelo qual *out-of-bag* fornece uma estimativa para o erro de generalização (BREIMAN, 2001).

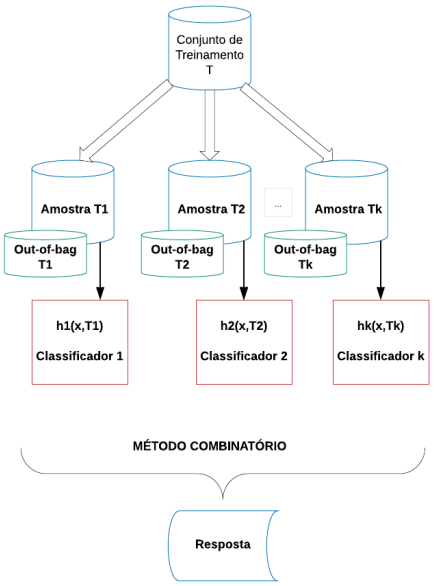


Figura 9: Método Bagging (b).

O método de *Bagging* foi utilizado como inspiração para a implantação de uma das estratégias de aprendizado adaptativo utilizada neste trabalho: a formação do conjunto de treinamento. Foi usado o mecanismo de divisão do conjunto de dados original em subconjuntos *bagging* e *out-of-bag*. O mesmo subconjunto *bagging* é usado no treinamento de cada classificador no comitê. Da mesma maneira o mesmo subconjunto *out-of-bag* é usado para a estimativa de erro de generalização (BREIMAN,1996a) para cada classificador no comitê. Posteriormente, os votos fornecidos por cada um dos membros do comitê são agregados, fornecendo a resposta final agregada, conforme exibido na Figura 9. Maiores detalhes da aplicação do método poderão ser vistos na Seção 4.2.

4.1.2 Função de ajuste "*Hits-Then-Errors*"

A função "*Hits-Then-Errors"* ou "Acertos-Então-Erros" foi utilizada em (DIAS, 2010) como função de avaliação em problemas de classificação. Esta função nesta dissertação teve como objetivo auxiliar na medição do melhor classificador no comitê através da análise dos erros de previsão quando a medição do desempenho no comitê indica a necessidade de adoção de uma estratégia adaptativa, e os desempenhos individuais dos classificadores no comitê não divergem em valores. A função é composta por alguns passos que serão descritos a seguir:

1- Cálculo percentual dos acertos na previsão das classes pelos classificadores envolvidos conforme equação abaixo:

$$PA = 100 \text{ X } \frac{1}{N} \sum_1^n a_i \text{ ,} \quad (6)$$

onde a_i é igual a um para os casos onde o classificador acerta a classe *i* e zero para o caso de erro.

2- Caso os classificadores possuam percentual de acertos idênticos, deve-se prosseguir para o passo 3, caso contrário, o classificador no comitê com o maior percentual de acertos será considerado o melhor.

3- Calcula-se o erro médio absoluto (*Mean Absolute Error* - MAE) para as *n* amostras que devem ser classificadas, computando-se o erro médio das

respostas das classes previstas pelos classificadores para cada amostras, conforme equação descrita abaixo:

$$\text{MAE} = \frac{1}{N} \sum_{1}^{n} |Y_i - X_i|, \quad (7)$$

onde Y representa o valor real da classe e X, o valor previsto pelo classificador.

4.2 A solução proposta

4.2.1 Visão geral do método proposto

Sistemas *Fuzzy* Baseados em Regras (SFBR) são metodologias fundamentais para representar e processar informações linguísticas, com mecanismos para lidar com incerteza e imprecisão. Contudo, modelos baseados em regras podem se tornar complexos quando o número de variáveis no conjunto de dados é grande. As árvores de padrões *fuzzy* podem ser consideradas uma alternativa para lidar com esta questão pois não se baseia no sistema de regras e sua representação visual hierárquica facilita a compreensão.

Diversas áreas de estudo têm proposto metodologias com objetivo de construir sistemas *fuzzy* por meio de algoritmos genéticos ou programação genética (HONG et al., 2006), (TSANG et al., 2007). O método aqui descrito, seguindo tendências de pesquisas em sistemas f*uzzy* e programação genética (HUANG et al., 2008) e (BASTIAN, 2000), (GEYER-SCHULZ, 1997), (BERLANGA et al., 2006a), (SANTOS, 2014) propõe o uso de um caso particular da programação genética, a programação genética cartesiana (PGC), na indução das árvores de padrões *fuzzy*.

Como já mencionado na Introdução deste documento, com o objetivo de propor um modelo que seja acurado, porém interpretável e adequado para ambientes não estacionários, foi proposta uma metodologia para gerar um comitê de árvores de padrões *fuzzy* induzidos pela programação genética cartesiana. Esse modelo deve ser capaz de aprender e lidar com problemas de *concept drift* em ambiente de aprendizado supervisionado quando os rótulos das classes não estão disponíveis no momento do treino.

Nesse modelo, cada árvore de padrão *fuzzy* é treinada através da PGC gerando genótipos os quais deverão ser submetidos ao erro de estimativa. Se o desempenho das árvores em comitê não estiver de acordo com o esperado, devem ser aplicadas

estratégias para que o modelo se mantenha autoadaptável.

A seguir serão descritas as etapas necessárias para o funcionamento da solução proposta e as escolhas de projeto realizadas. A Figura 10 oferece uma visão geral das duas grandes fases do modelo: inicialização/monitoramento e atualização; e as etapas existentes em cada uma delas.

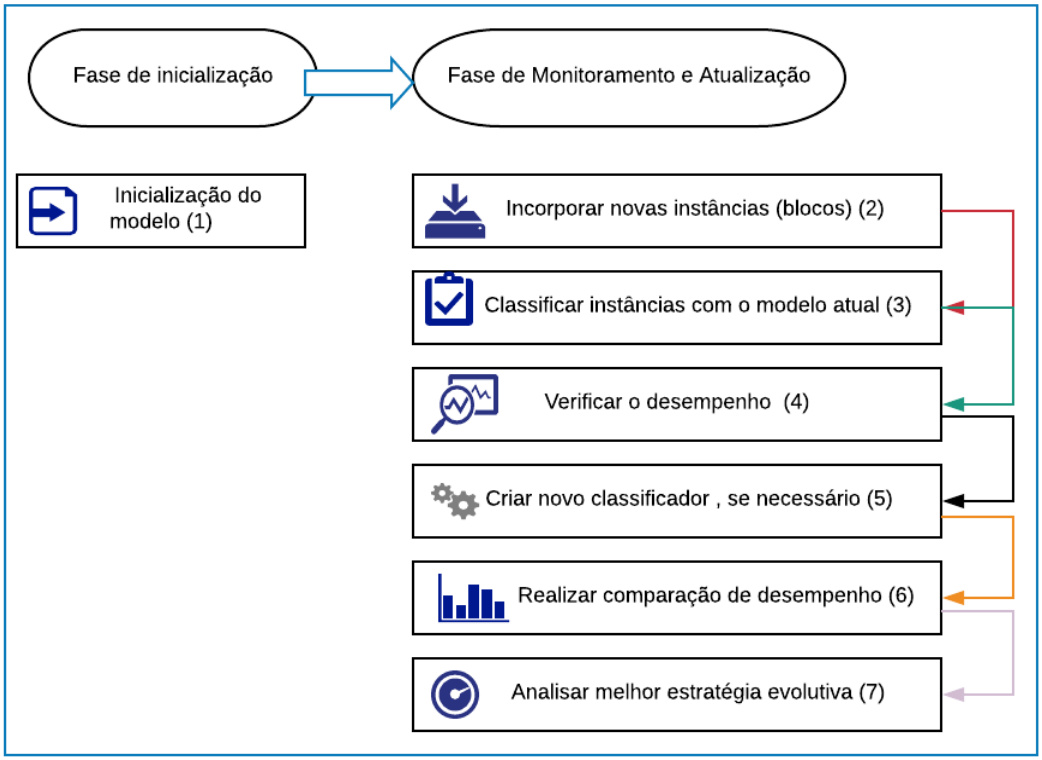


Figura 10: Visão geral da solução proposta.

1) Fase de inicialização: gerar um modelo de classificação inicial a partir de uma base de dados de treinamento com exemplos rotulados, chamado de base de dados histórica. Esse modelo inicial será usado para classificar as instâncias do fluxo de dados para obter a classe prevista da instância e compará-la com a classe real posteriormente.

2) Fase de monitoramento e atualização: monitorar o desempenho do modelo e determinar se o modelo precisa adaptar-se. Se afirmativo, fornecer os meios para atualizá-lo. O modelo é composto pelos módulos:

- Módulo *boostrap*: *bags* e *out-of-bag.*

- Módulo de criação do classificador.
- Módulo de Previsão.
- Módulo de Comparação.
- Módulo de Estratégias.

Cada um dos módulos descritos anteriormente irá desempenhar uma tarefa específica conforme ilustrado na Figura 11.

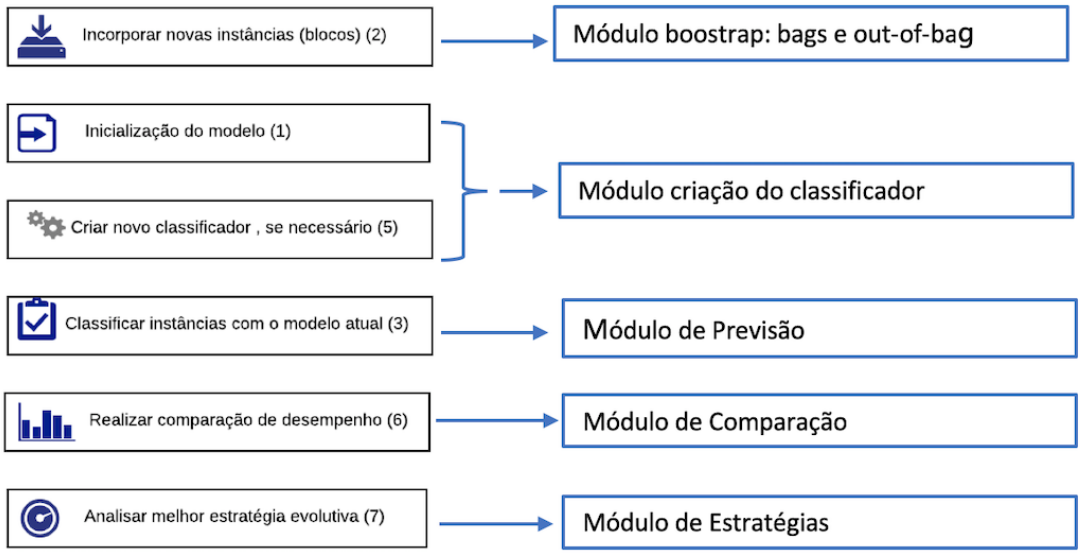


Figura 11: Módulos da fase de monitoramento e atualização.

4.2.2 O método proposto – descrição detalhada

Conforme exibido na Figura 12, o *input* para o funcionamento do método proposto é um conjunto de treinamento inicial, chamado de base de dados histórica, enquanto o resultado final é um modelo muito próximo dos conceitos atuais e recentes do fluxo de dados. A Figura 12 exibe a visão detalhada das etapas no decorrer do tempo, as quais são descritas a seguir de forma sequencial indicada pela numeração na Figura 10.

(1) Classificador Treinado PGC - Inicialização do modelo: a primeira etapa na solução proposta é criar um classificador treinado com uma base de dados histórica. Neste trabalho foi escolhido Árvores de Padrões *Fuzzy*, porém, poderia ser qualquer classificador – por exemplo, SVM. Desta maneira, foi induzida pela PGC um classificador do tipo Árvore de Padrões *Fuzzy* (APF1), treinado usando-se uma base

de dados histórica. Este classificador é adicionado ao comitê, sendo considerado o modelo inicial que será usado na classificação das amostras não rotuladas que serão apresentadas ao modelo.

(2) Incorporar novas instâncias (blocos): existem dois tipos de entrada de dados que os algoritmos podem usar: entrada individual e entrada em bloco. Na entrada individual o algoritmo aprende uma entrada por vez, já na entrada em blocos, as amostras são armazenadas sob a forma de bloco. Cada um dos tipos oferece vantagens e desvantagens; por exemplo, na entrada individual há mais plasticidade, porém, é mais suscetível a ruídos, já a entrada em blocos oferece mais estabilidade, porém o desafio é determinar o tamanho mais adequado para o bloco. Neste trabalho optou-se pela configuração da entrada de dados sob a forma de blocos de dados configuráveis, sendo estes do mesmo tamanho. Dessa maneira, nesta etapa, as instâncias são lidas uma a uma a partir de *stream* de dados e coletadas sob blocos. É nesta etapa também que se inicia o funcionamento da fase de monitoramento/atualização cujo primeiro passo é realizado pelo módulo *boostrap.* O módulo *boostrap* é acionado para a geração de subconjuntos *bagging* e *out-of-bag* dos dados de entrada organizados em blocos.

(3) Classificar instâncias com o modelo atual: o modelo atual classifica cada instância na forma de (x_i, y_i) e prevê sua classe c_k, onde x_i representa os atributos da instância, y_i seu rótulo e c_k é a classe prevista, sendo o módulo de Previsão o responsável por esta classificação. Cabe informar que a previsão é realizada sobre o bloco não rotulado. Neste trabalho pressupõe-se um atraso nos rótulos das classes. As respostas individuais dos classificadores no comitê são avaliadas e através de processo de votação majoritária é escolhida a classe que representa a resposta global do comitê.

(4) Verificar o desempenho: nesta etapa assume-se que os rótulos das amostras do bloco que foi classificado na etapa anterior (3) tornam-se disponíveis. Comparam-se todas as classes previstas com o rótulo da instância y_i. Se $(y_i \neq c_k)$, o módulo de Desempenho irá verificar se ocorreu *drift* através de uma medida de desempenho, pré-estabelecida. Pressupõe-se a ocorrência de *drift* caso esta medida esteja abaixo do configurado. Para este trabalho, dentre as várias medidas de desempenho disponíveis tais quais *F-measure*, AUC, *Recall*, adotou-se a Acurácia, por ser esta uma das medidas de desempenho mais utilizadas em aprendizado de máquinas, incluindo os autores escolhidos para os estudos de caso que serão analisados na Seção 5 . A taxa de

acertos mínima configurada para todos os estudos de caso foi de 85% pois estudos preliminares indicaram que este limiar era possível de ser atingido embora este valor tenha sido fixado, pode-se tratá-lo como um parâmetro no modelo. O modelo deverá adaptar-se caso a acurácia média do comitê obtida na classificação das instâncias na etapa 3 esteja abaixo do pré-estabelecido.

(5) Indução PGC - Criar novo classificador, se necessário: o desempenho do comitê medido na etapa 4 irá indicar se é necessário criar um novo classificador para atualizar o comitê. Se o desempenho for ruim, um novo classificador Árvore de Padrões *Fuzzy* (APF2) será criado e treinado como o subconjunto *bagging*, e testado com o subconjunto *out-of-bag* do bloco previsto na etapa 3, gerando desta maneira uma estimativa de erro. Esta nova APF2 também será avaliada com o subconjunto *out-of-bag* do bloco anterior, gerando uma segunda estimativa de erro. Esta atividade é realizada pelo módulo Criar Classificador.

(6) Realizar comparação de desempenho: esta tarefa é desempenhada pelo módulo de Comparação. Nesta etapa, pressupõe-se que o desempenho no comitê está aquém do esperado e houve a necessidade de se criar um novo classificador. Nesse caso, o módulo de Comparação verifica os valores obtidos na estimativa de erros no subconjunto *out-of-bag*. Essa comparação é feita tanto pelo resultado fornecido pelo comitê testado com *out-of-bag* do bloco atual e do bloco anterior, quanto pelo novo classificador criado (APF2) na etapa 5. É realizada a comparação descrita na Tabela 4, sendo:

$$\text{Erro 1} = \text{Val1} + \text{Val2} \text{ e } \text{Erro 2} = \text{Val3} + \text{Val4}. \quad (8)$$

Classificador/ Erros *Out-of-Bag*	**Erro *Out-of-Bag* subconjunto do bloco atual**	**Erro *Out-of-Bag* do bloco anterior (*)**
Comitê	Val1	Val2
Classificador novo	Val3	Val4

Tabela 4: Erros out-of-bag comitê X classificador novo.

Dessa maneira é possível verificar o desempenho do comitê e do novo classificador tanto sob a perspectiva de conceitos antigos quanto de conceitos atuais. De acordo com o resultado obtido em Erro1 e Erro2, uma estratégia adaptativa deverá ser realizada a seguir na etapa 7.

No primeiro ciclo de execução do método, o erro *out-of-bag* do bloco anterior será o erro *out-of-bag* obtido na previsão dos dados históricos.

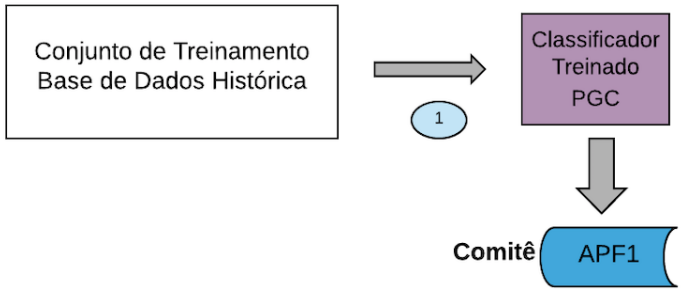
(7) Analisar a melhor estratégia de atualização do comitê: esta etapa define qual estratégia adaptativa deverá ser realizada para que o desempenho no modelo volte a crescer; o módulo Estratégias baseia-se na resposta obtida na etapa 6. Abaixo estão listadas as estratégias de atualização que tem por objetivo manter o modelo adaptável:

- (1) Adição: a condição para adição de um novo classificador é que exista espaço para a inserção do mesmo no comitê.
- (2) Troca: se o teste no Módulo de Comparação (etapa 6) apresentar um erro produzido pelo comitê na estimativa de erro *out-of-bag* for maior do que o erro *out-of-bag* produzido pelo novo classificador , ou seja, Erro1 > Erro2, então, o classificador no comitê com a pior resposta deverá ser substituído pelo novo classificador .
- (3) Exclusão: não sendo possível adicionar um classificador novo ao comitê, bem como trocar um classificador, a estratégia a ser adotada é a exclusão do classificador que tiver a pior resposta, ou seja, a menor acurácia. Este classificador deverá ser selecionado para ser excluído. Caso ocorra empate quanto à medição da acurácia, a função *hits-then-error* será usada para escolher o classificador que será eliminado. Um dos classificadores é eliminado e um teste é feito para medir se o comitê sem o classificador é melhor do que o comitê com a configuração anterior. Caso o comitê sem o classificador temporariamente excluído gere um resultado pior, o classificador excluído deverá retornar para o comitê.
- (4) Imutável: esta estratégia não altera a estrutura do comitê, apenas sinaliza que não foi realizada nenhuma mudança na estrutura do comitê, seja porque o desempenho do comitê está bom ou porque nenhuma das estratégias (adição, troca ou apagar) eram apropriadas.
- (5) Aleatório: esta estratégia foi criada para adicionar diversidade ao comitê. De forma aleatória são sorteados parâmetros para o funcionamento da APF e PGC a partir de uma lista previamente organizada. É gerado um novo classificador treinado com o subconjunto *bag* do bloco atual. Este novo classificador será adicionado ao comitê. Esta estratégia é selecionada a cada 5 estratégias do tipo (4) Imutável, esta medida foi definida através de experimentos preliminares não

documentados neste estudo.

O diagrama representado pela Figura 12 fornece uma visão geral da solução proposta em funcionamento.

Fase (1): inicialização



Fase (2): monitoramento e atualização

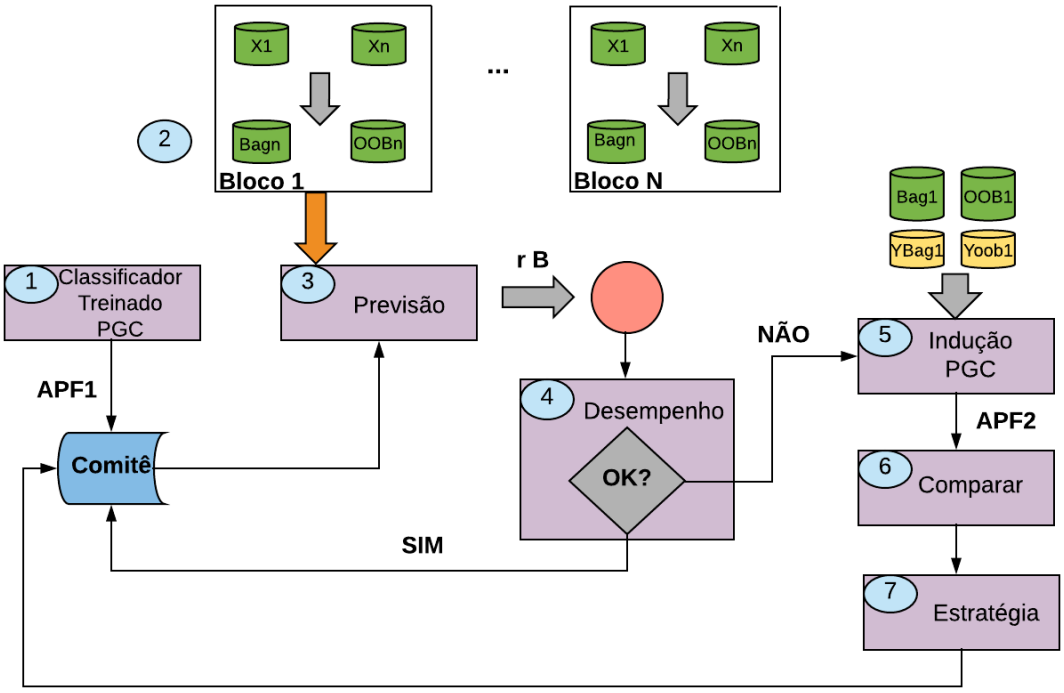


Figura 12: Visão geral do arcabouço.

O processo se inicia com um bloco de dados rotulado de uma base de dados histórica. Este bloco é processado pelo método *bagging* para gerar subconjuntos, dois novos blocos os quais foram denominados BBh (Bloco *Bag* Histórico) e OOBh (*Out-of-Bag* Histórico). O algoritmo da programação genética cartesiana (PGC) realiza a indução da nova Árvore de Padrões *Fuzzy* (APF1). Esta árvore, APF1 é treinada com amostras do bloco BBh e adicionada ao comitê, sendo este testado com o OOBh. Os erros gerados na previsão *Out-of-Bag* são armazenados pois serão usados mais adiante. Desta maneira, conclui-se a fase de inicialização.

Na fase de monitoramento/controle todos os blocos serão processados para gerar novos blocos de *Bag* e *Out-of-Bag*, denominados *Bag* do Bloco 1 (BB1), *Bag* do Bloco 2 (BB2) e sucessivamente até a chegada do n-ésimo bloco (BBn), da mesma maneira, OOB1 (*out-of-bag* do bloco1), OOB2 até OOBn.

Desta maneira ao chegar o bloco número 1, o comitê realiza a previsão deste bloco com base nos atributos destas amostras e os valores resultantes da previsão são armazenados. Como há apenas uma árvore no comitê, não há necessidade da determinação dos votos dos membros, e o resultado final do comitê é o próprio resultado da APF1. O desempenho do modelo é medido de acordo com o método escolhido para avaliação de desempenho; assume-se que neste ponto os rótulos do bloco 1 já estão disponíveis. Logo, desempenho ruim indica possível ocorrência de *drift.* Uma nova Árvore de Padrões *Fuzzy* (APF2) será criada usando para treinamento, amostras (*bag)* de BB1 (bloco atual). Esse novo classificador será testado com (*out-of-bag)* do bloco 1 (OBB1). Contudo, também deve ser avaliado como APF2 se comporta na previsão dos conceitos antigos. Para este primeiro ciclo isso é representado pelo *out-of-bag* da base histórica (OOBh).

A fim de verificar o resultado alcançado pela nova árvore em relação ao comitê, é feita uma comparação avaliando-se os resultados produzidos por APF2 sobre os erros OOB1 (conceitos atuais) e erros OOBh (conceitos antigos). Da mesma maneira é necessário avaliar como o comitê composto por APF1 se comporta na previsão de OOB1.

De posse dos resultados da avaliação dos conceitos novos e conceitos antigos gerados pela nova árvore APF2 e pelo comitê composto por APF1, é possível comparar quem teve o melhor resultado, ou seja, o menor erro. De maneira estratégica é decidido

o que deverá ser alterado no comitê para que o desempenho se mantenha estável. Se houver espaço disponível no comitê, a estratégia será a adição, caso contrário a APF1 poderá ser trocada por APF2, excluída ou o comitê permanecerá imutável.

Ao chegar o bloco2, a previsão do bloco é realizada pelo comitê e os resultados são medidos. O desempenho é novamente verificado e, caso a acurácia esteja abaixo do valor pré-estabelecido é gerado um novo classificador APF3, o qual deverá ser treinado com BB2 e testado com OBB2. Comparam-se os resultados obtidos pelo comitê sobre os erros OBB2 e erros OOB1, e APF2 sobre os erros OOB1 e os erros OBB2. A estratégia a ser tomada depende dos resultados obtidos acima.

Assim, o funcionamento do modelo pode ser generalizado em: ao chegar o bloco *t*, o comitê contendo um único classificador APF1 realiza a previsão usando-se os atributos deste bloco e o resultado final da classificação será dado pelo método de votação escolhido. O desempenho do comitê no boco *t* é medido, nesta etapa, quando então os rótulos reais já estão disponíveis. Caso este desempenho não esteja satisfatório, através da Programação Genética Cartesiana uma nova árvore de padrões *fuzzy* é gerada (APF2), a qual será treinada com BBt e testada com OOBt. Em seguida comparam-se os erros produzidos pelo novo classificador e pelo comitê, tanto na previsão em OOBt (conceitos atuais) quanto em OOBt-1(conceitos anteriores). O resultado desta avaliação, baseado na soma de quem errou menos, irá conduzir a estratégia a ser tomada em relação à configuração do comitê, sendo estas: adição, troca, exclusão, imutável ou aleatório.

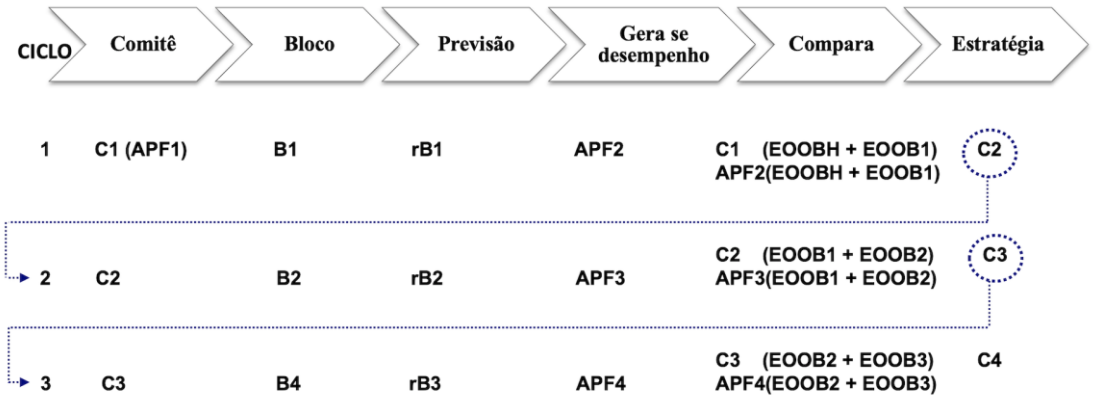


Figura 13: Evolução do comitê ao longo do tempo.

A Figura 13 exibe de forma gráfica o resumo do funcionamento da solução proposta no decorrer do tempo. Cabe citar: APFH - primeira árvore de padrão *fuzzy*

criada e treinada com a base de dados histórica; C1 – comitê inicial; EOOBH – erros na estimativa *out-of-bag* da base de dados histórica; EOOB1 – erro na estimativa *out-of-bag* do subconjunto gerado a partir do bloco número 1;

4.3 Particionamento e Operadores *Fuzzy*

Nesta etapa são definidas as variáveis linguísticas (termos *fuzzy*) bem como as funções de pertinência. Esta etapa ocorre com os dados disponíveis para treino e é utilizado no processo de formação da montagem da APF. Dessa maneira, a base de dados histórica, as amostras reunidas em blocos e os subconjuntos *Bag* e *Out-of-Bag* são fuzzificados assumindo termos *fuzzy* dentro de um universo de discurso com intervalo [0,1]. O valor de cada atributo ativa a função de pertinência de cada termo *fuzzy*. Cada atributo presente na base de dados terá um valor de pertinência associado a cada termo *fuzzy*. Portanto, o número de entradas a ser apresentada à PGC é de 5 vezes para cada atributo. A Tabela 5 apresenta os termos *fuzzy* utilizados, bem como os operadores *fuzzy* T-norma, T-conorma e os códigos inteiros que foram usados para representá-los no desenvolvimento da solução. Esses parâmetros seguiram as opções adotadas em (SANTOS, 2014).

Termos *Fuzzy*	**Operadores**		**Códigos**
Baixo	Máximo	Max(a, b)	0
Médio-Baixo	Mínimo	Min(a, b)	1
Médio	WA	x*a+(1-x)*b	2
Médio-Alto	OWA	x*Max(a,b)+(1-x)*Min(a, b)	3
Alto			

Tabela 5: Operadores.

A escolha adequada do particionamento agirá diretamente no desempenho do classificador, pois caso ocorram poucos conjuntos *fuzzy* por variável as amostras contidas no bloco de dados podem ser classificadas erroneamente, bem como se forem excessivos, podendo faltar amostras suficientes para o aprendizado.

Os operadores “média” (WA) e “média ordenada” (OWA) preenchem o espaço entre a maior interseção (o operador Min) e a menor união (o operador Max), sendo os valores de *a* e *b* as entradas dos nós que serão operados, e *x* é um valor aleatório parametrizado dentro intervalo [0,1], que pode ter seu valor ajustado pela mutação, trazendo o benefício de melhoria do desempenho do modelo.

4.4 Programação genética cartesiana

4.4.1 Parâmetros da PGC

Nesta etapa é realizado o treinamento do modelo. A grade computacional da PGC, representada pela quantidade de linhas e colunas exemplificado na Figura 4 irá receber o conjunto de dados fuzzificado na etapa anterior. A programação genética cartesiana exige que sejam definidos parâmetros para o seu funcionamento sendo estes: número de linhas, o número de colunas, o número de entradas e o número de saídas, o *levelback*, a quantidade máxima de gerações, o tamanho da população, a taxa de mutação do genótipo, a função utilizada para a avaliação do genótipo e os parâmetros desta. Estes parâmetros foram escolhidos de acordo com o estudo de caso descrito na seção 5.4.

Dentre as estratégias evolutivas que podem ser utilizadas na programação genética cartesiana, optou-se pela estratégia $1 + \lambda$ sendo $\lambda = 4$, mais tradicionalmente utilizada (RECHENBERG, 1971). Desta maneira, serão escolhidos 5 indivíduos para constituir a população que inicialmente será criada aleatoriamente, limitada pelos parâmetros definidos pelo programador. Este processo evolutivo irá ocorrer até que se atinja o número máximo de gerações ou que se encontre um indivíduo com erro (custo) nulo. A cada geração, realiza-se a decodificação e o cálculo de aptidão da população.

4.4.2 Genótipo

Neste trabalho optou-se por usar umas das duas possíveis variantes para a relação genótipo-classe utilizadas em (SANTOS, 2014), sendo esta representada por um genótipo para cada classe. Devido à estratégia evolutiva utilizada, a população neste caso será sempre igual a cinco vezes o número de classes, os genótipos são construídos por uma sequência de números inteiros e, por padrão, o parâmetro número de linhas da programação genética cartesiana foi fixado em 1, com o parâmetro número de colunas podendo variar. A quantidade total de genes será o número de colunas multiplicado por três, somado à quantidade de genes de saída.

Na Figura 14 é exibido um genótipo com *n* saídas. Cada coluna possui um nó e cada nó, 3 genes, onde o primeiro gene representa um dos operadores citados na Tabela 5, e o segundo e terceiro genes representam as coordenadas, ou seja, os genes de conexão de entrada e saída do nó, os nós 4, 5 e 6, por exemplo, representam casos onde ocorreram redundância. A redundância ocorre quando genes associados a um nó, não fazem

parte do grafo que está conectado, ou seja, dos nós que estão conectados a saída através de outros nós ou ligados diretamente. A redundância provoca a neutralidade e esta aumenta a variabilidade genética.

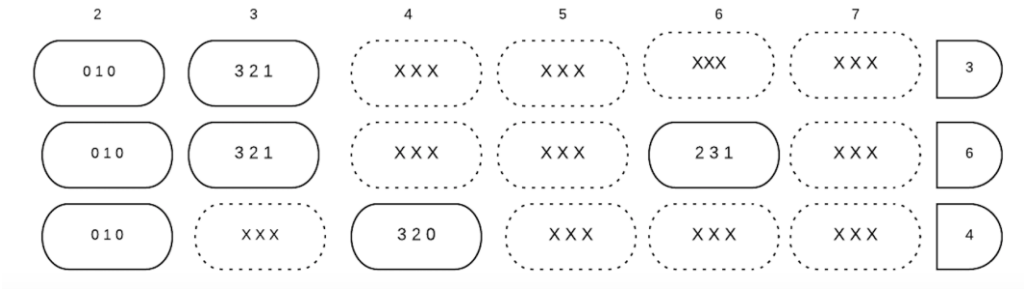


Figura 14: Figura 16: Genótipo com 3 saídas.

4.4.3 Função de Aptidão

O cálculo de custo para cada indivíduo da população é feito usando-se uma função de aptidão que visa a medir dois objetivos distintos: acurácia e interpretabilidade. A aptidão é calculada na Equação 9 usando-se pesos *w1* e *w2* para ponderar os objetivos. Esses pesos podem ser parametrizados, porém a soma deve ser igual a 1.

$$\text{Cálculo de Custo} = w1 * AP1 + w2 * AP2 \qquad (9)$$

A primeira parcela mede a acurácia e diversas funções podem ser utilizadas. Contudo, através de experimentos realizados neste estudo e medição dos resultados obtidos, optou-se pela Área sob a Curva ROC AUC (FAWCETT, 2006) para representar o termo AP1. Já AP2 representa a interpretabilidade, ou seja, ela relaciona o número total de genes ativos X número total de genes, conforme descrito na Equação 10:

$$AP2 = \frac{\text{NúmeroTotalGenes} - \text{GenesAtivo}}{\text{NúmeroTotalGenes}} \qquad (10)$$

A medida de desempenho utilizada na avaliação dos genótipos foi AUC, a qual não deve ser confundida com a medida de desempenho utilizada na etapa 4 onde é feita a verificação do desempenho do comitê.

4.5 Fenótipo da solução

O cromossoma da solução é lido gene por gene através de um método de decodificação que parte da saída e rastreia todo o genótipo para encontrar os genes que estão conectados a esta saída. Nesse processo, de forma recursiva, as coordenadas dos pontos de entrada e saída são lidas até que seja possível formar a solução, e obter a informação para a construção da árvore. A Tabela 5 exibe os códigos utilizados para os operadores e a Figura 15 exibe as 3 árvores fornecidas como saída para o exemplo da Seção 4.4.2 com um genótipo para cada classe.

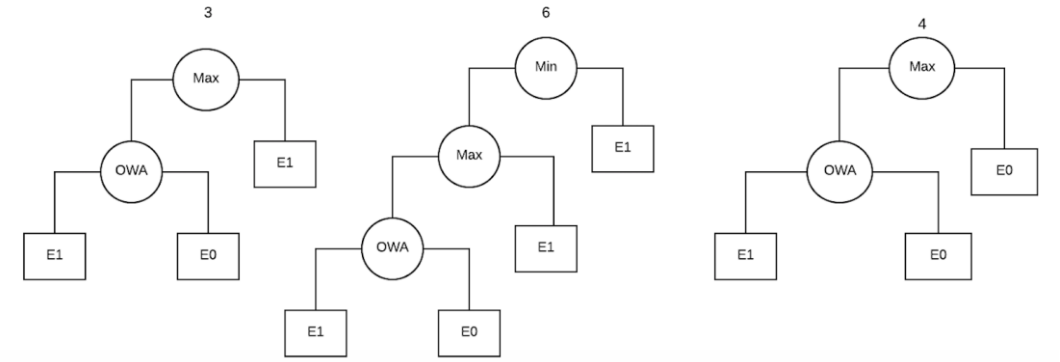


Figura 15: Representação em árvore.

5. ESTUDO DE CASO

Este capítulo apresenta os resultados experimentais realizados com o método apresentado nesta dissertação. Foram realizados experimentos com o objetivo de avaliar o comportamento do método *fuzzy*-genético, bem como a melhor configuração do modelo em termos de acurácia e interpretabilidade.

5.1 Visão Geral

A Seção 5.3 descreve experimentos que consistem na utilização do método proposto em bases de dados estáticas. Na Seção 5.4 são apresentados outros experimentos, onde o método é testado com diferentes bases de dados reais e artificiais na presença de *concept drift*. No total, sete bases de dados foram selecionadas: duas estáticas e cinco não estacionárias sendo: três bases de dados artificiais e duas bases de dados reais. Estas bases de dados são conhecidas na literatura e já foram empregadas em diversos trabalhos (ROKACH, 2010), (YIN et al., 2015), (DITZLER & POLIKAR 2013), (JACKOWSKI, 2013), (GAMA et al., 2014) entre outros.

Por fim, a Seção 5.4.6 exibe um gráfico consolidado com as acurácias obtidas nos experimentos bem como as considerações finais.

5.2 Detalhes da execução

O arcabouço proposto neste estudo foi desenvolvido usando-se MATLAB, com o apoio de uma toolbox denominada PRtools4 [1] que funciona em conjunto com o MATLAB para reconhecimento de padrões.

Conforme descrito nos Capítulos 2 e 3, o funcionamento da solução proposta requer a definição de dois conjuntos de parâmetros: um dos conjuntos relaciona-se com as definições necessárias para as árvores de padrões *fuzzy*, sendo estes: o número de partições e operadores *fuzzy*. O outro conjunto está relacionado com a programação genética cartesiana: definição de λ, tamanho da população, número de gerações, função de aptidão, taxa de mutação, número de linhas, número de colunas e *levelback*. Estes parâmetros serão definidos no estudo de caso detalhado na Seção 5.4.1 e Seção 5.4.2.

Decidiu-se pela utilização da abordagem em blocos para o treinamento e avaliação do modelo. Foram escolhidos valores utilizados em outros trabalhos da

[1] Prtool4 está disponível para download e uso livre podendo ser encontrado em: http://www.37steps.com

literatura, como por exemplo (ELWELL & POLIKAR, 2011) e (BRZEZINSKI & STEFANOWSKI 2014). A medida de desempenho utilizada foi a acurácia, e como forma de validar o resultado obtido pare as bases de dados que possuem desbalanceamento, adotou-se área sob a curva ROC (FAWCETT, 2006).

Para todos os estudos de caso e bases de dados o algoritmo foi executado 30 vezes. Ao final calculou-se a acurácia média, AUC média e o desvio padrão dos resultados.

5.2.1 Teste Estatístico

O teste de hipóteses escolhido para este estudo foi o Wilcoxon (HANLEY & MCNEIL, 1982). Foi determinada a hipótese nula:

(H_0) = não há diferença estatisticamente significativa entre os métodos avaliados.

Para o nível de significância adotou-se o valor mais comumente usado de α = 0,05. O teste de Wilcoxon foi aplicado sobre a acurácia média obtida nos estudos de caso pelo comitê e os valores informados pelos autores dos métodos comparativos dos demais trabalhados utilizados. Para cada base de dados, foi calculada a diferença de desempenho entre os classificadores e o comitê contendo diferentes quantidades de APF's de acordo com estudo de caso. Seguindo a descrição do teste de Wilcoxon, o resultado desta diferença foi ordenado sem considerar os sinais. Posteriormente foi calculada a soma positiva (S_P) e a soma negativa (S_N) das ordens, sendo o valor do teste estatístico representado por $T_{wilcox} = \min(S_P, S_N)$. Para a determinação do valor crítico foi utilizada uma tabela de apoio disponível online em websites sobre assuntos relacionados a estatística, como indicada em (MEIER et. al, 2014), e pode-se por fim comparar se o valor crítico foi menor ou maior do que o valor do teste estatístico T_{wilcox}. Se o valor do teste estatístico foi menor do que o valor crítico, rejeitou-se H_0.

5.3 Base de dados estática

Nesta seção é apresentado um estudo de caso realizado com bases de dados artificiais e estáticas (sem a presença de *concept drift*). Estas bases de dados são de uso amplo na literatura e foram utilizadas para os algoritmos de aprendizado em (WEIHS,

2010), (SANTOS, 2014), (BREIMAN,1996), entre outros. O objetivo neste experimento foi analisar o comportamento do comitê de classificadores e validar o funcionamento da solução para casos sem ocorrência de *drift.* A ideia é garantir que os resultados são equivalentes aos de (SANTOS, 2014).

As bases de dados escolhidas estão listadas na Tabela 6 juntamente com os resultados obtidos no trabalho de (SANTOS, 2014). Desta maneira, os parâmetros utilizados neste experimento seguiram a configuração que apresentou o melhor resultado no estudo mencionado anteriormente. Deseja-se comparar uma solução baseada em comitê contra um classificador simples, logo, o tamanho do comitê foi fixado em 1. Como medida de desempenho foram utilizadas a área sob a curva, do inglês AUC (FAWCETT, 2006), e a taxa média de acerto entre as classes.

Base de dados	Total de Pontos	Atributos	Classes	AUC	Taxa média de acertos
Duas Espirais	900	3	2	0,67	0,76
Difficult2	300	2	2	0,77	0,89

Tabela 6: Base de dados estática – resultados de (SANTOS, 2014).

A Tabela 7 apresenta os resultados com o método proposto nesta dissertação, usando a mesma quantidade de amostras para classificação utilizada em (SANTOS, 2014). Cabem algumas informações sobre essa tabela para melhor compreensão dos resultados:

- Total de Pontos = total de amostras disponível na base de dados.
- Tamanho da Base de dados Histórica = total de amostras que foram selecionadas aleatoriamente na divisão dos conjuntos; utilizou-se a proporção 60/40 nessa divisão de dados, que é a mais comumente utilizada na literatura, onde 60% do total foi dedicado à base de dados para a fase de inicialização do modelo.
- Tamanho do bloco de dados = para execução do arcabouço, como mencionado na Seção 4.2.2, os dados são agrupados em blocos para serem processados, sendo assim para este estudo de caso, 40% do total de amostras resultante da divisão do conjunto de dados foram utilizados como teste para o modelo.
- Quantidade de amostras no bloco = número de amostras que um bloco de dados poderá conter.

- Quantidade de blocos = número de blocos escolhido para simulação. A Quantidade de blocos multiplicada pela quantidade de amostras contidas no bloco resulta no tamanho máximo de 40% para o teste do modelo.

Base de dados	Total de Pontos	Tamanho da Base de dados Histórica	Quantidade de amostras no bloco	Quantidade de blocos	AUC	Taxa média de acertos
Duas Espirais	900	540	360	1	0,8306	0,8306
Duas Espirais	900	540	180	2	0,8333	0,8333
Duas Espirais	900	540	120	3	0,8455	0,8455
Duas Espirais	900	540	90	4	0,8240	0,8240
Difficult2	300	240	60	2	0,7833	0,7833
Difficult2	300	240	120	1	0,6238	0,6416

Tabela 7: Bases de dados artificiais e estáticas –abordagem com classificador simples no comitê.

Base de dados	Total de Pontos	Tamanho da Base de dados Histórica	Quantidade de amostras no bloco	Quantidade de blocos	AUC	Taxa média de acertos
Duas Espirais	10.000	6.000	4.000	1	0,7886	0,7900
Duas Espirais	10.000	6.000	400	10	0,8225	0,8242
Duas Espirais	10.000	6.000	300	20	0,8325	0,8333
Duas Espirais	10.000	6.000	200	30	0,8382	0,8375
Duas Espirais	10.000	6.000	150	40	0,8259	0,8233
Difficult2	30.000	24.000	600	10	0,7488	0,7467
Difficult2	30.000	24.000	300	20	0,7833	0,7846
Difficult2	30.000	24.000	200	30	0,8025	0,7962
Difficult2	30.000	24.000	150	40	0,8267	0,8421
Difficult2	30.000	24.000	100	60	0,7760	0,7760

Tabela 8: Base de dados artificiais e estáticas - abordagem por comite com classificador simples e quantidade maior de amostras.

A Tabela 8 apresenta os resultados para as mesmas bases de dados. Porém, foi usada uma quantidade maior de amostras, afim de avaliar o desempenho do arcabouço sob a execução de uma quantidade maior de blocos e sua evolução nos ciclos.

Embora a quantidade de amostras tenha aumentado, os resultados indicam que o método é competitivo, uma vez que não houve diferença estatisticamente significativa no teste de Wilcoxon. No entanto, houve um acréscimo da acurácia conforme aumentava-se a quantidade de blocos; contudo pode-se observar que parece haver um limiar máximo para a divisão do conjunto em blocos de dados sem que o desempenho dos indicadores seja afetado. A hipótese para isto reside na quantidade menor de amostras no bloco, ou seja, um bloco com menos de 150 amostras não foi o suficiente para o aprendizado do modelo. Definir o tamanho ideal de um bloco é o mesmo dilema pelo qual autores que utilizam técnicas de "*windowing*" devem lidar quanto à definição do tamanho da janela, ou se a mesma será fixa ou variável. Um bloco de tamanho maior é mais estável, porém reage de maneira mais lenta quando ocorre *concept drift* (GAMA et al., 2014).

Neste experimento, foi realizada uma análise do comportamento do modelo, simulando um classificador único em situações onde não há ocorrência de *drift.* Mas, sobretudo a tentativa de compreender a influência da manipulação do conjunto de entrada no desempenho do método e aprimorá-lo para que possa ser utilizado em modelos de aprendizagem em ambientes não estacionários.

5.4 Base de dados não estacionárias

Esta seção descreve os experimentos realizados com base de dados reais e artificiais. Foram escolhidos alguns autores que trabalham com tarefas de classificação em problemas de *concept drift* para validar o resultado obtido pelo método proposto neste trabalho. Este estudo gerou tabelas individuais para cada base de dados com as medidas de desempenho escolhidas, acurácia e AUC.

5.4.1 Electricity

Esta base de dados real é de ampla utilização em problemas de *concept drift*. Os dados foram coletados do fornecedor de energia elétrica de New South Wales, Austrália. Nesse mercado, as tarifas não são fixas e são afetados pela demanda e oferta do mercado. A base de dados consiste em 45.312 instâncias coletadas em intervalos de

30 minutos. Cada instância consiste em oito atributos e um rótulo de classe *up* ou *down*. Os preços são definidos a cada cinco minutos e o rótulo da classe define a mudança do preço relacionada a uma média móvel das últimas 24 horas.

O objetivo do classificador é usar os valores dos atributos para prever se o preço da eletricidade irá aumentar ou diminuir.

Como esta é uma base de dados derivada de um fenômeno do mundo real, não se pode saber de forma definitiva se ou quando ocorreu *concept drift*; o mesmo se aplica para a base de dados Nebraska.

- **1º Estudo de caso**

Primeiramente, foi realizado um estudo de caso que teve por objetivo o ajuste dos dois conjuntos de parâmetros apresentados neste estudo. Um dos conjuntos está relacionado com a programação genética cartesiana, e o outro com a síntese das APF.

Na programação genética cartesiana temos o número de gerações, a taxa de mutação, função de aptidão, quantidade de colunas e *levelback*. O segundo conjunto está relacionado a APF no que diz respeito ao número de partições *fuzzy*. Os operadores *fuzzy* foram deixados estáticos assumindo os valores mais convenientes utilizados em (SANTOS, 2014).

Para a variação dos parâmetros da programação genética cartesiana, este experimento também partiu da configuração ótima utilizada em (SANTOS, 2014): Linhas = 1; Colunas = 400; *Levelback* = 399; Tamanho da população = 5; Quantidade de gerações = 600; Partições = 5; Taxa de mutação = 0,1; Função de aptidão = Similaridade a partir do RMSE; Partição = 5; Pesos (critério de avaliação *w1* = 0,7 e *w2* = 0,3).

Todas as execuções foram feitas fixando-se parâmetros e variando-se somente aquele desejado no momento. Dessa maneira, de forma combinatória os parâmetros foram configurados na solução e os resultados apresentados avaliados através da acurácia e AUC. Para o parâmetro de quantidade de linhas da programação genética cartesiana, foi mantido o padrão = 1. Os operadores *fuzzy* também não variaram, seguindo o conjunto que apresentou melhor resultado em (SANTOS, 2014).

A **Erro! Fonte de referência não encontrada.** contém os parâmetros e a forma como estes foram organizados em cada execução de teste

Tabela 9: Parâmetros avaliados.

Execução	Número Gerações	Taxa Mutação	Função Aptidão	Quantidade de colunas	Levelback	Pesos (w1 e w2)	Partição
1º	Fixo	Fixo	Fixo	Fixo	Fixo	Fixo	Variável
2º	Fixo	Variável	Fixo	Fixo	Fixo	Fixo	Fixo
3º	Variável	Fixo	Fixo	Fixo	Fixo	Fixo	Fixo
4º	Fixo	Variável	Fixo	Fixo	Fixo	Fixo	Fixo
5º	Fixo	Fixo	Variável	Fixo	Fixo	Fixo	Fixo
6º	Fixo	Fixo	Fixo	Variável	Variável	Fixo	Fixo
8º	Fixo	Fixo	Fixo	Fixo	Fixo	Variável	Fixo

Para que o arcabouço pudesse executar mais rapidamente, primeiro foi utilizado somente um classificador no comitê e posteriormente a quantidade mínima de classificadores para um comitê com votação majoritária; ou seja, três classificadores base sob um total de 30 execuções.

A Tabela 10 exibe os valores que foram selecionados para serem avaliados nos conjuntos de parâmetros e a Tabela11 os parâmetros escolhidos.

Tabela 10: Valores dos parâmetros testados.

Parâmetro	Valor
Partição	(3, 5, 7)
Taxa de Mutação	(0,1; 0,2; 0,3; 0,5)
Número de Gerações	(300, 600, 800)
Quantidade de Colunas	(400, 800, 1600)
Função Aptidão	(AUC, Entropia, RMSE)
Critérios de aptidão (pesos)	(*w1* =1; *w2* = 0); (*w1* = 0,5; *w2* = 0,5); (*w1* = 0,7; *w2* = 0,3);

Logo, a combinação de parâmetros que forneceu o melhor resultado foi selecionada e utilizada para todos os demais experimentos com as outras bases de dados, além da própria base de dados Electricity.

Parâmetro	Valor
Partição	5
Taxa de Mutação	0,2
Número de Gerações	600

Quantidade de Colunas	800
Função Aptidão	AUC
Critérios de aptidão (pesos)	(*w1* = 0,7; *w2* = 0,3);

A partir do conjunto de parâmetros estabelecido na Tabela 11 foram realizados outros experimentos com bases de dados artificiais e reais descritas na Seção 5.4.

2º Estudo de caso

Dentre tantos trabalhos disponíveis na literatura, como exemplo (MINKU & YAO, 2012), (GAMA et al, 2004), (JACKOWSKI, 2013) etc., para este estudo de caso optou-se por comparar os resultados obtidos com (KOLTER& MALOOF, 2007) e (ESCOVEDO, 2015), pois estes trabalhos propõem uma solução para o problema de *concept drift* baseado na utilização de comitê de classificadores.

O método proposto por (KOLTER & MALOOF, 2007) cria e remove um classificador do comitê de forma dinâmica. Um classificador inicialmente é criado com peso = 1 e tem o seu valor reduzido em caso de erro. As estratégias de exclusão e adição são baseadas na avaliação da resposta local e global do comitê. Este algoritmo foi chamado de *Dynamic Weighted Majority* (DWM).

Por outro lado, o trabalho desenvolvido em (ESCOVEDO, 2015) utilizou um modelo neuroevolutivo (NEVE) com inspiração quântica baseado em comitê de redes neurais do tipo *multi-layer perceptron*. Nesse trabalho também foi apresentado um novo mecanismo de detecção de *concept drift*, denominado DetectA.

O objetivo deste estudo de caso foi comparar a acurácia entre os métodos: sem atribuição de peso, solução proposta nesta dissertação, e um método que atribui peso ao classificador baseado em seu desempenho (ESCOVEDO, 2015); (KOLTER& MALOOF, 2007).

Tabela 11:Parâmetros escolhidos - 1º Estudo de Caso.

A Tabela 12 apresenta o desvio padrão para as medidas de desempenho Acurácia e AUC dos resultados obtidos neste estudo, além do desvio padrão informado em (ESCOVEDO, 2015) e relaciona qual método pertence a qual autor. A Figura 16 exibe de forma gráfica a Acurácia média para base de dados Electricity.

Método	Peso	Autor(es)	Acurácia Desvio Padrão	AUC Média	AUC Desvio Padrão
Comitê com 3 classificadores	Não	(Patricia, 2018)	0,0106	0,7138	0,0105
Comitê com 5 classificadores	Não	(Patricia, 2018)	0,0109	0,7381	0,0108
Comitê com 11 classificadores	Não	(Patricia, 2018)	0,0109	0,7326	0,0107
Comitê com 21 classificadores	Não	(Patricia, 2018)	0,0108	0,7134	0,0109
MLP (1) - MLP simples com 10 neurônios	Sim	(ESCOVEDO, 2015)	< 0,02	Não informado	Não informado
MLP (2) - MLP detecção reativa (esquece o passado) com 10 neurônios	Sim	(ESCOVEDO, 2015)	< 0,02	Não informado	Não informado
NEVE (1) - NEVE sem detecção (melhor caso)	Sim	(ESCOVEDO, 2015)	< 0,02	Não informado	Não informado
NEVE (2) - NEVE detecção reativa (melhor caso)	Sim	(ESCOVEDO, 2015)	< 0,02	Não informado	Não informado
DWM	Sim	(KOLTER& MALOOF, 2007)	Não informado	Não informado	Não informado

Tabela 12: Detalhes Electricity.

O trabalho de (KOLTER& MALOOF, 2007) apresenta as curvas de desempenho para um período de 1 semana, ou seja, 2.352 amostras que equivalem a 48 períodos de 30 minutos em uma semana. Com 22 classificadores em média no comitê, os autores atingiram com o DWB acurácia média em torno de 80,75%. Como trabalho de referência os autores utilizaram os resultados obtidos em (HARRIES, 1999), que adotaram janelas deslizantes de diferentes tamanhos e atingiram acurácia entre 66% e 67,7%.

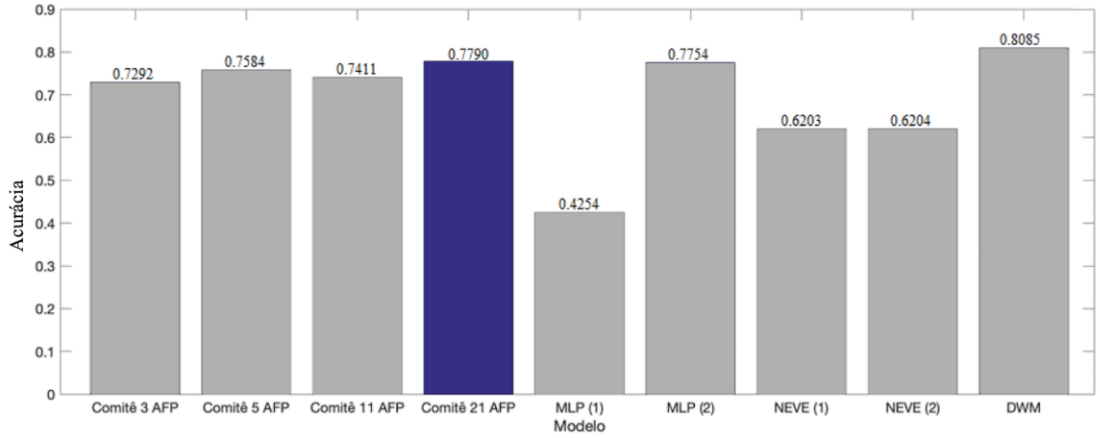


Figura 16: Resultado Electricity.

Para este experimento foram utilizados diferentes tamanhos de comitês iniciando-se com 3 classificadores, passando-se a 5, 11 e posteriormente 21, quantidade essa a mais próxima do utilizado pelos autores do DWM. Conforme pode-se observar na Figura 16, não houve muita variação na acurácia ao se aumentar o número de classificadores. Porém, o tempo de execução torna-se maior ao se elevar este número. Desta maneira, conclui-se que não é muito vantajoso para o arcabouço um comitê com grande número de classificadores.

Em termos de acurácia, o percentual atingido não conseguiu superar o apresentado pelo DWM. Contudo, o método demonstra-se promissor por possuir interpretabilidade, conforme exibido nas Figura 17 e 18, com uma quantidade menor de classificadores base. Ao se comparar os resultados obtidos em (ESCOVEDO, 2015) nos quatro métodos propostos pela autora, pode-se perceber que, em todas as situações, o método proposto nesta dissertação demonstrou uma maior taxa de acerto. No entanto, a aplicação do teste de Wilcoxon nos resultados obtidos em (KOLTER& MALOOF, 2007), (ESCOVEDO, 2015) e neste experimento indicam que não há diferença estatisticamente significativa no desempenho dos classificadores para $p \leq 0{,}05$.

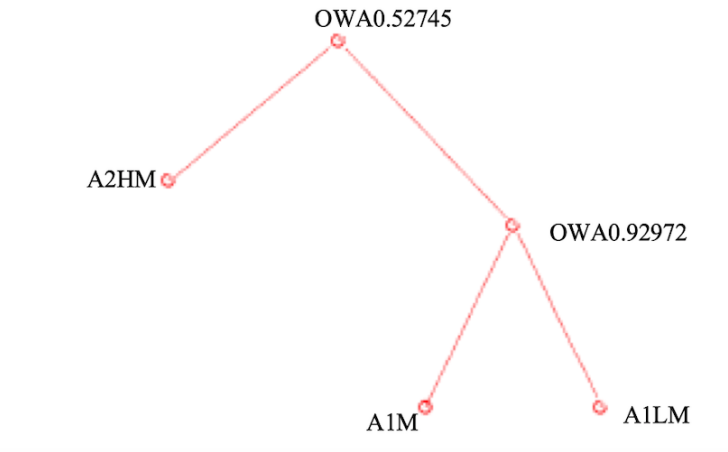


Figura 17: APF saída 1 Electricity – 2º Estudo de Caso.

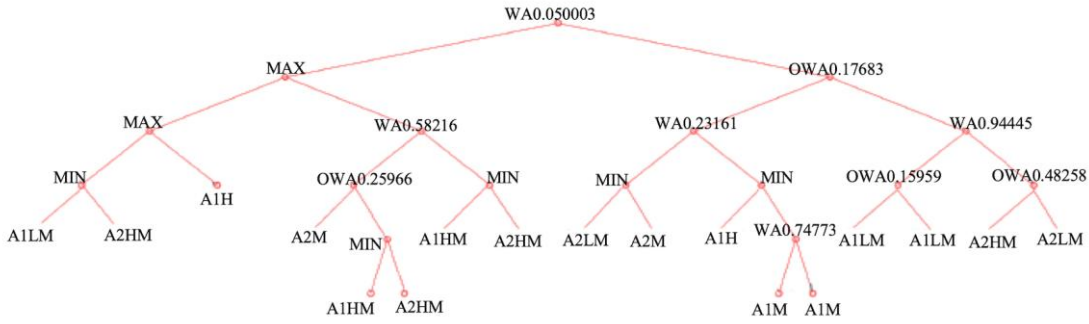


Figura 18: APF saída 2 Electricity – 2º Estudo de Caso.

A Figura 17 e a Figura 18 representam graficamente as saídas do classificador para a base de dados Electricity. Podemos observar que dos oito atributos de entrada que compõem a base os únicos utilizados foram "A1" (atributo *fuzzy* 1) e "A2" (atributo *fuzzy* 2). Estes dois atributos representam o dia da semana e o período do dia. Desta maneira os demais parecem não fornecer informações relevantes para o classificador.

5.4.2 Base de dados gerada através do MOA

Com o objetivo de consolidar os parâmetros indicados pela base de dados Electricity e confirmar as escolhas adotadas no estudo de caso 5.4.1 foi realizado um segundo estudo de caso para a determinação de parâmetros. Para tal, foi utilizada uma base de dados artificial gerada usando os recursos disponíveis no MOA[1]. No MOA é possível gerar arquivos com ruído, escolher como será a opção de *drift* através da

[1] MOA – É um Framework *opensource* que permite criar e executar experimentos de aprendizado de máquina ou mineração de dados em fluxos de dados contínuo com ou sem a presença de *concept drift.*

configuração da largura da "janela" *W*, ou seja, a largura da mudança de conceito, além do ponto central do início do *concept drift* e quantos pontos de mudança. O MOA permitiu a geração de uma base de dados contendo 200.000 amostras sendo 10 atributos, duas classes e a presença de concept drift configurado para uma largura de janela no total de 1000 amostras. Dentre as opções disponíveis foi escolhida a classe generators.RandomTreeGenerator devido à sua facilidade de configuração dos parâmetros para a geração do arquivo de dados com *drift.*

O procedimento para a execução dos testes de determinação dos melhores parâmetros para os dois conjuntos (*fuzzy* e PGC) seguiu os mesmos passos realizados na Seção 5.4.1; foram combinados os parâmetros da Tabela 13.

Execução	Número Gerações	Taxa Mutação	Função Aptidão	Quantidade de colunas	Levelback	Pesos (w1 e w2)	Partição
1º	Fixo	Fixo	Fixo	Fixo	Fixo	Fixo	Variável
2º	Fixo	Variável	Fixo	Fixo	Fixo	Fixo	Fixo
3º	Variável	Fixo	Fixo	Fixo	Fixo	Fixo	Fixo
4º	Fixo	Variável	Fixo	Fixo	Fixo	Fixo	Fixo
5º	Fixo	Fixo	Variável	Fixo	Fixo	Fixo	Fixo
6º	Fixo	Fixo	Fixo	Variável	Variável	Fixo	Fixo
8º	Fixo	Fixo	Fixo	Fixo	Fixo	Variável	Fixo

Tabela 13: Parâmetros combinados.

O resultado obtido foi muito similar ao exibido na Tabela 11 com exceção do parâmetro Taxa de Mutação que pareceu fornecer melhor resultado quando igual a 0,01, conforme pode ser verificado na Tabela 14.

Parâmetro	Valor
Partição	5
Taxa de Mutação	0,1
Número de Gerações	600
Quantidade de Colunas	800
Função Aptidão	AUC
Critérios de aptidão (pesos)	(*w1* = 0,7; *w2* = 0,3);

Tabela 14: Parâmetros selecionados - 2º Estudo de Caso.

Desta maneira, verificou-se que os resultados obtidos no 1º estudo de caso para a determinação dos melhores parâmetros usando-se uma base de dados real não foi ao acaso, visto que, para uma base uma base de dados artificial com *concept drift* houve pouca variação nos resultados dos parâmetros escolhidos. Logo, os demais estudos de caso que foram conduzidos utilizando-se os melhores parâmetros obtidos através da base de dados Electricity indicam que o modelo caminhou na direção correta.

5.4.3 Nebraska

Esta base de dados pertence ao órgão americano U.S. National Oceanic and Atmospheric Administration, que coletou, a partir de 9.000 estações climáticas, medidas como temperatura, pressão, velocidade do vento, precipitação etc. Devido ao volume da base de dados, assim como foi realizado em (POLIKAR & ELWELL, 2011), os dados escolhidos para este experimento foram os da subestação Offutt localizada em Bellevue, Nebraska. Este subconjunto possui informações de padrões climáticos bastante diversos dos anos de 1949 a 1999, sendo dessa maneira um problema típico real de classificação com a presença de *concept drift*. Oito atributos são selecionados de acordo com (ELWELL & POLIKAR, 2011) em um total de 18.159 amostras. Os rótulos fornecem a leitura diária indicando se choveu ou não. Existem 5.698 indicações positivas para chuva e 12.461 para não chuva.

O estudo de caso realizado com a base de dados real Nebraska teve como referências (ELWELL & POLIKAR, 2011) com o algoritmo Learn++.NSE e (ESCOVEDO, 2015) com os mesmos métodos descritos no 2º estudo de caso da Seção 5.4.1.

O algoritmo Learn++.NSE (ELWELL & POLIKAR, 2011) funciona de forma incremental baseado em um comitê de classificadores, o qual usa votação majoritária ponderada. Os pesos dos classificadores são dinamicamente atualizados em função dos ajustes dos erros cometidos pelos mesmos no ambiente atual e nos anteriores. O algoritmo usa apenas os dados atuais para treinamento, assumindo que todos os dados observados anteriormente (relevantes ou não para o aprendizado) não estão acessíveis ou não é possível armazená-lós. Assim, toda a informação relevante sobre os dados anteriores deve estar armazenada nos parâmetros dos classificadores previamente gerados.

O Learn++.NSE utiliza mecanismo passivo de detecção de *drift*, assumindo que, a cada passo, pode ou não ter ocorrido mudança. Dependendo da natureza da mudança, o algoritmo retém, constrói ou temporariamente descarta conhecimento, para que os novos dados possam ser categorizados.

Portanto, assim como no trabalho dos autores mencionados anteriormente o objetivo foi prever se choverá ou não.

A Tabela 15 apresenta o Desvio Padrão obtido no cálculo das medidas de desempenho Acurácia e AUC. Devido à falta de informação dos autores, este dado foi classificado como "Não informado". A Figura 19 exibe os resultados obtidos para a medida de desempenho Acurácia.

Método	**Autor(es)**	**Acurácia**	**AUC**	
		Desvio Padrão	**Média**	**Desvio Padrão**
Comitê com 11 classificadores	(PATRICIA, 2018)	0,2001	0,7026	0,2099
MLP (1) - MLP simples com 10 neurônios	(ESCOVEDO, 2015)	< 0,02	Não informado	Não informado
MLP (2) - MLP detecção reativa (esquece o passado) com 10 neurônios	(ESCOVEDO, 2015)	< 0,02	Não informado	Não informado
NEVE (1) - NEVE sem detecção (melhor caso)	(ESCOVEDO, 2015)	< 0,02	Não informado	Não informado
NEVE (2) - NEVE detecção reativa (melhor caso)	(ESCOVEDO, 2015)	< 0,02	Não informado	Não informado

Tabela 15: Detalhes Nebraska.

(*) *Naive Bayes* = NB; *Support Vector Machines* = SVM e *Classification and Regression Trees*=CART.

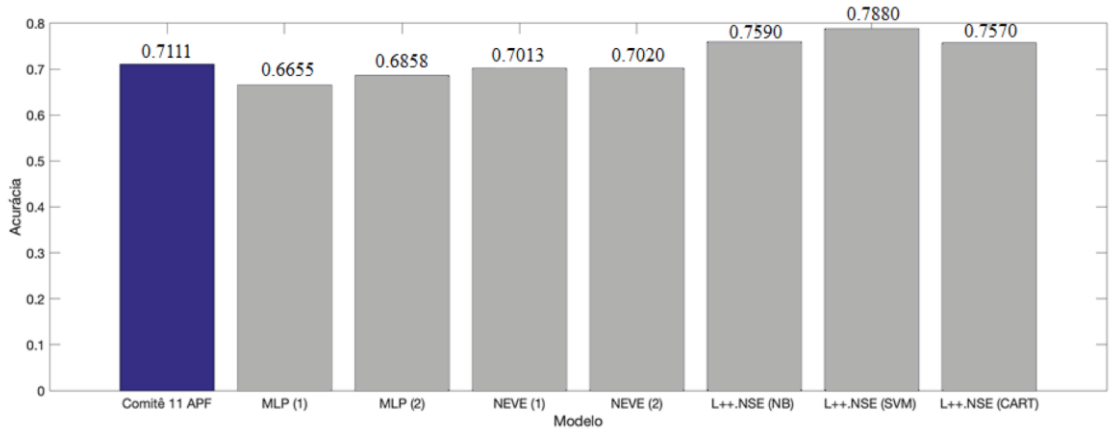


Figura 19: Resultado Nebraska.

Neste estudo de caso, optou-se por manter no comitê 11 classificadores considerando o resultado obtido para a base de dados Electricity, onde o resultado obtido na Acurácia não justificou o tempo computacional para a execução da base de dados conforme demonstrado na Figura 16 do primeiro estudo de caso.

A aplicação do teste de Wilcoxon confirmou não haver diferença estatisticamente significativa entre os classificadores. Isto indica que o modelo é competitivo frente a outros métodos rebuscados com classificadores base conhecidos por sua acurácia, como *Support Vector Machines* (SVM) e *Naive Bayes* (NB) utilizados de forma conjunta com o algoritmo L++.NSE. Da mesma forma o modelo também demonstrou habilidade para aprender a base de dados Nebraska sem a necessidade de um detector de *drift.*

5.4.4 SEA Concepts

A base de dados SEA Concepts foi desenvolvida artificialmente por (STREET & KIM, 2011) e tem sido usada por vários algoritmos como um teste padrão para avaliação de *concept drift.* O conjunto de dados é caracterizado por longos períodos sem qualquer desvio com mudanças bruscas ocasionais na classe de fronteira, isto é, desvio súbito ou mudança de conceito. O conjunto de dados inclui duas classes e três atributos, com apenas dois sendo relevantes e o terceiro sendo ruído. Os rótulos de classe são atribuídos com base na soma dos atributos relevantes. Um determinado padrão pertence à classe 1 se a soma dos atributos relevantes for maior que um limiar; e pertence à classe 2, caso contrário. Os dados são distribuídos uniformemente entre 0 e 10, em

intervalos regulares, e o limite é alterado, criando uma mudança abrupta na classe de fronteira, podendo-se variar intensivamente os valores (8→9→7,5→9,5).

- **1º Estudo de caso**

Neste estudo de caso adotou-se novamente os trabalhos de (ELWELL & POLIKAR, 2011) e (ESCOVEDO, 2015) como base de comparação. O procedimento de treinamento que foi utilizado por (ELWELL & POLIKAR, 2011) é idêntico ao descrito por (STREE & KIM, 2001): 50.000 amostras são introduzidas como dados de treinamento (25.000 pontos por classe), em 200 intervalos de tempo (*time steps*) e 250 amostras/*time step*. Também como realizado por (STREET & KIM, 2001), 10% de ruído de classe foi adicionado aos dados de treinamento. Para verificação do algoritmo, um conjunto separado de 50.000 amostras de dados de cada ambiente (sem ruído) foi usado para teste. Já (ESCOVEDO, 2015) utilizou 400 blocos com 250 amostras.

A Tabela 16 exibe algumas informações para este estudo de caso com a configuração de blocos mencionada anteriormente e a Figura 20 os resultados obtidos com este estudo de caso.

Método	**Autor(es)**	**Acurácia**	**AUC**	
		Desvio Padrão	**Média**	**Desvio Padrão**
Comitê com 11 classificadores	(PATRICIA,2018)	0,0224	0,9015%	0,0428
MLP (1) - MLP simples com 10 neurônios	(ESCOVEDO,2015)	<0,02	Não informado	Não informado
MLP (2) - MLP detecção reativa (esquece o passado) com 10 neurônios	(ESCOVEDO,2015)	<0,02	Não informado	Não informado
NEVE (1) - NEVE sem detecção (melhor caso)	(ESCOVEDO,2015)	< 0,02	Não informado	Não informado
NEVE (2) - NEVE detecção reativa (melhor caso)	(ESCOVEDO,2015)	<0,02	Não informado	Não informado

Tabela 16: Detalhes SEA – 1º Estudo de Caso.

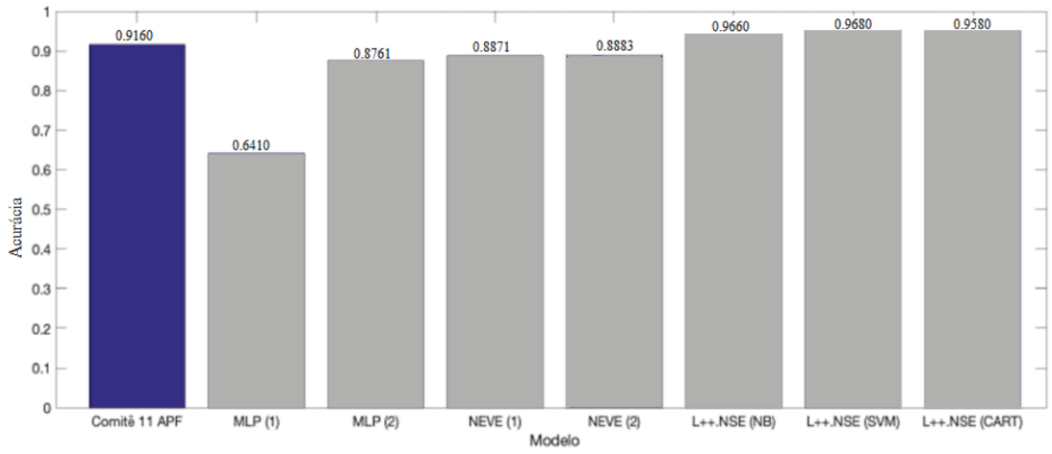


Figura 20: Resultado SEA - 1º Estudo de Caso.

O trabalho de (ELWELL & POLIKAR, 2007) com Learn ++.NSE traz uma abordagem que se baseia no uso estratégico de classificadores atuais e passados. Esses são combinados com pesos de votação atualizados dinamicamente, com base em seus erros ajustados ao longo do tempo e em ambientes recentes. Esse mecanismo de ponderação permite que o Learn++.NSE aprenda novos conhecimentos criando novos classificadores, enquanto usa o conhecimento existente quando esse conhecimento ainda é relevante. Além disso Learn++.NSE varia seu comportamento de acordo com o classificador base utilizado, exemplo: Learn++.NSE com *Naive Bayes* (NB) ou Learn++.NSE com SVM.

Pode-se considerar que neste estudo, particularmente, o ganho não está na superioridade do classificador base escolhido (as APF), mas sim nas estratégias para que permitiram ao comitê manter sua acurácia sem a necessidade de um mecanismo de atribuição de pesos utilizado com o Learn++.NSE. O tempo de aprendizado deste algoritmo inclui o tempo para treinar novos classificadores e repesar os membros do comitê. Esse tempo de aprendizado do Learn++.NSE aumenta linearmente, à medida que todos os classificadores são mantidos e reponderados nos dados de treinamento mais recentes. Como já dito, outra vantagem do uso de APF em relação aos métodos apresentados são as possíveis informações que as árvores podem gerar. O teste de Wilcoxon não gerou diferença estatisticamente significativa entre os resultados.

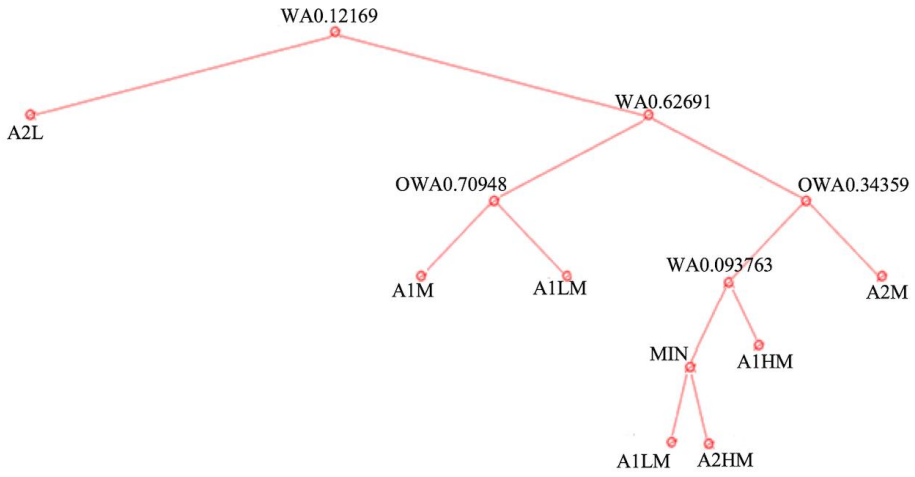


Figura 21: AFP saída 1 SEA - 1º Estudo de Caso.

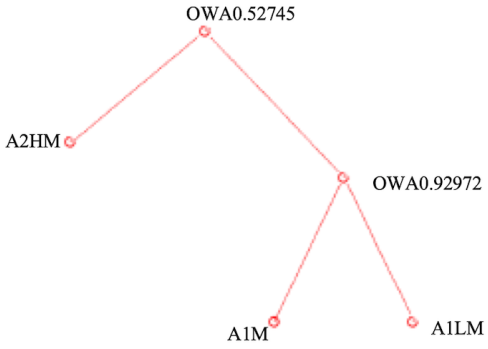


Figura 22: APF saída 2 SEA – 1º Estudo de Caso.

A interpretação das árvores geradas pelo método *Fuzzy*-Genético para a base de dados SEA Concepts sugere que os atributos de entrada "A1" na partição *fuzzy* referente a valores médios *"Mean (M)"* tendem a produzir melhores resultados. Já o atributo de entrada "A2" teve melhor aproveitamento para valores mais baixos, na partição *"Low (L)"* conforme pode ser observado na Figura 21 e Figura 22. O atributo A2 em ambas as saídas produzidas pelas APF possui o rótulo "A2L" muito próximo do topo da árvore, exercendo mais fortemente uma pressão sob o resultado final.

- **2º Estudo de caso**

Para este experimento foi gerada uma base de dados com 200.000 amostras, a partir do MOA. Para este estudo de caso, foram aplicados 2 pontos de mudança seguindo o esquema: 50.000 amostras geradas com a primeira função de classificação (

f1 + *f2* ≤ 9) e 50.000 amostras geradas com a segunda função de classificação (*f1* + *f2* ≤ 9,5) de forma similar a (DIAZ et al., 2014) com o algoritmo FAE. Contudo o modelo foi avaliado somente com $W = 0$, simulando desta maneira um *drift* abrupto. Os resultados são apresentados na Figura 23 e a Tabela 17 indica o desvio padrão obtido com o modelo proposto neste estudo.

Método	Autor(es)	Acurácia Desvio Padrão	AUC Média	AUC Desvio Padrão
Comitê com 11 classificadores	(Patricia,2018)	0,0301	0,6450%	0,0334

Tabela 17: Detalhes SEA – 2º Estudo de Caso.

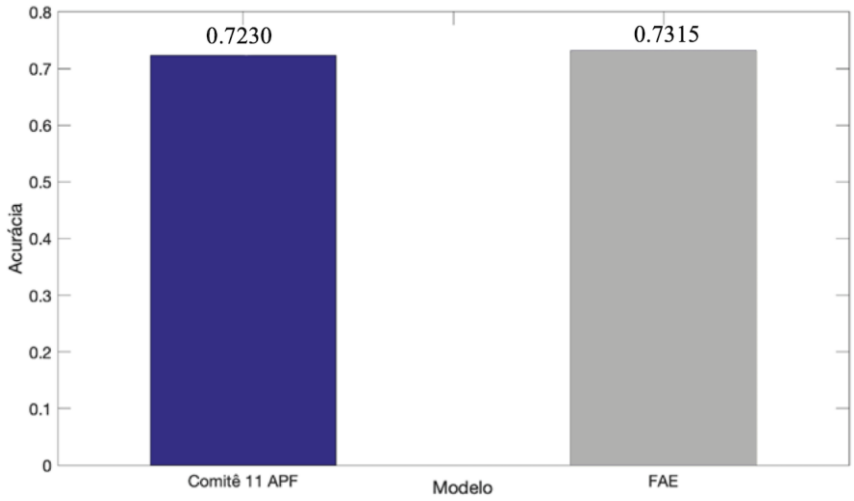


Figura 23: Resultado SEA – 2º Estudo de Caso.

O algoritmo FAE está apto a lidar com mudanças de conceito graduais e abruptas, pois utiliza método de detecção em conjunto com o comitê. Esta característica é um facilitador para o funcionamento do método pois evita a criação desnecessária de classificadores, além de favorecer os classificadores anteriores que representam outros conceitos remanescentes dentro do comitê. Contudo, o resultado fornecido pelo teste de Wilcoxon não indicou diferença estatisticamente significativa entre os dois classificadores, mesmo tendo FAE uma vantagem por possuir um método de detecção.

5.4.5 Rotating Checkerboard

Esta base de dados é um conjunto de dados não gaussiano derivado do problema canônico XOR, que se assemelha a um tabuleiro de xadrez rotativo. A rotação

torna esse problema enganosamente simples, já que o ângulo e a localização dos limites de decisão mudam drasticamente em intervalos de tempo. A fim de evitar o treinamento de blocos idênticos de dados e aumentar a complexidade, 10% de ruído aleatório foi introduzido de forma análoga a (POLIKAR & ELWELL, 2013). Existem quatro possíveis cenários de mudanças de taxas de *drift* disponíveis para esta base de dados: 1) taxa de desvio constante de 2π / 400 = 0,016 rad/time step; 2) aumentando exponencialmente a taxa de *drift* (o tabuleiro gira cada vez mais rápido); 3) variando a taxa de *drift* de forma senoidal e 4) taxa de desvio em forma de pulso gaussiano (lento-rápido-muito-rápido-rápido-lento). Os arquivos de dados e vídeos que descrevem esses experimentos podem ser encontrados em (POLIKAR & ELWELL, 2013).

As Figuras 24, 25 e 26 exibem os melhores resultados atingidos por Learn++.NSE, utilizado de forma conjunta com outros classificadores base, e o estudo comparativo realizado por (ESCOVEDO, 2015), além dos resultados obtidos pelo método proposto nesta dissertação.

Os valores obtidos para o desvio padrão com o método *Fuzzy*-Genético para a base de dados Rotating (Constante, Exponencial e Pulso) ficaram entre 0,0055 e 0,0069.

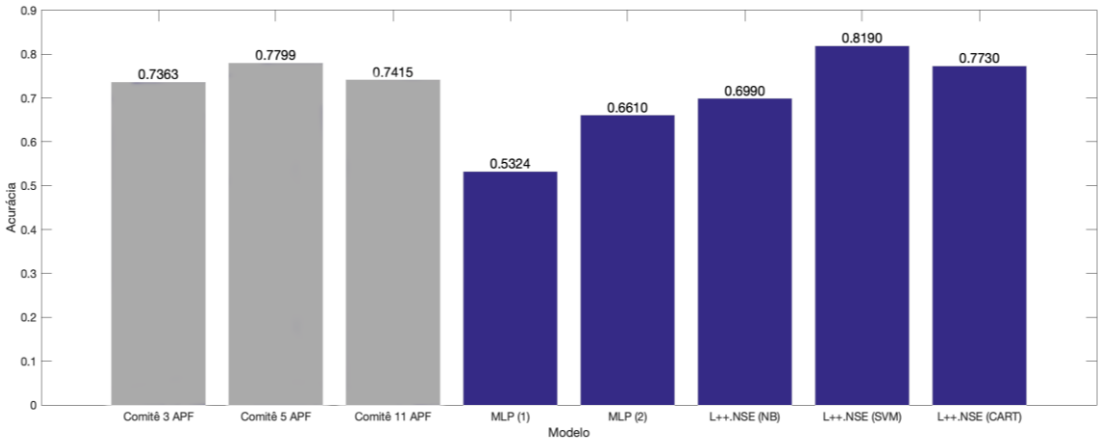


Figura 24: Resultado Rotating - Constante.

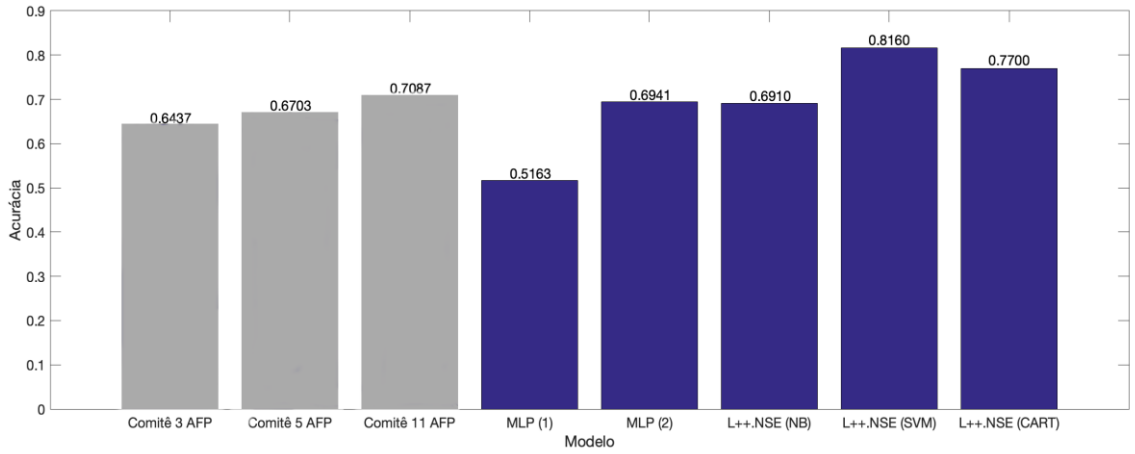


Figura 25: Resultado Rotating - Pulso.

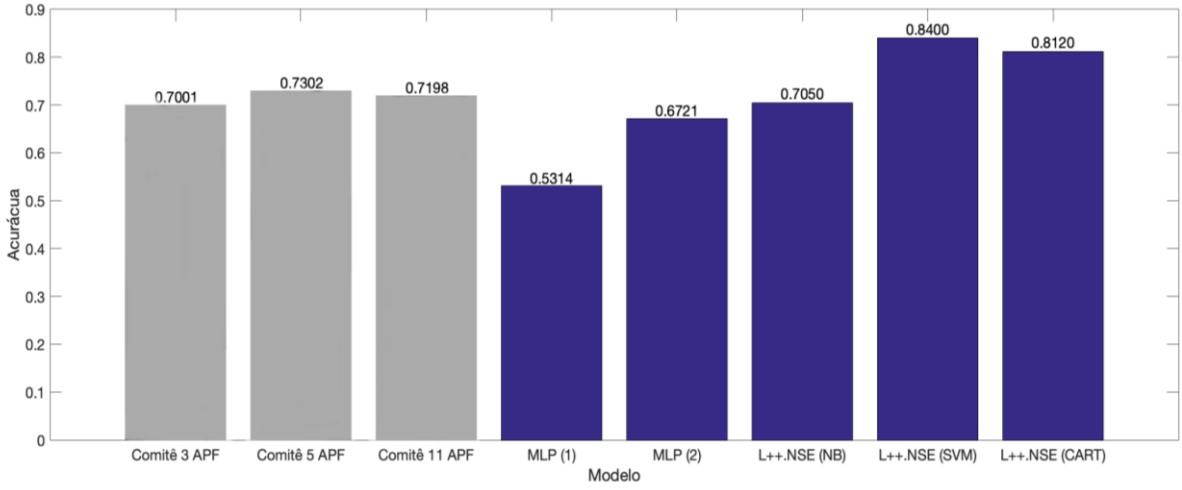


Figura 26: Resultado Rotating - Exponencial.

Este estudo de caso foi realizado sequencialmente após o experimento indicado na subseção 5.4.1 para a base de dados Electricity. Por isso também se optou por manter somente 11 classificadores de APF no comitê, visto que a Acurácia obtida com mais classificadores no comitê não justificou o tempo de execução mais elevado. Os resultados apresentados nas Figuras 24, 25 e 27 juntamente com o resultado obtido pelo teste de Wilcoxon (ausência de diferença estatisticamente significativa), confirma que o modelo possui desempenho similar a outros métodos. Isto também demonstra sua habilidade em aprender com as variações de *drifts* da base de dados Rotating e também a estabilidade do algoritmo. O L++.NSE com o SVM, dentre todos, é o que mais se destaca em relação aos resultados obtidos pelo método *Fuzzy*-Genético.

5.4.6 Considerações finais

A análise dos experimentos demonstrou acurácia similar as outras propostas utilizadas como método comparativo nesta dissertação. Porém, a informação gráfica fornecida pelas árvores APF oferece a possibilidade de validar e aperfeiçoar o método utilizado. Desta maneira foi possível gerar conclusões como no caso da base de dados Electricity. Ou no caso da base de dados SEA, no qual as árvores geradas pelo modelo não utilizaram o 3º atributo, caracterizado na base de dados como ruído. Desta maneira, a representação gráfica validou o desempenho obtido em termos de acurácia.

Pode-se dizer que o modelo contribuiu de forma secundária para a seleção de atributos de maneira automática conforme exibido na Figura 27.

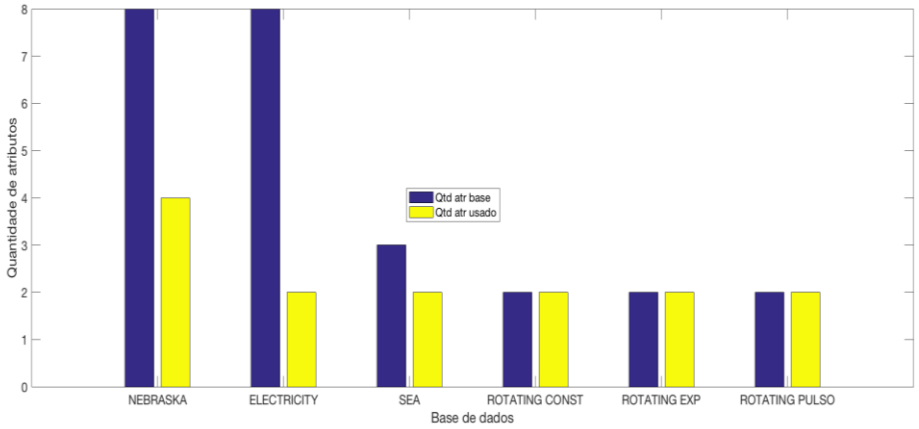


Figura 27: Quantidade de atributos selecionados.

CONCLUSÃO

Este trabalho apresentou uma metodologia para sintetizar comitê de classificadores de árvores de padrões *fuzzy* (APF), induzido pela programação genética cartesiana (PGC) para atuar em ambientes não estacionários.

O modelo desenvolvido foi capaz de atender satisfatoriamente às diretrizes de um algoritmo de aprendizagem de máquina para ambientes não estacionários, como proposto em (ELWELL & POLIKAR, 2011): os blocos de dados foram apresentados apenas uma vez para fins de treinamento e posteriormente o conhecimento foi generalizado; a medida que novos blocos chegaram o modelo foi capaz de se adaptar para absorver conhecimento do ambiente atual, bem como descartar informação que não era mais relevante.

Esse modelo foi construído sobre uma estrutura modular, a qual gerou um arcabouço flexível e adaptável. Além dos resultados satisfatórios e promissores, é possível que mudanças estruturais possam ser realizadas de acordo com o desejo do programador, tais quais: uso de qualquer algoritmo como classificador base no comitê, extensão do modelo para o uso de detector de *drift*, métodos de seleção de instâncias, uso de diferentes medidas de desempenho e ampliação das regras de estratégias evolutivas do comitê.

Foram realizados diversos experimentos com o modelo com bases de dados estacionárias e não estacionárias. O primeiro experimento indicou que a proposta possui desempenho satisfatório no aprendizado supervisionado tradicional, para duas bases de dados para ambientes estacionários, tanto quando atuou como um classificador simples quando em comitê. Isso demonstrou a capacidade de generalização do modelo para bases de dados sem a presença de *concept drift.*

Os demais experimentos foram realizados com o modelo utilizando-se duas bases de dados reais e três artificiais, todas essas com a presença de *concept drift.* A análise dos resultados mostrou que o modelo é robusto e eficiente para blocos de tamanho variável, bases de dados de alta dimensionalidade e diferentes balanceamentos de classes e bases de dados com diferentes proporções de *drifts.*

A comparação com os algoritmos propostos na literatura foi feita com base na acurácia informada pelos autores e dadas as condições do teste de Wilcoxon foi possível verificar que o método é tão competitivo quanto os demais, porém sem a necessidade de um método de detecção de *drift.*

Embora em aplicações de aprendizado de máquinas a acurácia preditiva seja a medida mais usual, algumas experiências em aprendizado de máquinas em determinados domínios indicam a importância da interpretabilidade das descrições induzidas. Neste sentido, as árvores geradas pelo método proposto podem gerar novas ideias potencialmente úteis sobre o domínio e, assim, melhorar o entendimento do usuário sobre este domínio. As questões importantes são: como tornar o conhecimento de histórico específico do domínio utilizável pelo sistema de aprendizado; e como interpretar os resultados à luz dessa experiência em segundo plano. Por exemplo, as árvores geradas na classificação da base de dados Electricity, indicou que dos oito atributos de entrada, somente dois foram realmente úteis no processo de classificação. De maneira similar as árvores geradas para a base de dados SEA possuíam somente os dois atributos relevantes para o problema, descartando o terceiro que foi adicionado à base como ruído.

Por fim, conclui-se que as estratégias de manutenção do comitê foram essenciais no aprendizado adaptativo do comitê, sendo estas estratégias possivelmente, um dos fatores chave para o desempenho satisfatório apresentado pelo modelo proposto.

Dessa maneira, como possibilidades de trabalhos futuros, pode-se destacar:

- Propor novas formas de combinação de regras, diferentes da votação majoritária, responsáveis pela decisão final do comitê, de maneira que estas possam explorar os pontos fortes dos membros do comitê.
- Utilização de um detector de *drift.*
- Realizar novos experimentos com outras bases de dados artificiais, para as quais se possa saber previamente o momento de ocorrência do *drift* e entender melhor o comportamento do modelo para aperfeiçoá-lo.
- Aperfeiçoar o método de auto ajuste de parâmetros do modelo pode levar a um melhor poder preditivo, levando-se em consideração a contribuição individual de cada classificador no comitê e a avaliação global do grupo no processo de obtenção de modelos mais complementares (KRAWCZK et al., 20.17).
- Desenvolver um mecanismo que possa avaliar o desempenho individual do classificador ao longo do tempo e a ocorrência de *concept drift*. Cada classificador teria um *timestamp* desde sua inserção no comitê até o momento de sua saída. A ideia seria a avaliação das APF de acordo com o seu tempo de permanência

no comitê, dessa maneira procura-se extrair conhecimento a respeito da variação dos termos *fuzzy*. Estes termos representariam o conceito atual. A mudança de termos poderia indicar a ocorrência de *concept drift.*

- Utilização do algoritmo de Florestas Aleatórias como classificador base para o comitê.

REFERÊNCIAS

BACH HS, MALOOF M. *Paired Learners for Concept Drift*. *In Proceedings of the Data Mining*, ICDM'08, Eighth IEEE International Conferenceon.23-32, 2008.;

BAENA-GARCIA M, CAMPO-ÁVILA J, FIDALGO R, BIFET A, GÀLVADA R, MORALES-BUENO R. *Early drift detection method*. In Proceedings of the In Fourth International Workshop on Knowledge Discovery from Data Streams, 2006;

BANZHA W. *Genotype-Phenotype-Mapping and Neutral Variation - A Case Study in Genetic Programming*. Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature. Anais...: PPSN III.London, UK, UK: Springer-Verlag, 1994;

BASTIAN A. *Identifying Fuzzy Models utilizing Genetic Programming*. Fuzzy Sets and Systems 113, 333–350, 2000;

BAUER E, KOHAVI R. *An empirical comparison of voting classification algorithms: Bagging, boosting, and variants*, Machine Learning 36(1-2), 105 139, 1999.

BERLANGA F, JESUS M, GONZÁLEZ P, HERRERA F, MESONERO M. *Multiobjective Evolutionary Induction of Subgroup Discovery Fuzzy Rules: A Case Study in Marketing.* Proceedings of the 6th Industrial Conference on Data Mining Conference on Advances in Data Mining: Applications in Medicine, Web Mining, Marketing, Image and Signal Mining. Anais. ICDM'06. Berlin, Heidelberg: Springer-Verlag, 2006;

BIFET A, GALVADA R. *Kalman filters and adaptive windows for learning in data streams*. In Proceedings of the Proceedings of the 9th international conference on Discovery Science Barcelona, Spain, Springer-Verlag.29-40, 2006;

BIFET A, GALVADA R. *Adaptive Learning from Evolving Data Streams*. In Proceedings of the Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIIIL yon, France, Springer-Verlag.249-260, 2009;

BIFET A, READ J, HOLMES G, PAHRINGER B. *Streaming Data Mining with Massive Online Analytics (MOA)*. Data Mining in Time Series and Streaming Databases. Ed. World Scientific, pp. 1-25 (2018) – Chapter 1. https://doi.org/10.1142/9789813228047_0001, 2018;

BREIMAN L. *Bagging predictors*. Machine Learning 24(2), 123 140, 1996;

BREIMAN L. BREIMAN: *Out-of-Bag Estimation*, 1996a*;*

BREIMAN L. *Random forests*. Machine Learning 45(1), 5 32, 2001;

BROWN G, WYATT J, HARRIS R, YAO X. *Diversity creation methods: a survey and categorization*. Inf. Fusion 6 (1) (2005) 5–20; 2005;

BRZEZINSKI D, STEFANOWSKI J. *Reacting to different types of concept drift: The accuracy updated ensemble algorithm*. Neural Networks and Learning Systems, IEEE Transactions on 25.1, pp. 81-94, 2014;

CELES W, CERQUERIA R, RANGEL J.L. Introdução à Estrutura de Dados. Editora Campus, 2014;

CHEN TC WU Y, LEE Y. *A GA-based Fuzzy Mining Approach to Achieve a Trade-off Between Number of Rules and Suitability of Membership Functions*". Soft Comput., v. 10, n. 11, p. 1091–1101, 2006;

DHARWADKER A, PIRZADA S. *Applications of Graph Theory*. [s.l.] Create Space Independent Publishing Platform, 2011;

DIAS D. "Programação Genética Linear com Inspiração Quântica", Tese de Doutorado – Departamento de Engenharia Elétrica – PUC – Rio, 2011;

DIETTERICH TG. *Ensemble Methods in Machine Learning*, Lecture Notes in Computer Science, 2000;

DITZELER G, POLIKAR R. *Incremental Learning of Concept Drift from Streaming Imbalanced Data*, 2013;

DUDA RO. *Pattern Classification*. Wiley, 2001;

EFRON B. *Bootstrap Methods: Another Look at the Jackknife*,.The Annals of Statistics 7(1), 1 26, 1979;

EIBEN, SMITH. *Introduction to Evolutionary Computing*. Stringer, 2003;

ELWEL RL, POLIKAR R. *Incremental Learning of Concept drift in Nonstationary Environments*. IEEE Transactions on Neural Networks 22(10): pp. 1517-1531, 2011.

ESCOVEDO T. Aprendizagem Neuroevolutiva e Detecção de Concept Drift em Ambientes não estacionários. Tese de Doutorado - Departamento de Engenharia Elétrica PUC-Rio, 2015;

FAN W. *StreamMiner: a classifier ensemble-based engine to mine concept drifting data streams*. In Proceedings of the 30th International Conference on Very Large Data Bases, pp. 1257–1260, 2004;

FAN W. *Systematic data selection to mine concept-drifting data streams*. In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 128–137, 2004b;

FAWCETT T. *An Introduction to ROC Analysis*. Pattern Recogn. v. 27, n. 8, p. 861 – 874, 2006;

GAMA J, MENDAS P, CASTILHO G, RODRIGUES. *Learning with drift detection*. Proc. of the 7th Brazilian Symposium on Artificial Intelligence (SBIA'04) - Lecture Notes in Computer Science, vol. 3171. São Luiz do Maranhão, Brazil: Springer, pp. 286–295, 2004;

GAMA J, FERNANDES R, ROCHA R. *Decision trees for mining data streams*. Intell Data Anal.,10,1,23-45, 2006;

GAMA J. ZLIOBAITE I, BIFET A, PECHENIZKIY M, BOUCHACHIA A. *A Survey on Concept Drift Adaptation*. ACM Computing Surveys,46,4,35, 2014;

GAO J, FAN W, HAN J, YU P. *A general framework for mining concept-drifting data streams with skewed distributions*. Proceedings of the 2007 SIAM International Conference on Data Mining, Minneapolis, Minnesota, 2007;

GEYER-SHULZ A. *Fuzzy rule-based expert systems and genetic machine learning*. [s.l.] Physica-Verlag, 1997;

GOMES J. MENASALVEAS E, SOUZA P. *Learning recurring concepts from data streams with a context-aware ensemble*. In Proceedings of the Proceedings of the 2011ACM Symposium on Applied Computing Tai Chung, Taiwan, ACM.994-999, 2011;

GUYON I, WESTON J, BARNHILL S, VAPNIK V. *Gene selection for cancer classification using support vector machines*. Machine Learning 46: 389. https://doi.org/10.1023/A: 1012487302797, 2012;

HANLEY A, MCNEIL B. *The meaning and use of the Area under a Reicever Operating Characteristic (ROC) curve*. Radiology Society of North America. Vol. 143. No. 1. Pages 29, 36, 1982;

HARRIES M, SAMMUT C, HORN K. *Extracting Hidden Context*. Mach. Learn.,32,2, 101-126, 1998;

HARRIES M. *SPLICE-2 Comparative Evaluation: Electricity Pricing*. Mach. Learn.,32,2, 101-126, 1999;

HEWAK N, KOHAIL S. *Learning Concept Drift Using Adaptive Training Set Formation Strategy*. International Journal of Technology Diffusion (IJTD),4,1,33-5, 2013;

HOLSINGER K. *The neutral theory of molecular evolution*. In proceedings, 2011;

IKONOMOVSKA E, GAMA J, DZERSKI S. *Learning model trees from evolving data streams*. Data Min. Knowl. Discov.,23,1,128-168, 2011;

JACKOWSKI K. *Fixed-size ensemble classifier system evolutionarily adapted to a recurring context with an unlimited pool of classifiers*. Pattern Anal Applic (2014) 17: 70. https://doi.org/10.1007/s10044-013-0318-x.

JOACHIMS T. *Text Categorization with Support Vector Machines: Learning with many relevant features*. ECML'98 Proceeding of the 10th European Conference of Machine Learning - Pages 137-142, 1998;

KADWE Y, SURVAWANSHI V. *A Review on Concept Drift*. IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 17, Issue 1, Ver. II (Jan – Feb. 2015), PP 20-26, 2015;

KARNICK M, AHISKALI M, MUHLBAIER MD, POLIKAR R. *Learning concept drift in nonstationary environments using an ensemble of classifiers based approach*. In IJCNN, pp. 3455–3462, 2008;

KLINKENBER R. *Learning drifting concepts: Examples election vs. example weighting*. Intell.DataAnal.,8,3,281-300, 2004;

KOH JL, LIN CY. *Concept Shift Detection for Frequent Item sets from Sliding Windows over Data Streams*. Springer Berlin Heidelberg,334-348., 2009;

KOLTER J, MALOOF M. *Dynamic weighted majority: an ensemble method for concept drifts*, 3th IEE International Conference on Data Mining ICDM 2003.

KOLTER J, MALOOF M. *Using additive expert ensembles to cope with concept drift*. In Proceedings of the 22nd International Conference on Machine Learning, pp. 449–456, 2005b;

KOLTER J, MALOOF M. *Dynamic weighted majority: An ensemble method for drifting concepts*, Journal of Machine Learning Research, vol. 8, pp. 2755–2790, 2007;

KOZA JR. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992;

KRAWACZYK B, MINKU L, GAMA J , STEFANOWSKI J, WOZNIAK M. "Ensemble leaning for data stream analysis: A survey". https://doi.org/10.1016/j.inffus.2017.02.004; 1566-2535 Elsevier B.V, 2017;

KROGH A. VEDELSBY J. *Neural network ensembles, cross validation, and active learning*. Advances in Neural Information Processing Systems, 7, 1995, pp. 231–238, 1995;

KUNCHEVA LI. *Classifier ensemble for changing environments*. In Multiple Classifier Systems, vol. 3077. New York: Springer- Verlag, 2004;

KUNCHEVA LI. *Classifier ensemble for detecting concept change in streaming data: Overview and perspectives*. In Proc. Eur. Conf. Artif. Intell, pp. 5–10, 2008;

KUNCHEVA LI. *Using control charts for detecting concept change in streaming data*. School of Computer Science, Bangor University,UK, 2009;

LU J, DONG F, LIU A, GU F. *Learning under Concept Drift: A Review*. In IEEE Transactions on Knowledge and Data Engineering. doi: 10.1109/TKDE.2018.2876857, 2018;

LUDIMILA I, SÀNCHEZ J: *Nearest Neighbour Classifiers for Streaming Data with Delayed Labelling*. Eighth IEEE International Conference on Data Mining icdm, pp.869, 2008;

MEIER J. K, BRUDNEY L. J, BOHTE J. *Applied Statistics for Public and Nonprofit Administration.* Cengage Learning, ISBN-13: 978-1-285-73723-2, 2014.

MICHLEWICZ Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Artificial Intelligence Series. Springer-Verlag: Berlin, 1992;

MILLER J. *What bloat? Cartesian Genetic Programming on Boolean problems*. Genetic and Evolutionary Computation Conference late breaking papers. Anais 2001.

MILLER J. *Cartesian Genetic Programming*. Edition ed. Heidelberg: New York: Springer, 2011;

MINKU L, WHITE A, YAO X. *The impact of diversity on online ensemble learning in the presence of concept drift*. IEEE Trans. Knowl. Data Eng., vol. 22, no. 5, pp. 730–742, 2010;

MINKU L, YAO X. *DDD: A New Ensemble Approach for Dealing With Concept Drift*. IEEE Transactions on Knowledge and Data Engineering, IEEE, v. 24, n. 4, pp. 619-633, 2012;

MITCHELL M. *An Introduction to Genetic Algorithms*. MIT Press. Disponível em: <http://www.boente.eti.br/fuzzy/ebook-fuzzy-mitchell.pdf>. Acesso em: jan. 2018.

MORENO-TORRES J G, RAEDER T, ALAIZ-RODRIGUEZ R, CHAWLA N, HERRRA F. *A unifying view on data sets shift in classification*. PatternRecogn.,45,1,521-530, 2012;

MOUSS H, MOUSS D, SEFOUHI L. *Test of Page-Hinckley, an approach for fault detection in an agro-alimentary production system*. In Proceedings of the Control Conference, 2004.5th Asian. 815-818 Vol.812., 2004;

MUTHUKRISHNAN S, BERG E, WU Y. *Sequential Change Detection on Data Streams*. In Proceedings of the Data Mining Workshops.ICDM Workshop. Seventh IEEE International Conferenceon.551-550, 2007;

NISHIDA K. *Learning and Detecting Concept Drift*. *PhD thesis*, Hokkaido University, Japan., 2008. URL: http://lis2.huie.hokudai.ac.jp/~knishida/paper/nishida2008-dissertation.pdf

NISHIDA K, YAMAUCHI K. *Learning, detecting, understanding, and predicting concept changes*. International Joint Conference on Neural Networks (IJCNN) (pp. 2280-2287, 2009;

LINDEN R. Algoritmos Genéticos. EditoraCiência Moderna, 3a edição, 2012;

PINTO N. Detecção de Alterações Respiratórias na Fibrose Cística Através da Técnica de Oscilações Forçadas e Algoritmos de Aprendizado de Máquinas. Dissertação de Mestrado – Faculdade de Engenharia – UERJ. 2008;

POLI R, MCPHEE N, LANGDON W. *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd: United Kingdom. 2008;

POLIKAR R, ELWELL R. *Datasets for Evaluating Concept drift/NSE Algorithms*. IEEE Transactions on Knowledge and Data Engineering. October - DOI: 10.1109/TKDE.2012.136.2013;

RECHENBERG I. *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Tese de P.h.D , Universidade Técnica de Berlim, Alemanha, 1971;

ROKACH L. *Ensemble-based classifiers*. Springer Science+Business Media B.V. 2009 DOI 10.1007/s10462-009-9124-7. 2009;

SANTOS A. Síntese de Árvores de Padrões Fuzzy através de Programação Genética Cartesiana. Dissertação de Mestrado – Faculdade de Engenharia – UERJ. 2014;

SCHLIMMER J, GRAGER R. *Incremental learning from noisy data*. Machine Learning, 1(3): pp. 317-354. 1986;

SCHOLZ M, KLINKENBERG R. *An ensemble classifier for drifting concept.* In Proceedings of the 2nd International Workshop on Knowledge Discovery in Data Stream, pp. 53–64.2005;

SCHOLZ M, KLINKENBERG R. *Boosting classifiers for drifting concepts*. Intelligent Data Analysis, vol. 11, no. 1, pp. 3–28.

SENGE R, HULLERMEIR E. *Top-Down Induction of Fuzzy Pattern Trees*, Trans. Fuz Sys., v. 19, n. 2, p. 241–252, 2011;

SENGE R, HULLERMEIR E. *n IEEE Transactions on Fuzzy Systems, vol. 23, no. 6, pp. 2024-2033, Dec. 2015. doi: 10.1109/TFUZZ.2015.2396078*

SHARKEY AJC, SHARKEY NE. *Combining diverse neural nets*. Knowl. Eng. Rev. 12 (3) (1997) 231–247.1997;

SINGH D, SWAROOP V, MALAVIVA M. *Data Security and Privacy in Data Mining: Research Issues & Preparation*. International Journal of Computer Trends and Technology – volume 4 Issue 2, 2013;

STANLEY KO. *Learning concept drift with a committee of decision trees*, Department of Computer Sciences, University of Texas at Austin, Tech. Rep. AI-03-302, 2003;

STREET NW, KIM Y. *A streaming ensemble algorithm (SEA) for large-scale classification*. In Proceedings of the Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining San Francisco, California, ACM.377-382, 2001.

TSANG C, KWON S, WANG H. *Genetic-fuzzy Rule Mining Approach and Evaluation of Feature Selection Techniques for Anomaly Intrusion Detection*. Pattern Recogn., v. 40, n. 9, p. 2373–2391, 2007;

TSYMBAL A. *The problem of concept drift: Definitions and related work*, Tech. Rep, 2004;

UTGOFF P, BERKMAN N, CLOUSE J. *Decision tree induction based on efficient tree restructuring*. Machine Learning, 29:5–44, 1997;

VORBURGER P, BERNSTEIN A. *Entropy-based Concept Shift Detection*. In Proceedings of the Data Mining, 2006. ICDM'06. Sixth International Conferenceon.1113-1118, 2006;

ZENOBI G, CUNNINCGHAM P. *Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error*, in: Machine Learning: ECML 2001, 2001, pp. 576–587, 2001;

ZLIOBAITE I. *Learning under Concept Drift: an Overview*. Computing Research Repository (CoRR),1,4784, 2010;

ZLIOBAITE I, BIFET A, READ J, PFAHRINGE B, HOLMES G. *Evaluation methods and decision theory for classification of streaming data with temporal dependence*, Machine Learning, 455–482, 2015. http://doi.org/10.1007/s10994-014-5441-4.

ZUANG Z, GEDEON T, NIKRAVESH N. *Pattern Trees Induction: A New Machine Learning Method*. IEEE Transactions on Fuzzy Systems, v. 16, n. 4, p. 958–970, 2008;

WANG H, FAN W, YU P, HAN J. *Mining Concept-Drifting Data Streams Using Ensemble Classifiers*. In Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Disc. and Data Mining (KDD). ACM, 226, 2003;

WEIHS C. *Classification as a Tool for Research*. InProceedings of the 11th IFCS Biennial Conference and 33rd Annual Conference of the Gesellschaft für Klassifikation e.V., Dresden, March 13-18, 2009. [s.l.] Springer, 2010;

WIDMER G, KUBAT M. *Learning in the presence of concept drift and hidden contexts*. Mach. Learn., vol. 23, no. 1, pp. 69-101, 1996;

YIN X, HUGANG K, HAO H. *DE2: dynamic ensemble of ensembles for learning nonstationary data* Neurocomputing, 2015;
http://dx.doi.org/10.1016/j.neucom.2014.06.092

APÊNDICE - REPRESENTAÇÃO GRÁFICA DAS ÁRVORES DE PADRÕES *FUZZY*

Neste apêndice são apresentadas as demais árvores geradas na classificação das bases de dados não estacionárias.

- Base de dados Rotating Checkerboard - Constante

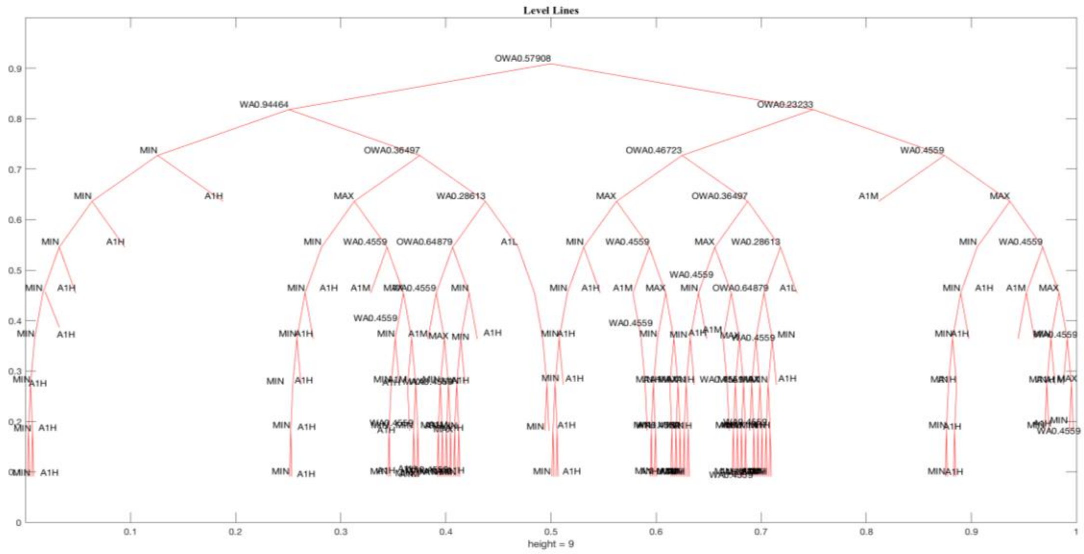


Figura 28: APF saída 1 - Rotating Checkerboard Constante.

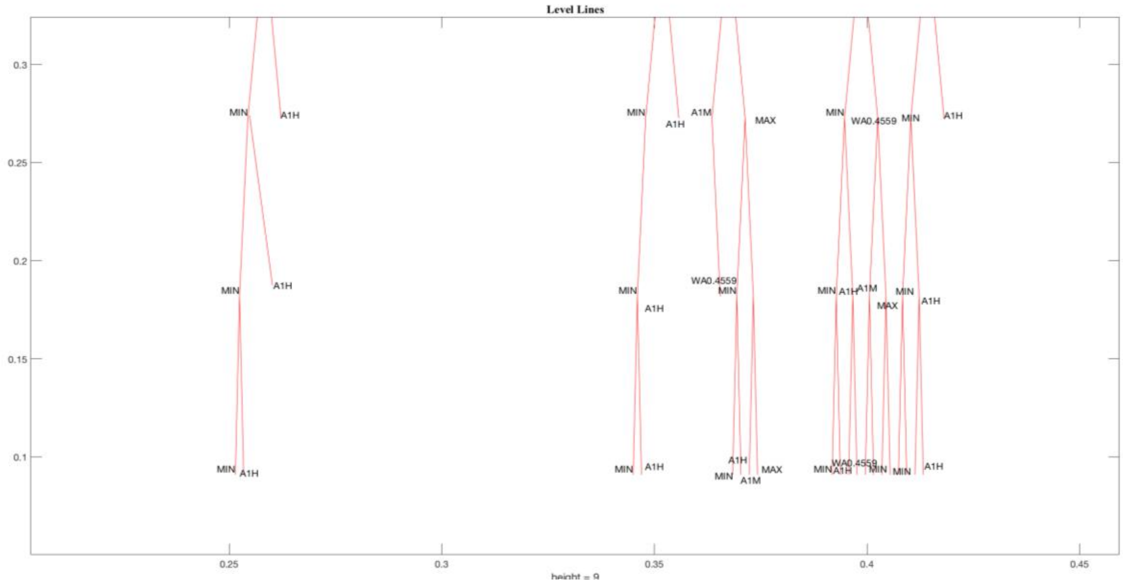


Figura 29: APF saída 1 - Rotating Checkerboard Constante – aproximação do intervalo entre 0.2 e 0.4.

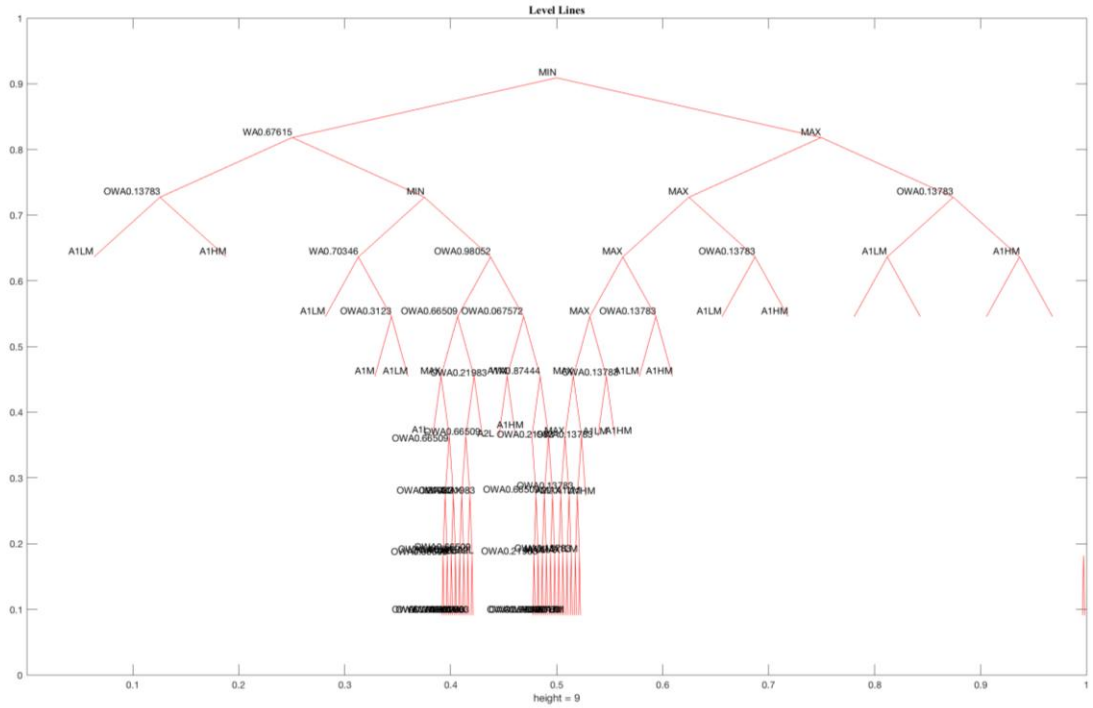


Figura 30: AFP saída 2 - Rotating Checkerboard Constante.

Na árvore representante da classe 1 (Figura 28) prevaleceram duas variações do atributo 1, representando por A1. Este atributo variou somente entre A1H (*high)* e A1M (*mean*). Lembrando-se que o arcabouço foi executado com 5 partições *fuzzy.* Já a representante da classe 2 (Figura 29) podem ser observadas variações do atributo A1, porém com tendência a valores em torno da partição média A1LM (*low mean*) e A1HM (*high mean*).

- Base de dados Rotating Checkerboard – Pulso

Para a variação da base de dados Rotating tipo Pulso, ambas Figura 31 e Figura 32 possuem indicado o uso dos dois atributos disponíveis, A1 e A2. Ocorreu uma tendência do atributo A1 para a partição *fuzzy* de valores médios (A1LM, A1HM) e o atributo A2 para a partição *fuzzy* de valores baixos representados por A2L.

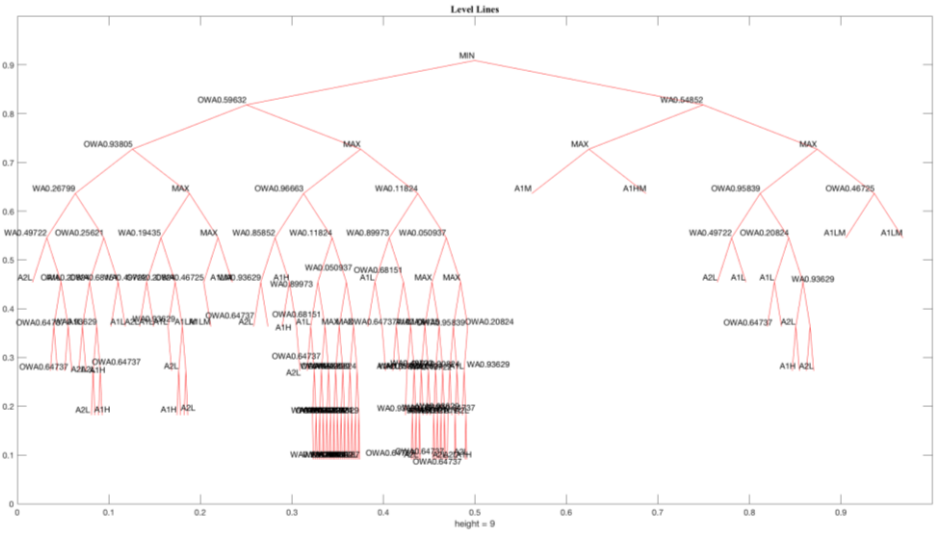


Figura 31: AFP saída 1 - Rotating Checkerboard Pulso.

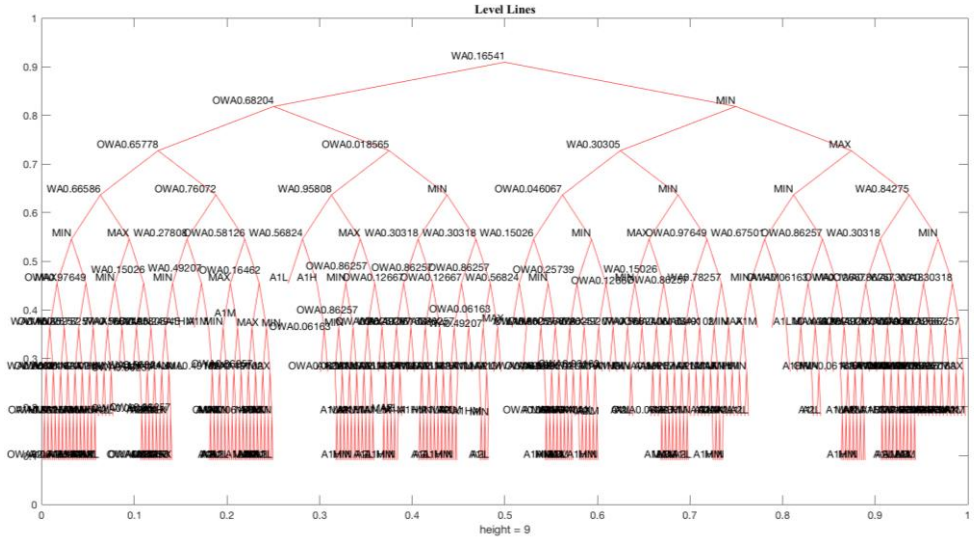


Figura 32: AFP saída 2 - Rotating Checkerboard Pulso.

- Base de dados Rotating Checkerboard – Exponencial

Para a variação da base de dados Rotating tipo Exponencial, ambas Figura 33 e Figura 34 possuem indicado o uso dos dois atributos disponíveis, A1 e A2. Ocorreu uma tendência do atributo A1 para a partição *fuzzy* de valores baixo, médio e alto. Já o atributo A2 se concentrou na partição de valores baixo A2L.

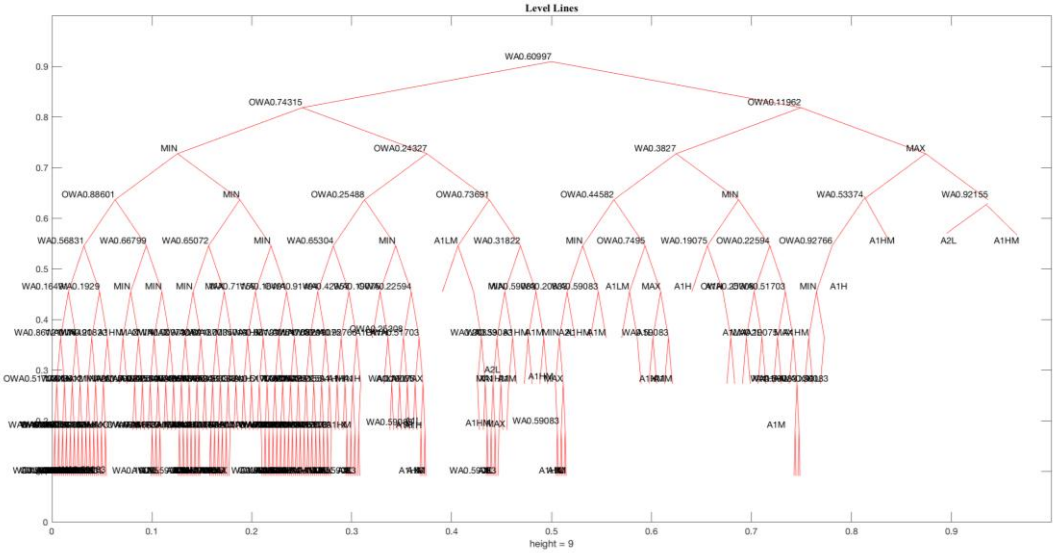


Figura 33: APF saída 1 - Rotating Checkerboard Exponencial.

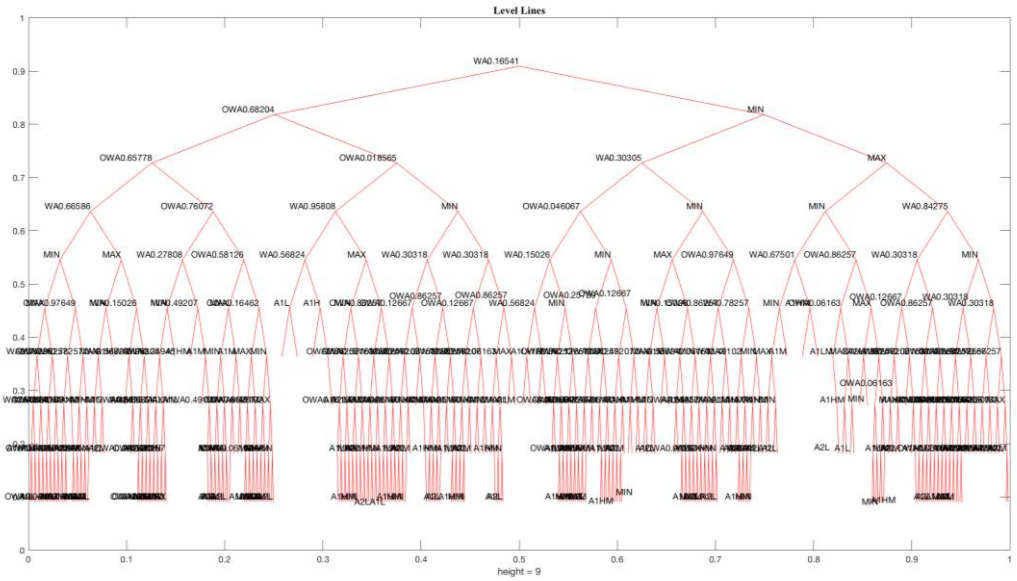


Figura 34: AFP saída 2 – Rotating Checkerboard Exponencial.

- Base de dados Nebraska

A Figura 35 e Figura 36 representam as saídas da AFP para a base de dados Nebraska. Diferentemente das árvores geradas para a base de dados Rotating, o classificador usou 5 dos 8 atributos da base de dados Nebraska na geração das representações das classes. Além disso em ambas as saídas das árvores observou-se a

influência da partição dos valores médios na construção da árvore, da mesma forma os valões médios exerceram maior influência no resultado final devido a proximidade com o topo da árvore.

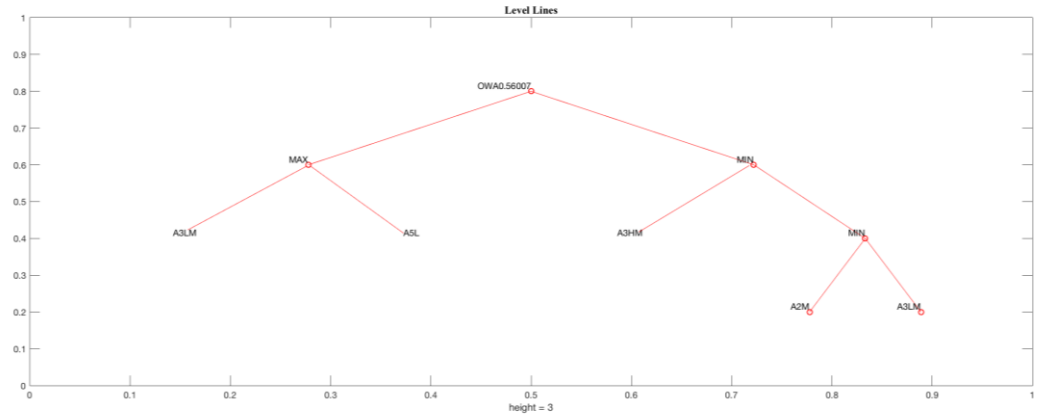


Figura 35: APF saída 1 - Nebraska.

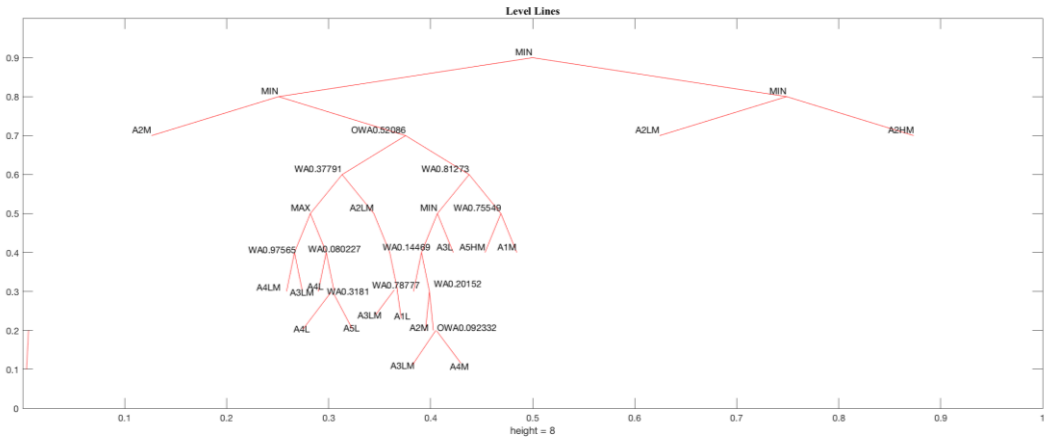


Figura 36: APF saída 2 - Nebraska.