



UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Tecnologia

THIAGO EDUARDO GOUVÊA ANDRADE

MELHORIA NA DETECÇÃO DE CONCEPT DRIFT EM FLUXOS DE DADOS
ONLINE

*IMPROVEMENT ON CONCEPT DRIFT DETECTION FOR ONLINE DATA
STREAMS*

LIMEIRA
2018

THIAGO EDUARDO GOUVÊA ANDRADE

MELHORIA NA DETECÇÃO DE CONCEPT DRIFT EM FLUXOS DE DADOS
ONLINE

Dissertação apresentada à Faculdade de
Tecnologia da Universidade Estadual de
Campinas como parte dos requisitos
exigidos para a obtenção do título de
Mestre em Tecnologia, na Área de
Sistemas de Informação e Comunicação.

Orientadora: Profa. Dra. Ana Estela Antunes da Silva
Coorientador: Prof. Dr. André Leon Sampaio Gradvohl

ESTE EXEMPLAR CORRESPONDE À VERSÃO
FINAL DA DISSERTAÇÃO DEFENDIDA PELO
ALUNO THIAGO EDUARDO GOUVÊA
ANDRADE, E ORIENTADA PELA PROFA. DRA.
ANA ESTELA ANTUNES DA SILVA

LIMEIRA

2018

Ficha catalográfica

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Faculdade de Tecnologia
Felipe de Souza Bueno - CRB 8/8577

An24m Andrade, Thiago Eduardo Gouvêa, 1986-
An24m Melhoria na detecção de *concept drift* em fluxos de dados online / Thiago
Eduardo Gouvêa Andrade. – Limeira, SP : [s.n.], 2018.

Orientador: Ana Estela Antunes da Silva.
Coorientador: André Leon Sampaio Gradvoohl.
Dissertação (mestrado) – Universidade Estadual de Campinas,
Faculdade de Tecnologia.

1. Mineração de dados (Computação). 2. Fluxo de dados
(Computadores). I. Silva, Ana Estela Antunes da, 1965-. II. Gradvoohl, André
Leon Sampaio, 1973-. III. Universidade Estadual de Campinas. Faculdade de
Tecnologia. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Improvement on concept drift detection for online data streams

Palavras-chave em inglês:

Data mining

Data flow computing

Área de concentração: Tecnologia e Inovação

Titulação: Mestre em Tecnologia

Banca examinadora:

Ana Estela Antunes da Silva [Orientador]

Fabricio Aparecido Breve

Ivan Luiz Marques Ricarte

Data de defesa: 12-07-2018

Programa de Pós-Graduação: Tecnologia



FOLHA DE APROVAÇÃO

Abaixo se apresentam os membros da comissão julgadora da sessão pública de defesa de dissertação para o Título de Mestre em Tecnologia na área de concentração de Sistemas de Informação e Comunicação, a que submeteu o aluno Thiago Eduardo Gouvêa Andrade, em 12 de julho de 2018 na Faculdade de Tecnologia- FT/ UNICAMP, em Limeira/SP.

Profa. Dra. Ana Estela Antunes da Silva

Presidente da Comissão Julgadora

Prof. Dr. Fabricio Aparecido Breve

Universidade Estadual Paulista - UNESP

Prof. Dr. Ivan Luiz Marques Ricarte

FT - Unicamp

A Ata da defesa com as respectivas assinaturas dos membros encontra-se no processo de vida acadêmica da aluna na Universidade.

AGRADECIMENTOS

A Deus, por manter as chamas da coragem, da humildade, da persistência e do equilíbrio sempre acesas para superar todos os desafios e obstáculos que foram impostos pela vida durante essa jornada.

Ao meu pai (*in memorian*), Pedro Lúcio Leone Andrade, por ser um exemplo de que o caminho da honestidade e o caminho da coragem são os caminhos do sucesso mesmo que esse sejam os caminhos mais difíceis.

À minha mãe, Angela Maria Gouvêa Andrade, por ser o maior suporte nos momentos em que a vida mais exigiu equilíbrio e por me ensinar que ‘quando o servidor está pronto o serviço aparece’.

À minha noiva, Vanessa Marinello de Souza, por ser uma inspiração de persistência e por estar presente em todos os finais de semana e feriados estudando ao meu lado.

À Professora Ana Estela Antunes da Silva, orientadora do trabalho, pelo conhecimento compartilhado, pela orientação no trabalho, por acreditar no meu trabalho, pela persistência em me manter focado nessa trajetória e pela dedicação na tarefa de ensinar a ciência.

Ao Professor André Leon Sampaio Gradvohl, coorientador, pelo conhecimento compartilhado, pela orientação no trabalho, por ser uma referência e inspiração na arte do saber e por demonstrar que se você trabalhar com o que ama, você não terá que trabalhar um único dia de sua vida.

Ao Professor Guilherme Palermo Coelho, pela contribuição técnica valiosa que foi dada desde a elaboração da proposta de projeto até a conclusão do trabalho, pela facilidade que conseguiu transmitir o conhecimento de uma maneira cristalina e por ser uma referência e inspiração na atividade docente.

Aos meus amigos, pela ajuda prestada para revisar o texto desse trabalho e pela motivação dada quando as dificuldades apareceram.

RESUMO

Algoritmos clássicos de mineração de dados podem apresentar uma capacidade limitada quando são utilizados em fluxos de dados *online*. Isso ocorre porque esse tipo de fluxos de dados não apresenta um comportamento estático, *i.e.* a quantidade de dados que chegará, a velocidade de chegada dos dados e a duração dos fluxos costumam ser fatores desconhecidos e podem mudar ao longo do tempo. Além disso, em ambientes de aplicações reais o padrão de dados também pode mudar ao longo do tempo. Essa mudança que ocorre no padrão dos dados é chamada de *Concept Drift* e torna desaconselhável a utilização dos algoritmos clássicos de mineração de dados para essa tarefa. Por isso, é importante desenvolver algoritmos que sejam capazes de lidar com situações em que os algoritmos clássicos de mineração de dados não apresentam um desempenho satisfatório. Com base nesses desafios pesquisadores têm buscado desenvolver algoritmos que sejam capazes de identificar *Concept Drifts* de maneira rápida, já que isso previne que ocorra uma perda grande de acurácia que é motivada por erros de identificação de um novo padrão das instâncias de dados. Também é importante que o algoritmo seja rápido para que não seja necessário armazenar em memória temporária algumas instâncias de dados que ainda não foram processadas. Motivado por esses desafios esse trabalho propõe três propostas de melhoria na tarefa de detecção de *Concept Drift* em fluxos de dados *online*: o *Fading*, o *Reduced Boundary* e uma melhoria no gerenciamento da janela de dados do algoritmo-base que é utilizado nesse trabalho, o EDIST2 (KHAMASSI, SAYED-MOUCHAWEH *et al.*, 2015). Com essas propostas de melhoria foi possível, em alguns cenários de execução, reduzir o tempo de CPU, o consumo de memória RAM e a acurácia média em relação ao EDIST2. Os resultados que foram encontrados podem ser considerados promissores já que o algoritmo EDIST2 teve um desempenho superior ao desempenho de algoritmos conhecidos em mineração de dados como DDM, EDDM e ADWIN em termos de acurácia média, tempo de CPU e consumo de memória RAM.

Palavras-chave: mineração de dados; fluxos de dados *online*; *concept drift*

ABSTRACT

Classic data mining algorithms can show a limited capacity whenever used with online data streams. It happens because an online data stream does not show a static behavior, i.e. the data quantity, the velocity of arriving data and the stream duration use to be unknown factors and can change over time. Besides that, in real application environments data pattern can change over time as well. This data pattern change is called Concept Drift and it is not advisable use classic data mining algorithms for this task. Therefore, it is important to develop algorithms capable of handle situations whenever classic data mining algorithms does not have enough performance. Based on these challenges, researchers have been seeking develop algorithms capable of quickly identify Concept Drifts, since it avoids an accuracy lost that is caused by identification errors of a new data instance pattern. It is also important that the algorithm would be quick enough in order to avoid allocating temporary memory spaces for some data instances were not processed yet. Motivated by these challenges, this work proposes three different approaches for detecting Concept Drift patterns within online data streaming: Fading, Reduced Boundary and the enhancement on managing data-window from the base algorithm used into this work, EDIST2 (KHAMASSI, SAYED-MOUCHAWEH et al., 2015). Given these enhancement proposals it was possible, in some implementation scenarios, to reduce CPU time and RAM memory consuming, and improve the average accuracy relative to EDIST2 algorithm. Results were found can be considered promising, since EDIST2 algorithm had a superior performance against known data mining algorithms, such as DDM, EDDM and ADWIN in terms of average accuracy, CPU speed and RAM memory consumption.

Keywords: *data mining, online data streams, concept drift*

LISTA DE ABREVIATURAS

ADWIN	<i>Adaptive window</i>
ARH	<i>Abrupt Rotating Hyperplane</i>
AS	<i>Abrupt Stagger</i>
ASEA	<i>Abrupt Streaming Ensemble Algorithm</i>
CPU	<i>Central Processing Unit</i>
D	Distância mínima aceitável entre erros de classificação na proposta de melhoria <i>Reduced Boundary</i>
d	Distância entre erros de classificação na proposta de melhoria <i>Reduced Boundary</i>
D_{perr}	Distância média entre erros de classificação na proposta de melhoria <i>Fading</i>
DDM	<i>Drift Detect Method</i>
DMA	Distância Mínima Aceitável na proposta de melhoria <i>Fading</i>
DP	Desvio Padrão de uma janela de dados
d_{recente}	Distância de um erro de classificação recente para o erro de classificação anterior
EDDM	<i>Early Drift Detect Method</i>
GRH	<i>Gradual Rotating Hyperplane</i>
GS	<i>Gradual Stagger</i>
GSEA	<i>Gradual Streaming Ensemble Algorithm</i>
KDD	<i>Knowledge Discovery from Data</i>
L	Quantidade de erros de classificação que o <i>Fading</i> irá lembrar (Lembrança)
MERPID	Modelo Estatístico que Representa o Padrão das Instâncias de Dados
MOA	<i>Massive Online Analysis</i>
N	Limite de erros de classificação do EDIST2 para verificar se ocorreu um <i>Concept Drift</i>
P_i	Peso de cada lembrança(L) da proposta de melhoria <i>Fading</i>
R	Repetitividade da condição D na proposta de melhoria <i>Reduced Boundary</i>
RAM	<i>Random Access Memory</i>
SEA	<i>Streaming Ensemble Algorithm</i>
μ_{WG}	Valor médio da janela de dados global do EDIST2
μ_{WL}	Valor médio da janela de dados local do EDIST2

LISTA DE ILUSTRAÇÕES

FIGURA 1 - EXEMPLO DE ÁRVORE DE DECISÃO	21
FIGURA 2 - ALGORITMO <i>HOEFFDING TREES</i>	22
FIGURA 3 - TIPOS DE FLUXOS DE DADOS: (A) <i>OFFLINE</i> (B) <i>ONLINE</i>	23
FIGURA 4 - TIPOS DE <i>CONCEPT DRIFT</i> : (A) REPENTINO, (B) INCREMENTAL, (C) GRADUAL, (D) RECORRENTE, (E) LIGEIRO E (F) RUÍDO.....	25
FIGURA 5 - MÁQUINA DE ESTADOS DO EDIST1 E DO EDIST2	28
FIGURA 6 – TRATAMENTO ATUAL DAS JANELAS DE DADOS DO EDIST2 NO ESTADO <i>INCONTROL</i>	30
FIGURA 7 – TRATAMENTO ATUAL DAS JANELAS DE DADOS DO EDIST2 NO ESTADO <i>WARNING</i>	31
FIGURA 8 - TRATAMENTO ATUAL DAS JANELAS DE DADOS DO EDIST2 NO ESTADO <i>DRIFT</i>	32
FIGURA 9 – HISTÓRICO DE ACURÁCIA DOS ALGORITMOS (A) EDIST2, (B) EDIST1, (C) DDM, (D) ADWIN E (E) EDDM PARA CLASSIFICAR INSTÂNCIAS DO FLUXO DE DADOS <i>ROTATING HYPERPLANE</i> QUE CONTÉM <i>CONCEPT DRIFT</i> DO TIPO GRADUAL.....	33
FIGURA 10 – PROPOSTA DE MELHORIA NO TRATAMENTO DAS JANELAS DE DADOS DO EDIST2	42
FIGURA 11 - EXEMPLO DE FUNCIONAMENTO DA PROPOSTA DE MELHORIA <i>FADING</i>	46
FIGURA 12 - FUNCIONAMENTO DO <i>REDUCED BOUNDARY</i> PARA IDENTIFICAR UM <i>CONCEPT DRIFT</i>	48
FIGURA 13 - DESEMPENHO DA ACURÁCIA DAS PROPOSTAS DE MELHORIA EM RELAÇÃO AO EDIST2 NO CENÁRIO DE DADOS EXPANDIDO.....	74
FIGURA 14 - DESEMPENHO DO TEMPO DE CPU DAS PROPOSTAS DE MELHORIA EM RELAÇÃO AO EDIST2 NO CENÁRIO DE DADOS EXPANDIDO.....	75
FIGURA 15 - DESEMPENHO DO CONSUMO DE RAM DAS PROPOSTAS DE MELHORIA EM RELAÇÃO AO EDIST2 NO CENÁRIO DE DADOS EXPANDIDO.....	77



LISTA DE QUADROS

QUADRO 1 - CRITÉRIO DE AVALIAÇÃO DO <i>FADING</i> PARA DETECÇÃO DE <i>CONCEPT DRIFT</i>	44
QUADRO 2 - CONDIÇÃO PARA QUE O <i>REDUCED BOUNDARY</i> IDENTIFIQUE UM <i>CONCEPT DRIFT</i>	47
QUADRO 3 - CONFIGURAÇÕES DO HARDWARE UTILIZADO PARA DESENVOLVIMENTO DE CÓDIGO E EXECUÇÃO DOS EXPERIMENTOS	49
QUADRO 4 – FLUXOS DE DADOS UTILIZADOS NA ANÁLISE DE SENSIBILIDADE	55
QUADRO 5 - CONFIGURAÇÃO UTILIZADA PARA A REALIZAÇÃO DOS EXPERIMENTOS EM CENÁRIO DE DADOS EXPANDIDO	72

LISTA DE TABELAS

TABELA 1 – CONFIGURAÇÕES DO EDIST2 PARA DETECÇÃO DE <i>CONCEPT DRIFT</i>	56
TABELA 2 – CENÁRIOS DE TESTE UTILIZADOS EM CADA EXPERIMENTO DA PROPOSTA DE MELHORIA <i>FADING</i>	57
TABELA 3 - RESULTADOS DA ANÁLISE DE SENSIBILIDADE DA PROPOSTA DE MELHORIA <i>FADING</i> PARA FLUXOS DE DADOS QUE POSSUEM <i>CONCEPT DRIFT</i> DO TIPO REPENTINO	60
TABELA 4 - RESULTADOS DA ANÁLISE DE SENSIBILIDADE DA PROPOSTA DE MELHORIA <i>FADING</i> PARA FLUXOS DE DADOS QUE POSSUEM <i>CONCEPT DRIFT</i> DO TIPO GRADUAL	62
TABELA 5 - CENÁRIOS DE TESTES UTILIZADOS EM CADA EXPERIMENTO DA PROPOSTA DE MELHORIA <i>REDUCED BOUNDARY</i>	64
TABELA 6 - RESULTADOS DA ANÁLISE DE SENSIBILIDADE DA PROPOSTA DE MELHORIA <i>REDUCED BOUNDARY</i> PARA FLUXOS DE DADOS QUE POSSUEM <i>CONCEPT DRIFT</i> DO TIPO REPENTINO	66
TABELA 7 - RESULTADOS DA ANÁLISE DE SENSIBILIDADE DA PROPOSTA DE MELHORIA <i>REDUCED BOUNDARY</i> PARA FLUXOS DE DADOS QUE POSSUEM <i>CONCEPT DRIFT</i> DO TIPO GRADUAL.....	68

Sumário

1. INTRODUÇÃO	14
1.1. OBJETIVOS	17
2. REFERENCIAL TEÓRICO	18
2.1. MINERAÇÃO DE DADOS.....	18
2.1.1. <i>Problemas de classificação</i>	19
2.1.2. <i>Árvores de decisão</i>	20
2.2. FLUXOS DE DADOS ONLINE	22
2.3. <i>CONCEPT DRIFT</i>	23
2.3.1. <i>Tipos de Concept Drift</i>	25
2.4. ALGORITMOS DETECTORES DE <i>CONCEPT DRIFT</i>	26
2.4.1. <i>EDIST1</i>	26
2.4.2. <i>EDIST2</i>	29
2.5. CONSIDERAÇÕES.....	34
3. REVISÃO DA LITERATURA	35
3.1. TRABALHOS RELACIONADOS	35
3.2. CONSIDERAÇÕES.....	38
4. METODOLOGIA.....	40
4.1. PROPOSTAS PARA MELHORIA NA DETECÇÃO DE <i>CONCEPT DRIFT</i> EM FLUXOS DE DADOS ONLINE.....	40
4.1.1. <i>Melhoria no gerenciamento de janela de dados intermediária</i>	40
4.1.2. <i>Fading</i>	43
4.1.3. <i>Reduced boundary</i>	46
4.2. AMBIENTE DE INVESTIGAÇÃO	48
4.2.1. <i>Linguagem de programação</i>	48
4.2.2. <i>Hardware e softwares utilizados</i>	49
4.2.3. <i>Conjuntos de dados utilizados nos experimentos</i>	50
4.3. METODOLOGIA DE INVESTIGAÇÃO.....	51
5. ANÁLISE EXPERIMENTAL DE SENSIBILIDADE DAS PROPOSTAS DE MELHORIA	54
5.1. ANÁLISE DE SENSIBILIDADE DA PROPOSTA DE MELHORIA <i>FADING</i>	57
5.2. ANÁLISE DE SENSIBILIDADE DA PROPOSTA DE MELHORIA <i>REDUCED BOUNDARY</i>	63
5.3. CONCLUSÃO DA ANÁLISE DE SENSIBILIDADE	68
6. ANÁLISE EXPERIMENTAL DAS PROPOSTAS DE MELHORIA EM CENÁRIO EXPANDIDO	71
7. CONCLUSÕES, CONTRIBUIÇÃO E TRABALHOS FUTUROS	79
8. BIBLIOGRAFIA	81
APÊNDICE A - GLOSSÁRIO.....	85

APÊNDICE B – FLUXOGRAMA DO ALGORITMO EDIST2.....	89
APÊNDICE C – PSEUDOCÓDIGO DO ALGORITMO EDIST2.....	93

1. Introdução

Em computação, há uma área de pesquisa denominada aprendizado de máquinas que é destinada à pesquisa e ao desenvolvimento de algoritmos que sejam capazes de reconhecer padrões e conceitos ao longo do tempo. O aprendizado de máquinas é dividido em três principais tipos, que são: o aprendizado supervisionado, o aprendizado não-supervisionado e o aprendizado por reforço. Uma área que tem relação com a área de aprendizado supervisionado é a mineração de dados. A mineração de dados consiste na busca de padrões em bases de dados brutos por meio de tarefas como associações, classificação, agrupamento e estimativa (BRAMER, 2013). Ela pode ser aplicada sobre dados persistentes, ou *offline*, como bancos de dados, arquivos de *logs* e registros de transações, dentre outros. A mineração de dados também pode ser aplicada sobre dados *online*, que são dados produzidos em tempo real ou muito próximo disso, ou fluxos de dados *online*, gerados por fontes dinâmicas como sensores, *sites* na *internet*, placas de rede de computadores, etc.

Um aspecto importante que deve ser observado ao pesquisar novos algoritmos de aprendizado para mineração de dados *online*, é que a evolução tecnológica do *hardware* tem possibilitado o aumento da velocidade de geração e de transmissão de dados. Com isso, o volume de dados em movimento em um intervalo de tempo (vazão ou *throughput*) pode ser tão grande que seria impraticável ou até mesmo impossível armazenar estes dados para realizar uma mineração neles posteriormente. Portanto, é primordial que os algoritmos de mineração de dados para fluxos de dados *online* sejam tão rápidos que consigam detectar padrões em tempo real sem que necessitem armazenar muitos ou todos os dados do fluxo.

Apesar das técnicas tradicionais de mineração de dados oferecerem bons resultados quando são aplicadas a ambientes estáticos, isto é, em fluxos de dados *offline*, elas podem não apresentar desempenho tão satisfatório quando aplicadas a ambientes dinâmicos, que é o caso dos fluxos de dados *online* (KHAMASSI, SAYED-MOUCHAWEH *et al.*, 2015). Isso ocorre por um conjunto de fatores, dentre os quais podem-se destacar o grande volume de dados no fluxo, a velocidade de chegada dos dados, a impossibilidade de

determinar previamente o fim de um fluxo de dados e a possibilidade de ocorrer mudança no padrão dos dados, de forma inesperada, ao longo do tempo. Essa mudança no padrão de dados é o que se define por mudança de conceito ou, em inglês, *Concept Drift* (MITCHELL, 1997).

O *Concept Drift* ocorre quando uma fonte de dados dinâmica muda de comportamento de forma inesperada e passa a transmitir dados com um padrão diferente do padrão que estava presente no fluxo de dados (KHAMASSI e SAYED-MOUCHAWEH, 2014). Com isso, é essencial que um algoritmo de mineração para fluxos de dados *online* seja capaz de atualizar continuamente o seu modelo de aprendizado com padrões novos que surgirem no fluxo (WANG, WEI *et al.*, 2003). Caso isso não seja realizado, o algoritmo passará a apresentar uma acurácia baixa e se tornará obsoleto para a tarefa de detecção de padrões.

Para exemplificar uma ocorrência de *Concept Drift* imagine que uma fábrica de sorvetes deseja melhorar o processo produtivo com o objetivo de aumentar a produção de sorvetes no período do ano em que a venda de sorvetes aumenta e de diminuir a produção de sorvetes no período do ano em que a venda de sorvetes diminui. Para isso, a fábrica de sorvetes decide contratar uma empresa de inteligência comercial que dispõe de um sistema que é capaz de identificar mudanças no padrão de vendas de produtos. Então, após realizar a análise por um período de um ano o sistema identifica que a venda de sorvete começa a aumentar no mês de setembro e atinge o ápice no mês de janeiro. Por outro lado, a venda de sorvete começa a cair no mês de março e atinge o menor valor no mês de abril. Além disso, o sistema conseguiu identificar que mesmo no período de muitas vezes ocorreu, algumas vezes, uma queda muito grande na venda de sorvetes. Para estes dias foram inseridas observações de que o dia estava chuvoso. Também, foi possível verificar que duas ao menos duas vezes por semana o número de vendas aumentava. Estes aumentos ocorreram sempre que não era um dia útil, isto é, nos finais de semana e nos feriados.

Com base nas informações que foram coletadas pelo sistema perceba que ocorre *Concept Drift* de duas maneiras. Uma destas maneiras é um aumento gradual na venda de sorvetes no mês de setembro, ou uma redução gradual na venda de sorvetes no mês de abril. A outra maneira que o *Concept*

Drift ocorre é quando a venda de sorvetes aumenta de maneira repentina nos finais de semana e feriados, ou quando a venda de sorvetes diminui de maneira repentina nos dias de chuva.

Veja que para que um algoritmo seja considerado bom para minerar fluxos de dados *online* ele precisa ter uma acurácia boa, ou seja, o algoritmo precisa identificar corretamente o padrão dos dados que surgem. Caso isso não aconteça o algoritmo poderá identificar, de maneira incorreta, um ou mais dados como sendo de um padrão diferente do padrão real.

Além de uma acurácia boa é importante que um algoritmo de mineração de dados em fluxos de dados *online* tenha um custo computacional baixo. Isso porque é inviável, e muitas vezes impossível, redimensionar um *hardware* constantemente para suportar um aumento do consumo de memória principal (*Random Access Memory* – RAM) ou uma demanda crescente de processamento. O aumento de memória RAM pode ocorrer por dois motivos, que são: o algoritmo precisar armazenar em espaço de memória temporário (*buffer*), instâncias de dados que não consegue minerar em tempo real ou o algoritmo utiliza um modelo de aprendizado muito grande para representar o conhecimento do algoritmo (HUANG, 2012). Já a demanda por mais processamento pode ocorrer pela chegada de um volume muito grande de dados, através do fluxo de dados online, de modo que o algoritmo precise ser executado de forma rápida o suficiente para que não seja necessário salvar em *buffer* muitas instâncias de dados para serem analisadas posteriormente (DESAI e JOSHI, 2015).

Tendo em vista as características que foram apresentadas nos parágrafos anteriores, e que são importantes para algoritmos de mineração de dados em fluxos de dados *online*, este trabalho enfrenta os desafios desta área para apresentar propostas de melhoria na detecção de *Concept Drift* em fluxos de dados *online*. Para isso, foi realizada uma investigação sobre um algoritmo detector de *Concept Drift*, o EDIST2 (KHAMASSI, SAYED-MOUCHAWEH et al., 2015), para identificar e propor melhorias na tarefa de detecção de *Concept Drift* em fluxos de dados *online*. O EDIST2 foi escolhido porque apresentou resultados satisfatórios em termos de acurácia e custo computacional quando foi comparado com algoritmos conhecidos da área de mineração de dados

como DDM (GAMA, MEDAS *et al.*, 2004), EDDM (BAENA-GARCÍA, CAMPO-AVILA *et al.*, 2006) e ADWIN (BIFET e GAVALDÀ, 2007).

Para apresentar as investigações que foram realizadas, bem como as propostas de melhoria que são sugeridas ao EDIST2, este trabalho foi organizado da seguinte maneira: o Capítulo 2 traz um referencial teórico dos principais conceitos envolvidos nesse trabalho; o Capítulo **Erro! Fonte de referência não encontrada.** apresenta os trabalhos relacionados ao tema deste trabalho; o Capítulo 4 mostra a metodologia utilizada para investigar o EDIST2 e também introduz as melhorias que são propostas nesse trabalho; no Capítulo 5 são detalhados os experimentos preliminares relacionados as propostas de melhoria; o Capítulo 6 discute os experimentos que foram realizados em cenários mais amplos; e, por fim, o Capítulo 7 aponta as conclusões relacionadas as propostas de melhoria, indica as contribuições relevantes desse trabalho e propõe sugestões de trabalhos futuros.

1.1. Objetivos

O objetivo principal deste trabalho é o de investigar se é possível reduzir o custo computacional e também aumentar a acurácia de algoritmos de detecção de *Concept Drift* em fluxos de dados *online*. Esta investigação foi realizada no algoritmo EDIST2 (KHAMASSI, SAYED-MOUCHAWEH *et al.*, 2015) para verificar pontos de melhoria na lógica original deste algoritmo e para verificar se é possível agregar técnicas complementares de identificação de *Concept Drift* que melhorem o desempenho deste algoritmo.

Como objetivos específicos este trabalho busca investigar pontos de melhoria no modo em que o algoritmo EDIST2 identifica a mudança no padrão dos dados, e também busca investigar se a inserção de critérios de identificação de *Concept Drift* novos, que sejam alternativos ao critério de identificação padrão deste algoritmo, são capazes de reduzir o tempo de CPU e o consumo de memória RAM e, ainda, aumentar a acurácia média deste algoritmo.

2. Referencial teórico

Antes de introduzir as ideias propostas por este trabalho, é importante trazer à luz do entendimento alguns conceitos importantes que serão utilizados adiante. Por isso, este capítulo será destinado ao detalhamento dos principais conceitos envolvidos neste trabalho, que são: mineração de dados, fluxos de dados *online*, *Concept Drift* e algoritmos detectores de *Concept Drift*.

2.1. Mineração de dados

Vivemos hoje em uma era de evolução tecnológica em que os dispositivos estão deixando de ser apenas objetos simples e passando a interagir com os seres humanos ou com outros objetos. Isso ocorre, por exemplo, em tecnologias do tipo *machine-to-machine* onde as máquinas e dispositivos comunicam entre si sem intervenção humana. Esta mudança no comportamento dos objetos tem contribuído cada vez mais para o aumento do volume de dados e de informações que trafegam na rede mundial de computadores.

Outro fator que contribui para o aumento de dados e informações na rede mundial de computadores é a facilidade que os seres humanos possuem para comunicarem-se entre si. Isso pode ocorrer através de troca de mensagem em aplicativos de mensagem instantânea, ou através de comentário em uma postagem de rede social, ou através de chamada de voz etc. Qualquer forma de interação entre seres humanos com seres humanos, ou de seres humanos com máquinas, ou ainda de máquinas com máquinas produz uma grande quantidade de dados e informações na internet.

Através da análise de informações que estão presentes na internet é possível, por exemplo, identificar o perfil de consumo das pessoas, quais atividades físicas elas praticam como *hobby*, preferências de programas de TV, preferências políticas, entre outras. Também é possível detectar fraude em operações financeiras, acionar dispositivos automaticamente com base na leitura de sensores, bloquear mensagens do tipo *spam* e várias outras aplicações que não envolvem interação humana. Todas estas tarefas são possíveis graças a uma técnica chamada mineração de dados (do inglês, *data*

mining), que também é conhecida como *Knowledge Discovery from Data* (KDD).

A mineração de dados é definida por Han, Kamber e Pei (2011) como sendo o processo de descoberta de padrões a partir de grandes quantidades de dados. Já Mitchell (1997) define mineração de dados como um campo de aprendizado de máquinas que aborda a questão de como utilizar melhor dados históricos para descobrir padrões e melhorar o processo de tomada de decisões. Uma outra definição para mineração de dados é feita por Cabena, Hadjinian *et al.* (1998) que diz que a mineração de dados é um campo interdisciplinar que traz diversas técnicas de aprendizado de máquina, reconhecimento de padrões, estatísticas, bancos de dados e visualização para abordar a questão da extração de informações de grandes quantidades de dados.

Alguns exemplos de fontes dos dados que podem ser utilizadas para encontrar padrões através da mineração de dados, são: bancos de dados, *data warehouses*, tráfego *web*, repositórios de dados, informações que são transmitidos dinamicamente, dentre outros. De acordo com Larose (2005), as tarefas mais comuns para realizar uma mineração de dados em fontes de dados, são: classificação, descrição, estimativa, agrupamento e associação. Apesar dessa variedade de tarefas de mineração de dados, o foco deste trabalho está na mineração de dados através da tarefa de classificação.

2.1.1. Problemas de classificação

Em geral, a classificação é um processo de aprendizado que é realizado através de uma função que mapeia um item de dados em uma entre várias classes possíveis. Toda classificação baseada em algoritmos de aprendizagem indutiva recebe como entrada um conjunto de amostras que consistem em um vetor de valores de atributos (também chamado de vetor de características) e uma classe correspondente a este conjunto de características.

O objetivo do aprendizado é criar um modelo de classificação, conhecido como classificador, que irá predizer, com os valores de atributos de entrada disponíveis, a classe para alguma entidade (uma amostra dada). Em outras palavras, a classificação é o processo de atribuir um valor discreto de rótulo (classe) a um registro não rotulado, e um classificador é um modelo (resultado

da classificação) que prediz um atributo – classe de uma amostra – quando outros atributos são dados. Ao fazê-lo, as amostras são divididas em grupos. Por exemplo, uma classificação simples pode agrupar os registros de faturamento do cliente em duas classes específicas: aqueles que pagam suas contas em 30 dias e aqueles que levam mais de 30 dias para pagar.

Diferentes metodologias de classificação são aplicadas hoje em dia em quase todas as disciplinas onde a tarefa de classificação, devido à grande quantidade de dados, requer automação do processo. Exemplos de métodos de classificação usados como parte de aplicações de mineração de dados incluem classificar tendências no mercado financeiro e identificar objetos em grandes bancos de dados de imagens (KANTARDZIC, 2011).

Han, Kamber e Pei (2011) descrevem a classificação como sendo o processo de encontrar um modelo de representação de dados que descreva e distinga classes ou padrões de dados. O modelo de representação de dados, ou classificador, é descrito com base na análise de um conjunto de dados de treinamento, isto é, através de instâncias de dados em que os rótulos de classes são conhecidos.

O classificador é utilizado para prever o rótulo de classe de instâncias de dados em que o rótulo da classe é desconhecido. Este modelo de representação de dados pode ser obtido através de técnicas diferentes, como: as regras de classificação, árvores de decisão, fórmulas matemáticas ou redes neurais. Neste trabalho os modelos de representação dos dados são obtidos através de árvores de decisão.

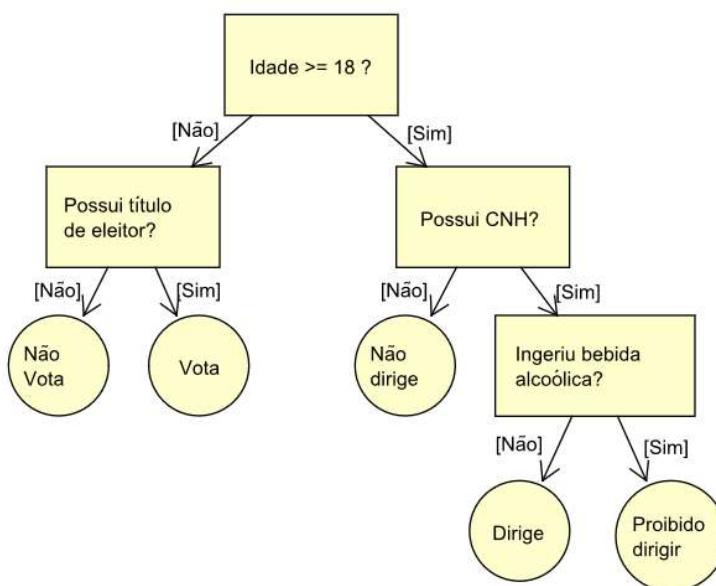
2.1.2. Árvores de decisão

Han, Kamber e Pei (2011) definem árvore de decisão como sendo uma espécie de fluxograma em estrutura de árvores onde cada nó intermediário, isto é, o nó que não é nó folha, representa um teste em um atributo; cada ramo de um nó intermediário representa uma saída para o teste que é realizado no nó intermediário; e, por fim, cada nó folha é responsável por representar o rótulo de uma classe. A Figura 1 ilustra uma árvore de decisão que possui cinco classes ou nós folha, que são: vota, não vota, não dirige, dirige e proibido dirigir. Esta árvore de decisão também possui quatro atributos ou nós

intermediários, que são: maior de 18 anos, possui título de eleitor, possui carteira nacional de habilitação (CNH) e ingeriu bebida alcoólica.

Han, Kamber e Pei (2011) explicam ainda que a construção de classificadores através de árvore de decisão não requer nenhum conhecimento de domínio ou configuração de parâmetros e, portanto, é apropriada para a descoberta de conhecimento de forma exploratória, como é o caso de fluxos de dados *online*. Como exemplos reais os autores citam que os algoritmos de árvores de decisão têm sido usados para classificação em áreas de aplicação, como: medicina, manufatura e produção, análise financeira, astronomia e biologia molecular.

Figura 1 - Exemplo de árvore de decisão



(Do autor)

Neste trabalho todos os experimentos foram realizados utilizando como base o algoritmo classificador *Hoeffding trees*. O algoritmo *Hoeffding trees* é uma árvore de decisão suportada matematicamente pelo limite de *Hoeffding* (HOEFFDING, 1963) que explora o fato de que uma pequena quantidade de amostras, geralmente, é suficiente para escolher um atributo de divisão ideal. Isto significa que algoritmo *Hoeffding trees* é capaz de escolher os nós intermediários, ou nós de decisão, através de uma quantidade pequena de amostragem de dados. Com isso, o classificador *Hoeffding trees* é capaz de aprender a partir de fluxos de dados massivos desde que a fonte geradora de

dados não mude ao longo do tempo. Para ilustrar como é o funcionamento do algoritmo *Hoeffding Trees* foi criada a Figura 2.

Figura 2 - Algoritmo *Hoeffding Trees*

Algorithm Hoeffding tree induction algorithm.

```

1: Let HT be a tree with a single leaf (the root)
2: for all training examples do
3:   Sort example into leaf l using HT
4:   Update sufficient statistics in l
5:   Increment nl, the number of examples seen at l
6:   if nl mod nmin = 0 and examples seen at l not all of same class then
7:     Compute  $\bar{G}_l(X_i)$  for each attribute
8:     Let Xa be attribute with highest  $\bar{G}_l$ 
9:     Let Xb be attribute with second-highest  $\bar{G}_l$ 
10:    Compute Hoeffding bound  $\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n_l}}$ 
11:    if Xa ≠ X∅ and ( $\bar{G}_l(X_a) - \bar{G}_l(X_b) > \epsilon$  or  $\epsilon < \tau$ ) then
12:      Replace l with an internal node that splits on Xa
13:      for all branches of the split do
14:        Add a new leaf with initialized sufficient statistics
15:      end for
16:    end if
17:  end if
18: end for

```

Fonte: (STAHL, GABER *et al.*, 2011)

De acordo com Hulten, Spencer e Domingos (2001) uma característica teoricamente mais atraente da *Hoeffding Trees* em relação a outros algoritmos incrementais de árvore de decisão é que a *Hoeffding Trees* possui garantias sólidas de desempenho. Isso ocorre porque através do limite de *Hoeffding* é possível mostrar que a saída de uma *Hoeffding Trees* é quase idêntica a saída de um aprendiz não incremental quando muitas amostras são utilizadas.

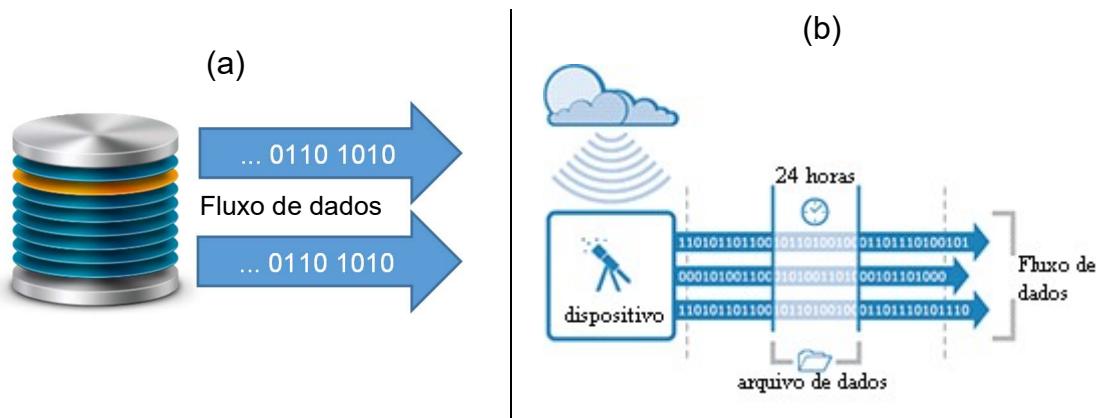
2.2. Fluxos de dados *online*

Os fluxos de dados são vistos como uma sequência de tuplas relacionais, que chegam continuamente ao longo do tempo (DONGRE e MALIK, 2014). Essas tuplas podem conter leituras de sensores, dados de imagens de satélites e de cliques em páginas na web, entre outros. Diferente dos sistemas de bancos de dados que são considerados fluxos de dados *offline*, pois os dados estão disponíveis a qualquer momento, nos fluxos de dados *online*, se os dados não forem processados ou armazenados imediatamente quando surgem, estes dados são perdidos para sempre. Além disso, a velocidade de chegada dos dados poderá ser tão alta que não será

possível armazenar tudo em armazenamento ativo (*i.e.*, em um banco de dados convencional) para posteriormente interagir com os dados no momento da nossa escolha (LESKOVEC, RAJARAMAN e ULLMAN, 2014). Por isso, é essencial que exista algoritmos capazes de tratar a maior quantidade de dados possível existente nos fluxos de dados, dentro das limitações existentes de processamento e de memória para armazenamento temporário. A Figura 3 (a) ilustra os fluxos de dados *offline* vindos de um banco de dados, a Figura 3 (b) mostra o comportamento de fluxos de dados *online* vindos de sensores.

Outra característica dos fluxos de dados *online* é que os dados podem apresentar um comportamento dinâmico, *i.e.* os dados podem mudar de padrão por diversas vezes ao longo do tempo. Isto ocorre porque as fontes geradoras de dados dos fluxos de dados *online* podem sofrer influência do ambiente em que estão situadas ou podem ser motivadas por interação humana. Esta mudança que ocorre no padrão dos dados de um fluxo de dados *online* é chamada *Concept Drift* e pode afetar o desempenho de algoritmos de aprendizado de máquinas baseados em mineração de dados.

Figura 3 - Tipos de fluxos de dados: (a) *offline* (b) *online*.



Fonte (a): (Do autor)

Fonte (b): (U.S. Department of Energy (DOE), 2015)

2.3. *Concept Drift*

Em aplicações do mundo real, muitas vezes, o padrão dos dados não é estável e muda ao longo do tempo. Essa mudança que ocorre no padrão dos dados é o que chamamos de *Concept Drift*. Com o *Concept Drift* em um fluxo de dados, se um modelo de aprendizado não passa por uma atualização periódica ele acaba se tornando obsoleto (TSYMBAL, 2004). Dongre e Malik

(2014) definem que *Concept Drift* é um termo usado para descrever mudanças na estrutura de aprendizado que ocorrem ao longo do tempo, levando a uma queda drástica de acurácia do modelo de aprendizado.

Exemplos da ocorrência de *Concept Drift* na vida real incluem: sistemas de monitoramento, de detecção de spam, de previsões meteorológicas, de categorização e de preferências de clientes, etc. Em sistemas de monitoramento de um banco, por exemplo, ocorre um *Concept Drift* quando o ambiente que está vazio passa a ser ocupado por pessoas, ou seja, antes do início do expediente bancário o padrão de dados é uma imagem estática e quando o expediente bancário inicia o ambiente passa a ter movimentação de pessoas. O mesmo ocorre quando o expediente bancário termina, isto é, o padrão de dados era de imagens com movimentação de pessoas e passa a ser um ambiente sem movimentação alguma.

Outro exemplo utilizando o sistema de monitoramento bancário é a identificação de um invasor. Por exemplo, fora do expediente bancário o padrão de dados deveria ser composto por imagens estáticas, isto é, sem movimentação alguma. Entretanto, imagine que neste banco tenha um guarda noturno que faça uma ronda a cada uma hora durante o período em que o banco está fechado. Então, a cada uma hora o sistema de monitoramento terá uma pequena alteração no padrão de dados das imagens. Com isso, ocorrerá um *Concept Drift* caracterizado por uma duração curta.

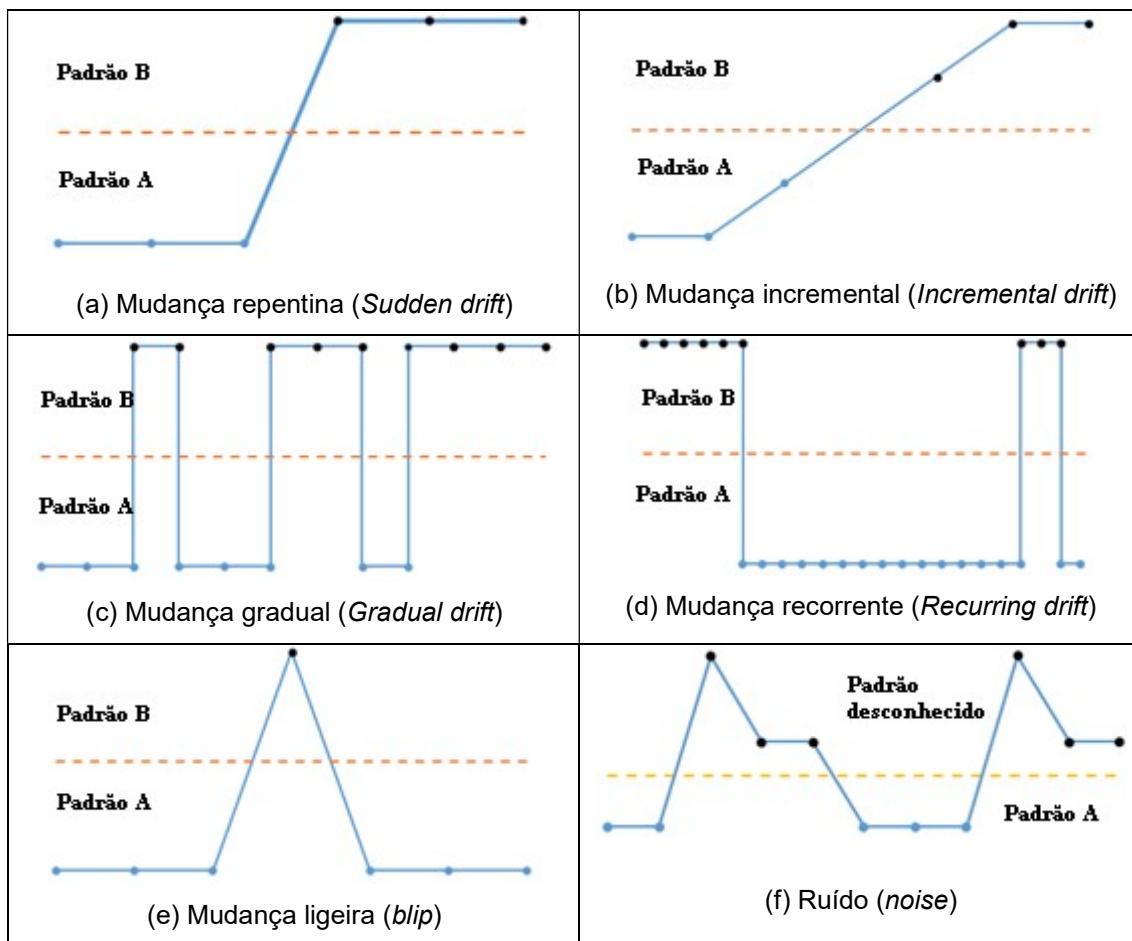
Com base nos exemplos que foram apresentados nos parágrafos anteriores nota-se que o *Concept Drift* pode ocorrer de maneiras diferentes. Para que ele ocorra basta que o padrão de uma ou mais instâncias de dados de um fluxo de dados seja divergente do padrão da maioria das instâncias de dados que estavam presentes no fluxo. Por isso, é importante conhecer as formas diferentes que o *Concept Drift* pode aparecer em um fluxo de dados, quais os tipos de *Concept Drift* que promovem uma mudança completa no padrão dos dados e quais os tipos de *Concept Drift* que promovem uma mudança temporária. Para isso, a Seção 2.3.1 apresentará os tipos de *Concept Drift* mais comuns e também quais são as mudanças que cada tipo de *Concept Drift* promove nos fluxos de dados *online*.

2.3.1. Tipos de *Concept Drift*

O *Concept Drift* pode ser classificado em diferentes tipos de acordo com o comportamento da sua ocorrência. Os tipos de *Concept Drift* foram classificados por Dongre e Malik (2014) como sendo: repentino, gradual, incremental, recorrente, ligeiro (do inglês, *blip*) e ruído. É possível separar os tipos de *Concept Drift* em duas categorias: os de mudança completa do padrão do fluxo de dados, em que estão presentes os tipos repentino, gradual e incremental; e os de mudança temporária, em que estão presentes os tipos recorrente, ligeiro e ruído.

Na mudança completa, se o padrão dos dados muda completamente a partir de determinada instância para outro padrão, significa que ocorreu uma mudança repentina, conforme Figura 4 (a).

Figura 4 - Tipos de *Concept Drift*: (a) repentino, (b) incremental, (c) gradual, (d) recorrente, (e) ligeiro e (f) ruído



Fonte: (Do autor)

Entretanto, se a mudança do padrão atual do fluxo ocorreu gradativamente e de forma contínua, migrando do padrão atual para outro padrão, e a partir de determinado momento começa a surgir apenas instâncias deste outro padrão, então essa mudança caracteriza a ocorrência da mudança incremental, conforme ilustra a Figura 4 (b).

Por fim, se o fluxo de dados começa a alternar instâncias do padrão atual com instâncias de outro padrão, e aos poucos o fluxo começa a apresentar menos instâncias do padrão atual do que do outro padrão, até ser completamente substituído pelo novo padrão, então essa mudança caracteriza a ocorrência de mudança incremental, conforme Figura 4 (c).

Para os casos de *Concept Drift* temporários, o surgimento de algumas instâncias de um padrão de dados antigo em meio às instâncias do padrão de dados atual caracteriza a ocorrência da mudança recorrente. Este tipo de *Concept Drift* está representado pela Figura 4 (d). Entretanto, se surgir apenas uma instância de um outro padrão que já passou no fluxo de dados, significa que ocorreu uma mudança ligeira, conforme ilustrado na Figura 4 (e). E ainda, se ocorrer o surgimento de instâncias que não apresentem um padrão igual entre si, e que o padrão destas instâncias não tenha ocorrido no fluxo de dados, i.e. um padrão desconhecido, é considerada a ocorrência de um ruído, Figura 4 (f).

2.4. Algoritmos detectores de *Concept Drift*

Os dois algoritmos para detecção de *concept drift* da família EDIST são detalhados nesta seção. Inicialmente, comenta-se sobre o algoritmo EDIST1 e, em seguida, sobre o algoritmo EDIST2.

2.4.1. EDIST1

O EDIST1 (KHAMASSI, SAYED-MOUCHAWEH *et al.*, 2013) é um algoritmo de mineração de dados que realiza a detecção de *Concept Drift* através da distância entre erros de classificação em fluxos de dados. Isto é, quando o EDIST1 percebe que ocorreram erros de classificação muito próximos de instâncias de um fluxo de dados é porque o padrão dos dados pode estar mudando ou até mesmo já ter mudado. Assim, se o EDIST1 identificar que o padrão dos dados mudou ele descarta o modelo de

aprendizado que representa o padrão de dados antigo do fluxo de dados e adota um modelo de aprendizado atualizado que contém o padrão de dados novo do fluxo de dados.

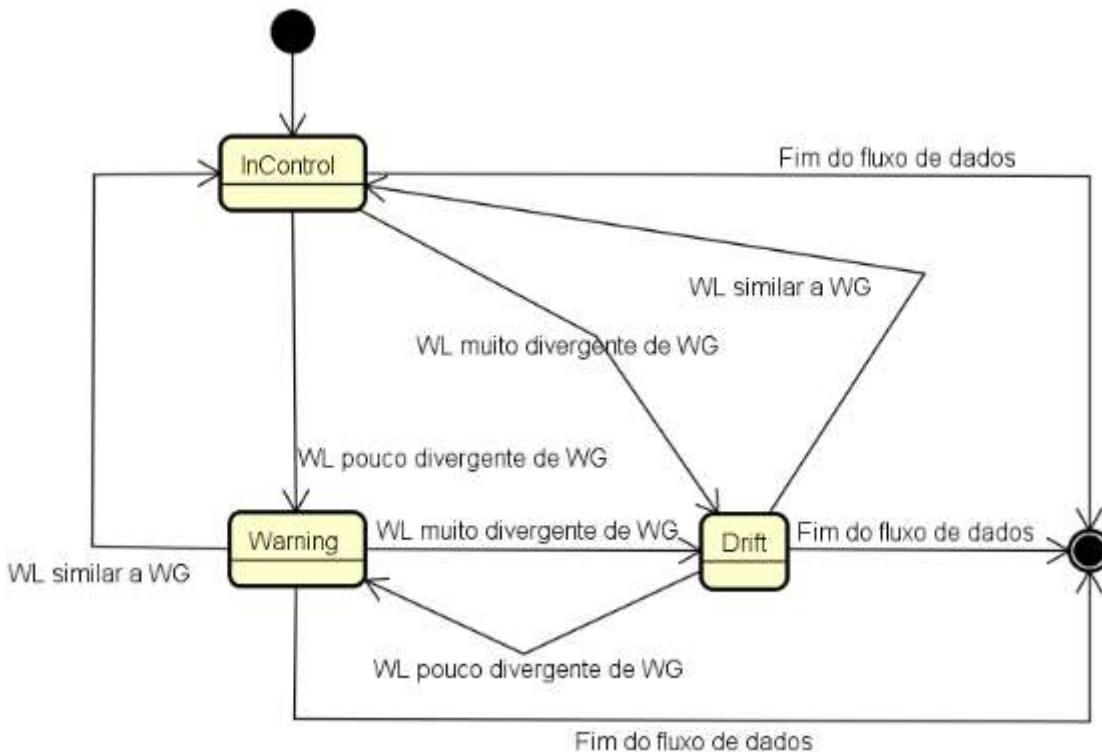
No EDIST1 a detecção de mudança de padrão dos dados é realizada graças a um sistema de janela de dados. Nesse sistema de janela de dados do EDIST1 o padrão dos dados que já passaram pelo fluxo de dados é representado em uma janela de dados chamada de janela global. Já o padrão dos dados que estão chegando no fluxo de dados é representado em uma janela de dados chamada de janela local. Tanto na janela de dados global quanto na janela local, o padrão dos dados é representado através de um Modelo Estatístico que Representa o Padrão das Instâncias de Dados (MERPID). Neste MERPID, o padrão dos dados é descrito por uma função normal padrão, de modo que cada uma destas janelas de dados possui um valor para a variável média e um valor para a variável desvio padrão. Assim, se o MERPID da janela de dados local apresentar uma divergência para o MERPID da janela de dados global, de modo que esta divergência ultrapasse uma tolerância de erro, o EDIST1 entende que o padrão dos dados mudou, i.e. que ocorreu um *Concept Drift* de mudança completa.

No EDIST1 também é possível evitar que *Concept Drifts* de mudança temporária façam com que o algoritmo abandone um modelo de aprendizado já consolidado para dar lugar ao modelo de aprendizado que representa esta instabilidade temporária no padrão dos dados. Isso ocorre porque o EDIST1 consegue distinguir a ocorrência de uma mudança rápida no padrão dos dados de uma mudança completa no padrão dos dados. A mudança completa é detectada conforme foi explicado no parágrafo anterior.

A mudança temporária é detectada da seguinte maneira: se as características da janela de dados local apresentarem divergência das características da janela de dados global, no entanto, dentro de uma margem de tolerância, significa que pode estar ocorrendo um *Concept Drift* rápido, como é o caso do *Concept Drift* do tipo ligeiro, do tipo recorrente ou até mesmo que possa estar acontecendo um ruído. Com isso, o EDIST1 evita que ocorra uma queda de acurácia do algoritmo, já que não adotará um modelo de aprendizado contaminado por *Concept Drift*.

O controle interno do EDIST1 para identificar se o padrão de dados não mudou, ou para verificar se o padrão de dados está mudando, ou, ainda, para verificar se o padrão dos dados mudou completamente, é feito através de uma máquina de estados conforme está representado na Figura 5. Nesta máquina de estados, o EDIST1 se encontra no estado chamado *InControl*, quando as características da janela de dados local (WL) apresentam similaridade com as características da janela de dados global (WG). Já quando as características da janela de dados local apresentam uma diferença de similaridade das características da janela de dados global, porém, dentro de uma tolerância o EDIST1 se encontra no estado *Warning*. Por fim, se as características da janela de dados local não tiverem similaridade com as características da janela de dados global o EDIST1 se encontra no estado *Drift*.

Figura 5 - Máquina de Estados do EDIST1 e do EDIST2



Uma característica importante do EDIST1 é que este algoritmo não necessita de uma grande quantidade de memória RAM para ser executado. Isso ocorre por dois motivos: 1) porque a janela de dados local e a janela de dados global não armazenam as instâncias de dados de um fluxo de dados, mas representam as características destas instâncias de dados através da

variável média e da variável de desvio padrão; e 2) porque a memória RAM é utilizada para armazenar apenas o modelo de aprendizado, que é uma árvore de decisão e, também, para armazenar variáveis estatísticas das janelas de dados.

2.4.2. EDIST2

Na técnica EDIST2 (KHAMASSI, SAYED-MOUCHAWEH *et al.*, 2015) o funcionamento do algoritmo é similar ao do seu antecessor, o EDIST1. Entretanto, existem duas diferenças importantes entre estes algoritmos, que são: as regras de decisão para identificação do estado *Warning* e do estado *Drift*, e o fato do EDIST2 não descartar os dados da janela de dados intermediária após o algoritmo sair do estado de *Warning* e retornar para o estado *InControl*, como ocorria no EDIST1.

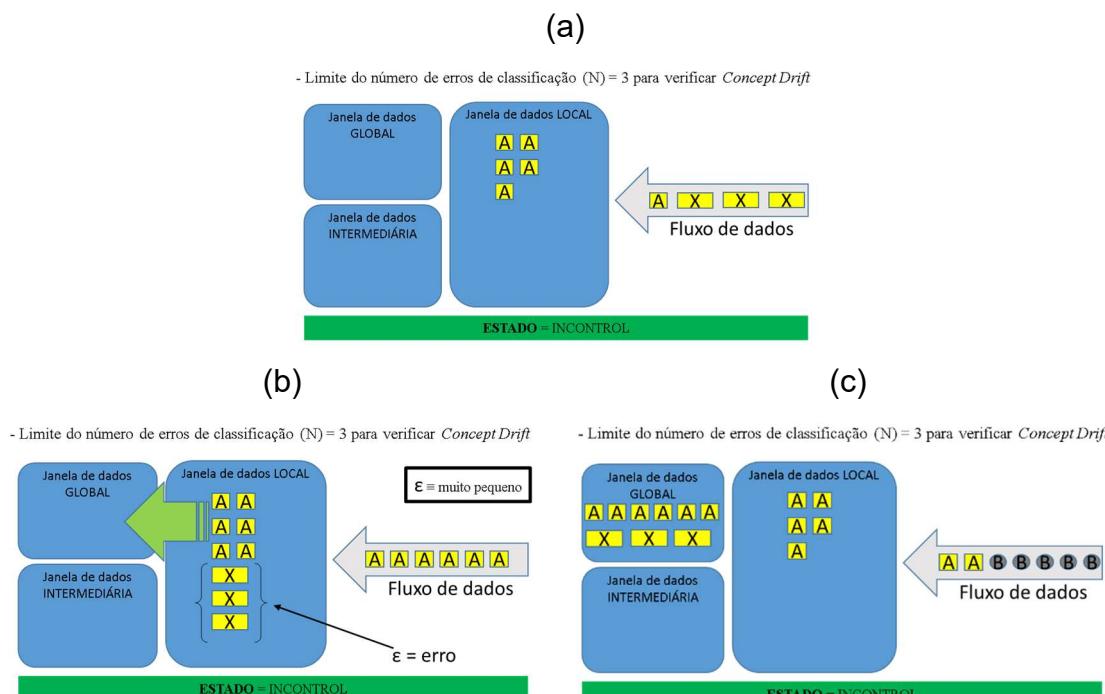
Para explicar como é o funcionamento do EDIST2 foi criado um fluxograma, que está representado no Apêndice B, e também um pseudocódigo, que pode ser visualizado no Apêndice C – Pseudocódigo do algoritmo EDIST2. Além disso, com o objetivo de ilustrar a maneira que o EDIST2 utiliza as janelas de dados para detectar o *Concept Drift* foram criadas a Figura 6, a Figura 7 e a Figura 8 que serão referenciadas ao longo dos próximos parágrafos. Nessas figuras o padrão atual dos dados do fluxo de dados foi representado pela letra A. Já o *Concept Drift* de mudança temporária foi representado pela letra X. Por fim, a letra B representa o padrão novo dos dados do fluxo de dados, *i.e.* a letra B representa um *Concept Drift* de mudança completa.

Quando o EDIST2 realiza a tarefa de classificação de instâncias de fluxos de dados, todas as instâncias são representadas inicialmente na janela de dados local. Enquanto o EDIST2 estiver classificando corretamente as instâncias de um fluxo de dados ele permanecerá no estado *InControl*, conforme está ilustrado pela Figura 6 (a). Caso ocorra uma quantidade de erros de classificação igual ao limite de erros que foi configurado no algoritmo (N), e também se estes erros (ε) forem considerados pequenos pelo critério de decisão do EDIST2, o modelo estatístico que representa o padrão das instâncias de dados (MERPID) da janela de dados local é movido para a janela de dados global, conforme está representado na Figura 6 (b) e na Figura 6 (c).

Assim, o algoritmo continua a analisar novas instâncias do fluxo de dados, de acordo com a Figura 7 (a), e a representar, através do MERPID, essas instâncias na janela de dados local.

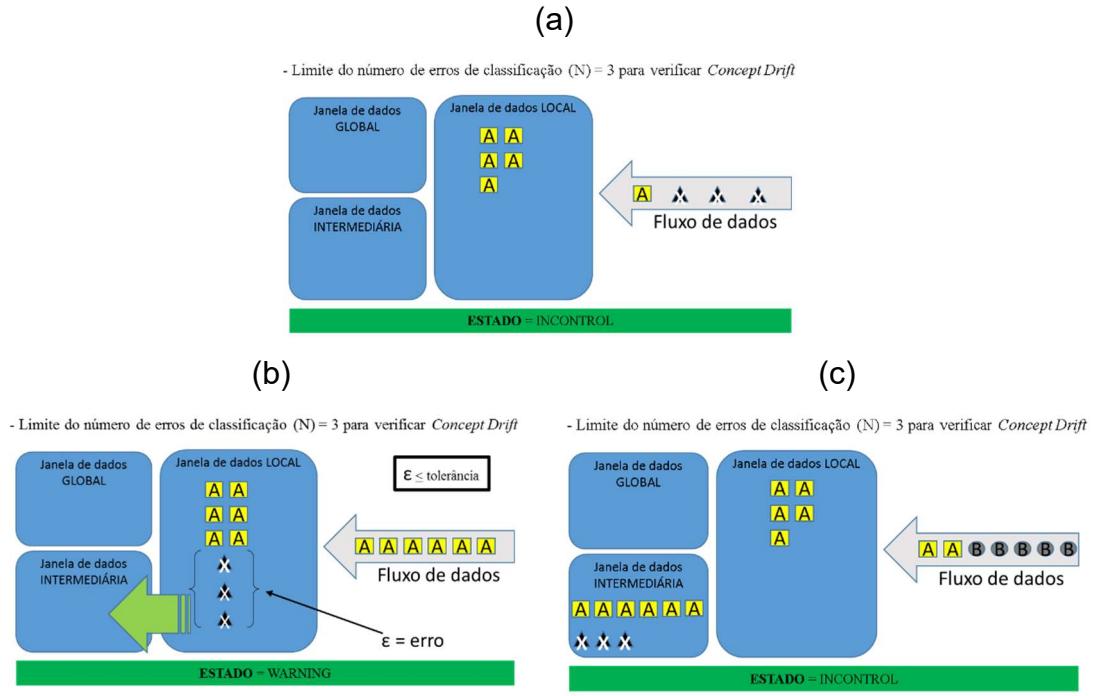
Se ocorrer novamente uma quantidade de erros de classificação igual ao limite de erros que foi configurado no algoritmo, e também se estes erros não forem considerados pequenos pelo critério de decisão do EDIST2, mas estiverem dentro de uma margem de tolerância deste algoritmo, o EDIST2 entrará no estado *Warning* e moverá o MERPID da janela de dados local para a janela de dados intermediária, conforme está ilustrado na Figura 7 (b) e na Figura 7 (c). Depois disso, o algoritmo continuará a classificar as novas instâncias de um fluxo de dados, de acordo com a Figura 8 (a), e representará estas instâncias na janela de dados local.

Figura 6 – Tratamento atual das janelas de dados do EDIST2 no estado *InControl*



Fonte: (Do autor)

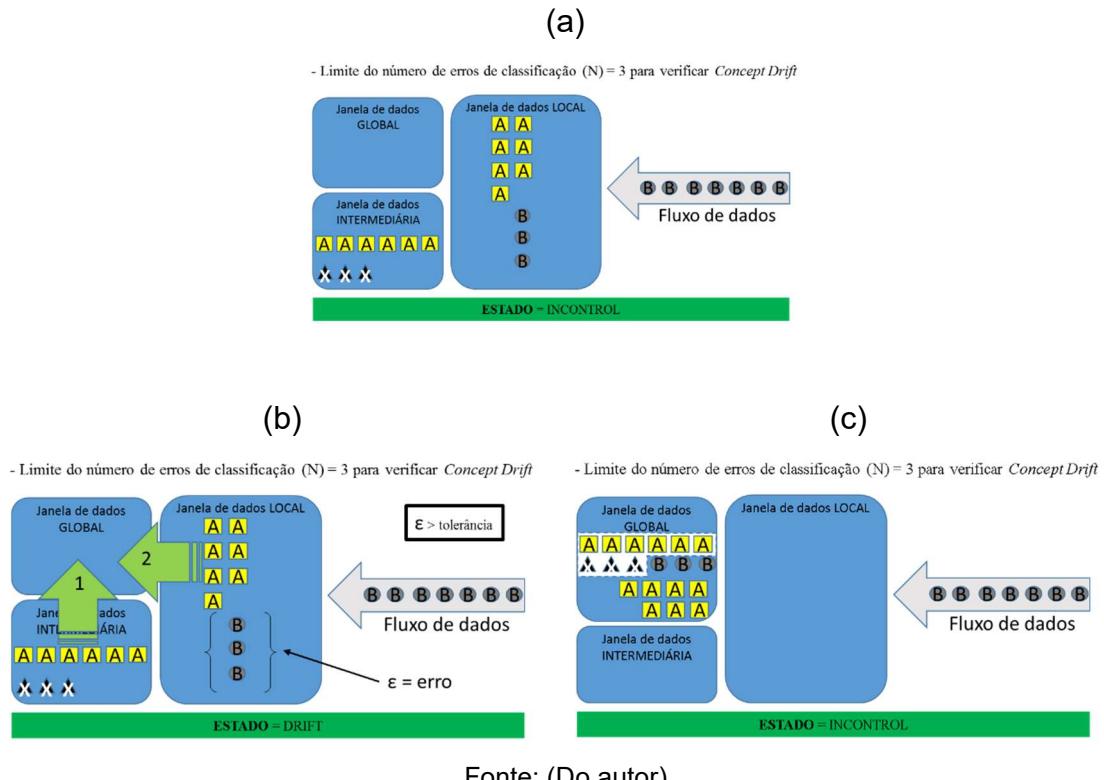
Figura 7 – Tratamento atual das janelas de dados do EDIST2 no estado *Warning*



Fonte: (Do autor)

Por fim, se ocorrer uma quantidade de erros de classificação igual ao limite de erros que foi configurado no algoritmo, e também se esses erros de classificação estiverem acima da tolerância do EDIST2, o algoritmo entra no estado *Drift*. Com isso, ele remove o MERPID da janela de dados global, se existir, move o MERPID da janela de dados intermediária para a janela de dados global e também move o MERPID da janela de dados local para a janela de dados global. Estas ações são ilustradas pela Figura 8 (b). Assim, o novo padrão dos dados do fluxo de dados é representado pelo MERPID que estava na janela de dados intermediária e também pelos MERPID que estava na janela de dados local, conforme está representado na Figura 8 (c). Em seguida, o EDIST2 volta a realizar a classificação de novas instâncias do fluxo de dados.

Figura 8 - Tratamento atual das janelas de dados do EDIST2 no estado *DRIFT*



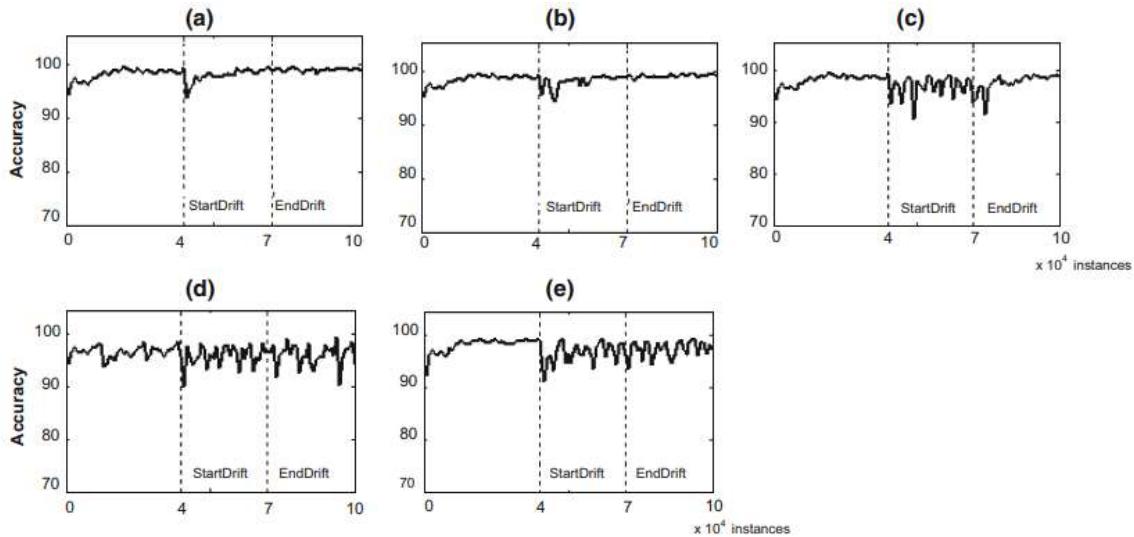
Fonte: (Do autor)

De acordo com Khamassi, Sayed-Mouchaweh *et al.* (2015) estas alterações fazem com que o EDIST2 consiga construir um modelo de aprendizado melhor e, com isso, o algoritmo reagirá mais rapidamente à ocorrência de um *Concept Drift*. Para comprovar esta afirmação, os autores criaram os gráficos que são apresentados na Figura 9. Nesses gráficos estão representados os históricos de acurácia que foram obtidos, respectivamente, pelo EDIST2, na Figura 9 (a), pelo EDIST1, na Figura 9 (b), pelo DDM, na Figura 9 (c), pelo ADWIN, na Figura 9 (d) e pelo EDDM, na Figura 9 (e), após realizar a classificação de instâncias de dados do fluxo de dados *Rotating Hyperplane* (HULTEN, SPENCER e DOMINGOS, 2001). No experimento que deu origem ao gráfico da Figura 9 todos os algoritmos utilizaram como classificador o algoritmo *Hoeffding Trees*.

Veja que o EDIST2, na Figura 9 (a), apresentou uma variação de acurácia pequena durante o início do *Concept Drift*. Já o EDIST1, que também apresentou uma variação de acurácia pequena, teve a duração desta variação de acurácia por um número de instâncias maior do que o que foi observado no EDIST2. Por fim, os outros três algoritmos, DDM, ADWIN e EDDM,

apresentaram uma variação de acurácia grande, com duração de mais instâncias do que o EDIST1, e também de mais instâncias do que o EDIST2.

Figura 9 – Histórico de acurácia dos algoritmos (a) EDIST2, (b) EDIST1, (c) DDM, (d) ADWIN e (e) EDDM para classificar instâncias do fluxo de dados *Rotating Hyperplane* que contém *Concept Drift* do tipo gradual.



Fonte: (KHAMASSI, SAYED-MOUCHAWEH et al., 2015)

Nota-se que o comportamento da acurácia do EDIST2 durante o período de *Concept Drift*, e também após o período de *Concept Drift*, indica que este algoritmo possui a capacidade de se adaptar a um padrão de dados novo de forma mais rápida do que os outros algoritmos que foram comparados com ele. Essa é uma característica importante para algoritmos que são utilizados com fluxos de dados *online*, já que nesse ambiente a dinamicidade das fontes geradoras de dados pode fazer com que ocorram sucessivos *Concept Drifts* ao longo do tempo.

Outra característica que torna o EDIST2 um algoritmo atual está relacionada com a flexibilidade que esse algoritmo oferece para que ele seja utilizado em outras tarefas de mineração de dados. Isso significa que é possível substituir o algoritmo que constrói o modelo de aprendizado por outro algoritmo que constrói outro tipo de modelo de aprendizado. Por exemplo, podemos substituir um modelo de aprendizado baseado em classificação por um modelo de aprendizado baseado em regressão.

Além de permitir a alteração do algoritmo construtor do modelo de aprendizado, o EDIST2 possibilita que seja alterada a função estatística que é

a utilizada para verificar se o padrão dos dados mudou no fluxo de dados. Isso significa que podemos substituir, por exemplo, a função normal padrão, que é a função originalmente utilizada pelo EDIST2, por uma função linear ou por uma função logarítmica, dentre outras.

Essas flexibilidades de modificação na estrutura do EDIST2 permitem que sejam realizadas alterações rápidas no núcleo desse algoritmo. Por isso, esse trabalho apresentará três propostas de alteração do EDIST2 com o objetivo de melhorar a detecção de *Concept Drift* em fluxos de dados *online* em termos de aumento da acurácia média, de redução do tempo de CPU e de redução do consumo de memória RAM.

2.5. Considerações

A melhoria de algoritmos é um processo contínuo e pode contribuir bastante com a evolução da área em que o algoritmo é empregado. Um exemplo disso é o trabalho que foi apresentado na Seção 2.4.2 em que o algoritmo proposto por este trabalho, o EDIST2, conseguiu obter um desempenho melhor do que o desempenho de algoritmos conhecidos na área de mineração de dados como é o caso do DDM, do EDDM e do ADWIN.

Com o objetivo de desenvolver algoritmos cada vez melhores pesquisadores têm buscado melhorar técnicas de mineração de dados de algoritmos já existentes e, também, têm buscado criar algoritmos novos com base em paradigmas diferentes de paradigmas que são encontrados nos algoritmos já existentes. Por isso, no Capítulo **Erro! Fonte de referência não encontrada.** serão apresentados alguns trabalhos que estão sendo desenvolvidos para melhorar a detecção de *Concept Drift* em fluxos de dados.

3. Revisão da literatura

3.1. Trabalhos relacionados

A identificação de padrões em fluxos de dados atraiu a atenção de muitos pesquisadores de algoritmos de aprendizado de máquinas nos últimos anos. Isso ocorreu devido à grande aplicação que essa tarefa pode ser empregada no dia-a-dia das pessoas. Alguns exemplos, são: o reconhecimento facial, a identificação de padrões de comportamento, preferências de temas e de notícias, interesse em produtos, dentre outros.

Entretanto, o principal desafio nessa tarefa é desenvolver um algoritmo que consiga identificar corretamente o padrão dos dados presentes em um fluxo de dados *online*, com uma acurácia próxima de 100%. Além disso, o algoritmo precisa identificar de maneira quase instantânea qualquer mudança que ocorra no padrão dos dados e aprender o novo padrão de dados rapidamente. Em seguida, o algoritmo deve passar a identificar o padrão correto das instâncias novas que chegarem no fluxo de dados com uma acurácia alta.

Para enfrentar estes desafios os pesquisadores têm buscado construir algoritmos novos com base em algoritmos já existentes, como é o caso de Cabral e Barros (2018) que propuseram três abordagens novas chamadas de *Fisher Proportions Drift Detector* (FPDD), *Fisher Square Drift Detector* (FSDD) e *Fisher Test Drift Detector* (FTDD), para melhorar o algoritmo *Statistical Test of Equal Proportions* (STEPD) através de um *Fisher's Exact test* (FISHER, 1934; YATES, 1934) que se baseia em uma tabela de contingência de erros e em classificações corretas nas duas janelas de dados. Esses três métodos funcionam de maneira muito parecida com o STEPД: eles monitoram os resultados de um classificador básico, usam uma janela de dados para as instâncias de dados recentes e outra janela de dados para as instâncias de dados antigas, contam com testes estatísticos para sinalizar a possibilidade do padrão dos dados mudar ou uma mudança no padrão dos dados. As únicas diferenças estão relacionadas aos arranjos de testes estatísticos que são adotados. Além de usar testes diferentes dos testes que são utilizados no algoritmo STEPД, os novos métodos medem as diferenças nos erros, enquanto o STEPД usa o número de classificações corretas das janelas de dados.

Nos experimentos que foram realizados, o FPDD, o FSDD e o FTDD tiveram um desempenho inferior ao dos algoritmos DDM, ECDD, SEED, FHDDM e STEPD em apenas um cenário. Na maioria dos cenários o desempenho foi parecido com o desempenho desses algoritmos e, em alguns cenários, as três propostas de melhoria tiveram um desempenho superior ao desempenho do DDM, do ECDD, do SEED, do FHDDM e do STEPD. Com isso, o FPDD, o FSDD e o FTDD tiveram um desempenho geral melhor do que os algoritmos que foram utilizados na comparação de desempenho.

Outro trabalho que utilizou como base um algoritmo de detecção de *Concept Drift* já existente foi proposto por Costa, Duarte *et al.* (2017). Nesse trabalho os autores criaram o algoritmo *Multidimensional Fourier Transform* (MDFT) para estender a abordagem do algoritmo *Unidimensional Fourier Transform* (1DFT) para fluxos de dados multidimensionais, que são fluxos de dados com características *Online Analytical Processing* (OLAP) em que as instâncias de dados podem ser analisadas sob perspectivas, ou dimensões diferentes (e.g. uma instância de dados pode ser analisada, de uma só vez, pela perspectiva temporal, pela perspectiva geográfica, pela perspectiva de custo e pela perspectiva comportamental, entre outras).

Além de criar um algoritmo para fluxos de dados multidimensionais Costa, Duarte *et al.* (2017) propuseram uma forma mais robusta para analisar *Concept Drifts* em fluxos de dados com base na entropia de Shannon (SHANNON, 2001) e na teoria dos jogos de Von Neumann (NEUMANN e MORGENSTERN, 1944) com o objetivo de quantificar as variações de padrão dos dados.

Para verificar a viabilidade do algoritmo proposto pelos autores foram realizados testes com sete configurações diferentes do algoritmo, foram utilizados seis fluxos de dados diferentes e foram utilizados os algoritmos classificadores *Hoeffding Trees* e *Naïve Bayes*. As medidas que foram utilizadas para avaliar o algoritmo foram a taxa de detecção perdida (MDR), o tempo médio de detecção (MTD), o tempo médio entre os falsos alarmes (MTFA) e o tempo de execução. Quando o parâmetro MDR foi testado o algoritmo MDFT obteve o melhor desempenho porque apresentou o maior valor de precisão na detecção de *Concept Drift* em relação aos outros algoritmos testados. Já para a medida MTFA o algoritmo MDFT foram percebidos alguns

falsos alarmes na detecção de *Concept Drift*. Além disso, o MDFT foi o algoritmo que consumiu o maior tempo de CPU entre os algoritmos que foram testados.

Um dos trabalhos mais recentes que propõe um algoritmo novo para a detecção de *Concept Drift* é o *Active Fuzzy Weighting Ensemble for Dealing with Concept Drift* (DONG, LU et al., 2018). Neste trabalho os autores propõem um algoritmo chamado de *Active Fuzzy Weighting Ensemble* (AFWE). O AFWE é um algoritmo de agrupamento adaptativo composto pelo método detector de *Concept Drift Early Drit Detection Method* (EDDM), por um modelo *Fuzzy*, e por uma estratégia de votação dinâmica.

O método EDDM foi utilizado com o objetivo de reduzir o custo computacional do AFWE, já que o EDDM tem a capacidade de identificar quando ocorre um *Concept Drift* e pode prover um classificador-base novo sob demanda. Já o modelo *Fuzzy* é utilizado em conjunto com uma estratégia de votação dinâmica para organizar todos os classificadores-base existentes com o objetivo de construir um modelo de aprendizado conjunto. Então, o algoritmo pode encurtar o tempo de recuperação de uma queda de precisão após um *Concept Drift*, pode se adaptar a diferentes tipos de *Concept Drift* e, também, obter um desempenho melhor em termos de custo computacional.

O algoritmo AFWE de Dong, Lu et al. (2018) tem um princípio de funcionamento parecido com o princípio de funcionamento do algoritmo-base deste trabalho, o EDIST2. Isso porque o AFWE também utiliza uma máquina de estados composta pelo estado *Normal* (que no EDIST2 é o estado *InControl*), pelo estado *Warning* e pelo estado *Drift*.

Quando o algoritmo AFWE está no estado *Normal* ele limpa as instâncias temporárias que são armazenadas quando o AFWE entra no estado *Warning*, caso existam, e treina o classificador-base. Já quando está no estado *Warning* o AFWE memoriza as instâncias que parecem ter um padrão diferente do padrão atual do fluxo de dados, e treina o classificador-base. Depois de concluir o treinamento do classificador-base nestes dois estados o AFWE atualiza o algoritmo de agrupamento. Por fim, quando o AFWE entra no estado *Drift* ele utiliza um modelo *Fuzzy* para ponderar as instâncias de dados que foram armazenadas temporariamente quando o AFWE entrou no estado *Warning*, sem seguida, cria e treina um classificador-base novo utilizando as

instâncias temporárias que foram ponderadas pelo modelo *Fuzzy*, por fim, atualiza o modelo de aprendizado do algoritmo de agrupamento.

Na verificação experimental Dong, Lu *et al.* (2018) utilizaram um total de sete conjuntos de dados, de modo que dois conjuntos de dados sintéticos, o *Rotating Hyperplane* e o SEA também são utilizados neste trabalho. Além disso, os autores compararam experimentalmente o desempenho do algoritmo que é proposto por eles, o AFWE, com o desempenho dos algoritmos *Drift Detection Method* (DDM) e EDDM, que também foram utilizados neste trabalho, e com os algoritmos *Dynamic Weighted Majority* (KOLTER e MALOOF, 2007) e *Learn++.NSE* (ELWELL e POLIKAR, 2011).

Os resultados que foram obtidos pelo AFWE parecem promissores, entretanto, os fluxos de dados que foram testados são pequenos (o maior fluxo de dados tem 45312 instâncias) e podem criar uma visão bastante limitada do desempenho deste algoritmo numa perspectiva temporal maior. Portanto, seria importante que os autores tivessem verificado o desempenho do AFWE em um universo de dados mais amplo, já que isso poderia ajudar a entender se ocorre uma tendência de comportamento nas curvas de desempenho que foram utilizadas como critério de avaliação do algoritmo. Esta verificação em um cenário de dados mais amplo foi realizada neste trabalho e através dela foi possível verificar a tendência de comportamento do algoritmo ao longo do tempo.

3.2. Considerações

Veja que na Seção 3.1 foram citados alguns trabalhos que buscam melhorar a tarefa de detecção de *Concept Drift*. Perceba que o desenvolvimento de algoritmos melhores pode ser realizado de maneiras diferentes, isto é, ele pode ocorrer através da melhoria de técnicas de detecção de *Concept Drift* já existentes, e também pode ocorrer através da alteração de paradigma de algoritmos já existentes. Com isso, perceba que há espaço para que trabalhos novos sejam desenvolvidos sob óticas de metodologias diferentes, mas com o objetivo comum de produzir algoritmos cada vez mais eficientes para a detecção de *Concept Drift*.

Este trabalho propõe duas maneiras diferentes de melhorar a detecção de *Concept Drift*. Uma delas, Seção 4.1.1, sugere uma melhoria na técnica

utilizada pelo algoritmo EDIST2. As outras duas propostas sugeridas por este trabalho, Seção 4.1.2 e Seção 4.1.3, propõe alterações na maneira que o algoritmo EDIST2 identifica um *Concept Drift* através da mudança no paradigma que é utilizado para identificar uma mudança de padrão nos dados. Por isso, o Capítulo 4 irá apresentar como cada uma destas propostas de melhoria, e também como foi realizada a análise de viabilidade destas propostas de melhoria.

4. Metodologia

Neste capítulo são apresentados: as ferramentas e os recursos que foram utilizados, a metodologia quantitativa de investigação que foi aplicada e os critérios de avaliação que foram utilizados para verificar se as propostas de melhoria promovem, efetivamente, uma melhoria de desempenho do algoritmo EDIST2. Assim, este capítulo está dividido da seguinte maneira: a Seção 4.1 apresentará as propostas de melhoria que são sugeridas ao algoritmo EDIST2 por este trabalho; e a Seção 4.2 elencará os recursos físicos e os recursos lógicos que foram utilizados para realizar os experimentos.

4.1. Propostas para melhoria na detecção de *Concept Drift* em fluxos de dados *online*

Uma das propostas de melhoria no algoritmo detector de *Concept Drift* EDIST2 é apenas uma sugestão de alteração no tratamento da janela de dados intermediária que é utilizada pelo EDIST2. As outras duas propostas introduzem novos conceitos à lógica original do algoritmo e ainda adicionam parâmetros de entrada novos a este algoritmo.

Assim, a Seção 4.1.1 apresentará as alterações que foram realizadas no EDIST2 para melhorar a janela intermediária de dados, a Seção 4.1.2 trará os principais conceitos da proposta de melhoria *Fading* e também como esta proposta de melhoria será utilizada juntamente com o EDIST2. A Seção 4.1.3 apresentará os principais conceitos envolvidos na proposta de melhoria *Reduced Boundary* e também como ela será utilizada junto ao EDIST2.

4.1.1. Melhoria no gerenciamento de janela de dados intermediária

Conforme abordado no Capítulo 2, Seção 2.4.2, o algoritmo EDIST2 utiliza três janelas de dados para realizar a detecção de *Concept Drift* em fluxos de dados *online*: a janela global, a janela intermediária e a janela local. Através da comparação do MERPID de cada janela de dados, o EDIST2 consegue identificar se está ocorrendo um *Concept Drift*.

Caso ocorra um *Concept Drift* de mudança completa, o EDIST2 descarta o MERPID obsoleto, que está na janela de dados global, e adota o MERPID da janela de dados intermediária e o MERPID da janela de dados local. Por outro lado, se ocorrer um *Concept Drift* de mudança temporária o EDIST2 não

descarta o MERPID que contém este *Concept Drift*, que é o MERPID da janela de dados intermediária. Com isso, o modelo de aprendizado do EDIST2, que é representado pelo MERPID da janela de dados global, poderá conter as características do padrão deste *Concept Drift*.

Pelas investigações que foram realizadas foi possível perceber que isso pode funcionar bem quando ocorre *Concept Drift* de mudança completa, que é o caso do *Concept Drift* do tipo gradual, do tipo incremental e do tipo repentino. Entretanto, quando o *Concept Drift* é de mudança temporária, que é o caso do *Concept Drift* do tipo recorrente, do tipo ligeiro e do tipo ruído, a acurácia do algoritmo pode ser prejudicada. Isso porque as instâncias pertencentes a estes tipos de *Concept Drift* poderão ir, em algum momento, para a janela de dados intermediária. Com isso, caso o EDIST2 entre no estado *Drift*, as instâncias da janela de dados intermediária serão movidas para a janela de dados global e, então, o *Concept Drift* fará parte do modelo de aprendizado do EDIST2, que é representado pela janela de dados global na Figura 8 (c).

Durante a etapa de investigação do funcionamento do EDIST2 foi possível perceber que este problema ocorre quando o EDIST2 sai do estado *InControl*, vai para o estado *Warning*, e depois retorna do estado *Warning* para o estado *InControl*. Quando o EDIST2 entra no estado *Warning* ele move o MERPID da janela de dados local para a janela de dados intermediária. Com isso, está sendo movido o MERPID que possui o mesmo padrão de instâncias anteriores do fluxo de dados, e também o MERPID do *Concept Drift* que gerou o estado *Warning*. Para entender melhor veja que na Figura 7 e na Figura 8 que o padrão do *Concept Drift* é o ‘X’ e que o padrão do fluxo de dados é o ‘A’.

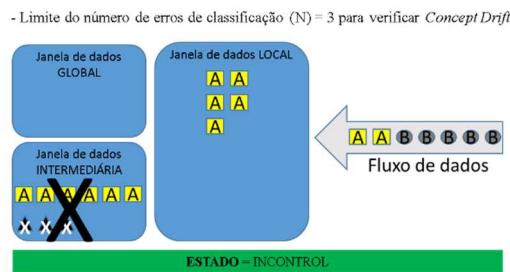
Depois de entrar no estado *Warning* o algoritmo continua a classificar as novas instâncias que chegam do fluxo de dados. Assim, se o algoritmo retorna para o estado *InControl*, o MERPID da janela de dados local é movido para a janela de dados global. Perceba que a janela de dados intermediária ainda é representada por instâncias do padrão ‘A’ e por instâncias do padrão ‘X’. Dessa forma, imagine que comece a chegar instâncias de um padrão ‘B’ e, com isso, o EDIST2 entre no estado *Drift*. A partir deste momento o MERPID da janela de dados global é descartado porque é considerado um padrão de dados obsoleto e, em seguida, o MERPID da janela intermediária que contém instâncias do padrão ‘A’ e do padrão ‘X’ é movido para a janela de dados global.

Consequentemente, o MERPID do novo padrão, 'B', que está na janela de dados local também é movido para a janela de dados global. Com isso, veja que a janela de dados global é composta agora pelo MERPID do padrão 'A', pelo MERPID do padrão 'B', e pelo MERPID do padrão 'X', que foi um *Concept Drift* de mudança temporária (e.g., um *Concept Drift* do tipo recorrente).

Na verdade, a janela de dados global deveria possuir um MERPID composto por poucas instâncias do padrão 'A', que são as instâncias de transição, e por instâncias do padrão 'B', que é o novo padrão do fluxo de dados. Isto teria ocorrido se na transição do estado *Warning* para o estado *InControl* o EDIST2 tivesse descartado o MERPID da janela de dados intermediária, conforme é ilustrado pela Figura 10 (a), pela Figura 10 (b), pela Figura 10 (c) e pela Figura 10 (d). Portanto, este trabalho propõe remover o MERPID da janela de dados intermediária sempre que o EDIST2 entrar no estado *InControl*.

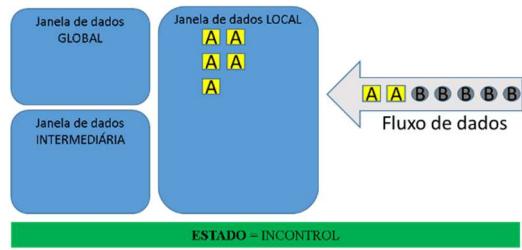
Figura 10 – Proposta de melhoria no tratamento das janelas de dados do EDIST2

(a) Retorno do EDIST2 ao estado *InControl* com a limpeza da janela de dados intermediária.



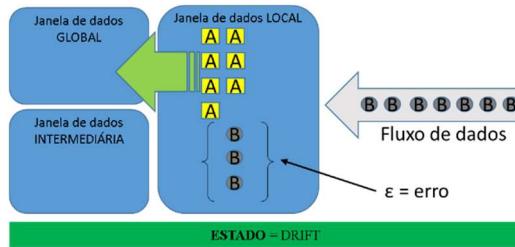
(b) O EDIST2 classifica instâncias novas do fluxo de dados.

- Limite do número de erros de classificação (N) = 3 para verificar *Concept Drift*



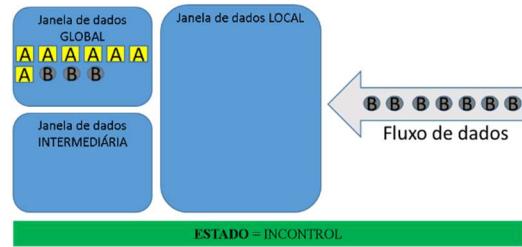
(c) O MERPID da janela de dados local passa para a janela global com a alteração do padrão de dados.

- Limite do número de erros de classificação (N) = 3 para verificar *Concept Drift*



(d) A janela de dados global representa o padrão de dados novo conhecido pelo EDIST2.

- Limite do número de erros de classificação (N) = 3 para verificar *Concept Drift*



Fonte: (Do autor)

4.1.2. *Fading*

Na Seção 4.1.1, apresentou-se uma proposta de melhoria no tratamento da janela de dados intermediária do EDIST2. Naquela proposta de melhoria, a lógica original do EDIST2 foi preservada e a mudança ocorreu apenas no modo de tratar o MERPID, que é representado pelas janelas de dados. Na proposta de melhoria *Fading* foi sugerida uma alteração maior para o EDIST2. Nessa alteração foi adicionado um critério de avaliação novo para verificar se a distância entre erros de classificação das instâncias mais recentes do fluxo de dados apresenta uma tendência de queda.

Esse novo critério de avaliação foi adicionado com o objetivo de tentar antecipar a detecção de *Concept Drift* em relação a lógica original do EDIST2, já que a lógica original do EDIST2 só verifica se um *Concept Drift* está ocorrendo após N erros de classificação. Apesar disso, a lógica original do EDIST2 foi preservada porque apresenta um desempenho satisfatório para a tarefa de detecção de *Concept Drift*.

Como a lógica original do algoritmo EDIST2 verifica se ocorreu um *Concept Drift* somente a cada N erros de classificação, se o padrão dos dados mudar antes que ocorram os N erros de classificação o EDIST2 demorará algumas instâncias para identificar que um *Concept Drift* ocorreu. Entretanto, quando o *Fading* é utilizado, a verificação de *Concept Drift* é realizada a cada erro de classificação. Com isso, não é preciso aguardar que ocorram os N erros de classificação para perceber que um *Concept Drift* ocorreu. Além disso, se o *Fading* demorar para identificar que um *Concept Drift* ocorreu, o maior atraso que poderá ser observado será exatamente o mesmo atraso do EDIST2, já que a lógica original desse algoritmo foi preservada.

O funcionamento da proposta de melhoria *Fading* ocorre da seguinte maneira: o EDIST2 calcula a distância de um erro de classificação recente ($d_{recente}$), i.e. calcula o número de instâncias do fluxo de dados que existem entre um erro de classificação novo e o erro de classificação anterior a este erro. Em seguida, a distância $d_{recente}$ é utilizada junto com as últimas L-1 distâncias mais recentes (d_i), que são memorizadas pelo algoritmo, para que seja calculada a distância média entre erros de classificação ($D_{\mu err}$), conforme Equação (1).

$$D\mu_{err} = \frac{d_{recente} \cdot P_L + \sum_{i=0}^{L-2} (d_i \cdot P_i)}{L} \therefore P_i = P + \left(\frac{1-P}{L-1}\right) \cdot i \quad (1)$$

Por fim, se a $D\mu_{err}$ for menor que um limiar pré-definido, chamado de distância mínima aceitável (DMA), significa que pode estar ocorrendo um *Concept Drift*. Com isso, o EDIST2 passa do estado *InControl* para o estado *Warning* e a tolerância a erros de classificação é reduzida em 25%. Essa redução de 25% da tolerância é realizada para sinalizar para o algoritmo que o padrão dos dados pode estar mudando no fluxo de dados. Com isso, se muitas instâncias de dados novas não apresentarem características mais próximas das características do padrão de dados atual do fluxo o *Fading* irá identificar como um *Concept Drift* e irá mudar o EDIST2 para o estado *Drift*.

Lembre-se que a detecção de *Concept Drift* no EDIST2 é feita com base na curva normal padrão, de modo que um *Concept Drift* é identificado sempre que $\Delta\mu$ é maior que um valor de erro que é calculado, mais um desvio padrão (DP). Este DP representa a tolerância do EDIST2 para distinguir se os erros representam um *Concept Drift*. Para ilustrar o comportamento deste critério de avaliação no *Fading* veja o Quadro 1, a seguir:

Quadro 1 - Critério de avaliação do *Fading* para detecção de *Concept Drift*.

Algoritmo	<i>Concept Drift</i> ocorre quando	Em qual estado do algoritmo?
EDIST2 ou <i>Fading</i> ($D\mu_{err} \geq DMA$)	$\Delta\mu > erro + DP$	<i>InControl</i> ou <i>Warning</i>
<i>Fading</i> ($D\mu_{err} < DMA$)	$\Delta\mu > erro + (\frac{3}{4} \cdot DP)$	<i>Warning</i> (forçado pelo <i>Fading</i>)

O nome *Fading* foi escolhido porque o novo critério de avaliação que é inserido no EDIST2 realiza uma espécie de desvanecimento (do inglês, *fading*) de cada d_i através de um peso (P_i). Este desvanecimento é realizado para reduzir artificialmente o valor da distância média entre erros de classificação. O objetivo desta redução artificial é o de fazer com que o algoritmo fique mais sensível à ocorrência de erros de classificação próximos, já que uma distância

pequena entre erros de classificação pode indicar que está ocorrendo uma mudança no padrão dos dados do fluxo.

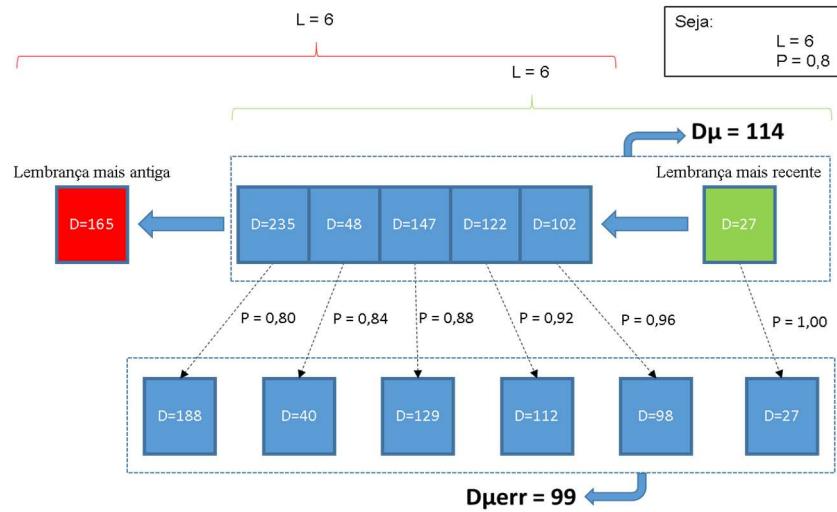
A proposta *Fading* requer a inserção de três parâmetros de entrada novos no EDIST2, que são: a quantidade de distâncias entre erros de classificação que o algoritmo irá memorizar (chamado de Lembrança, L), o peso (P ou P0) que será atribuído à lembrança mais antiga, e a distância mínima aceitável (DMA). O parâmetro L tem um comportamento do tipo *First In First Out* (FIFO), *i.e.* quando ocorre um erro de classificação o *Fading* descarta a lembrança mais antiga para que a lembrança mais nova seja memorizada. Então, para a lembrança mais nova é atribuído um peso igual 1, para a lembrança mais antiga é atribuído um peso P, e para cada lembrança intermediária é atribuído um peso Pi, que é obtido através da Equação (2).

$$P_i = P + \left(\frac{1-P}{L-1} \right) \cdot i \quad (2)$$

Para ilustrar o funcionamento do *Fading* foi criada a Figura 11. Nesta figura veja que após ocorrer um erro de classificação, com distância D igual a 27, o *Fading* descarta a lembrança mais antiga, D igual a 165, e insere a distância mais recente, D igual a 27, no conjunto de lembranças. Com isso, a média aritmética das distâncias entre erros de classificação, D_{μ} , sem a aplicação de peso é de 114. Então, o *Fading* aplica um peso sobre cada lembrança e o valor de D_{μ} cai para 99 e passa a ser chamado de $D_{\mu_{err}}$. Portanto, a distância média entre erros de classificação foi artificialmente reduzida de 114 para 99 pelo *Fading*.

Como o *Fading* insere três parâmetros de entrada novos ao EDIST2 foi realizada uma análise de sensibilidade para esta proposta de melhoria. Essa análise de sensibilidade será apresentada na Seção 5.1 e mostra qual é o comportamento da acurácia, do tempo de processamento e do consumo de memória RAM do EDIST2 quando é realizada a alteração de valor dos parâmetros de entrada novos, L, P e DMA. Além disso, na Seção 5.2 será apresentada uma análise experimental mais ampla que foi realizada para verificar qual é o desempenho do *Fading* em relação ao desempenho do EDIST2 em termos de acurácia, de tempo de CPU e de consumo de memória RAM.

Figura 11 - Exemplo de funcionamento da proposta de melhoria *Fading*.



Fonte: (Do autor)

4.1.3. Reduced boundary

Outra melhoria que este trabalho propõe para o algoritmo classificador EDIST2 é o *Reduced Boundary*. No *Reduced Boundary* a lógica original do EDIST2 foi preservada e ainda foi adicionado um critério de avaliação a esse algoritmo para verificar se ocorre um aumento da frequência de erros de classificação. Esse critério de avaliação introduz dois novos parâmetros de entrada no EDIST2, que são: a distância mínima aceitável entre erros de classificação (D) e a repetitividade (R) da condição D , i.e. a quantidade máxima de vezes que a distância entre erros de classificação, d , pode ser menor que o valor do parâmetro D para que não seja considerado um *Concept Drift*. Além destes dois parâmetros de entrada novos o *Reduced Boundary* utiliza um parâmetro interno, peso, para regular a tolerância de divergência da janela de dados local em relação a janela de dados global para que seja detectado um *Concept Drift*. Este parâmetro interno foi fixado com o valor 10%. Isto significa que as características da janela de dados local precisarão apresentar uma divergência superior a 10% das características da janela de dados global para que o *Reduced Boundary* entenda que está ocorrendo um *Concept Drift*.

O critério de avaliação que é introduzido no EDIST2 pelo *Reduced Boundary* detecta um possível *Concept Drift* quando a frequência de erros de classificação está alta, ou seja, quando os erros de classificação estão muito próximos entre si. Em outras palavras, o *Reduced Boundary* indica ao EDIST2

que pode estar ocorrendo um *Concept Drift* quando a distância entre os erros de classificação (d) é menor ou igual ao valor do parâmetro D e, ainda, que esta condição se repetiu R vezes. A partir daí o *Reduced Boundary* induz o EDIST2 a uma verificação de *Concept Drift*, de forma antecipada, para averiguar se está acontecendo uma mudança no padrão dos dados. Vale destacar que a verificação será feita de forma antecipada porque o EDIST2 não precisará aguardar que ocorram os N erros de classificação para verificar se está ocorrendo um *Concept Drift*.

Depois que o EDIST2 é induzido pelo *Reduced Boundary* a verificar se está ocorrendo um *Concept Drift* ele compara o valor ‘média’ da janela de dados local (μ_{WL}) com o valor ‘média’ da janela de dados global (μ_{WG}). Caso μ_{WL} seja 10% maior do que μ_{WG} significa que pode estar ocorrendo um *Concept Drift*. Caso contrário pode ter ocorrido um ruído, por exemplo.

O Quadro 2 ilustra o teste que o *Reduced Boundary* faz para verificar se há um *Concept Drift* no fluxo de dados. Também foi criada a Figura 12 para ilustrar o funcionamento do *Reduced Boundary* na detecção de *Concept Drift*.

Quadro 2 - Condição para que o *Reduced Boundary* identifique um *Concept Drift*

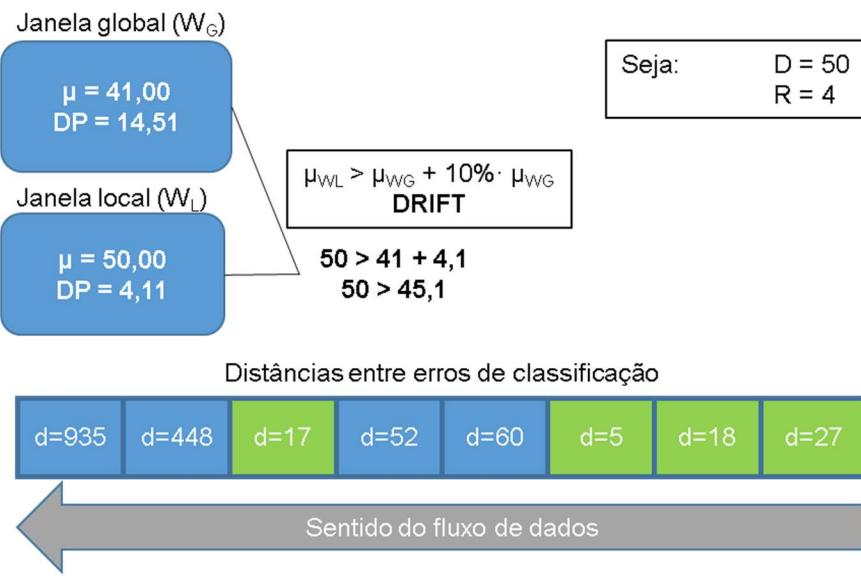
Condição	EDIST2 vai para o estado
Se $\mu_{WL} > \mu_{WG} + \mu_{WG} \cdot 10\%$	Drift
Senão	InControl

Na Figura 12 veja que o parâmetro D foi definido com o valor 50 e o parâmetro R foi definido com o valor 4. Isso significa que quando d for menor ou igual a 50, por 4 vezes, o *Reduced Boundary* irá induzir o EDIST2 a realizar uma verificação de *Concept Drift*. Assim, de acordo com o exemplo da Figura 12, o valor de d foi menor do que o valor de D em quatro oportunidades. Isso ocorreu quando o valor de d foi igual a 17, igual a 5, igual a 18 e igual a 27. Com isso, o EDIST2 verifica se o valor de μ_{WL} é 10% maior do que o valor de μ_{WG} . Percebe-se que neste exemplo o valor de μ_{WL} vale 50 e o valor de μ_{WG} vale 41. Portanto, como o valor de μ_{WL} é 22% maior do que o valor de μ_{WG} o EDIST2 irá entender que está ocorrendo um *Concept Drift*.

Como esta proposta de melhoria também insere dois parâmetros de entrada novos ao EDIST2, foi realizada uma análise de sensibilidade para verificar qual é a influência dos parâmetros novos, D e R , sobre o desempenho

do EDIST2 em termos de acurácia, tempo de processamento e consumo de memória RAM. Além disso, na Seção 5.2 será apresentada uma análise experimental mais ampla que foi realizada para verificar qual é o desempenho do *Reduced Boundary* em relação ao desempenho do EDIST2 em termos de acurácia, de tempo de CPU e de consumo de memória RAM.

Figura 12 - Funcionamento do *Reduced Boundary* para identificar um *Concept Drift*



Fonte: (Do autor)

4.2. Ambiente de investigação

Essa seção lista os recursos que foram utilizados para realizar os experimentos de análise de sensibilidade, do Capítulo 5, e também os experimentos em cenário expandido, do Capítulo 6. Na Seção 4.2.1, são indicados os softwares necessários para replicar os experimentos. Na Seção 4.2.2, descreve-se o hardware em que os experimentos foram testados. E, por fim, na Seção 4.2.3, são indicadas as bases de dados que foram utilizadas como fluxos de dados e, também, as características de cada uma destas bases de dados.

4.2.1. Linguagem de programação

Para replicar o algoritmo detector de *Concept Drift* que é objeto de estudo deste trabalho foi utilizada a linguagem de programação Java. Nessa linguagem de programação foi implementada a replicação do algoritmo EDIST2, foram implementadas as propostas de melhoria *Fading* e *Reduced*

Boundary e também foi implementada a melhoria no gerenciamento da janela intermediária de dados do EDIST2. Essa linguagem de programação foi escolhida porque a biblioteca *Massive Online Analysis*, conhecida como MOA (MOA Project Website, 2016), que foi utilizada como biblioteca-base nesse trabalho foi desenvolvida em Java.

4.2.2. Hardware e softwares utilizados

Para desenvolver os algoritmos que são utilizados neste trabalho, bem como para executar todos os experimentos foi utilizando um computador com as configurações que estão expressas no Quadro 3, a seguir:

Quadro 3 - Configurações do hardware utilizado para desenvolvimento de código e execução dos experimentos

Sistema Operacional	Microsoft Windows 10 Pro
Memória RAM	6,00 GB de RAM
Processador	Intel Core i5 M480 2,67 GHz

A implementação da replicação do algoritmo detector de *Concept Drift* estudado neste trabalho e também das propostas de melhoria que sugerimos para este algoritmo foram realizadas na *Integrated Development Environment* (IDE) Eclipse na versão Neon, *release* 4.6.0. As bibliotecas externas à linguagem de programação Java e que foram incorporadas a este projeto foram a biblioteca *commons-math* na versão 3.5, da Apache, e a biblioteca MOA na *release* 2014.11, da Universidade de Waikato. Como ferramentas auxiliares foram utilizados o software *Notepad++*, da *Notepad++ Team*, e o software *Excel*, da *Microsoft*.

O *Notepad++* foi utilizado para formatar os resultados que foram coletados de cada experimento. Nele foi alterado o delimitador de colunas gerado pelo EDIST2, vírgula, para outro delimitador de colunas, o *pipe*, já que a vírgula é o separador decimal do padrão brasileiro. Além disso, o *Notepad++* foi utilizado para converter o separador decimal, ponto, que é gerado pelo EDIST2 para o separador decimal vírgula que é do padrão brasileiro. Já o software *Excel* foi utilizado para a criação de gráficos que auxiliaram na análise dos resultados dos experimentos que foram realizados.

4.2.3. Conjuntos de dados utilizados nos experimentos

Os conjuntos de dados que foram utilizados para a realização dos experimentos deste trabalho foram o *Rotating Hyperplane*, o *Streaming Ensemble Algorithm* e o *Stagger*. Estes conjuntos de dados não possuem balanceamento de classes e foram gerados a partir da biblioteca MOA, que foi modificada e fornecida (KHAMASSI, SAYED-MOUCHAWEH *et al.*, 2015), e serão utilizados para simular fluxos de dados que posteriormente serão submetidos ao algoritmo. A biblioteca MOA sem as modificações que foram realizadas está disponível na internet¹.

Para os experimentos que serão apresentados no Capítulo 5 foram utilizados três fluxos de dados que possuem *Concept Drift* do tipo repentino, e também três fluxos de dados que possuem *Concept Drift* do tipo gradual. Estes seis conjuntos de dados foram gerados através do *framework* MOA, e são: o *Abrupt Rotating Hyperplane* (ARH), o *Abrupt Streaming Ensemble Algorithm* (ASEA), o *Abrupt Stagger* (AS), o *Gradual Rotating Hyperplane* (GRH), o *Gradual Streaming Ensemble Algorithm* (GSEA) e o *Gradual Stagger* (GS). Para cada fluxo de dados foram inseridos *Concept Drifts* entre as instâncias 40 mil e 70 mil. A geração do *Concept Drift* em cada fluxo de dados foi realizada pela biblioteca MOA. Para isso, foram especificados apenas a instância de início do *Concept Drift* e a instância de fim do *Concept Drift*. As características de cada um desses fluxos de dados serão descritas nos itens (a), (b) e (c), a seguir:

(a) Rotating Hyperplane

O fluxo de dados *Rotating Hyperplane* é um fluxo de dados composto por 100 mil instâncias, possui dois atributos e duas classes chamadas *class1* e *class2*. Os atributos deste fluxo de dados são valores numéricos que variam entre 0 e 1. As classes de instâncias deste fluxo de dados respeitam a Equação (3), a seguir:

$$\begin{aligned} \sum_{i=1}^2 a_i x_i &\geq \frac{\sum a_i}{2} \equiv \text{class1} \\ \sum_{i=1}^2 a_i x_i &< \frac{\sum a_i}{2} \equiv \text{class2} \end{aligned} \tag{3}$$

¹ <https://moa.cms.waikato.ac.nz/downloads/>

(b) Streaming Ensemble Algorithm

Streaming Ensemble Algorithm (SEA) é um fluxo de dados composto por 100 mil instâncias. Esse fluxo possui três atributos e duas classes chamadas *groupA* e *groupB*. Os atributos desse fluxo de dados são valores numéricos que variam entre 0 e 10. A classe de uma instância deste fluxo de dados segue a seguinte regra: se a soma dos dois primeiros atributos resultar em um valor menor ou igual a 8, então a instância de dados pertence à classe *groupA*, caso contrário a instância de dados pertence à classe *groupB*.

(c) Stagger

Stagger é um fluxo de dados composto por 100 mil instâncias de dados, possui três atributos e uma classe do tipo booleana que pode assumir dois valores, que são *true* ou *false*. Os atributos deste fluxo de dados são características de formas geométricas. Estas características são tamanho, cor e forma. A característica tamanho pode assumir três valores: *small*, *medium* e *large*. Já a característica cor por assumir os valores *red*, *blue* e *green*. Por último, a característica forma pode assumir os valores *circle*, *square* e *triangle*. Entretanto, a característica forma não foi utilizada na construção do fluxo de dados Stagger que foi utilizado neste experimento. Com isso, a classe de uma instância deste fluxo de dados segue a seguinte regra: se a instância de dados for do tamanho *small* e da cor *red*, ou então se for do tamanho *medium* e da cor *blue* ou ainda se for do tamanho *large* e for da cor *green*, então o valor da classe será *false*. Caso contrário o valor da classe será *true*. É importante destacar que apesar do conjunto de dados Stagger possuir uma classe do tipo booleana, as instâncias de dados não são equiprováveis para a classe *true* e para a classe *false*.

4.3. Metodologia de investigação

Com o objetivo de verificar se as propostas que foram apresentadas na Seção 4.1 melhoraram o desempenho do EDIST2, foram realizados experimentos para cada uma destas propostas. Nos experimentos foram coletados os valores das variáveis de acurácia do algoritmo, de tempo de CPU

e de consumo de memória RAM. A acurácia e o consumo de memória RAM foram obtidos através de métodos disponíveis no algoritmo classificador. Já o tempo de CPU foi obtido através da classe *TimingUtils* da biblioteca MOA (MOA Project Website, 2016). Estas variáveis foram utilizadas em análises quantitativas para verificar qual é o cenário de cada experimento que cada proposta de melhoria obteve o melhor desempenho conjunto para estas variáveis.

Para escolher o cenário que obteve o melhor desempenho em cada experimento foram obedecidos os seguintes critérios:

- As variáveis de saída acurácia média, tempo de CPU e consumo de RAM tem a mesma importância numa avaliação, ou seja, a importância de cada variável de saída é de 33%.
- Será considerado como melhor cenário de um experimento o cenário que obtiver o maior ganho de valor para as variáveis de saída e, ao mesmo tempo, tiver a menor perda de valor para as variáveis de saída.

Cada proposta de melhoria foi verificada através de seis experimentos diferentes, sendo um experimento para cada conjunto de dados da Seção 4.2.3 que possui *Concept Drift* do tipo repentino, e um experimento para cada conjunto de dados da Seção 4.2.3 que possui *Concept Drift* do tipo gradual. Com isso, foi possível verificar através de uma análise de sensibilidade se uma mudança de característica de um conjunto de dados influencia na configuração dos parâmetros de entrada do algoritmo, mesmo que o tipo de *Concept Drift* não mude. Também, foi possível verificar se a configuração dos parâmetros de entrada do algoritmo muda quando o tipo de *Concept Drift* de um conjunto de dados é alterado.

É importante destacar que os experimentos de cada proposta de melhoria são compostos por quantidades diferentes de cenários de verificação. O número de cenários de cada experimento depende do número de parâmetros de entrada que cada proposta de melhoria possui, e também do conjunto de valores que serão testados para cada parâmetro de entrada. No Capítulo 5 será possível verificar com mais clareza quais são os cenários que compõe cada experimento para cada proposta de melhoria.

Depois que cada cenário foi testado foram coletados os dados de acurácia, de tempo de CPU e de consumo de memória RAM. Então, estes dados foram formatados através do *software Notepad++* e, depois, foram enviados para o *software Excel*. No *Excel* foi criado um gráfico para cada cenário de modo que cada gráfico é composto pelo valor da acurácia média, do tempo de CPU e do consumo de memória RAM. Então, obedecendo o critério de escolha de melhor cenário foi encontrado o cenário de cada experimento que apresentou o melhor desempenho geral.

Depois que o melhor cenário de cada experimento foi encontrado as propostas que são sugeridas por este trabalho passaram por uma avaliação de efetividade de melhoria. Nesta verificação nova, o melhor cenário de cada experimento é testado novamente com conjuntos de dados quinhentas vezes maiores do que os conjuntos de dados que foram utilizadas na análise de sensibilidade. Na verdade, os conjuntos de dados novos são compostos pela repetição sequenciada de dados dos conjuntos de dados que foram utilizados durante a análise de sensibilidade. Isso significa que as instâncias de 1 a 100.000 são exatamente iguais as instâncias de 100.001 até 200.000 e assim por diante. O objetivo desta verificação é o de validar se as propostas sugeridas por este trabalho irão melhorar de desempenho ao longo do tempo, já que 100.00 instâncias podem ser insuficientes para que se concretize alguma tendência de melhorar ou alguma tendência de piorar o desempenho do EDIST2.

5. Análise experimental de sensibilidade das propostas de melhoria

Na Seção 4.1 foram apresentadas as três propostas de melhoria que este trabalho sugere para o algoritmo detector de *Concept Drift*, EDIST2. Duas dessas propostas, o *Fading* e o *Reduced Boundary*, introduzem parâmetros de entrada novos que podem alterar o comportamento desse algoritmo. Essa alteração de comportamento ocorre porque a mudança de valor dos parâmetros de entrada pode fazer com que o EDIST2 consuma mais memória RAM, ou ocupe a CPU por mais tempo, ou ainda, passe a errar mais vezes a classificação de instâncias do fluxo de dados. Por isso, este capítulo apresentará os experimentos que foram realizados para verificar qual é a sensibilidade que as variáveis de saída dessas propostas de melhoria têm quando os valores dos parâmetros de entrada novos são alterados.

Em todos os experimentos que são apresentados neste capítulo as grandezas acurácia, tempo de CPU e consumo de RAM são chamadas de variáveis de saída do algoritmo. É com base nessas variáveis de saída que será verificado o desempenho do algoritmo em termos de custo computacional, i.e. em tempo de processamento (ou tempo de CPU) e em consumo de memória RAM, e também em termos de precisão através da medida acurácia. O cálculo de desempenho de cada variável de saída do algoritmo é descrito pela Equação (4). De acordo com a Equação (4), se o resultado for um valor negativo significa que a variável de saída da proposta de melhoria apresentou um desempenho inferior ao desempenho da mesma variável de saída do EDIST2. Caso contrário, significa que a variável de saída da proposta de melhoria apresentou um desempenho superior ao da mesma variável de saída do EDIST2.

$$Desempenho = \frac{\text{variável_saída}_{\text{proposta_melhoria}} - \text{variável_saída}_{\text{EDIST2}}}{100} \quad (4)$$

O valor desempenho de uma variável de saída é calculado com o objetivo de ajudar na identificação das seguintes características: verificar se uma variável de saída tem uma alteração de valor proporcional a alteração de valor de um dos parâmetros de entrada novos; verificar se a alteração de valor dos parâmetros de entrada novos faz com que a variável de saída apresente

um padrão de comportamento, e.g. tendência de queda de valor; e verificar se a alteração do tipo de *Concept Drift* de um fluxo de dados faz com que seja necessário ajustar o valor dos parâmetros de entrada novos.

Para verificar se o tipo de *Concept Drift* influencia no ajuste dos parâmetros de entrada novos foram utilizados fluxos de dados que possuem *Concept Drift* do tipo repentino (do inglês, *abrupt*) e fluxos de dados que possuem *Concept Drift* do tipo gradual (do inglês, *gradual*), conforme estão apresentados no Quadro 4. Veja que foram utilizados fluxos de dados que possuem *Concept Drift* do tipo repentino e fluxos de dados que possuem *Concept Drift* do tipo gradual. Para cada tipo de *Concept Drift* foram testados três fluxos de dados. O objetivo de testar três fluxos de dados com o mesmo tipo de *Concept Drift* foi o de verificar se os parâmetros de entrada novos precisariam ser ajustados para que a proposta de melhoria apresentasse um desempenho melhor.

Quadro 4 – Fluxos de dados utilizados na análise de sensibilidade

Tipo de Concept Drift	Nome do fluxo de dados
Repentino	<ul style="list-style-type: none"> - <i>Abrupt Rotating Hyperplane (ARH)</i> – Seção 4.2.3 (a) - <i>Abrupt Streaming Ensemble Algorithm (ASEA)</i> – Seção 4.2.3 (b) - <i>Abrupt Stagger (AS)</i> – Seção 4.2.3 (c)
Gradual	<ul style="list-style-type: none"> - <i>Gradual Rotating Hyperplane (GRH)</i> – Seção 4.2.3 (a) - <i>Gradual Streaming Ensemble Algorithm (GSEA)</i> – Seção 4.2.3 (b) - <i>Gradual Stagger (GS)</i> – Seção 4.2.3 (c)

A alteração do tipo de *Concept Drift* do fluxo de dados foi realizada com dois objetivos principais, que são: o de verificar se os parâmetros de entrada novos precisariam ser ajustados para que a proposta de melhoria apresentasse um desempenho melhor, e o de identificar se as variáveis de saída do algoritmo apresentariam alguma alteração significativa de comportamento motivadas pela mudança do tipo de *Concept Drift* do fluxo de dados. Com isso, foi possível verificar entre os valores que foram testados se existe uma configuração ótima para os parâmetros de entrada novos, i.e. se existe um valor para cada parâmetro de entrada novo que permita ao algoritmo obter o melhor desempenho independentemente da característica do fluxo de dados ou do tipo de *Concept Drift* que o fluxo de dados possui.

Em todos os experimentos deste capítulo os algoritmos EDIST2, *Fading* e *Reduced Boundary* foram configurados conforme apresentado na Tabela 1, a seguir. O algoritmo classificador que foi utilizado e que é responsável por criar o MERPID foi o *Hoeffding Trees*. Já o limite de erros de classificação para verificar se ocorreu um *Concept Drift* (N), que é o critério de verificação de *Concept Drift* do EDIST2 e que é o pior caso do *Fading* e do *Reduced Boundary*, foi definido para 30 erros de classificação. Por fim, cada um dos seis fluxos de dados do Quadro 4 que foram utilizados nos experimentos são compostos por 100 mil instâncias de dados. É importante destacar que as 30 mil instâncias existentes entre a instância 40 mil e a instância 70 mil representam mudanças que ocorrem no padrão dos dados, ou seja, são instâncias de *Concept Drift*. Lembre-se que em três conjuntos de dados do Quadro 4 o *Concept Drift* ocorre de maneira repentina e em outros três conjuntos de dados o *Concept Drift* ocorre de maneira gradual.

Tabela 1 – Configurações do EDIST2 para detecção de *Concept Drift*

Parâmetro	Valor
Algoritmo classificador	<i>Hoeffding Trees</i>
Erros de classificação para verificação de <i>Concept Drift</i> (N)	30
Localização do <i>Concept Drift</i>	[40.000 a 70.000]
Instâncias de dados por fluxo de dados	100.000

Como as propostas de melhoria *Fading* e *Reduced Boundary* introduzem parâmetros de entrada diferentes ao EDIST2, as investigações serão apresentadas em seções separadas. Com isso, o Capítulo 5 foi organizado da seguinte maneira: a Seção 5.1 mostra uma investigação que foi realizada para compreender a sensibilidade das variáveis de saída da proposta de melhoria *Fading* quando os parâmetros de entrada novos são alterados; a Seção 5.2 apresenta uma investigação que foi realizada para compreender a sensibilidade das variáveis de saída da proposta de melhoria *Reduced Boundary* quando os parâmetros de entrada novos são alterados; e a Seção 5.3 discutirá as conclusões dos experimentos que foram realizados na Seção 5.1 e na Seção 5.2.

5.1. Análise de sensibilidade da proposta de melhoria *Fading*

A Seção 4.1.2 mostrou que a proposta de melhoria *Fading* introduz três parâmetros de entrada novos ao algoritmo EDIST2, que são: a quantidade de distâncias entre erros de classificação que o algoritmo lembrará (*L*), o peso que a lembrança mais antiga terá em relação ao peso da distância mais recente (*P*), e a distância mínima aceitável entre erros de classificação (*DMA*). Como a alteração de valor desses parâmetros de entrada poderá modificar o comportamento das variáveis de saída do algoritmo, essa seção mostrará uma análise de sensibilidade que foi realizada para o *Fading* com o objetivo de verificar o que ocorre com as variáveis de saída desse algoritmo quando os valores dos parâmetros de entrada são modificados.

A análise de sensibilidade que foi realizada nesta seção foi dividida em seis experimentos, sendo um experimento para cada fluxo de dados do Quadro 4. Cada experimento é composto por dez cenários, conforme a Tabela 2, e foram criados através da seleção distributiva dos valores de *L* com os valores de *P*. Os valores que foram atribuídos ao parâmetro *L* são 15, 50, 100, 200 e 500. Já para o parâmetro *P* foram atribuídos os valores 0,3 e 0,8.

O parâmetro *DMA* teve o valor fixado em 50 para todos os cenários dos experimentos por dois motivos, que são: cada valor novo que fosse testado para este parâmetro aumentaria dez cenários nos experimentos; além disso, é importante realizar uma investigação nova com o objetivo de verificar se o valor de *DMA* precisa ser alterado de acordo com a característica do fluxo de dados.

Tabela 2 – Cenários de teste utilizados em cada experimento da proposta de melhoria *Fading*.

Cenário	1	2	3	4	5	6	7	8	9	10
L	15	15	50	50	100	100	200	200	500	500
P	0.3	0.8	0.3	0.8	0.3	0.8	0.3	0.8	0.3	0.8
DMA	50	50	50	50	50	50	50	50	50	50

Os valores do parâmetro *L* foram atribuídos com o objetivo de verificar se as variáveis de saída do *Fading* apresentam um crescimento proporcional ao aumento do valor de *L*. Já os valores do parâmetro *P* foram atribuídos com o objetivo de verificar se uma atenuação maior ou uma atenuação menor das lembranças mais antigas do *Fading* acelera o processo de detecção do

Concept Drift, já que o parâmetro P reduz artificialmente o valor de cada distância entre erros de classificação que é lembrada pelo *Fading* e, com isso, faz com que a distância média erros de classificação seja menor.

Como o objetivo deste trabalho é propor melhorias para a detecção de *Concept Drift* em fluxos de dado *online*, não será apresentado aqui um estudo acerca do melhor ajuste de valor para os parâmetros de entrada, L e P, do *Fading*. Por isso, esse tema é um desafio novo que pode ser enfrentado em uma proposta de trabalho futuro. Uma sugestão de estudo é verificar a possibilidade do *Fading* ajustar automaticamente o valor dos parâmetros de entrada, L e P, de acordo com a característica de cada fluxo de dados que está sendo classificado, isto é, sem que seja necessário que o usuário forneça valores para os parâmetros L e P.

Depois que o número de experimentos e a quantidade de cenários de cada experimento foram definidos, foram testados todos os cenários com dois objetivos: o objetivo de verificar qual é o comportamento das variáveis de saída do *Fading* quando os parâmetros de entrada novos são alterados; e também com o objetivo de comparar o desempenho de cada cenário do *Fading* com o EDIST2. Como resultado dessa comparação foi selecionado o melhor cenário do *Fading* de cada experimento. Essa informação é importante para verificar se existe uma configuração ótima entre os valores que foram testados para os parâmetros de entrada novos.

Como os fluxos de dados do Quadro 4 estão divididos em dois grupos, i.e. um grupo com fluxos de dados que possuem *Concept Drift* do tipo repentino e um grupo com fluxos de dados que possuem *Concept Drift* do tipo gradual, os experimentos também foram separados nesses dois grupos. O grupo de experimentos que foram realizados com fluxos de dados que possuem *Concept Drift* do tipo repentino é composto pelo Experimento 1, que foi realizado com o fluxo de dados ARH, e pelo Experimento 2, que foi realizado com o fluxo de dados ASE A e pelo Experimento 3, que foi realizado com o fluxo de dados AS.

Já o grupo dos experimentos que foram realizados utilizando fluxos de dados que possuem *Concept Drift* do tipo gradual é composto pelo Experimento 4, que foi realizado com o fluxo de dados GRH, pelo Experimento 5, que foi realizado com o fluxo de dados GSEA, e pelo Experimento 6, que foi realizado com o fluxo de dados GS.

Depois que o EDIST2 e o *Fading* foram utilizados para classificar as instâncias dos fluxos de dados conforme especificado no parágrafo anterior, foram coletadas as variáveis de saída para cada cenário testado e os resultados foram apresentados na Tabela 3 para os fluxos de dados que possuem *Concept Drift* do tipo repentino, e na Tabela 4 para os fluxos de dados que possuem *Concept Drift* do tipo gradual.

Primeiramente, são analisados os resultados que foram obtidos nos testes realizados com os fluxos de dados que possuem *Concept Drift* do tipo repentino. Nesse grupo de resultados estão presentes os cenários do Experimento 1, os cenários do Experimento 2 e os cenários do Experimento 3. Todos esses resultados estão representados na Tabela 3, a seguir.

Observam-se nos valores apresentados na Tabela 3 algumas informações importantes: 1) o *Fading* consumiu mais tempo de CPU do que o EDIST2 em 27 cenários de um total de 30 cenários; 2) em todos os 30 cenários o consumo de memória RAM do *Fading* foi menor do que o consumo de memória RAM do EDIST2; 3) a acurácia média do EDIST2 é maior ou igual a acurácia média do *Fading* em todos os cenários.

No Experimento 1, o melhor desempenho do *Fading* ocorreu no Cenário 7. Isso porque, nesse cenário, apesar da acurácia ser 0,88% menor do que a acurácia do EDIST2 e do tempo de CPU ser 6% maior que o tempo de CPU do EDIST2, o *Fading* conseguiu reduzir o consumo de RAM em quase 73% em relação ao EDIST2. Nos outros cenários desse experimento, apesar da acurácia média ser maior do que a acurácia média do Cenário 7, o tempo de CPU e o consumo de memória RAM desses cenários foram maiores do que o tempo de CPU e do que o consumo de memória RAM do Cenário 7.

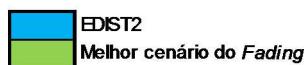
Já no Experimento 2, o melhor desempenho do *Fading* ocorreu no Cenário 8. Isso porque nesse cenário o *Fading* consumiu o menor tempo de CPU e também a menor quantidade de memória RAM. Veja que o consumo de recursos computacionais no Cenário 8 foi menor do que o consumo de recursos computacionais do EDIST2.

Por outro lado, foi nesse cenário que o *Fading* apresentou o pior desempenho de acurácia média entre os três experimentos. No Cenário 8 a acurácia média do *Fading* foi 1,11% menor do que a acurácia média do EDIST2. Embora a perda de acurácia seja uma característica ruim para o

algoritmo, essa perda não foi tão significativa quando o ganho computacional desse cenário foi levado em consideração, já que o tempo de CPU foi 1,76% menor do que o tempo de CPU do EDIST2 e o consumo de memória RAM foi aproximadamente 53% menor do que o consumo de memória do EDIST2.

Tabela 3 - Resultados da análise de sensibilidade da proposta de melhoria *Fading* para fluxos de dados que possuem *Concept Drift* do tipo repentino

Experimento	Fluxo de dados	Cenário	Parâmetros de entrada		Acurácia média	Tempo total de CPU	Consumo médio de RAM	Comparação do melhor cenário em relação ao EDIST2
			L	P				
1	Rotating Hyperplane	EDIST2	-	-	98,41	1,31	21.044	
		1	15	0,3	98,39	1,53	18.471	
		2	15	0,8	98,41	1,55	18.324	
		3	50	0,3	98,39	1,63	18.656	
		4	50	0,8	98,16	1,50	14.638	
		5	100	0,3	97,60	1,48	7.136	
		6	100	0,8	98,03	1,53	13.246	
		7	200	0,3	97,53	1,39	5.694	
		8	200	0,8	98,38	1,28	18.736	
		9	500	0,3	97,62	1,42	7.858	
		10	500	0,8	97,57	1,47	5.606	
2	SEA	EDIST2	-	-	97,84	1,70	22.548	
		1	15	0,3	97,08	2,20	14.716	
		2	15	0,8	97,36	2,00	16.071	
		3	50	0,3	96,89	2,06	12.422	
		4	50	0,8	97,11	1,94	13.291	
		5	100	0,3	97,79	1,94	20.824	
		6	100	0,8	96,83	1,80	13.004	
		7	200	0,3	97,22	1,86	14.561	
		8	200	0,8	96,73	1,67	10.720	
		9	500	0,3	97,59	1,89	18.524	
		10	500	0,8	96,84	1,72	11.420	
3	STAGGER	EDIST2	-	-	85,41	2,66	9.046	
		1	15	0,3	85,36	2,81	8.424	
		2	15	0,8	85,36	2,83	8.424	
		3	50	0,3	85,39	2,72	6.886	
		4	50	0,8	85,39	3,11	6.886	
		5	100	0,3	85,38	2,86	6.640	
		6	100	0,8	85,38	2,64	6.640	
		7	200	0,3	85,34	3,08	5.969	
		8	200	0,8	85,34	3,13	5.969	
		9	500	0,3	85,26	3,02	6.403	
		10	500	0,8	85,26	3,14	6.403	



No Experimento 3, o melhor desempenho do *Fading* ocorreu no Cenário 6, quando o *Fading* apresentou, aproximadamente, o mesmo valor de acurácia média que o EDIST2, com um de tempo de CPU 0,75% menor do que o tempo de CPU do EDIST2, e uma redução do consumo de memória RAM de 26,60% em relação ao EDIST2.

Além disso, nos outros cenários do *Fading* quando uma das variáveis de saída apresentou um desempenho superior ao desempenho da mesma variável do Cenário 6, as outras duas variáveis de saída apresentaram um desempenho inferior ao desempenho das mesmas variáveis do Cenário 6. Um exemplo disso é o Cenário 7 que consumiu menos memória RAM do que o Cenário 6, mas por outro lado, o Cenário 7 consumiu mais tempo de CPU que o Cenário 6 e ainda teve uma acurácia média menor que a acurácia média do Cenário 6.

Destaca-se que o melhor cenário do Experimento 2 e o melhor cenário do Experimento 3 apresentaram uma redução do custo computacional em relação ao EDIST2. Isso pode indicar que a proposta de melhoria *Fading* pode melhorar a tarefa de detecção de *Concept Drift* em fluxos de dados *online* através da redução do custo computacional. Entretanto, quando o tipo de *Concept Drift* do fluxo de dados foi alterado para um *Concept Drift* do tipo gradual, o *Fading* apresentou uma queda de desempenho significativa porque passou a consumir mais tempo de CPU em relação ao EDIST2.

Para ilustrar os resultados obtidos quando o EDIST2 e o *Fading* foram utilizados para classificar instâncias de fluxos de dados que possuem *Concept Drift* do tipo gradual foi criada a Tabela 4. De acordo com os resultados da Tabela 4, de modo geral, o *Fading* aumentou consideravelmente o tempo de CPU em relação ao EDIST2. Observa-se, por exemplo, que os cenários do Experimento 6 apresentaram um tempo de CPU bastante discrepante do tempo de CPU do EDIST2. Note que o Cenário 5 do Experimento 6 chegou a ser 43,25% mais oneroso em tempo de CPU do que o EDIST2.

Apesar do tempo de CPU do *Fading* ser elevado em relação ao tempo de CPU do EDIST2, o consumo de memória RAM dessa proposta de melhoria foi significativamente inferior ao consumo de memória RAM do EDIST2 em 15 dos 30 cenários. Por exemplo, o Cenário 9 do Experimento 4 consumiu 74,33% menos memória RAM do que o EDIST2. Além disso, em 23 dos 30 cenários o *Fading* conseguiu reduzir o consumo de memória RAM em relação ao EDIST2.

Tabela 4 - Resultados da análise de sensibilidade da proposta de melhoria *Fading* para fluxos de dados que possuem *Concept Drift* do tipo gradual

Experimento	Fluxo de dados	Cenário	Parâmetros de entrada		Acurácia média	Tempo total de CPU	Consumo médio de RAM	Comparação do melhor cenário em relação ao EDIST2
			L	P	(%)	(segundos)	(bytes)	
4	Rotating Hyperplane	EDIST2	-	-	98,44	1,03	22.257	
		1	15	0,3	98,37	1,38	18.865	
		2	15	0,8	98,41	1,19	18.333	
		3	50	0,3	97,68	1,36	9.977	
		4	50	0,8	98,39	1,31	18.668	
		5	100	0,3	97,61	1,22	7.142	
		6	100	0,8	98,00	1,19	13.245	
		7	200	0,3	98,16	1,19	14.700	
		8	200	0,8	98,16	1,20	14.649	
		9	500	0,3	97,50	1,13	5.713	
		10	500	0,8	98,18	1,11	14.614	
5	SEA	EDIST2	-	-	96,90	1,75	22.632	
		1	15	0,3	96,59	2,33	18.987	
		2	15	0,8	96,77	2,02	22.699	
		3	50	0,3	96,84	2,14	18.858	
		4	50	0,8	96,06	2,25	13.212	
		5	100	0,3	96,87	2,45	20.952	
		6	100	0,8	96,59	2,23	17.504	
		7	200	0,3	96,57	2,64	18.665	
		8	200	0,8	96,57	2,45	18.665	
		9	500	0,3	96,11	1,91	13.467	
6	STAGGER	EDIST2	-	-	79,21	2,52	10.984	
		1	15	0,3	79,20	2,94	11.879	
		2	15	0,8	79,20	3,33	11.879	
		3	50	0,3	79,33	3,23	8.163	
		4	50	0,8	79,33	3,52	8.163	
		5	100	0,3	79,27	3,61	3.517	
		6	100	0,8	79,27	3,38	3.517	
		7	200	0,3	79,50	3,13	12.347	
		8	200	0,8	79,50	3,33	12.347	
		9	500	0,3	79,39	3,44	11.430	
		10	500	0,8	79,39	3,48	11.430	

 EDIST2
 Melhor cenário do *Fading*

Outro aspecto importante que pode ser observado quando comparamos os resultados que estão apresentados na Tabela 3 com os resultados que estão apresentados na Tabela 4, é que o melhor cenário de cada fluxo de dados mudou quando o tipo de *Concept Drift* mudou. Quando o fluxo de dados *Rotating Hyperplane* possuía *Concept Drift* do tipo repentino, i.e. no Experimento 1, o melhor cenário ocorreu quando o valor de L foi igual a 200 e quando o valor de P foi igual a 0,3. Entretanto, quando o tipo de *Concept Drift* do fluxo de dados *Rotating Hyperplane* foi alterado para o tipo gradual, i.e. no Experimento 4, o melhor cenário ocorreu quando o valor de L foi igual a 500 e quando o valor de P foi igual a 0,3. O mesmo ocorreu para o fluxo de dados SEA, que é representado pelo Experimento 2 e pelo Experimento 5, e para o

fluxo de dados Stagger, que é representado pelo Experimento 3 e pelo Experimento 6. Portanto, o tipo de *Concept Drift* influencia no ajuste dos parâmetros de entrada novos do *Fading*. Com isso, conclui-se que não haverá um valor ótimo para que os parâmetros de entrada que são introduzidos pelo *Fading*.

De um modo geral, os resultados que foram obtidos pelo *Fading* nos seis experimentos mostram que não há uma relação proporcional entre os parâmetros de entrada, L e P, e as variáveis de saída acurácia, tempo de CPU e consumo de RAM. Isso porque o aumento do valor de L não gerou um aumento proporcional de acurácia e nem uma redução proporcional do custo computacional. Entretanto, a alteração do valor desses parâmetros de entrada produziu uma alteração de valor das variáveis de saída do *Fading*.

Como os experimentos deste capítulo foram destinados à verificação do comportamento das variáveis de saída do *Fading* quando os valores dos parâmetros de entrada novos são alterados, foram utilizados fluxos de dados pequenos. Com isso, o universo de análise ficou limitado e, por isso, não é possível afirmar se há ou não efetividade de melhoria quando o *Fading* é utilizado.

Portanto, no Capítulo 6 será apresentada uma investigação que foi realizada com essa proposta de melhoria em um cenário mais amplo, com um fluxo de dados maior. Assim, foi possível verificar ao longo do tempo se o *Fading* melhora a detecção de *Concept Drift* em relação ao EDIST2.

5.2. Análise de sensibilidade da proposta de melhoria *Reduced Boundary*

Na mesma linha de investigação dos experimentos que foram realizados com a proposta de melhoria *Fading* na Seção 5.1, esta seção apresenta uma investigação que foi realizada com a proposta de melhoria *Reduced Boundary* para identificar qual é a influência que os parâmetros de entrada novos, Distância Mínima Aceitável (D) e Repetitividade de D (R), têm sobre o comportamento desse algoritmo. Além disso, foi realizada uma análise comparativa de desempenho entre o *Reduced Boundary* e o EDIST2 com o objetivo de verificar se o *Reduced Boundary* melhora o desempenho do EDIST2. Por fim, foi selecionado o melhor cenário do *Reduced Boundary* com o

objetivo de encontrar, entre os valores que foram testados, qual é o melhor ajuste global para os parâmetros de entrada dessa proposta de melhoria.

Assim como foi feito na Seção 5.1, a análise de sensibilidade foi realizada utilizando os fluxos de dados que estão apresentados no Quadro 4. Portanto, os experimentos desta também foram agrupados conforme o tipo de *Concept Drift*. Com isso, para os fluxos de dados que possuem *Concept Drift* do tipo repentino foram criados o Experimento 7, que foi realizado com o fluxo de dados ARH, o Experimento 8, que foi realizado com o fluxo de dados ASEA, e o Experimento 9, que foi realizado com o fluxo de dados AS. Já para os fluxos de dados que possuem *Concept Drift* do tipo gradual foram criados o Experimento 10, que foi realizado com o fluxo de dados GRH, o Experimento 11, que foi realizado com o fluxo de dados GSEA, e o Experimento 12, que foi realizado com o fluxo de dados GS.

Para os experimentos que foram realizados nessa seção foram criados 9 cenários de testes, conforme a Tabela 5. Cada cenário de teste foi criado a partir de uma seleção distributiva que foi realizada entre os valores que foram atribuídos para o parâmetro de entrada D, que são 5, 10 e 50, com os valores que foram atribuídos para o parâmetro de entrada R, que são 10, 20 e 50.

Tabela 5 - Cenários de testes utilizados em cada experimento da proposta de melhoria *Reduced Boundary*.

Cenário	1	2	3	4	5	6	7	8	9
D	5	5	5	10	10	10	50	50	50
R	10	20	50	10	20	50	10	20	50

Assim como ocorreu na escolha de valores para os parâmetros L e P, do *Fading*, a escolha de valores para o parâmetro D e a escolha de valores para o parâmetro R, do *Reduced Boundary*, foi realizada com o objetivo de verificar se o aumento de valor dos parâmetros de entrada seria replicado na mesma proporção pelas variáveis de saída dessa proposta de melhoria. Portanto, não foi realizado um estudo para identificar quais seriam os melhores valores para os parâmetros de entrada, já que o objetivo dos experimentos deste capítulo é entender qual é o comportamento das variáveis de saída quando os parâmetros de entrada novos são alterados.

Portanto, este trabalho sugere como proposta de trabalho futuro um estudo para verificar se existe um valor ótimo para os parâmetros de entrada da proposta de melhoria *Reduced Boundary*, e também verificar se é possível que o *Reduced Boundary* ajuste automaticamente o valor desses parâmetros de entrada novos de acordo com as características do fluxo de dados que está sendo classificado.

Depois que foram definidos a quantidade de experimentos e a quantidade de cenários que cada experimento possui, foram realizados os testes e foram coletados os valores das variáveis de saída do *Reduced Boundary* do EDIST2. Os resultados do Experimento 7, do Experimento 8 e do Experimento 9 que foram realizados com fluxos de dados que possuem *Concept Drift* do tipo repentino estão representados na Tabela 6. Já os resultados do Experimento 10, do Experimento 11 e do Experimento 12 que foram realizados com fluxos de dados que possuem *Concept Drift* do tipo gradual estão representados na Tabela 7.

Nos resultados apresentados na Tabela 6 a alteração de valor dos parâmetros de entrada novos, D e R, provocam uma mudança de comportamento nas variáveis de saída do *Reduced Boundary*. Entretanto, as variáveis de saída do *Reduced Boundary* não apresentaram uma alteração de valor proporcional à alteração de valor dos parâmetros de entrada. Com isso, fica demonstrado que não há relação direta dos parâmetros de entrada com as variáveis de saída dessa proposta de melhoria.

Outra característica importante que deve ser observada nos experimentos da Tabela 6 é que o Cenário 7 e o Cenário 8 foram os cenários que apresentaram o menor consumo de memória RAM entre os 9 cenários de todos os experimentos. Esse comportamento demonstra que há motivação para a realização de trabalhos futuros com o objetivo de buscar um valor ótimo para os parâmetros de entrada D e R como é sugerido por este trabalho.

Quando os resultados dos cenários de cada experimento são comparados entre si verifica-se que a alteração da característica do fluxo de dados altera o comportamento do *Reduced Boundary*. Isso foi demonstrado quando o valor dos parâmetros de entrada, D e R, precisaram ser alterados para obter o melhor cenário de cada experimento. Com isso, o melhor cenário

do Experimento 7 foi o Cenário 8, o melhor cenário do Experimento 8 foi o Cenário 1 e o melhor cenário do Experimento 9 foi o Cenário 9.

Tabela 6 - Resultados da análise de sensibilidade da proposta de melhoria *Reduced Boundary* para fluxos de dados que possuem *Concept Drift* do tipo repentino

Experimento	Fluxo de dados	Cenário	Parâmetros de entrada		Acurácia média	Tempo total de CPU	Consumo médio de RAM	Comparação do melhor cenário em relação ao EDIST2								
			D	R												
7	Rotating Hyperplane	EDIST2	-	-	98,41	1,31	21.044	<p>The chart displays three metrics: Accuracy (blue), CPU Time (orange), and RAM Consumption (grey). The Y-axis ranges from -20% to 80%. The X-axis shows values for Accuracy, CPU Time, and RAM Consumption. The legend indicates: Acurácia (blue square), Tempo CPU (orange square), and Consumo RAM (grey square).</p> <table border="1"> <thead> <tr> <th>Metric</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Acurácia</td> <td>-0,80%</td> </tr> <tr> <td>Tempo CPU</td> <td>3,57%</td> </tr> <tr> <td>Consumo RAM</td> <td>68,42%</td> </tr> </tbody> </table>	Metric	Value	Acurácia	-0,80%	Tempo CPU	3,57%	Consumo RAM	68,42%
Metric	Value															
Acurácia	-0,80%															
Tempo CPU	3,57%															
Consumo RAM	68,42%															
1	5	10	98,41	1,39	18.631											
2	5	20	98,41	1,34	18.631											
3	5	50	98,41	1,45	18.631											
4	10	10	98,41	1,25	18.631											
5	10	20	98,41	1,22	18.631											
6	10	50	98,41	1,39	18.631											
7	50	10	97,60	1,42	5.571											
8	50	20	97,61	1,27	6.645											
9	50	50	98,41	1,42	18.631											
8	SEA	EDIST2	-	-	97,84	1,58	22.548	<p>The chart displays three metrics: Accuracy (blue), CPU Time (orange), and RAM Consumption (grey). The Y-axis ranges from 0% to 15%. The X-axis shows values for Accuracy, CPU Time, and RAM Consumption. The legend indicates: Acurácia (blue square), Tempo CPU (orange square), and Consumo RAM (grey square).</p> <table border="1"> <thead> <tr> <th>Metric</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Acurácia</td> <td>0,00%</td> </tr> <tr> <td>Tempo CPU</td> <td>0,99%</td> </tr> <tr> <td>Consumo RAM</td> <td>12,44%</td> </tr> </tbody> </table>	Metric	Value	Acurácia	0,00%	Tempo CPU	0,99%	Consumo RAM	12,44%
Metric	Value															
Acurácia	0,00%															
Tempo CPU	0,99%															
Consumo RAM	12,44%															
1	5	10	97,84	1,56	19.743											
2	5	20	97,84	1,67	19.743											
3	5	50	97,84	1,61	19.743											
4	10	10	97,84	1,66	19.743											
5	10	20	97,84	1,58	19.743											
6	10	50	97,84	1,72	19.743											
7	50	10	95,39	1,88	3.083											
8	50	20	96,97	1,70	12.118											
9	50	50	97,84	1,72	19.743											
9	STAGGER	EDIST2	-	-	85,41	2,14	9.046	<p>The chart displays three metrics: Accuracy (blue), CPU Time (orange), and RAM Consumption (grey). The Y-axis ranges from -20% to 40%. The X-axis shows values for Accuracy, CPU Time, and RAM Consumption. The legend indicates: Acurácia (blue square), Tempo CPU (orange square), and Consumo RAM (grey square).</p> <table border="1"> <thead> <tr> <th>Metric</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Acurácia</td> <td>-0,06%</td> </tr> <tr> <td>Tempo CPU</td> <td>-0,73%</td> </tr> <tr> <td>Consumo RAM</td> <td>37,54%</td> </tr> </tbody> </table>	Metric	Value	Acurácia	-0,06%	Tempo CPU	-0,73%	Consumo RAM	37,54%
Metric	Value															
Acurácia	-0,06%															
Tempo CPU	-0,73%															
Consumo RAM	37,54%															
1	5	10	85,32	2,08	7.250											
2	5	20	85,42	2,30	9.106											
3	5	50	85,41	2,50	9.179											
4	10	10	84,65	2,39	2.034											
5	10	20	85,39	2,09	8.740											
6	10	50	85,41	2,31	9.179											
7	50	10	84,33	2,67	1.833											
8	50	20	84,33	2,44	1.785											
9	50	50	85,36	2,16	5.650											



A escolha do melhor cenário de cada experimento foi realizada com base no cenário que apresentou o melhor desempenho geral para as variáveis de saída. Os experimentos da Tabela 6 mostram que os cenários que foram selecionados como sendo os melhores cenários apresentaram, de maneira conjunta, menor tempo de CPU e menor consumo de memória RAM. Nos outros cenários apesar de ter ocorrido um desempenho melhor para uma variável de saída, o valor de outra variável representou um desempenho inferior, e.g. o Cenário 7 do Experimento 7 consome menos memória RAM do que o Cenário 8 do mesmo experimento, mas por outro lado, o Cenário 7 do Experimento 7 consome mais tempo de CPU do que o Cenário 8 do mesmo

experimento. No Experimento 8 e no Experimento 9 a escolha do melhor cenário obedeceram a mesma motivação do Experimento 7. Ou seja, o melhor cenário desses experimentos foi escolhido com base no melhor desempenho apresentado pelas variáveis de saída, de maneira conjunta, mesmo que apenas uma dessas variáveis de saída tenha apresentado um valor melhor em outro cenário.

Quando o tipo de *Concept Drift* dos fluxos de dados foi alterado do tipo repentino para o tipo gradual o *Reduced Boundary* apresentou alteração de comportamento das variáveis de saída. Com isso, os valores dos parâmetros de entrada novos, D e R, que representam o melhor desempenho dessa proposta de melhoria foram diferentes dos valores que foram encontrados nos experimentos da Tabela 6. Por exemplo, a Tabela 6 e a Tabela 7 mostram que a mudança do tipo de *Concept Drift* do fluxo de dados *Rotating Hyperplane* fez com que o melhor cenário do Experimento 7, que era o Cenário 8, passasse a ser no Experimento 10 o Cenário 9. Esse mesmo comportamento pode ser observado nos outros experimentos que utilizaram o mesmo fluxo de dados, porém, com o tipo de *Concept Drift* diferente que é o caso do Experimento 8 em relação ao Experimento 11 e do Experimento 9 em relação ao Experimento 12.

De maneira geral, a proposta de melhoria *Reduced Boundary* também apresentou indícios de que pode melhorar a tarefa de detecção de *Concept Drift* em fluxos de dados *online*. Isso porque os melhores cenários dos seis experimentos conseguiram reduzir o custo computacional do EDIST2. Por outro lado, a acurácia média dessa proposta de melhoria não apresentou uma melhoria em nenhum dos experimentos. Mas como os objetivos deste capítulo são o de verificar se o comportamento do algoritmo modifica quando os parâmetros de entrada novos são alterados, e também o de verificar se as propostas de melhoria conseguem melhorar o desempenho do EDIST2, os experimentos foram realizados em um cenário limitado. Por isso, o Capítulo 6 apresentará uma investigação que foi realizada para essa proposta de melhoria em um cenário mais amplo, com um fluxo de dados bem maior do que o fluxo de dados que foi utilizado nesse capítulo.

Tabela 7 - Resultados da análise de sensibilidade da proposta de melhoria *Reduced Boundary* para fluxos de dados que possuem *Concept Drift* do tipo gradual

Experimento	Fluxo de dados	Centro	Parâmetros de entrada		Acurácia média	Tempo total de CPU	Consumo médio de RAM	Comparação do melhor cenário em relação ao EDIST2
			D	R	(%)	(segundos)	(bytes)	
10	Rotating Hyperplane	EDIST2	-	-	98,44	1,00	22.257	
		1	5	10	98,44	1,05	19.626	
		2	5	20	98,44	1,08	19.626	
		3	5	50	98,44	0,98	19.626	
		4	10	10	98,40	1,00	18.667	
		5	10	20	98,44	1,06	19.626	
		6	10	50	98,44	1,00	19.626	
		7	50	10	97,75	1,08	5.376	
		8	50	20	97,49	1,06	5.651	
		9	50	50	98,44	0,94	19.626	
11	SEA	EDIST2	-	-	96,90	1,59	22.632	
		1	5	10	96,90	1,67	20.750	
		2	5	20	96,90	1,70	20.750	
		3	5	50	96,90	1,73	20.750	
		4	10	10	96,90	1,72	20.750	
		5	10	20	96,90	1,63	20.750	
		6	10	50	96,90	1,69	20.750	
		7	50	10	94,56	1,98	1.958	
		8	50	20	95,47	1,95	7.985	
		9	50	50	96,90	1,59	20.750	
12	STAGGER	EDIST2	-	-	79,21	2,31	10.984	
		1	5	10	79,12	2,45	3.957	
		2	5	20	79,14	2,66	8.253	
		3	5	50	79,21	2,55	10.544	
		4	10	10	78,55	2,89	1.813	
		5	10	20	79,11	2,33	6.188	
		6	10	50	79,27	2,58	11.098	
		7	50	10	78,08	2,81	1.502	
		8	50	20	78,35	2,48	1.628	
		9	50	50	79,20	2,48	4.492	

█ EDIST2
█ Melhor cenário do *Reduced Boundary*

5.3. Conclusão da análise de sensibilidade

Na análise de sensibilidade que foi realizada para as propostas de melhoria *Fading* e *Reduced Boundary* foi possível perceber que o EDIST2 apresentou uma grande redução do consumo de memória RAM quando uma dessas propostas de melhoria foi utilizada. No *Fading* apesar da redução do consumo de memória RAM o algoritmo passou a consumir mais tempo de CPU em relação ao tempo de CPU do EDIST2. Esse aumento do tempo de CPU já era esperado porque o *Fading* verifica mais vezes do que o EDIST2 se está ocorrendo um *Concept Drift*.

Embora o aumento do tempo de CPU seja esperado alguns dos cenários das duas propostas de melhoria apresentaram redução de valor dessa

variável de saída em relação ao EDIST2. Isso ocorreu porque o *Fading* e o *Reduced Boundary* puderam identificar a ocorrência de um *Concept Drift* antes do EDIST2. Quando essa detecção de maneira antecipada acontece o *Fading* e o *Reduced Boundary* passam a errar menos vezes a classificação de novas instâncias do fluxo de dados. Com isso, esses algoritmos precisaram ocupar menos a CPU para calcular se uma nova instância de dados é de padrão diferente, já que esta verificação é feita apenas quando o algoritmo erra a classificação de alguma instância do fluxo de dados.

Outro aspecto importante que deve ser observado é que com a utilização do *Fading* e do *Reduced Boundary* a acurácia do algoritmo apresentou uma pequena redução. Apesar da redução de acurácia ser pequena em relação a acurácia do EDIST2, um algoritmo de classificação não deve apresentar uma tendência de queda de acurácia ao longo do tempo porque a atenuação da acurácia tornaria a utilização do algoritmo inviável, já que ele passará a errar cada vez mais a classificação de instâncias novas do fluxo de dados.

Por essa razão, é importante verificar também qual será o comportamento da acurácia ao longo do tempo com o objetivo de identificar se há uma tendência de atenuação dessa variável de saída. Para isso, é preciso que sejam realizados experimentos maiores, com fluxos de dados maiores que possibilitem uma visualização mais clara de qual é a tendência de comportamento dessa variável de saída.

Outra perspectiva sobre os experimentos que foram realizados está relacionada ao ajuste dos parâmetros de entrada. Isso porque foi possível perceber que à medida que os valores dos parâmetros de entrada foram alterados as variáveis de saída também mudaram. É verdade que alguns cenários apresentaram um comportamento bem parecido para as variáveis de saída mesmo depois que os valores dos parâmetros de entrada foram alterados, mas em nenhum desses casos as variáveis de saída apresentaram exatamente o mesmo valor para cenários diferentes. Portanto, o ajuste de valor dos parâmetros de entrada influencia no comportamento das variáveis de saída e também no desempenho do *Fading* e no desempenho do *Reduced Boundary*.

De maneira geral os objetivos deste capítulo, foram: o de verificar se é possível melhorar o desempenho do EDIST2 com a utilização do *Fading* e do

Reduced Boundary; e também o de verificar se o comportamento das variáveis de saída dessas propostas de melhoria é influenciado pela alteração de valor dos parâmetros de entrada novos. Por se tratarem de verificações pontuais foram utilizados fluxos de dados pequenos. Isso limitou a perspectiva de análise acerca da efetividade de melhoria que estas propostas podem oferecer ao EDIST2.

Portanto, o Capítulo 6 apresentará investigações que foram realizadas em cenários expandidos, com fluxos de dados maiores. Com isso, foi possível responder questões importantes relacionadas a tendência das variáveis de saída ao longo do tempo e também relacionadas a efetividade de melhoria que o *Fading* e o *Reduced Boundary* podem oferecer ao EDIST2. Também no Capítulo 6, será apresentado um experimento em cenário expandido que foi realizado para verificar se a proposta de alteração na janela de dados intermediária irá aumentar o desempenho do EDIST2. Essa proposta de melhoria não foi verificada nesse capítulo porque não adiciona parâmetros de entrada novos ao EDIST2.

6. Análise experimental das propostas de melhoria em cenário expandido

No Capítulo 5 foram realizados experimentos com dois objetivos: verificar a sensibilidade que as variáveis de saída das propostas de melhoria têm quando os valores dos parâmetros de entrada novos são alterados; e verificar se o desempenho do EDIST2 melhorou com essas propostas. Além disso, no Capítulo 5 também foi selecionado o melhor cenário de cada experimento com o objetivo de identificar qual é o melhor valor para os novos parâmetros de entrada dentre aqueles que foram testados.

Já neste capítulo o objetivo é discutir se as propostas conseguem melhorar o desempenho do EDIST2. Para isso, os novos parâmetros de entrada do *Fading* e do *Reduced Boundary* foram configurados com os valores encontrados nos melhores cenários dos experimentos do Capítulo 5. Portanto, essas propostas de melhoria foram utilizadas para analisar fluxos de dados bem maiores do que os fluxos de dados que foram utilizados no Capítulo 5. O objetivo de realizar experimentos em fluxos de dados maiores é o de verificar se o *Fading* e o *Reduced Boundary* apresentam um desempenho superior ao desempenho do EDIST2 ao longo do tempo.

Além de apresentar resultados de experimentos que foram realizados para as propostas de melhoria *Fading* e *Reduced Boundary*, este capítulo discute os resultados dos experimentos que foram realizados para a proposta de melhoria no gerenciamento da janela de dados intermediária do EDIST2, chamado nesse capítulo de EDIST2-MJI. É importante lembrar que essa proposta de melhoria não introduz novos parâmetros de entrada no EDIST2 e, por isso, não foram realizados experimentos para essa proposta de melhoria no Capítulo 5.

Neste capítulo é apresentada uma comparação dos resultados que foram obtidos nas variáveis de saída das três propostas de melhoria que foram sugeridas por esse trabalho com o valor das variáveis de saída do EDIST2. Para isso, é importante caracterizar a configuração que foi utilizada nos experimentos para obter o valor das variáveis de saída de cada algoritmo. Essas configurações dizem respeito a quais fluxos de dados foram utilizados, ao número de instâncias que compõe cada fluxo de dados, a localização do

Concept Drift em cada fluxo de dados, ao algoritmo classificador que é responsável pela construção do MERPID e quais as configurações dos parâmetros de entrada N, L, P, D e R. Todas essas configurações estão especificadas no Quadro 5, a seguir. Assim como ocorreu no Capítulo 5, cada fluxo de dados que foi utilizado deu origem a um experimento. Entretanto, conforme foi tratado na Seção 4.3 os fluxos de dados que foram utilizados nos experimentos a seguir são 500 vezes maiores, de modo que cada conjunto de dados é formado pela repetição sequenciada dos fluxos de dados que foram utilizados no Capítulo 5. Além disso, conforme já foi abordado na Seção 4.2.3 estes conjuntos de dados não possuem balanceamento de classes, isto é, o surgimento de instâncias de dados de cada classe no fluxo não é equiprovável.

Quadro 5 - Configuração utilizada para a realização dos experimentos em cenário de dados expandido

Fluxos de dados	- <i>Abrupt Rotating Hyperplane (ARH)</i> – Seção 4.2.3 (a) - <i>Abrupt Streaming Ensemble Algorithm (ASEA)</i> – Seção 4.2.3 (b) - <i>Abrupt Stagger (AS)</i> – Seção 4.2.3 (c) - <i>Gradual Rotating Hyperplane (GRH)</i> – Seção 4.2.3 (a) - <i>Gradual Streaming Ensemble Algorithm (GSEA)</i> – Seção 4.2.3 (b) - <i>Gradual Stagger (GS)</i> – Seção 4.2.3 (c)
Tamanho do fluxo de dados	50 milhões de instâncias
Localização do <i>Concept Drift</i> nos fluxos de dados	Instâncias 40.000-70.000, Instâncias 140.000-170.000, Instâncias 240.000-270.000, ... Instâncias 49.840.000-49.870.000, Instâncias 49.940.000-49.970.000
Algoritmo classificador	<i>Hoeffding Trees</i>
Configuração EDIST2	N=30
Configuração <i>Fading</i>	L=200 e P=0,3
Configuração <i>Reduced Boundary</i>	D=50 e R=50

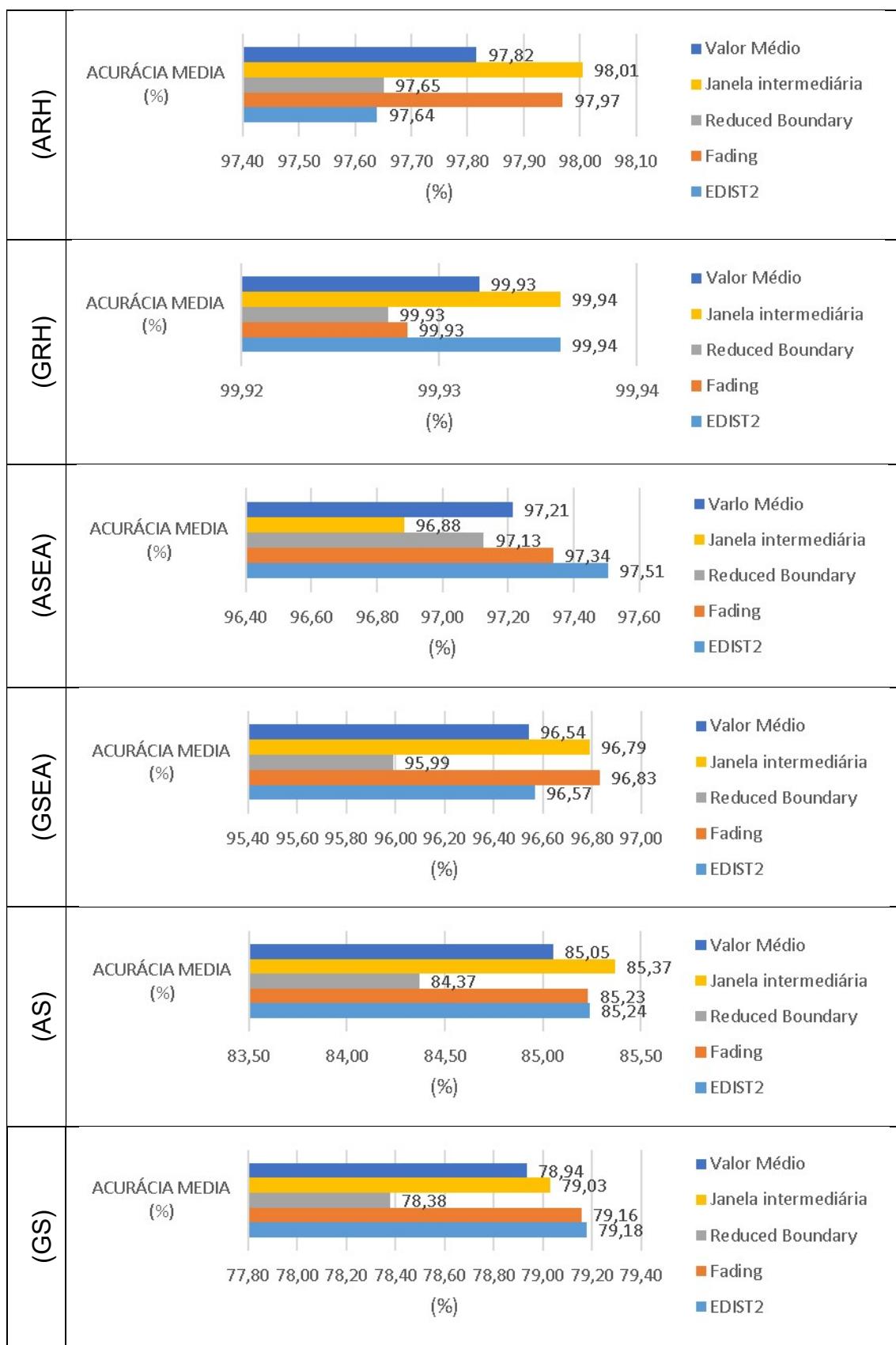
Depois que foi definido o número de experimentos e quais são as configurações utilizadas para esses experimentos, os algoritmos EDIST2, *Fading*, *Reduced Boundary* e EDIST2-MJI foram utilizados para classificar as

instâncias de cada fluxo de dados do Quadro 5. Depois que a tarefa de classificação foi concluída, foram coletados os valores das variáveis de saída acurácia média, tempo de CPU e consumo de memória RAM de cada um desses algoritmos. Assim, foram criadas a Figura 13, a Figura 14 e a Figura 15. Na Figura 13 foram agrupados os resultados obtidos para a variável de saída acurácia média em todos os seis experimentos. Na Figura 14 foram agrupados os resultados obtidos para a variável de saída tempo de CPU em todos os seis experimentos. Por fim, na Figura 15 foram agrupados os resultados obtidos para a variável de saída consumo de RAM dos seis experimentos.

Na Figura 13 está representado o desempenho da acurácia média de cada proposta de melhoria em relação à acurácia média do EDIST2 para cada fluxo de dados testado. A proposta de melhoria EDIST2-MJI (Janela intermediária) apresentou um desempenho de acurácia média melhor do que o desempenho de acurácia média do EDIST2 na maioria dos experimentos. Já o *Fading* obteve um desempenho de acurácia média parecido com o desempenho de acurácia média do EDIST2. Por outro lado, a proposta de melhoria *Reduced Boundary* não apresentou resultados significativos de acurácia média em relação a acurácia média do EDIST2 em nenhum dos seis experimentos.

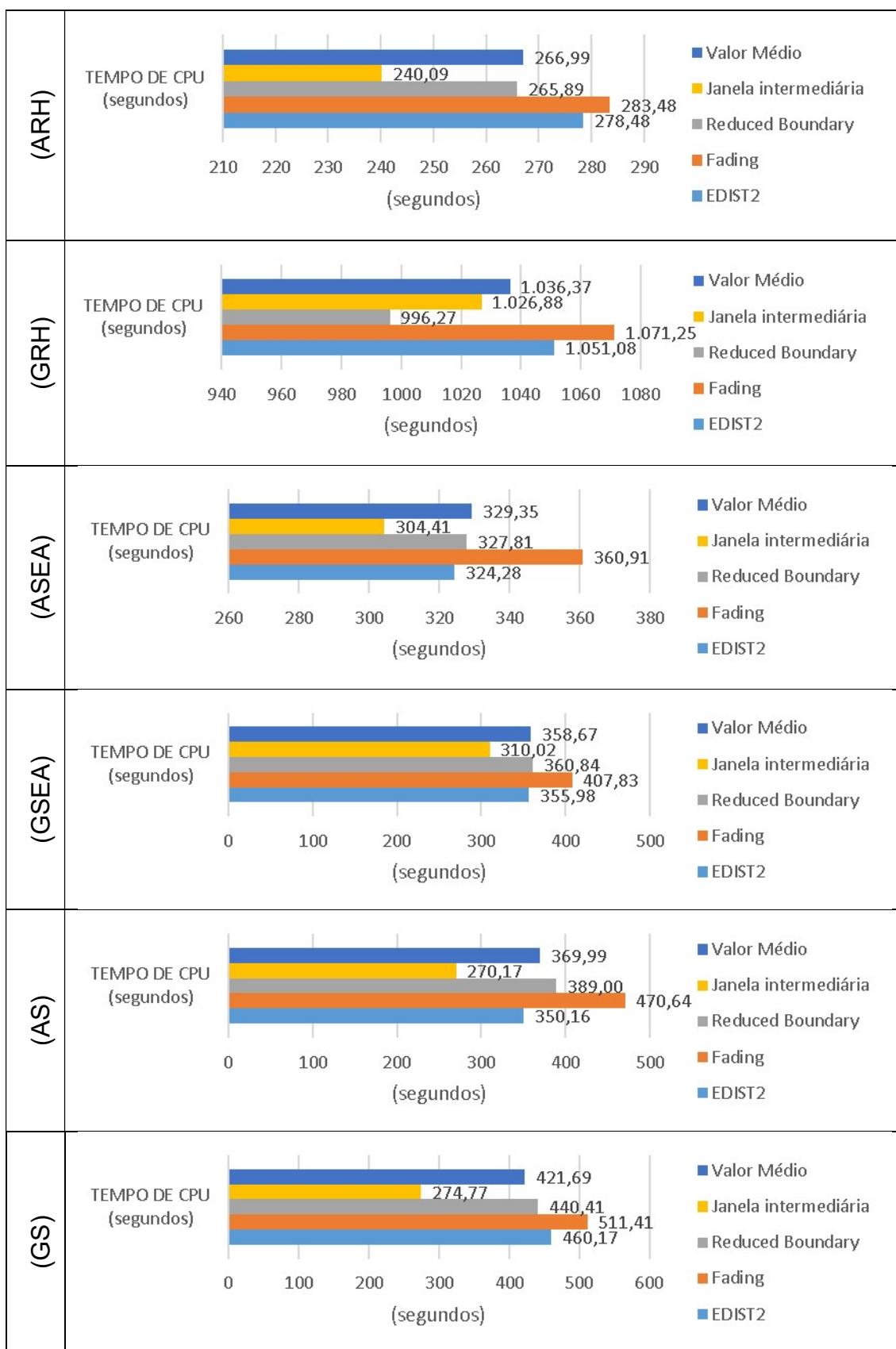
Quando a variável de saída tempo de CPU foi analisada, a proposta de melhoria EDIST2-MJI novamente apresentou um desempenho superior ao desempenho que foi obtido pela mesma variável de saída do EDIST2. Isso ocorreu porque em todos os experimentos essa proposta de melhoria consumiu menos tempo de CPU do que o EDIST2. Já a proposta de melhoria *Fading* que tinha apresentado um bom desempenho de acurácia média foi o algoritmo que consumiu o maior tempo de CPU e, por isso, apresentou o pior desempenho para essa variável de saída. Por fim, a proposta *Reduced Boundary* apresentou um desempenho superior ao desempenho do EDIST2 para essa variável de saída já que consumiu menos tempo de CPU do que o EDIST2. A Figura 12 ilustra essas comparações.

Figura 13 - Desempenho da acurácia das propostas de melhoria em relação ao EDIST2 no cenário de dados expandido



Fonte: (Do autor)

Figura 14 - Desempenho do tempo de CPU das propostas de melhoria em relação ao EDIST2 no cenário de dados expandido



Fonte: (Do autor)

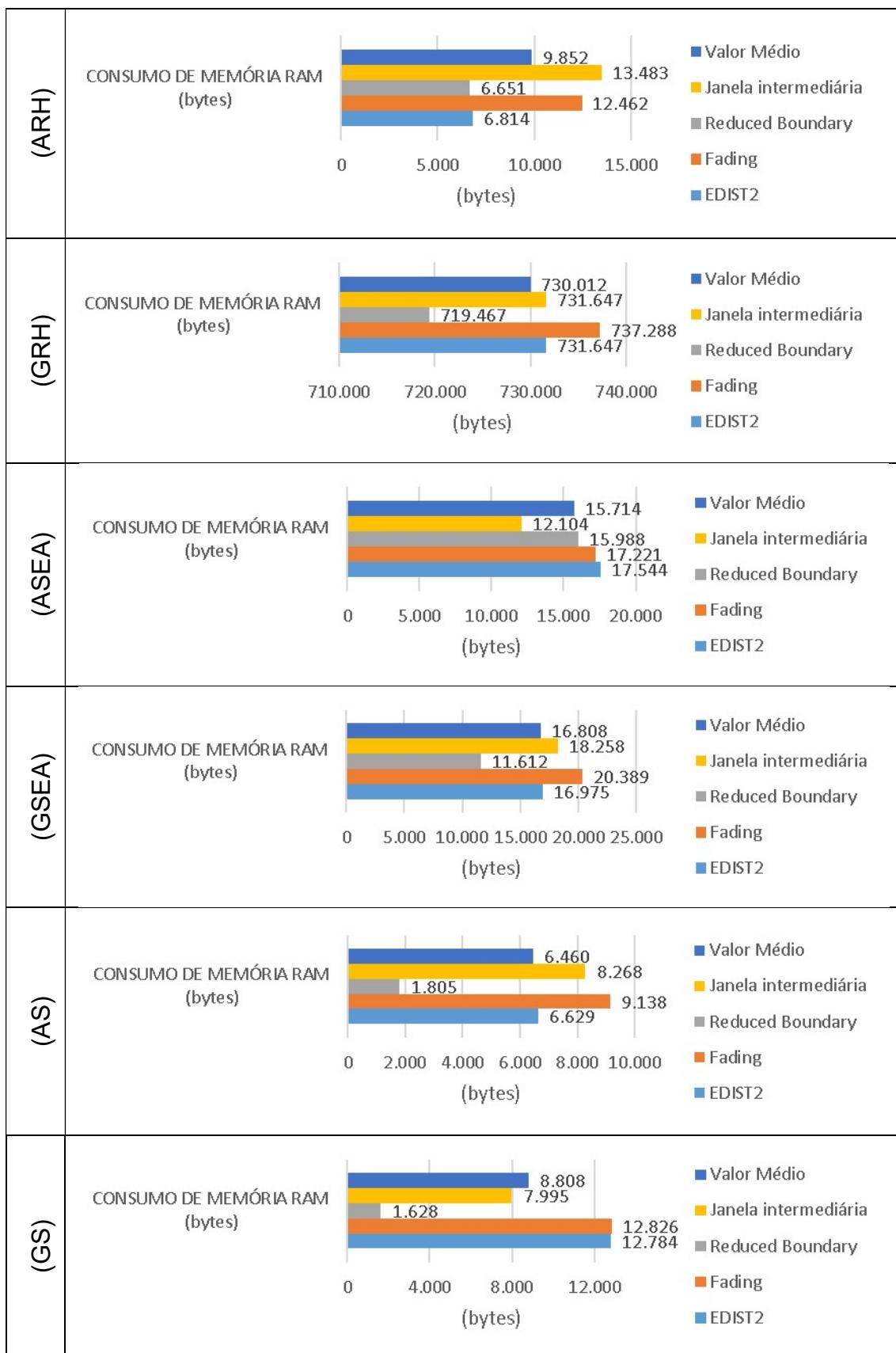
A última variável de saída que foi utilizada para comparar o desempenho das propostas de melhoria com o desempenho do EDIST2 é o consumo de memória RAM. A comparação de desempenho entre esses algoritmos está ilustrada na Figura 15.

Na Figura 15, o melhor desempenho dessa variável de saída ocorreu na proposta de melhoria *Reduced Boundary*. Isso porque essa proposta de melhoria consumiu a menor quantidade de memória RAM em 5 dos 6 experimentos. Nota-se que o consumo de memória RAM dessa proposta de melhoria foi bastante inferior ao consumo de memória RAM do que o EDIST2 e do que as outras propostas de melhoria em 4 dos 6 experimentos.

Já a proposta de melhoria EDIST2-MJI apresentou um desempenho para essa variável de saída próximo do desempenho do EDIST2. Isso ocorreu porque em 3 dos 6 experimentos o consumo de memória RAM do EDIST2-MJI foi superior ao consumo de memória RAM do EDIST2, em 1 experimento o consumo de memória RAM do EDIST2-MIJ foi igual ao consumo de memória RAM do EDIST2, e em 2 experimentos o consumo de memória do EDIST2-MJI foi inferior ao consumo de memória RAM do EDIST2.

Por fim, a proposta de melhoria *Fading* não apresentou resultados bons para essa variável de saída. Isso porque em 5 dos 6 experimentos o consumo de memória RAM do *Fading* foi bastante superior ao consumo de memória RAM do EDIST2 e das outras propostas de melhoria. No único experimento em que o *Fading* consome menos memória RAM do que o EDIST2 essa diferença é bastante insignificativa e, ainda, o *Fading* consome mais memória RAM o *Reduced boundary* e do que a proposta de melhoria EDIST2-MIJ.

Figura 15 - Desempenho do consumo de RAM das propostas de melhoria em relação ao EDIST2 no cenário de dados expandido



Fonte: (Do autor)

Nos experimentos que foram apresentados neste capítulo veja que as três propostas que foram sugeridas por esse trabalho melhoram o desempenho do algoritmo EDIST2. O EDIST2-MJI foi a proposta de melhoria que teve o melhor desempenho de acurácia média e também teve o melhor desempenho em relação ao tempo de CPU, apesar de ter um consumo grande de memória RAM. Por outro lado, a proposta de melhoria *Reduced Boundary* foi a proposta de melhoria que consumiu a menor quantidade de memória RAM e, ainda, conseguiu obter um bom desempenho para tempo de CPU. Já a proposta de melhoria *Fading*, apesar de apresentar um bom desempenho de acurácia, ela consumiu muito tempo de CPU e também uma quantidade muito grande de memória RAM.

Assim, com base nos resultados que foram obtidos nos experimentos em cenários maiores podemos verificar que a proposta de melhoria na janela de dados intermediária do EDIST2, a proposta de melhoria *Fading* e a proposta de melhoria *Reduced Boundary* podem melhorar a tarefa de detecção de *Concept Drift* em fluxos de dados *online* do algoritmo EDIST2.

7. Conclusões, contribuição e trabalhos futuros

Com base nos resultados que foram obtidos neste trabalho foi possível concluir que as propostas *Fading*, *Reduced Boundary* e a proposta de melhoria na janela intermediária de dados podem melhorar a tarefa de detecção de *Concept Drift* em fluxos de dados *online*. Isso porque essas propostas de melhoria apresentaram um desempenho superior em pelo menos uma das variáveis de saída, acurácia, tempo de CPU ou consumo de memória RAM. Entretanto, vale ponderar que a proposta de melhoria *Fading* demonstrou ser um algoritmo com um custo computacional alto, já que teve um tempo de CPU e também um consumo de memória RAM altos. Por outro lado, a proposta de melhoria *Reduced Boundary* e a proposta de melhoria na janela intermediária de dados conseguiram reduzir o tempo de CPU e o consumo de memória RAM em relação ao valor dessas variáveis de saída do EDIST2.

Outra perspectiva de análise está relacionada ao comportamento das variáveis de saída quando o tipo de *Concept Drift* do fluxo de dados foi alterado. Para esse caso, perceba que não ocorreu uma alteração significativa no padrão de comportamento das variáveis de saída após a alteração do tipo de *Concept Drift* dos fluxos de dados. Na Figura 13, de acurácia média, na Figura 14, de tempo de CPU e na Figura 15, de consumo de memória RAM, o desempenho das variáveis de saída das propostas de melhoria quando o tipo de *Concept Drift* era o repentino ficou parecido com o padrão de comportamento do desempenho das variáveis de saída desses algoritmos quando o tipo de *Concept Drift* do fluxo de dados foi o gradual.

Como contribuição para a área de pesquisa este trabalho propôs novas abordagens para identificar uma mudança no padrão dos dados: o *Fading* e o *Reduced Boundary*. Nessas abordagens novas foi possível reduzir o custo computacional do algoritmo detector de *Concept Drift* através do *Reduced Boundary* sem que ocorresse degradação significativa da acurácia do algoritmo-base desse trabalho. O fato do *Reduced Boundary* ajudar a antecipar a detecção do *Concept Drift* em relação ao EDIST2 fez com que o algoritmo classificador *Hoeffding Trees* precisasse construir um MERPID menor para representar o padrão dos dados dos fluxos de dados e, por isso, o *Reduced Boundary* precisou consumir menos memória RAM.

Como o *Fading* e o *Reduced Boundary* introduzem novos parâmetros de entrada ao EDIST2 e o ajuste desses parâmetros pode influenciar no desempenho do algoritmo, propõe-se como trabalho futuro uma investigação que consiga identificar se é possível utilizar um valor ótimo para os parâmetros de entrada dessas propostas de melhoria. Este trabalho ainda sugere, como um desafio maior, fazer com que o *Fading* ou o *Reduced Boundary* consigam ajustar automaticamente o valor dos parâmetros de entrada novos de acordo com as características do fluxo de dados.

8. Bibliografia

- BAENA-GARCÍA, M. et al. Early Drift Detection Method. **Fourth international workshop on knowledge discovery from data streams**, v. 6, p. 77-86, 2006.
- BIFET, A.; GAVALDÀ, R. Learning from Time-Changing Data with Adaptive Windowing. **SIAM International Conference on Data Mining (SDM'07)**, 2007. 443-448.
- BRAMER, M. Principles of Data Mining (Undergraduate Topics in Computer Science), 2007.
- BRAMER, M. **Principles of Data Mining**. 2nd. ed. Incorporated: Springer Publishing Company, 2013.
- CABENA, et al. **Discovering Data Mining: From Concept to Implementation**. Upper Saddle River, NJ: Prentice Hall, 1998.
- CABRAL, D. R. D. L.; BARROS, R. S. M. Concept drift detection based on Fisher's Exact test. **Information Sciences**, p. 220-234, Fevereiro 2018.
- CABRAL, L. D. S.; SIEBRA, S. D. A. Identificação de Competências em Currículos usando Ontologias: uma abordagem teórica, Faculdade Integrada do Recife, 2006.
- CAMILO, C. O.; DA SILVA, J. C. **Mineração de Dados: Conceitos, Tarefas, Métodos e Ferramentas**. Universidade Federal de Goiás. Goiânia. 2009.
- COSTA, F. G. D. et al. Multidimensional surrogate stability to detect data stream concept drift. **Expert Systems With Applications**, p. 15-29, 2017.
- DEHGHAN, M.; BEIGY, H.; ZAREMOODI, P. A novel concept drift detection method in data streams using ensemble classifiers. **Intelligent Data Analysis** **20**, p. 1329-1350, 2016.
- DELEN, ; OLSON, D. L. **Advanced Data Mining Techniques**. Heidelberg: Springer, 2008.
- DEMŠAR, J.; BOSNIC, Z. Detecting concept drift in data streams using model explanation. **Expert Systems With Applications**, 2018. 546-559.
- DESAI, D.; JOSHI, A. **A Deviant Load Shedding System for Data Stream Mining**. International Conference on Advanced Computing Technologies and Applications. Mumbai, India: Elsevier. 2015. p. 118 - 126.
- DONG, F. et al. Active Fuzzy Weighting Ensemble for Dealing with Concept Drift. **International Journal of Computational Intelligence Systems**, v. 11, n. Atlantis Press, p. 438 - 450, 2018.
- DONGRE, P. B.; MALIK, L. G. A review on real time data stream classification and adapting to various concept drift scenarios. **IEEE International Advance Computing Conference (IACC)**, Gurgaon, p. 533-537, Fevereiro 2014. ISSN 978-1-4799-2571-1.

ELWELL, R.; POLIKAR, R. Incremental learning of concept drift in nonstationary environments. **IEEE Transactions on Neural Networks**, p. 1517 - 1531, 2011.

FAN, W. **Systematic Data Selection to Mine Concept-Drifting Data**. 10th ACM SIGKDD International Conference of Knowledge Discovery. Seattle, WA, USA: ACM New York. 2004. p. 128-137.

FISHER, R. **Biological Monographs and Manuals**. London, England. 1934.

GAMA, J. et al. Learning with Drift Detection. In: BAZZAN, A. ; LABIDI, S. **Advances in Artificial Intelligence – SBIA 2004**. São Luis, Maranhão, Brasil: Springer, v. 3171 of the series Lecture Notes in Computer Science, 2004. p. 286-295.

GANOR, B. et al. **Terrorism Informatics - Knowledge Management and Data Mining for Homeland Security**. New York: Springer, 2008.

HAMBY, D. M. A review of techniques for parameter sensitivity analysis of environmental models. **Environmental Monitoring and Assessment**, v. 32, p. 135-154, September 1994.

HAN, J.; KAMBER, M.; PEI, J. **Data mining concepts and techniques**, 2011. ISSN 978-0-12-381479-1.

HOEFFDING, W. Probability inequalities for sums of bounded random variables. **Journal of the American Statistical Association**, p. 13-30, 1963.

HUANG, Q.-H. Sequential Patterns Mining on High-Dimensional Data Stream. **Intelligent Computing Theories and Applications**, Berlin, 2012. 447-454.

HULTEN, G.; SPENCER, L.; DOMINGOS, P. **Mining time-changing data streams**. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, California: ACM New York, NY, USA. 2001. p. 97-106.

KANTARDZIC, M. **Data Mining**: Concepts, Models, Methods, and Algorithms. Second Edition. ed. [S.I.]: IEEE Press, 2011.

KHAMASSI, I. et al. **Ensemble classifiers for drift detection and monitoring in dynamical environments**. Annual conference of the Prognostics and Health Management Society. New Orlean: [s.n.]. 2013. p. 199-224.

KHAMASSI, I. et al. Self-Adaptive Windowing Approach for Handling Complex Concept Drift. **Cognitive Computation**, v. 7, n. 6, p. 772-790, Dezembro 2015. ISSN 1866-9964.

KHAMASSI, I.; SAYED-MOUCHAWEH, M. Drift detection and monitoring in non-stationary environments. **IEEE Conference on Evolving and Adaptive Intelligent Systems**, Linz, Austria, 2014. 1-6.

- KOLTER, J. Z.; MALOOF, M. A. Dynamic weighted majority: An ensemble method for drifting concepts, *J. Machine Learning Research*, p. 2755 - 2790, 2007.
- LADEIRA, M.; DE OLIVEIRA, M. G.; DE ARAÚJO, M. E. C. **Lupa Digital: Agilização da Busca Decadactilar na Identificação Criminal Através de Mineração de Dados.** XXV Congresso da Sociedade Brasileira de Computação. São Leopoldo, RS: [s.n.]. 2005. p. 1945-1959.
- LAROSE, D. T. **Discovering Knowledge in Data: An Introduction to Data Mining.** [S.I.]: John Wiley & Sons, Inc, 2005.
- LESKOVEC, J.; RAJARAMAN, A.; ULLMAN, J. D. Mining Data Streams. In: LESKOVEC, J.; RAJARAMAN, A.; ULLMAN, J. D. **Mining of Massive Datasets.** 2a. ed. California: Cambridge University Press, v. Único, 2014. Cap. 4, p. 131. ISBN 9781107077232. Disponível em: <<http://www.mmds.org/>>. Acesso em: 01 Janeiro 2016.
- MASUD, M. M. et al. Addressing Concept-Evolution in Concept-Drifting Data Streams. **IEEE International Conference on Data Mining**, 2010. 929-934.
- MCCUE, C. **Data Mining and Predictive Analysis - Intelligence Gathering and Crime Analysis.** [S.I.]: Elsevier, 2007.
- MITCHELL, T. **Machine Learning.** New York, NY, USA: McGraw Hill, v. Único, 1997. ISBN 0070428077.
- MOA Project Website. **MOA Project Website**, 2016. Disponível em: <[http://moa.cms.waikato.ac.nz/.>](http://moa.cms.waikato.ac.nz/.). Acesso em: 01 Janeiro 2016.
- NEUMANN, J. V.; MORGENSTERN, O. **Theory of games and economic behavior.** [S.I.]: Princeton University Press, 1944.
- PONNIAH, P. **Data Warehousing Fundamentals:** a comprehensive guide for it professionals. [S.I.]: John Wiley and Sons, Inc., 2001.
- SADIK, S.; GRUENWALD, ; LEAL, E. In Pursuit of Outliers in Multi-dimensional Data Streams. **IEEE International Conference on Big Data**, 2016.
- SHALEV-SHWARTZ, S. Online Learning: Theory, Algorithms, and Applications. **Tese de doutorado**, Jerusalem, 2007.
- SHANNON, C. E. A mathematical theory of communication. **ACM SIGMOBILE Mobile Computing and Communications Review** , v. V, p. 3-55, Janeiro 2001.
- STRUMBELJ, E.; KONONENKO, I. An efficient explanation of individual classifications. **The Journal of Machine Learning Research**, 2010. 1-18.
- ŠTRUMBELJ, E.; KONONENKO, I.; ŠIKONJA, M. R. Explaining instance classifications with interactions of subsets of feature values. **Data & Knowledge Engineering**, 2009. 886-904.

TSYMBAL, A. **The problem of concept drift: definitions and related work, v. 6.** Computer Science Department, Trinity College Dublin. Dublin. 2004.

U.S. Department of Energy (DOE). **Atmospheric Radiation Measurement (ARM) Program**, 2015. Disponivel em:

<https://www.arm.gov/images/cms/data_steam_diagram.gif>. Acesso em: 17 Novembro 2015.

WANG, H. et al. **Mining Concept-Drifting Data Streams using Ensemble Classifiers**. KDD '03 Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. Washington, D.C.: ACM New York, NY, USA. 2003. p. 226-235.

WIDMER, G.; KUBAT, M. Learning in the Presence of Concept Drift and Hidden Contexts. **Machine Learning**, v. 23, n. Issue 1, p. 69 101, April 1996.

WITTEN, I. H.; FRANK, E. **Data Mining: Practical Machine Learning Tools and Techniques**. San Francisco, CA: Elsevier, 2005.

YATES, F. Contingency Tables Involving Small Numbers and the χ^2 Test. **Supplement to the Journal of the Royal Statistical Society**, v. 1, p. 217-235, 1934.

Apêndice A - Glossário

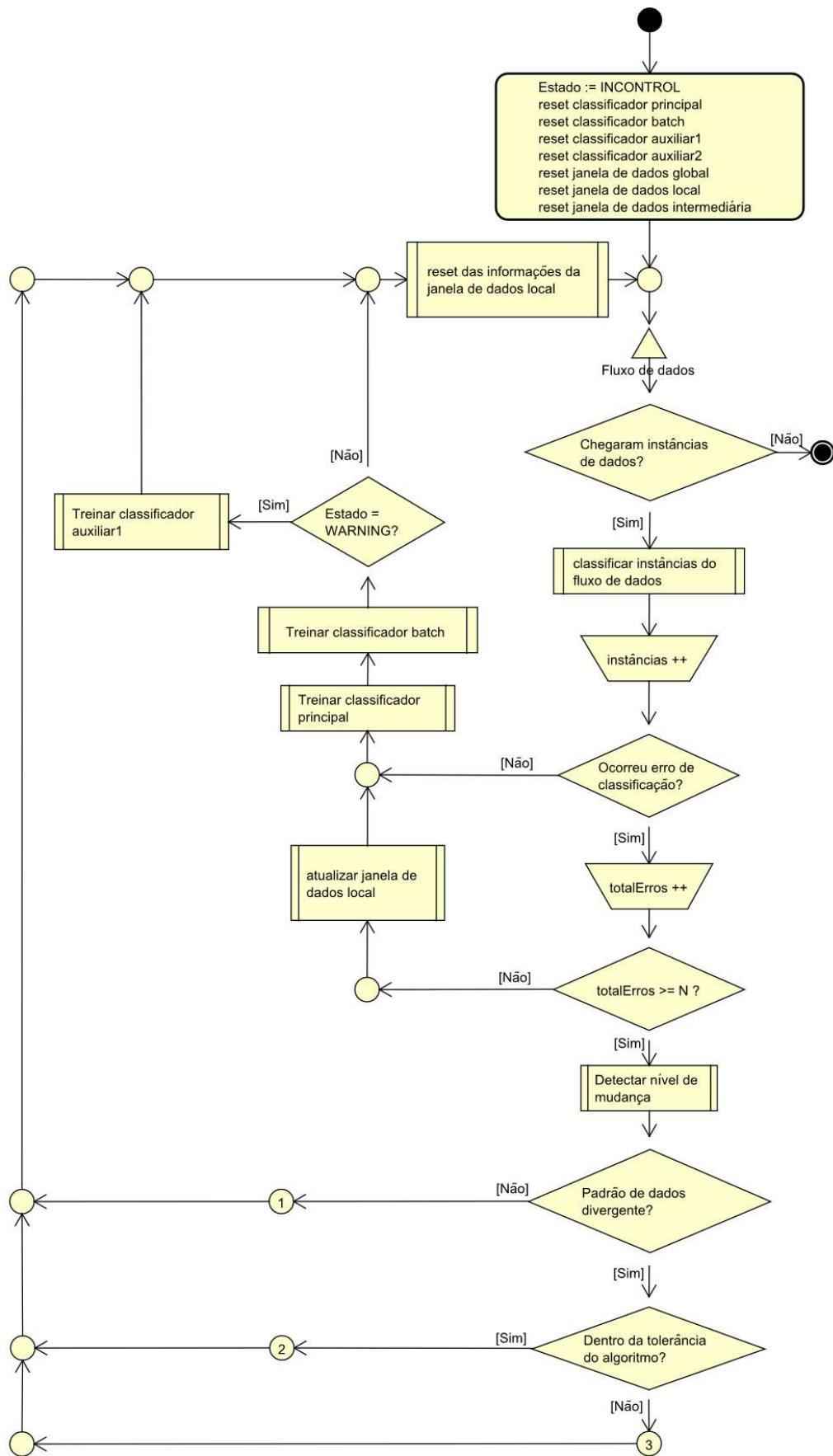
Termo	Significado	Referência
Adwin <i>(Adaptive windowing)</i>	É uma técnica que utiliza janela de dados de tamanho flexível para armazenar bits ou números reais. (BIFET e GAVALDÀ, 2007)	
Blip drift, mudança ligeira	É a ocorrência de um evento raro pertencente a padrão conhecido do modelo de aprendizado, em meio as instâncias do padrão atual do fluxo. (DONGRE e MALIK, 2014)	
Buffer	É um espaço temporário na memória de acesso aleatório para leitura e escrita de dados.	
Classificação	É o processo de descoberta de um modelo (ou função) que descreve e distingue classes de dados ou conceitos (HAN, KAMBER e PEI, 2011)	
Concept drift, Mudança de conceito	É a mudança que ocorre no padrão dos dados que chegam pelo fluxo de dados ao longo do tempo. (KHAMASSI, SAYED-MOUCHAWEH et al., 2015)	
Data mining, mineração de dados	É o processo de descobrir padrões e conhecimento interessantes de grandes quantidades de dados. (HAN, KAMBER e PEI, 2011)	
Data streams, fluxos de dados	É uma sequência de dados agrupados em pacotes que são transmitidos de uma origem para um destino.	
Decision trees	É um método que se baseia num diagrama em forma de árvore, onde cada nó interno representa um teste em um atributo, cada ramo representa o resultado do teste, e cada nó folha representa uma etiqueta classe que representa o atributo. (HAN, KAMBER e PEI, 2011)	
Drift Detection Method (DDM)	É uma técnica utilizada para auxiliar algoritmos de aprendizado de máquinas na detecção de <i>concept drift</i> em fluxos de dados. (GAMA, MEDAS et al., 2004)	

<i>Drift level</i>	No trabalho que introduz o método EDIST2, <i>Drift level</i> é um estado utilizado pelo algoritmo para identificar que está ocorrendo <i>concept drift</i> no fluxo de dados.
<i>Early Drift Detection Method (EDDM)</i>	É uma melhoria proposta no <i>Drift Detection Method</i> para auxiliar algoritmos de aprendizado de máquinas na detecção de <i>concept drift</i> em fluxos de dados. (BAENA-GARCÍA, CAMPO-AVILA et al., 2006)
<i>EDIST, EDIST2</i>	É uma técnica utilizada para auxiliar algoritmos de aprendizado de máquinas na detecção de <i>concept drift</i> em fluxos de dados que se baseia na distância entre erros consecutivos de classificação de dados. (KHAMASSI, SAYED-MOUCHAWEH et al., 2015)
<i>Gradual drift, mudança gradual</i>	É uma mudança no padrão dos dados do fluxo, em que aos poucos o padrão atual vai sendo substituído por outro padrão conhecido do modelo de aprendizado.
<i>Hoeffding trees</i>	É um algoritmo que, para tomada de decisão, se baseia no teorema da fronteira de Hoeffding e que é utilizado em inteligência artificial para auxiliar da detecção de <i>concept drift</i> .
<i>Incontrol level</i>	No trabalho que introduz o método detector de <i>concept drift</i> EDIST2, é denominado estado <i>Incontrol Level</i> se o fluxo de dados não apresentou nenhuma distorção no padrão dos dados do fluxo. (KHAMASSI, SAYED-MOUCHAWEH et al., 2015)
<i>Incremental drift, mudança incremental</i>	Ocorre quando os dados do fluxo mudam lentamente ao longo do tempo do padrão atual para um outro padrão conhecido do modelo de aprendizado. (DONGRE e MALIK, 2014)

Machine learning, aprendizado de máquinas	É o domínio de inteligência computacional voltado para a construção de algoritmos que melhoram o seu desempenho automaticamente com base na experiência adquirida.
MOA	<i>Massive Online Analysis</i> (MOA) (MOA Project Website, 2016) implementada na linguagem de programação Java para mineração de fluxos de dados que inclui uma coleção de algoritmos de aprendizado de máquina e ferramentas associadas.
Noise, ruído	São mudanças aleatórias no padrão do fluxo de dados, com instâncias que não pertençam a um padrão conhecido do modelo de aprendizado.
Online learning, aprendizado online	É o processo de responder a uma sequência de perguntas, em posse das respostas corretas para cada uma destas perguntas, e ainda ser capaz de aprender novos conceitos, oferecendo respostas corretas para novas perguntas. (SHALEV-SHWARTZ, 2007)
Outlier	É a ocorrência de um dado significativamente distante dos demais dados, com características que podem caracterizá-lo como gerado por uma fonte diferente da fonte geradora dos demais dados. (HAN, KAMBER e PEI, 2011)
Recurring drift, mudança recorrente	É a ocorrência temporária de mudança no padrão dos dados do fluxo, para instâncias de outro padrão conhecido do modelo de aprendizado.
Regressão	É uma metodologia estatística que é mais frequentemente usada para previsão numérica, identificando as tendências da distribuição com base nos dados disponíveis. (HAN, KAMBER e PEI, 2011)

<i>Sliding Window, windowing, data window, janelamento, janela deslizante</i>	Janela de dados que pode ter tamanho fixo ou flexível e que é utilizada para percorrer o fluxo de dados auxiliando na detecção do <i>concept drift</i> .
<i>Sudden drift, mudança repentina</i>	É a mudança no padrão do fluxo de dados onde no instante T_0 estão surgindo instâncias pertencentes ao padrão A, e no momento T_1 passam a surgir apenas instâncias do padrão B.
Warning level	É um estado intermediário utilizado pelos métodos EDIST e EDIST2 na detecção de <i>concept drift</i> com o objetivo de aguardar novas instâncias que confirmem a ocorrência de <i>concept drift</i> ou se foi apenas um ruído.

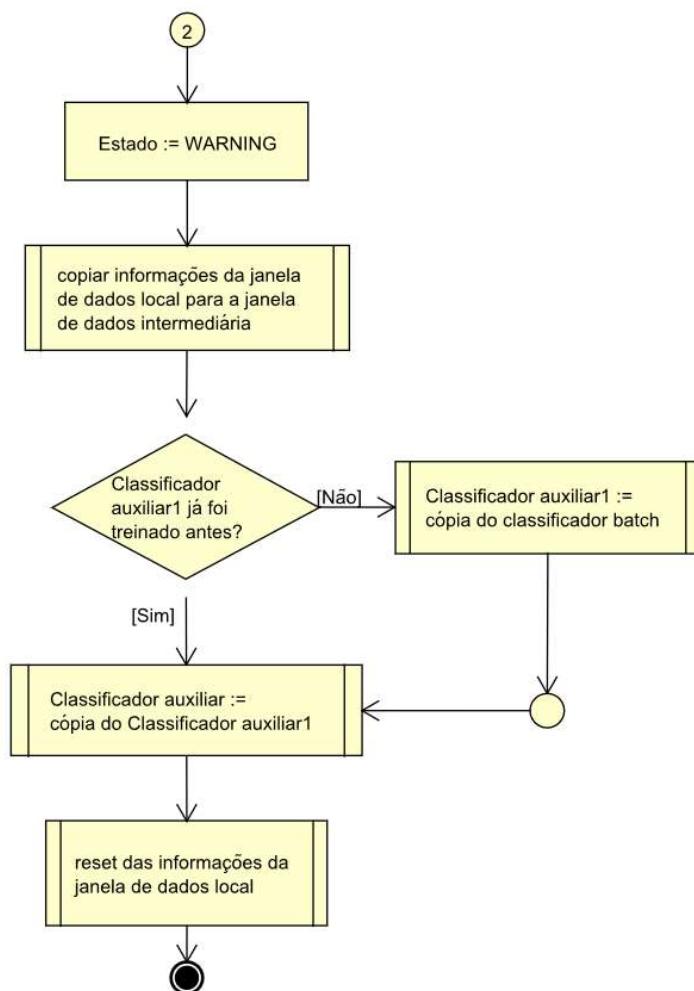
Apêndice B – Fluxograma do algoritmo EDIST2



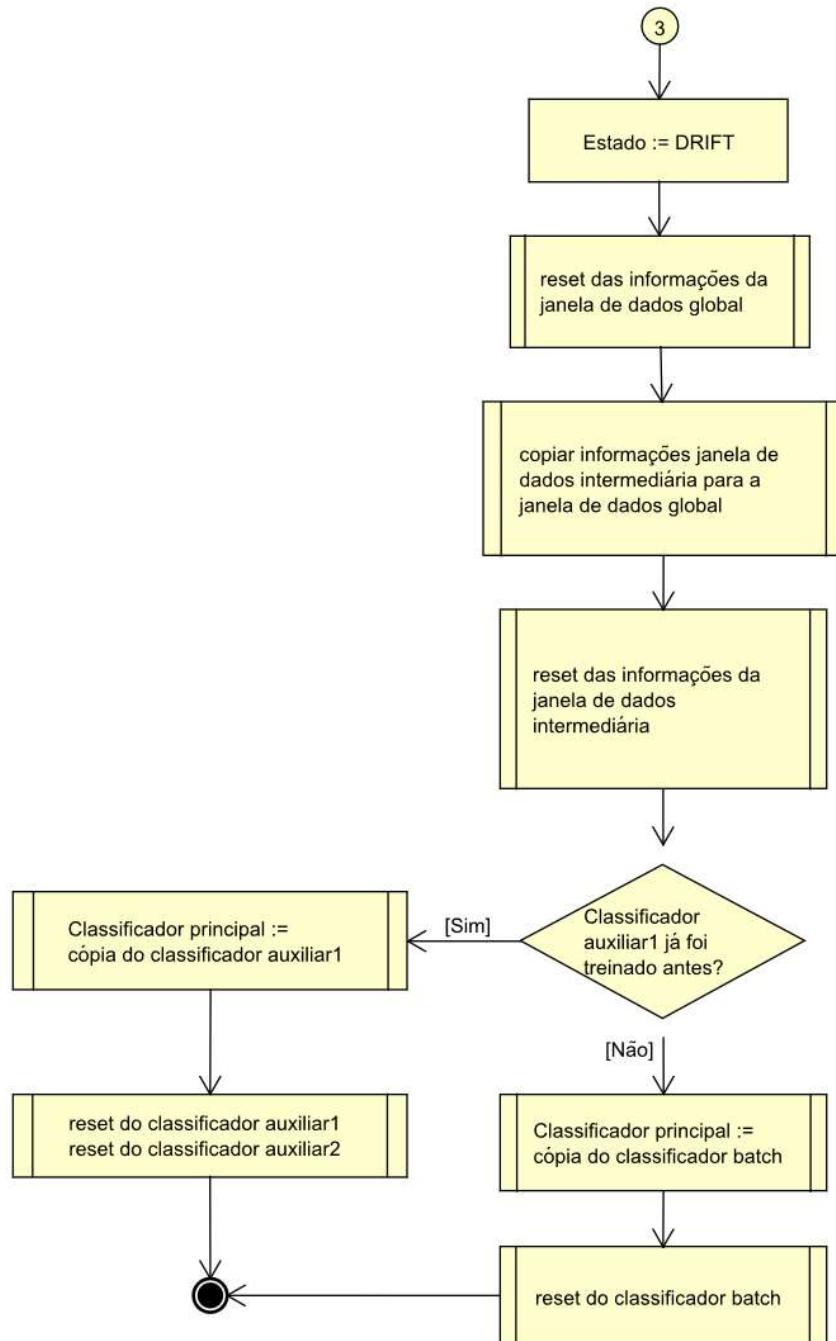
Essa função é chamada pelo algoritmo principal EDIST2 quando o padrão das instâncias de dados mais recentes do fluxo de dados não foi muito divergente do padrão de dados que estava presente no fluxo de dados. Com isso o EDIST2 permanece no estado INCONTROL e atualiza o MERPID antes de voltar a analisar novas instâncias de dados.



Essa função é chamada quando o padrão das instâncias de dados mais recentes do fluxo de dados é um pouco divergente do padrão das instâncias de dados mais antigas do fluxo de dados. Com isso, o EDIST2 entende que pode estar ocorrendo uma mudança no padrão dos dados e entra no estado WARNING. A partir daí armazena em um MERPID separado o padrão dessas últimas instâncias do fluxo de dados para o caso do padrão de dados mudar. Depois disso, o EDIST2 continua a analisar as instâncias novas que chegarem no fluxo de dados.



Quando o padrão das instâncias de dados mais recentes for muito divergente do padrão das instâncias de dados que estavam presentes no fluxo, o EDIST2 chamará essa função com o objetivo de descartar o MERPID que representa o padrão de dados obsoleto para utilizar o MERPID do padrão de dados novo que está presente no fluxo de dados. Depois disso, o EDIST2 continua a analisar novas instâncias do fluxo de dados.



Apêndice C – Pseudocódigo do algoritmo EDIST2

Algorithm *EDIST2*

Input: (x, y) : Data Stream
 N_0 : the minimum number of errors to construct the window
Output: Trained classifier C

1. *InitializeClassifier*(C)
2. $C' \leftarrow \text{CopyClassifier}(C)$
3. $W_G \leftarrow \text{CollectInstances}(C, N_0)$
4. $W'_G \leftarrow \emptyset$
5. **repeat**
6. $W_0 \leftarrow \text{CollectInstances}(C, N_0)$
7. $\text{Level} \leftarrow \text{DetectedLevel}(W_G, W_0)$
8. **switch** (Level)
9. **case 1: Incontrol**
10. $W_G \leftarrow W_G \cup W_0$
11. $\text{UpdateParameters}(W_G, W_0)$
12. The classifier C is incremented according to W_G
13. **end case 1**
14. **case 2: Warning**
15. $W'_G \leftarrow W'_G \cup W_0$
16. $\text{UpdateParameters}(W'_G, W_0)$
17. The classifier C' is incremented according to W'_G
18. **end case 2**
19. **case 3: Drift**
20. $W_G \leftarrow W'_G$
21. $W'_G \leftarrow \emptyset$
22. $\text{ResetClassifier}(C)$
23. $C \leftarrow \text{CopyClassifier}(C')$
24. $\text{ResetClassifier}(C')$
25. **end case 3**
26. **end switch**
27. **until** The end of the data streams

Algorithm *DetectedLevel*(W_G, W_0)

Input: W_G : Global data window characterized by:
 N_G : error number
 μ_G : error distance mean
 σ_G : error distance standard deviation
 W_0 : Current data window characterized by:
 N_0 : error number,
 μ_0 : error distance mean,
 σ_0 : error distance standard deviation
Output: *Level*: detection level

1. $\mu_d \leftarrow \mu_G - \mu_0$
2. $\sigma_d \leftarrow \sqrt{\frac{\sigma_G^2}{N_G} + \frac{\sigma_0^2}{N_0}}$
3. $\epsilon \leftarrow t_{1-\alpha} * \sigma_d$
4. **if** ($\mu_d > \epsilon + \sigma_d$)
5. $\text{Level} \leftarrow \text{Drift}$
6. **else if** ($\mu_d > \epsilon$)
7. $\text{Level} \leftarrow \text{Warning}$
8. **else** $\text{Level} \leftarrow \text{Incontrol}$
9. **end if**
10. **end if**
11. **return** (Level)

Algorithm *CollectInstances*(C, N_0)

Input: (x, y) : Data Stream
 N_0 : minimum error number
 C : the trained classifier
Output: W : Data window characterized by:
 N : error number
 μ : error distance mean
 σ : error distance standard deviation

1. $W \leftarrow \emptyset$
2. $N \leftarrow 0$
3. $\mu \leftarrow 0$
4. $\sigma \leftarrow 0$
5. **repeat** for each instance x_i
6. **if** ($\text{Prediction}(C, x_i) = \text{false}$)
7. $d_i \leftarrow \text{computeDistance}()$
8. (* incremental update of μ and σ *)
9. $\mu \leftarrow \frac{N}{N+1}\mu + \frac{d_i}{N+1}$
10. $\sigma \leftarrow \sqrt{\frac{N-1}{N}\sigma^2 + \frac{(d_i - \mu)^2}{N+1}}$
11. $N \leftarrow N + 1$
12. **end if**
13. $W \leftarrow W \cup \{x_i\}$
14. $\text{TrainClassifier}(C, x_i)$
15. **until** ($N = N_0$)
16. **return** (W)

Algorithm *UpdateParameters*(W_G, W_0)

Input: W_G : Global data window characterized by:
 N_G : error number
 μ_G : error distance mean
 σ_G : error distance standard deviation
 W_0 : Current data window characterized by:
 N_0 : error number,
 μ_0 : error distance mean,
 σ_0 : error distance standard deviation
Output: Updated parameters of W_G

1. $\mu_G \leftarrow \frac{1}{N_G + N_0} (N_G \cdot \mu_G + N_0 \cdot \mu_0)$
2. $\sigma_G \leftarrow \sqrt{\frac{N_G \sigma_G^2 + N_0 \sigma_0^2}{N_G + N_0} + \frac{N_G N_0}{(N_G + N_0)^2} (\mu_G - \mu_0)^2}$
2. $N_G \leftarrow N_G + N_0$