UNIVERSIDADE FEDERAL DO AMAZONAS - UFAM

INSTITUTO DE COMPUTAÇÃO - ICOMP

PROGRAMA PÓS-GRADUAÇÃO EM INFORMÁTICA - PPGI

DETECÇÃO E CLASSIFICAÇÃO DO DESVIO DE CONCEITO

Ramon Fava Souza

Manaus - AM

Janeiro 2025

Ramon Fava Souza

DETECÇÃO E CLASSIFICAÇÃO DO DESVIO DE CONCEITO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática do Instituto de Computação Universidade Federal do Amazonas, como parte dos requisitos necessários à obtenção do título de Mestre em Informática.

Orientador

Prof. Dr. Rafael Giusti

Universidade Federal do Amazonas - UFAM

Instituto de Computação - IComp

Manaus - AM

Janeiro 2025

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Souza, Ramon Fava
S729d Detecção e classificação de desvio de conceito / Ramon Fava Souza . 2024
89 f.: il. color; 31 cm.

Orientador: Rafael Giusti
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Desvio de conceito. 2. Rede prototípica. 3. Detecção. 4. Classificação. I. Giusti, Rafael. II. Universidade Federal do Amazonas III. Título

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Informática

FOLHA DE APROVAÇÃO

"DETECÇÃO E CLASSIFICAÇÃO DO DESVIO DE CONCEITO"

RAMON FAVA SOUZA

DISSERTAÇÃO DE MESTRADO DEFENDIDA E APROVADA PELA BANCA EXAMINADORA CONSTITUÍDA PELOS PROFESSORES:

Prof. Dr. Rafael Giusti - PRESIDENTE

Prof. Dr. Eduardo James Pereira Souto - MEMBRO INTERNO

Dr. Ricardo Marcondes Marcacini - MEMBRO EXTERNO

MANAUS, 20 de dezembro de 2024.

Documento assinado eletronicamente por **Rafael Giusti**, **Professor do Magistério Superior**, em 28/01/2025, às 13:40, conforme horário oficial de Manaus, com fundamento no art. 6°, § 1°, do Decreto n° 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Eduardo James Pereira Souto**, **Professor do Magistério Superior**, em 28/01/2025, às 19:19, conforme horário oficial de Manaus, com fundamento no art. 6°, § 1°, do Decreto n° 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **RICARDO MARCONDES MARCACINI**, **Usuário Externo**, em 30/01/2025, às 12:07, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2405606** e o código CRC **8A3E2C5F**.

Avenida General Rodrigo Octávio, 6200 - Bairro Coroado I Campus Universitário Senador Arthur Virgílio Filho, Setor Norte - Telefone: (92) 3305-1181 / Ramal 1193
CEP 69080-900, Manaus/AM, coordenadorppgi@icomp.ufam.edu.br

Dedico à minha família e ao meu orientador que me ajudaram a tornar esse sonho possível.

Agradecimentos

Agradeço a Deus por me ajudar a alcançar esse grande e desafiador objetivo; à minha família que foi essencial para que eu chegasse até aqui; à minha esposa Luana Marina que, ao longo dessa jornada, me apoiou e me incentivou a seguir em frente, mesmo nas dificuldades; ao Prof. Dr. Rafael Giusti pela orientação, paciência, dedicação, profissionalismo e disponibilidade e à UFAM e seus professores pelos ensinamentos na minha formação.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Este trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado do Amazonas - FAPEAM - por meio do projeto POSGRAD.

"Que os nossos esforços desafiem as impossibilidades. Lembrai-vos de que as grandes proezas da história foram conquistadas do que parecia impossível"

Charles Chaplin

DETECÇÃO E CLASSIFICAÇÃO DO DESVIO DE CONCEITO

Autor: Ramon Fava Souza

Orientador: Prof. Dr. Rafael Giusti

Resumo

Dados podem ser utilizados para extrair informações úteis. Alguns deles estão em um ambiente estático, no qual suas propriedades não mudam ao longo do tempo. Já outros estão inseridos em um ambiente dinâmico. Esse dinamismo gera um problema conhecido como desvio de conceito, no qual o conceito aprendido sobre os dados muda ao longo do tempo. A consequência desse fenômeno é que, um modelo treinado com dados antes da mudança de conceito estará potencialmente desatualizado, não refletindo mais o conceito e, portanto, apresentando baixo desempenho. Para resolver esse problema, existem diversos detectores de desvio, mas são raros os que identificam especificamente o tipo de mudança. Este trabalho analisa o estado da arte na classificação de desvio de conceito para ajudar os modelos a responderem melhor ao desvio e auxiliar os usuários na tomada de decisão, fornecendo mais detalhes sobre o que está acontecendo com os dados. Para isso, foi criada uma base sintética contendo diferentes tipos e causas de desvio de conceito. Depois, uma rede prototípica foi treinada com esse conjunto. A fim de testar o detector desenvolvido, vários experimentos foram executados, dentre eles estão: o comparativo do algoritmo feito com outros modelos, a detecção e identificação do desvio em um conjunto sintético e a detecção em conjuntos reais e sintéticos nas tarefas de classificação e agrupamento. Por fim, o algoritmo desenvolvido superou outros detectores em vários cenários.

Palavras-chave: Classificação, Identificação, Desvio de Conceito, Rede Prototípica.

DETECÇÃO E CLASSIFICAÇÃO DO DESVIO DE CONCEITO

Autor: Ramon Fava Souza

Orientador: Prof. Dr. Rafael Giusti

Abstract

Data can be used to extract useful information. Some of them are in a static environment, in which the data properties do not change over time. Others are in a dynamic environment. This dynamism produces a problem known as concept drift, when the concept learned about the data changes over time. The consequence of this phenomenon is that a model trained with data before the concept change will potentially be out of date, no longer reflecting the concept and, therefore, presenting low performance. To solve this problem there are several drift detectors, but few of them can classify the drift. Therefore, this work analyzes the state of the art in concept drift classification to help algorithms better respond to drift and assist users in decision making by providing more details about what is happening with the data. To this end, a synthetic database containing different types and causes of concept drift was created. Then, a prototypical network was trained with this dataset. In order to test the developed detector, several experiments were executed, including: the comparison of the algorithm with other models, the detection and identification of drift in a synthetic set and the detection in real and synthetic datasets in classification and clustering tasks. Finally, the developed algorithm outperformed other detectors in several scenarios.

Keywords: Classification, Identification, Concept Drift, Prototypical Network.

LISTA DE ILUSTRAÇÕES

Lista de tabelas

SUMÁRIO

1

INTRODUÇÃO

1.1 Contextualização

Em Aprendizado de Máquina (AM), o objetivo é extrair, de maneira automática ou semiautomática, um modelo que "explica" o conhecimento contido nos dados (MAHESH, 2020). Geralmente, os modelos são treinados considerando-se um ambiente de aprendizado estático, no qual as características dos dados não mudam com o tempo. Entretanto, nos ambientes dinâmicos, pode ocorrer o desvio de conceito, no qual essas características se modificam ao longo do tempo, tornando o modelo de AM obsoleto (JR et al., 2014). Um exemplo prático desse desvio pode ser observado em (DINO, 2023), uma publicação do jornal Globo no qual se observa uma mudança no hábito alimentar dos brasileiros na procura por alimentos saudáveis. Identificar esse tipo de mudança é importante, pois permite às empresas adaptar seus modelos de negócio para elevar suas vendas e, consequentemente, majorar seus faturamentos. Um exemplo seria a criação de um modelo de recomendação, nos aplicativos que vendem comidas, de restaurantes mais saudáveis ou a realização de promoções para esse público alvo que mudou seus hábitos alimentares.

Assim, o desvio de conceito afeta o desempenho de um modelo, pois ele treinou e aprendeu um conceito que, ao longo do tempo, mudou. Um jeito simples de não ser impactado pela mudança de conceito é treinar o modelo periodicamente. Entretanto, esse treinamento pode ter um custo alto. Uma boa prática é treinar só quando for necessário. Por isso, o desvio de conceito e sua detecção vem sendo estudado há algum tempo pela literatura, com alguns trabalhos anteriores ao ano de 1986 (SCHLIMMER;

GRANGER, 1986). Nesse contexto, é importante ressaltar que existe mais de um tipo de desvio porque há diferentes formas de alteração da distribuição dos dados em um ambiente dinâmico, quais sejam, a título de exemplo: abrupta, incremental, gradual e recorrente, que serão explicitadas mais adiante.

Desse modo, descobrir o tipo de desvio ajuda, dentre outros pontos, a solucionar, da melhor forma, um desvio de conceito que ocorre em um fluxo de dados (quando dados chegam continuamente sem um final definido e com variação ao longo do tempo de volume e velocidade) (GUO et al., 2022). Por exemplo, um varejista online utiliza um algoritmo para recomendar para o cliente um certo produto na compra de outro. Com o passar do tempo, a relação de consumo desses itens muda e eles não são mais consumidos juntos. Assim, há um desvio de conceito e o varejista terá seu lucro diminuído se não resolver esse problema. Então, ele pode fazer os ajustes necessários no modelo sabendo que há um desvio de conceito e qual o seu tipo.

Além disso, o tipo de desvio ajuda tanto em problemas supervisionados quanto nos não supervisionados. No último, tem-se menos informações sobre os dados. Um dos objetivos de analisar os dados pode ser para inseri-los em grupos que irão conter instâncias com características parecidas e diferentes de outros grupos. Utilizando o exemplo anterior do varejista, o agrupamento poderia juntar clientes com características semelhantes para oferecer os melhores produtos para essas pessoas, ao invés de ofertar produtos aleatórios.

Ademais, também há muitos estudos sobre detecção de desvio de conceito (YANG et al., 2021; CASTELLANI; SCHMITT; HAMMER, 2021; NAMITHA; KUMAR, 2020), sendo que a maioria foca, principalmente, em descobrir o ponto onde o desvio ocorreu e a minoria em descobrir o tipo de desvio (GUO et al., 2022). Além disso, não existem trabalhos que analisam as alterações na distribuição dos dados para informar como ela mudou.

Outro problema é que nos conjuntos de dados reais utilizados nos trabalhos de detectores de desvio não são marcados os pontos nos quais ocorrem as mudanças nem seus tipos. O que muitos deles fazem para avaliar o detector é utilizar conjuntos sintéticos, pois tem-se o controle das características das mudanças. Entretanto, não há

um conjunto que também marque a causa do desvio de conceito.

Portanto, levando em consideração os problemas citados - alto custo de retreinar em curtos intervalos um modelo, falta de mais informações sobre o desvio além de que ele ocorreu e a ausência de detectores que identificam a causa do desvio - o presente estudo irá desenvolver um algoritmo para detectar e classificar um desvio de conceito, visando alertar o usuário sobre mudanças no conjunto de dados no qual o seu modelo esteja operando. Além de criar um conjunto de dados que será utilizado para treinar esse algoritmo.

1.2 Objetivos

Tendo em vista os problemas apresentados, o objetivo geral deste trabalho é desenvolver um algoritmo para a detecção e classificação de desvios de conceito, visando identificar os tipos de desvios e as alterações nos dados para melhorar a adaptabilidade dos modelos.

Os objetivos específicos são:

- Criar um conjunto de dados para representar as causas do desvio de conceito;
- Detectar desvios de conceito em tarefas de agrupamento e classificação;
- Implementar um algoritmo para identificar o tipo de desvio e as mudanças associadas nos dados;
- Avaliar a generalização do algoritmo em domínios distintos daqueles utilizados no treinamento.

1.3 Organização do Trabalho

O restante deste texto é composto por mais 5 capítulos: Fundamentos Teóricos, Trabalhos Relacionados, Proposta de Trabalho, Experimentos e Resultados e Conclusão. O primeiro apresenta os conceitos que foram necessárias para o desenvolvimento do

trabalho e que são essenciais para o entendimento do trabalho: desvio de conceito (o que é, tipos, causas, como detectar), aprendizado de máquina (classificação e agrupamento) e como avaliar modelos.

O próximo capítulo expõe os trabalhos relacionados a este. Logo, ele trata sobre algoritmos de detecção de desvio de conceito e sua identificação. É importante para notar o estado da arte em detecção e identificação de desvios e as lacunas que existem. Em seguida, o quarto capítulo detalha a solução desenvolvida para alcançar os objetivos. Serão abordados tanto o conjunto de dados criado quanto o algoritmo produzido. Eles foram criados para detectar, identificar o seu tipo e sua causa.

Já o capítulo "Experimentos e Resultados" apresenta todos os protocolos dos testes que foram realizados e, em seguida, o resultado de cada um. Os experimentos foram pensados para testar a eficácia da solução comparando-a com outros detectores e com diversas bases e tarefas de AM.

Por fim, o último capítulo conclui todo o trabalho realizado. Serão abordadas as dificuldade encontradas, as contribuições e possíveis melhorias a serem feitas no futuro.

2

FUNDAMENTOS TEÓRICOS

Este capítulo aborda os conhecimentos utilizados ao longo deste trabalho: detecção e desvio de conceito e aprendizado de máquina e alguns de seus algoritmos. Eles são conceitos e teorias que serão utilizados para construir a ideia proposta no mestrado.

2.1 Desvio de Conceito

Em AM, o desvio acontece quando o conceito que foi adquirido no treinamento de um modelo muda com o passar do tempo. Isso significa que as propriedades estatísticas do domínio alvo mudam de modo imprevisível, de forma que, ao longo do tempo, um ruído pode se transformar em um dado relevante (LU et al., 2018).

Para uma definição mais formal, considere um período de tempo [0, t] e um conjunto de dados $S_{0,t} = \{ d_0, \ldots, d_t \}$, no qual $d_i = (X_i, y_i)$ é uma instância, X_i é o conjunto de atributos dessa instância e y_i é o seu rótulo. Se $S_{0,t}$ possui uma distribuição $F_{0,t}(X,y)$, definimos que o desvio ocorre no instante t + 1 se a distribuição $F_{0,t}(X,y)$ é diferente de $F_{t+1,\infty}(X,y)$.

Outra maneira de analisar o desvio é que ele pode ser entendido por meio da probabilidade conjunta de X e y no instante t (LU et al., 2018). Essa probabilidade pode ser decomposta em dois termos: $P_t(X, y) = P_t(X) \times P_t(y \mid X)$, de acordo com a regra do produto. Assim, se $P_t(X) \neq P_{t+1}(X)$ enquanto $P_t(y \mid X) = P_{t+1}(y \mid X)$, então há um desvio, mas ele não afeta os limites de decisão do modelo. Quando isso ocorre, é denominado desvio virtual. Esse cenário pode ser visto na Figura 2.1.1, que mostra os

dados distribuídos no tempo t_0 e depois, quando há o desvio virtual de conceito no tempo t_{t+1}.

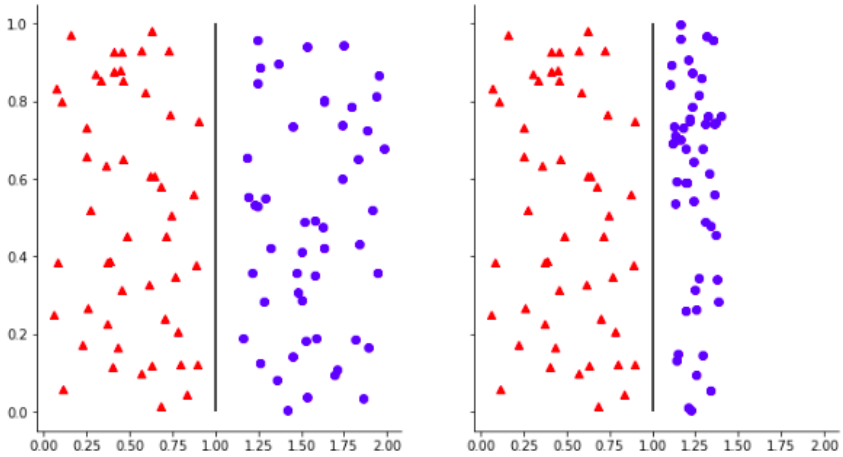

Figura 2.1.1 – Dados distribuídos no tempo t_0 e no t_{t+1}, o qual apresenta desvio virtual.

Se $P_t(y \mid X)$ é diferente de $P_{t+1}(y \mid X)$ enquanto $P_t(X) = P_{t+1}(X)$, então os limites de decisão irão mudar e tem-se um desvio de conceito real. Ele também ocorre quando ambos os componentes da equação mudam entre os dois instantes de tempo.

A Figura 2.1.2 mostra um exemplo de desvio real. Nela, mostra-se a distribuição dos dados nos instantes t_0 e t_{t+1}. Entre eles, acontece o desvio de conceito real e a linha de decisão foi movida no eixo horizontal de valor 1 para 0,6.

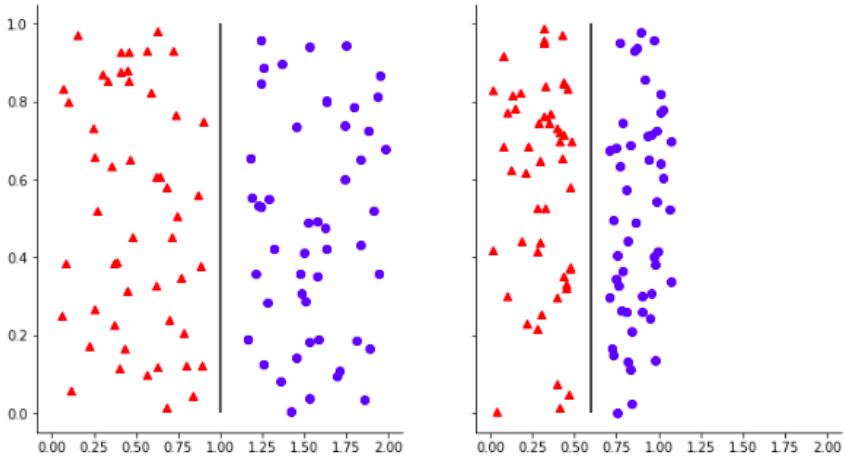

Figura 2.1.2 – Dados distribuídos no tempo t_0 e no t_{t+1}, o qual apresenta desvio real.

Como visto, uma maneira de classificar um desvio é entre virtual e real, mas ele também pode ser classificado de outro jeito: considerando como os dados mudaram

ao longo do tempo. Utilizando esse critério, alguns tipos de desvio de conceito são: abrupto, incremental, gradual, recorrente, blip e ruído (IWASHITA; PAPA, 2018). A seguir, tem-se a explicação de cada tipo e cada um deles pode ser visto na Figura 2.1.3.

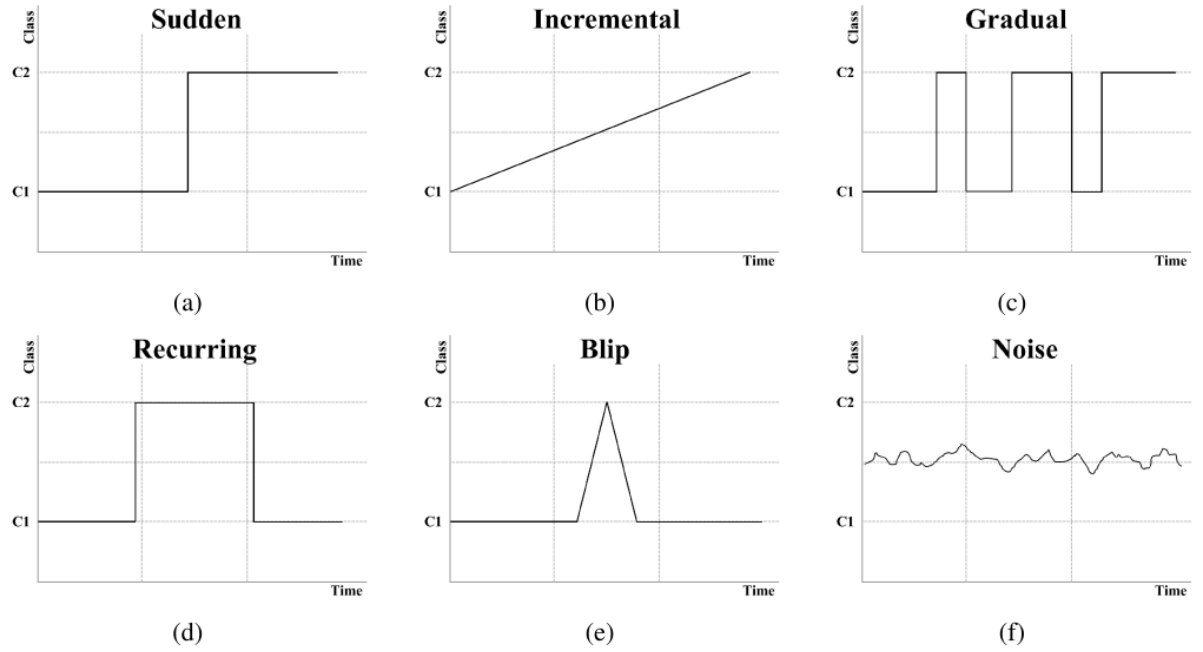

Figura 2.1.3 – Tipos de desvio de conceito: a Abrupto - conceito muda bruscamente; b Incremental - conceito muda devagar e os dados mudam incrementalmente com o tempo; c Gradual - conceito muda devagar e há alternância entre os conceitos até que o novo domine totalmente; d Recorrente - conceito desaparece, mas volta com o passar do tempo; e Blip - um evento raro, onde um conceito aparece e desaparece rapidamente; f Ruído - mudanças aleatórias nas instâncias que podem ser filtradas. Fonte: (IWASHITA; PAPA, 2018).

2.1.1 Desvio de Conceito em Agrupamento

Como mencionado anteriormente, na tarefa de agrupamento, os dados também podem sofrer desvio de conceito. A Figura 2.1.4 ilustra como o desvio pode ser visto no agrupamento. A seguir, o detalhamento de cada gráfico dela:

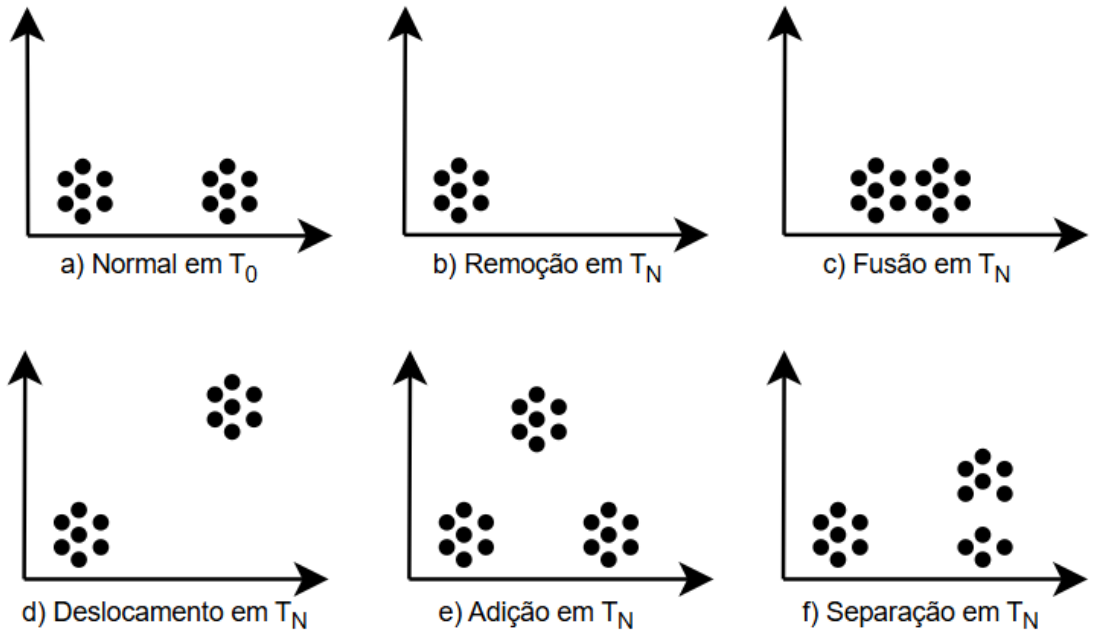

Figura 2.1.4 – Desvios de conceito em agrupamento quando $T \neq 0$: a Normal - distribuição normal, sem desvio de conceito; b Remoção - desaparece um grupo, não surgem mais dados para determinado grupo; c Fusão - dois ou mais grupos se juntam, dados referentes a dois grupos vão se aproximando no espaço; d Deslocamento - um grupo se desloca no espaço; e Adição - aparece um novo grupo; f Separação - um grupo é separado, dados referentes a um determinado grupo vão se separando no espaço.

2.1.2 Detecção de Desvio de Conceito

Analisando esses tipos de desvio de conceito, percebe-se que ele interfere no desempenho de um modelo ao aprender um conceito que não será válido para instâncias futuras. Assim, o modelo poderá realizar sua tarefa erroneamente, classificando ou agrupando incorretamente as instâncias. Por isso, pesquisadores têm estudado como detectar o desvio de conceito e como reagir a ele.

Existem alguns métodos para detectar o desvio de conceito em um fluxo de dados. Eles podem ser separados em três categorias (LU et al., 2018):

- Baseados na taxa de erro: mais comum das três, consiste em algoritmos que verificam se a variação dos erros do classificador é estatisticamente significante, ou seja, se a diferença observada em relação ao que é considerado esperado pelo modelo é suficientemente grande para não ser atribuída às flutuações normais dos dados. A ideia é que um aumento na taxa de erros pode indicar um desvio de

conceito.

- Baseados na distribuição dos dados: uma mudança na distribuição dos dados pode indicar um desvio de conceito. Assim, usa-se alguma métrica, como uma função de distância, para avaliar a semelhança entre os dados novos e os antigos analisando se a diferença entre eles é estatisticamente significante.

- Hipótese múltiplas: aplica algoritmos utilizados nas duas categorias anteriores. A ideia é melhorar a detecção de desvio ao combinar as diferentes estratégias; por exemplo, utiliza-se um algoritmo baseado na distribuição do dados e, quando ele detecta o desvio, utiliza-se um algoritmo baseado na taxa de erro para validar que o desvio de fato ocorreu.

Algoritmos baseados na taxa de erro não se aplicam a todas as situações, pois não é garantido o rótulo verdadeiro para calcular o erro do modelo. Por exemplo, na tarefa de agrupamento, pode-se até usar a taxa de erro, mas ela é substituída por alguma outra medida, como a distância da instância até o centroide do grupo ao qual ela foi atribuída pelo modelo de agrupamento (NAMITHA; KUMAR, 2020). Assim, este trabalho estuda medidas possíveis de se utilizar para detectar desvio de conceito na tarefa de agrupamento sem consultar o rótulo verdadeiro.

2.1.2.1 *Drift Detection Method*

O *Drift Detection Method* (DDM) é um algoritmo para detecção de mudança de conceito *online* e supervisionado (GAMA et al., 2004). Seu princípio fundamental é que o erro cumulativo do classificador, observado sobre uma amostra, diminui conforme novos exemplos da amostra são testados. Um aumento inesperado do erro indica uma mudança de conceito.

O algoritmo funciona em duas fases: alerta e detecção. Na primeira, os exemplos são coletados em uma janela, incluindo os rótulos verdadeiros. Na última, uma mudança de conceito é detectada e um novo modelo é treinado com os exemplos acumulados na janela.

Para identificar se houve mudança de conceito ou se teve início a fase de alerta, o DDM estabelece um intervalo de confiança sobre o erro do classificador. Dada uma instância $(\mathbf{x}_i, y_i)$, a probabilidade de erro do classificador segue uma distribuição de Bernoulli com parâmetro p_i. O erro padrão é estimado por s_i de acordo com a seguinte equação:

$$s_i = \sqrt{\frac{p_i(1-p_i)}{i}}.$$

Durante a inferência, o DDM guarda os menores valores de p_i e s_i, atualizando as variáveis p_{min} e s_{min} sempre que $p_i + s_i < p_{\text{min}} + s_{\text{min}}$. Se a probabilidade de erro atual ultrapassar o intervalo de confiança da probabilidade mínima com respeito a um parâmetro α, isto é, se

$$p_i + s_i \geqslant p_{\text{min}} + \alpha s_{\text{min}},$$

então o DDM inicia um estado de alerta ou identifica um desvio. O parâmetro α é definido de acordo com a confiança desejada para o sistema. Por exemplo, pode-se usar $\alpha = 2$ para nível de alerta e $\alpha = 3$ para identificação da mudança de conceito.

2.1.2.2 *Early Drift Detection Method*

O *Early Drift Detection Method* (EDDM) é parecido com o DDM. A maior diferença entre eles é que o EDDM utiliza as distâncias entre os erros de classificação ao invés do próprio erro (BAENA-GARCIA et al., 2006).

O algoritmo calcula a distância média entre dois erros (p'_i) e o desvio padrão das distâncias (s'_i). São guardados os valores máximos (p'_{max}, s'_{max}) obtidos por:

$$p'_i + 2s'_i$$

Assim como o DDM, o EDDM também possui duas fases: "Fase de Alerta" e "Fase de Detecção". O limiar de cada uma delas é, respectivamente, α e β, e suas características são:

- Fase de Alerta: os exemplos são coletados para serem utilizados caso uma mudança de conceito seja detectada e precise treinar o modelo novamente. A expressão utilizada para definir se o fluxo está nessa fase é

$$\frac{p'_i + 2s'_i}{p'_{max} + 2s'_{max}} < \alpha.$$

- Fase de Detecção: o desvio de conceito é detectado e precisa-se de um novo modelo. Os valores p'$_{\text{max}}$ e s'$_{\text{max}}$ também são descartados. A expressão utilizada para definir se o fluxo está nessa fase é

$$\frac{p'_i + 2s'_i}{p'_{max} + 2s'_{max}} < \beta.$$

O algoritmo começa a verificar as fases depois de 30 exemplos de erro para estimar a distribuição das diferenças entre dois erros consecutivos e compará-los com as futuras distribuições. Assim, $p'_{max} + 2s'_{max}$ representa 95% da distribuição. Os valores utilizados para α e β são, respectivamente, $0{,}95$ e $0{,}90$.

Se a similaridade entre os valores dos exemplos e os valores máximos aumentarem na fase de alerta, os exemplos coletados são removidos e o algoritmo retorna à sua normalidade.

2.1.2.3 *Adaptive Windowing*

ADaptive WINdowing (ADWIN) é um detector de desvio que utiliza janelas deslizantes de tamanho variável (BIFET; GAVALDA, 2007). A janela armazena os dados de entrada e aumenta quando não há mudança de desvio e diminui quando uma mudança é detectada.

A ideia é que, dada uma janela W que armazena os dados que chegam, sempre que duas subjanelas de W exibam estatísticas distintas acima de um valor limite ϵ_{corte}, pode-se concluir que os valores esperados correspondentes são diferentes, e a parte mais antiga da janela W é descartada, ou seja, uma das duas subjanelas de W é removida.

Pode-se representar as entradas do algoritmo como $\delta \in$ (0,1) e uma sequência de valores reais (x_1, x_2, ..., x_t). Cada x_t é gerado de acordo com uma distribuição D_t, podendo haver diferença entre D_t e D_{t+1} sempre que houver uma mudança de conceito.

O ADWIN mantém uma janela deslizante W de tamanho n que contém os valores mais recentes w_i, w_{i-1}, w_{i-2}, ..., w_{i-n+1}. O algoritmo utiliza duas subjanelas que são provenientes de W, W_0 e W_1 e calcula o valor de corte para essa partição:

- Sejam n_0 e n_1 os comprimentos de W_0 e W_1 e n o comprimento de W, então $n = n_0 + n_1$;
- Sejam μ'_{W_0} e μ'_{W_1} as médias dos valores em W_0 e W_1 e μ_{W_0} e μ_{W_1} seus valores esperados;
- σ^2_W é a variância observada dos elementos em W.

Assim, o valor de corte é dado por ϵ_{corte} conforme o seguinte:

$$m = \frac{1}{1/n_0 + 1/n_1}$$

$$\delta' = \frac{\delta}{n}$$

$$\epsilon_{corte} = \sqrt{\frac{2}{m} \cdot \sigma^2_W \cdot ln\frac{2}{\delta'}} + \frac{2}{3m} ln\frac{2}{\delta'}$$

Para detectar o desvio, o teste estatístico verifica se a média observada em ambas as subjanelas é maior do que o limite de corte. Se for maior, o desvio de conceito é detectado. Para esse método funcionar, ele tem que ser utilizado para cada dimensão, sendo que cada atributo consiste uma dimensão. Então, se os dados possuírem m atributos, será necessário usar m janelas.

2.1.2.4 Soma Cumulativa

A Soma Cumulativa (CUSUM) acumula a diferença entre o valor observado e o valor real da variável alvo para detectar desvios de conceito. Assim, para um maior detalhamento, seja o i-ésimo ponto, x_i, CUSUM é calculado como:

$$S_{\text{i}} = max(0, S_{\text{i-1}} + (x_{\text{i}} - K))$$

- S_{i} é a soma até o i-ésimo ponto;
- X_{i} é o valor do i-ésimo ponto;
- K é o valor de referência escolhido pelo usuário; se for muito pequeno, será mais sensível e pode gerar alarmes falsos, e se for grande, será menos sensível e não detectará mudanças pequenas.

A cada novo ponto, a soma é atualizada e, se a soma for maior que um intervalo de decisão definido pelo usuário, o desvio é detectado.

2.1.2.5 Teste Page-Hinkley

O teste tem como objetivo detectar mudanças na média de um fluxo analisando a soma das diferenças dos valores observados com o histórico de médias. Dada a média esperada das observações (μ), por exemplo a média dos dados de um conjunto inicial, e uma observação no tempo t (X_t), a soma cumulativa das diferenças é calculada (GAMA; SEBASTIAO; RODRIGUES, 2013):

$$m_t = \sum_{i=1}^{t} (X_i - \mu - \delta)$$

No qual: δ é a magnitude de mudança permitida (sensibilidade) e μ é a média das observações até o tempo t-1. O valor mínimo sempre será calculado e armazenado. O teste é calculado como:

$$PHT_t = m_t - m_{\text{min}}$$

A cada novo ponto, o valor de PHT_t é atualizado e o desvio de conceito é detectado quando o teste é maior que um valor limite definido pelo usuário.

2.2 Aprendizado de Máquina

O aprendizado de máquina é um campo da inteligência artificial que estuda e desenvolve algoritmos e modelos que aprendem e tomam decisões a partir de dados (MITCHELL, 1997). A maioria dos algoritmos de aprendizado de máquina pode ser dividida entre supervisionados e não supervisionados. A diferença entre eles é a forma como os dados se apresentam para o treinamento.

Algoritmos supervisionados utilizam dados associados a um rótulo. O rótulo indica um conceito associado a cada exemplo e o objetivo é aprender um mapeamento de novos exemplos para o conceito correto. Quando o rótulo é categórico, tem-se classificação; quando o rótulo é numérico, a tarefa chama-se regressão.

Já nos não supervisionados, os dados não tem rótulos e deve-se encontrar uma estrutura nos dados para obter algum conhecimento. Exemplos de tarefas não supervisionadas incluem agrupamento e redução de dimensionalidade.

2.3 Classificação

Classificação é uma tarefa de AM que rotula dados com base nas suas características. A seguir, serão apresentados algoritmos de classificação utilizados neste trabalho.

2.3.1 K-Vizinhos Mais Próximos

Um dos modelos de classificação utilizados neste trabalho é o k-Vizinhos Mais Próximos (*k-Nearest Neighbors* ou k-NN). Ele é um classificador “preguiçoso”, pois não possui uma etapa explícita de treinamento. Em vez disso, ele memoriza os exemplos do conjunto de treinamento, que são denominados protótipos. Na fase de inferência, ele encontra protótipos (vizinhos) mais próximos de um exemplo de consulta e os usa para determinar a classe dessa instância (CUNNINGHAM; DELANY, 2021). O "k"significa o número de vizinhos utilizados para determinar a classe do dado de consulta, que pode ser 1 ou mais.

Desse modo, depois de saber a distância de cada ponto para o de consulta, k pontos mais próximos são escolhidos para determinar a sua classe. Para determinar a proximidade, utiliza-se funções de distância, como, por exemplo, a distância euclidiana, cuja equação é

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2},$$

para dois vetores **x** e **y** que representam duas instâncias.

Depois de calculada todas as distâncias, a classe do novo ponto pode ser inferida. Existem alguns métodos para determinar qual a classe dele. Um deles é o do voto simples, no qual é contabilizada a classe dos k pontos vizinhos e a escolhida é a que está em maior quantidade.

Para melhor exemplificar o funcionamento do kNN é utilizado como exemplo a Figura 2.3.1, que mostra alguns dados em um espaço bidimensional. O ponto triângulo verde representa o dado de consulta, os pontos quadrados vermelhos representam uma classe e os círculos azuis outra e os dois fazem parte do conjunto de treinamento do classificador. O ponto de consulta pode ser classificado como vermelho ou azul dependendo do número "k"e do método de votação escolhido. Se o k for 3, ele é classificado como quadrado vermelho, já se k for 6, ele será classificado como círculo azul.

2.3.2 Naive Bayes

Naive Bayes (WEBB; KEOGH; MIIKKULAINEN, 2010) é um classificador probabilístico que aplica o teorema de Bayes de modo ingênuo (naive), pois pressupõe que os atributos são independentes entre si. Apesar de ser simples, tem um bom desempenho em várias tarefas, como classificação, particularmente em sistemas de recomendação e em processamento de texto com tabelas atributo-valor e sacos de palavras.

Para classificar uma nova instância, o classificador calcula a probabilidade a posteriori para cada classe e seleciona a classe com a maior probabilidade. Seja $\mathbf{x_i}$ uma instância de um conjunto X, a probabilidade posterior de uma classe y dado $\mathbf{x_i}$ é:

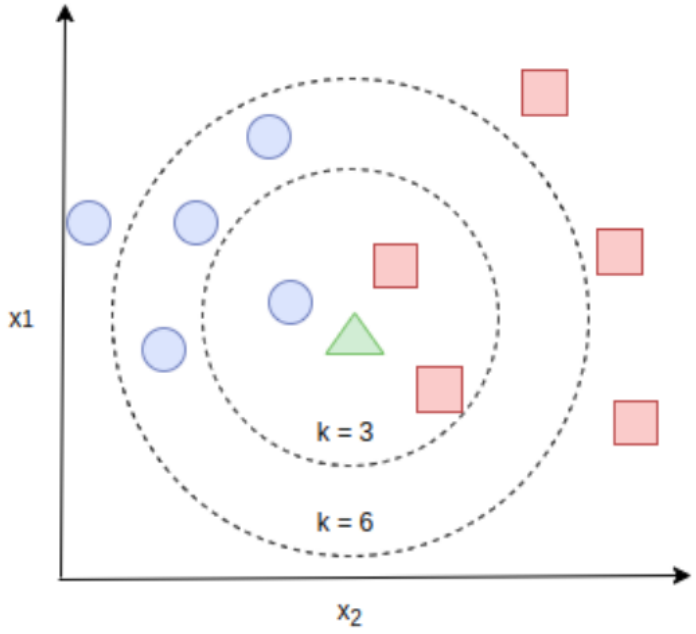

Figura 2.3.1 – Exemplo de pontos em um espaço 2D.

$$P(y|\mathbf{x_i}) = \frac{P(\mathbf{x_i}|y)P(y)}{P(\mathbf{x_i})}$$

A inferência ocorre escolhendo a classe com maior probabilidade, como na fórmula a seguir:

$$\hat{y} = \arg\max_y P(y|\mathbf{x}) = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Uma desvantagem do Naive Bayes é que o produtório das probabilidades $P(x_i|y)$ é obtido com a suposição de que os atributos do exemplo $\mathbf{x} = (x_1, x_2, \ldots, x_M)$ são independentes entre si. Isso permite calcular as verossimilhanças calculando-se a frequência com que cada valor de um atributo aparece nos exemplos de cada classe, mas nem sempre é uma suposição verdadeira. Na prática, os diferentes atributos possuem pelo menos algum grau de co-variância. Quanto maior a dependência entre os atributos, menor será o desempenho esperado do Naive Bayes. Apesar disso, na prática esse impacto não impede o uso do classificador.

2.3.3 AdaBoost

Adaptive Boosting ou *AdaBoost* (FREUND; SCHAPIRE, 1995) é um algoritmo de aprendizado na categoria de *ensembles*, que são conjuntos de modelos cujas saídas são combinadas em detrimento de usar a saída de um único modelo individualmente. A expectativa de um *ensemble* é que a combinação das saídas dos diferentes modelos produza um resultado mais confiável que a de um modelo individual, pois a probabilidade de múltiplos modelos errarem a inferência de um exemplo é menor do que a de um único modelo cometer um erro.

A abordagem do AdaBoost consiste em treinar diferentes classificadores sequencialmente, sempre focando mais nos exemplos que o classificador anteriormente treinado errou. Para isso, os exemplos devem ser ponderados. Os exemplos classificados incorretamente por um modelo têm seus pesos aumentados, ao passo que os exemplos corretamente classificados têm seus pesos reduzidos. O próximo classificador desse processo iterativo será treinado levando esses pesos em consideração.

Os passos do algoritmo podem ser quebrados nas seguintes etapas:

1. Todos os exemplos de treinamento são pesados igualmente;
2. Um classificador é treinado;
3. Calcula-se o seu erro;
4. Ajustam-se os pesos dos exemplos;
5. Calculam-se a contribuição do classificador na inferência final;
6. Refaz os itens 2 a 4 até que um critério de parada seja atingido;
7. Combinam-se os classificadores utilizando a acurácia obtida no treinamento para cada classificador.

2.3.4 Redes Neurais

Aqui serão apresentados, brevemente, os conceitos de rede neural. Ela é utilizada neste trabalho na rede prototípica (Seção 2.3.5), a qual foi a base do desenvolvimento do

detector e classificador do desvio de conceito.

A base de redes neurais é o neurônio artificial, assim chamado por se basear no neurônio biológico. De forma simplificada, um neurônio biológico está "conectado" a uma grande quantidade de outros neurônios, recebendo sinais de alguns e enviando sinais para outros. A comunicação entre os neurônios acontece pela transmissão de compostos químicos chamados neurotransmissores (*e.g.* dopamina e serotonina) e um neurônio que recebe sinais suficientemente fortes de outros neurônios envia um sinal para os neurônios com os quais está associado. Entende-se atualmente que é a partir dessa interação que processos emergentes como o raciocínio surgem. O neurônio artificial simula um neurônio biológico por meio de um vetor de pesos $\mathbf{w} = (w_0, w1, \ldots, w_N)$ e um vetor de entrada $\mathbf{x} = (x_0, x_1, \ldots, x_N)$. As entradas simulam a capacidade de um neurônio receber sinais e os pesos indicam a afinidade do neurônio com as fontes desses sinais. O neurônio "decide" se os sinais recebidos são suficientemente fortes com a seguinte combinação linear (RASCHKA; MIRJALILI, 2017):

$$u = \sum_{i=1}^{n} w_i x_i$$

As entradas $x_1, x_2, \ldots, x_N$ podem estar associadas a outros neurônios ou aos atributos da instância que está sendo processada. A entrada x_0 é um valor chamado viés, único para cada neurônio, que dá uma maior flexibilidade para aprender os padrões dos dados. O resultado dessa soma é usado como entrada de uma função de ativação (φ). Assim, a saída do neurônio é: $y = \varphi(u + b)$.

Dentre as várias funções de ativação existentes na literatura, a Unidade Linear Retificada (ReLU) foi escolhida neste trabalho por suas características, que são: é simples de calcular, ajuda a rede a convergir mais rapidamente e pode desativar neurônios. Essa função funciona comparando o valor da sua entrada (z) com zero. Assim, se o valor de z é menor que zero, então a saída será zero, caso contrário, o valor da saída é z. Ou seja: $g(z) = max(0, z)$.

Para resolver tarefas complexas, pode-se utilizar uma arquitetura de neurônios chamada de Rede Totalmente Conectada (*Fully Connected Network*—FCN). Nela, os neurônios são dispostos em camadas e a saída de cada neurônio de uma camada é

conectada com a entrada de todos os neurônios da camada seguinte. Isso permite que a rede tenha vários caminhos para aprender ao mudar o peso de cada conexão. Assim, a primeira camada recebe os dados e a sua saída percorre várias camadas intermediárias até chegar na última, a qual irá produzir a saída.

Além disso, para evitar *overfitting*, quando um modelo rede não generaliza e "decora" a amostra de treinamento, este trabalho utilizou a técnica de regularização Dropout. Ela funciona durante o treinamento da rede inativando, com uma probabilidade p definida pelo usuário, alguns neurônios da rede a cada iteração. A ideia é que, no treinamento, um neurônio pode compensar o erro de outro, assim, a desativação de um deles faz com que essa "dependência" acabe. Isso faz com que a rede consiga generalizar melhor.

2.3.5 Rede Prototípica

As redes prototípicas propostas por (SNELL; SWERSKY; ZEMEL, 2017) são modelos de classificação projetados para aprender padrões a partir de poucos exemplos por classe, característica, particularmente, vantajosa em cenários onde obter dados rotulados é custoso.

A ideia central delas é mapear as instâncias para um espaço de *embedding* no qual exemplos de uma mesma classe estejam próximos uns dos outros, formando grupos ao redor de um protótipo. O protótipo representa a média dos vetores de *embedding* das instâncias de uma classe. Assim, para classificar uma nova instância, calcula-se a distância entre seu vetor de *embedding* e os protótipos de todas as classes, atribuindo-lhe o rótulo do protótipo mais próximo.

A Figura 2.3.2 ilustra o agrupamento de instâncias em um espaço de embedding. Os pontos de cor laranja, verde e azul representam, cada um, uma classe específica e os pontos pretos, o protótipo dessas classes. Uma nova instância (X) é atribuída à classe laranja visto que o seu protótipo está mais próximo do X do que os outros.

A transformação de uma instância para o espaço de *embedding* é realizada por uma rede neural. Para encontrar um *embedding* que permita separar o espaço de modo

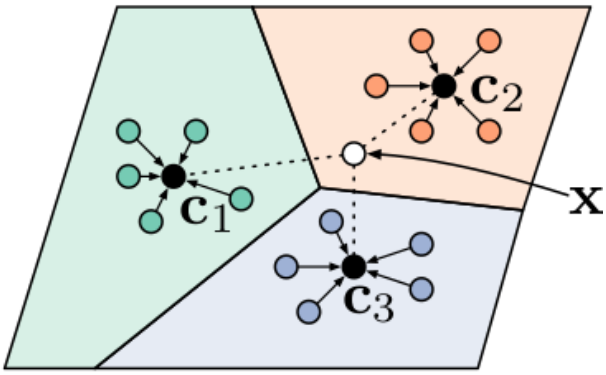

Figura 2.3.2 – Representação de uma rede prototípica extraído—Fonte: (SNELL; SWERSKY; ZEMEL, 2017)

que os protótipos particionem bem o espaço de decisão, o mecanismo de prototipagem é incorporado à função de custo (*loss function*) no treinamento da função de *embedding*.

O treinamento das redes prototípicas envolve dois subconjuntos de dados:

- Conjunto de suporte (Sk): usado para calcular os protótipos de cada classe;
- Conjunto de consulta (Qk): usado para avaliar a função de custo com base na distância entre as instâncias e seus protótipos.

Em cada episódio de treinamento da rede que gera o *embedding*, dois subconjuntos dos dados de treinamento são selecionados aleatoriamente para cada classe. O primeiro, denominado conjunto de suporte (S_k), é usado para encontrar um centroide c_k para uma classe. O segundo, denominado conjunto de consulta (Q_k), é utilizado para o cálculo da função de custo, que envolve a distância entre cada exemplo em Q_k com seu centroide c_k.

O custo é calculado utilizando a distância entre as instâncias de consulta e os protótipos, sendo posteriormente propagado para ajustar os pesos da rede que gera o *embedding*. O Algoritmo 1 apresenta as etapas detalhadas do treinamento.

Na fase de inferência, os exemplos são classificado pela regra do vizinho mais próximo. Primeiro a nova instância é transformada para o espaço de *embedding* e então ela é classificada de acordo com o protótipo (centroide) mais próximo.

Algoritmo 1 Treinamento de um episódio da rede prototípica para calcular a loss. N é o número de exemplos no conjunto de treinamento, K é o número de classes, N_C é o número de classes por episódio, N_S é o número de exemplos de suporte por classe, N_Q é o número de exemplos de consulta por classe. AmostrasAleatórias(S, N) significa um conjunto com N elementos escolhidos aleatoriamente do conjunto S

Entrada: Conjunto de treinamento $D = \{(x_1, y_1), ..., (x_N, y_N)\}, caday_i \in 1, ..., K$
Saída: Custo J para um episódio de treinamento

```
V ← seleciona aleatoriamente N_c classes de {1, ..., K}
para k ← 1 até N_C faça
    S_k ← N_S exemplos de suporte amostrados aleatoriamente de D_{V_k}
    Q_k ← N_Q exemplos de consulta amostrados aleatoriamente de D_{V_k} \ S_l
    c_k ← (1/N_C) Σ_{(x_i,y_i)∈S_k} fφ(x_i)        ▷ Calcula o protótipo com o conjunto de suporte
fim para
J ← 0
para k ← 1 até N_C faça
    para cada (x, y) ∈ Q_k faça                    ▷ Atualiza a função de custo.
        J ← J + (1/(N_C N_Q)) [ d(f_φ(x), c_k) + log Σ_{k'} exp(−d(f_φ(x), c_k)) ]
    fim para
fim para
```

2.4 Agrupamento

Agrupamento é a tarefa de distribuir elementos em grupos baseado nas suas propriedades. Os elementos do mesmo grupo devem ser similares entre si e diferentes dos elementos dos outros grupos. Assim, o agrupamento quer minimizar a distância intragrupo e maximizar a distância intergrupo (ZUBAROĞLU; ATALAY, 2021; SAXENA et al., 2017).

Os algoritmos de agrupamento podem ser categorizados de acordo com o seguinte:

- Hierárquico: cada elemento pertence a um grupo e cada par de grupos está contido em um grupo comum. Esse tipo de agrupamento frequentemente é visualizado na forma de um dendograma, um diagrama que remete uma árvore binária e mostra a hierarquia entre os diferentes grupos. Os algoritmos hierárquicos também podem ser divididos em dois: aglomerativos e divisivos. Os algoritmos aglomerativos começam com a suposição de que cada instância é um grupo em si e mesclam as instâncias para criar grupos. Por outro lado, algoritmos divisivos começam

assumindo que existe um único grupo com todos os dados e dividem os grupos em grupos menores. Os algoritmos dessa classe têm alta complexidade e são sensíveis aos *outliers*;

- Baseado em distância: divide as instâncias de dados em um número predefinido de grupos, com base na distância aos centroides dos grupos. Geralmente o algoritmo é de fácil implementação, mas o número de grupos deve ser predefinido;

- Baseado em densidade: utiliza microgrupo, que é um conjunto de instâncias de dados que estão muito próximas umas das outras. Um resumo do microgrupo é mantido em um vetor de características. Em seguida, esses microgrupos são mesclados e formam grupos de acordo com os conceitos de alcançabilidade e conectividade de densidade. Se a distância entre dois microgrupos for menor ou igual à soma de seus raios, então eles são diretamente alcançáveis pela densidade. Se quaisquer dois grupos adjacentes em um conjunto de microgrupos são diretamente alcançáveis por densidade, então o conjunto de microgrupos é também alcançável por densidade. Todos os microgrupos que são alcançáveis por densidade uns aos outros são conectados por densidade. Algoritmos dessa classe são robustos ao ruído e são capazes de encontrar grupos de formato arbitrário e detectar o número de grupos. Entretanto, devem ser escolhidos vários parâmetros;

- Baseado em modelo: utiliza o modelo de distribuição de dados que melhor se ajusta aos dados de entrada. É robusto ao ruído, mas seu desempenho depende do modelo selecionado.

Utilizar o agrupamento em um fluxo de dados é diferente de um agrupamento quando se tem todos os dados à disposição. Alguns desafios enfrentados são: o dado só pode ser lido uma vez, por questão de armazenamento e processamento; em uma certa ordem; e devem ser processados rapidamente. Uma solução para estes desafios que alguns algoritmos utilizam é apenas guardar as instâncias que melhor representam o conjunto dos dados. Além disso, dependendo da técnica utilizada, pode-se guardar apenas os metadados do agrupamento ou não guardar nenhum dado. Assim, o algoritmo pode analisar rapidamente os dados que chegam.

Outro desafio é o desvio de conceito que pode acontecer no agrupamento quando grupos previamente formados desaparecem ou novos grupos surgem. O limite de decisão dos grupos no espaço também podem sofrer alterações. Quando isso ocorre tem-se um desvio real.

2.4.1 K-Means

O algoritmo K-means realiza a tarefa de agrupamento, é simples de ser implementado e é computacionalmente eficiente (ARTHUR; VASSILVITSKII et al., 2007). Ele funciona escolhendo no espaço k pontos que irão representar os centroides dos k grupos. Depois que são selecionados, é realizado o cálculo da distância e cada elemento é atribuído ao centroide mais próximo Depois, os centroides são atualizados para refletir as mudanças que possivelmente ocorreram no grupo. Para encontrar a melhor solução, o algoritmo deve repetir essa etapa inúmeras vezes.

O K-means pode ser dividido em alguns passos:

- escolha inicial de k centroides (*e.g.*, k exemplos aleatoriamente selecionados ou k combinações aleatórias de coordenadas do espaço de decisão);
- agrupar os conjuntos de pontos que são mais próximos de cada centroide utilizando uma função de distância, como, por exemplo, a distância euclidiana;
- calcular o novo centroide fazendo o cálculo da média, ou seja, somando todos os pontos e dividindo pelo total;
- repetir os dois passos anteriores até que as mudanças nos centroides não sejam significativas.

2.4.2 Clustream

O k-means é mais utilizado em ambientes estáticos. Para fluxos de dados que evoluem, (AGGARWAL et al., 2003) apresenta o algoritmo Clustream. Ele é dividido em duas fases: microagrupamento (online) e macroagrupamento (offline). O primeiro tem como

objetivo guardar a informação de uma maneira compacta, já a última agrupa os micro grupos para ter uma representação mais significativa dos dados.

O algoritmo começa na fase online e o usuário deve definir o número de micro grupos k e um tempo t para determinar se o micro grupo é relevante. Assim, para cada novo ponto x_t as etapas nessa parte são:

- encontrar o micro grupo mais perto do ponto;
- atualizar as estatísticas desse micro grupo;
- criar um novo micro-grupo, caso x_t esteja muito longe dos demais micro grupos;
- remover micro grupos que não foram atualizados em um tempo maior do que t.

Depois, o algoritmo entra na fase offline e o usuário deve definir o número de macro grupos k. Assim, para cada execução de agrupamento nessa etapa, ele usa algum algoritmo de agrupamento para agrupar os micro grupos e formar os macro grupos.

2.4.3 Silhueta

A silhueta (ROUSSEEUW, 1987) surge como um método para medir quão similar uma instância é em seu próprio grupo comparada com os outros. Desse modo, ela é mede a qualidade do agrupamento. Também pode ser usada, via tentativa e erro, como uma forma de identificar o número ideal de grupos em algoritmos nos quais o número de grupos é um hiperparâmetro que precisa ser especificado pelo usuário.

O que essa métrica faz é analisar o quão próximos os dados do mesmo grupo estão entre si (coesão) e o quão distinto um elemento é dos elementos de outros grupos (separação). A fórmula da silhueta é:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\},}$$

no qual:

- $a(i)$ é a média da distância do ponto i para todos os outros pontos do mesmo grupo;

- $b(i)$ é a média mínima das distâncias do ponto i para todos os pontos de grupos diferentes do seu.

A diferença $b(i) - a(i)$ reflete a qualidade do i-ésimo ponto. O denominador normaliza a silhueta para o intervalo $[-1, 1]$. Um valor exatamente igual a -1 ou 1 significa que o ponto está extremamente mal-colocado ou extremamente bem-colocado (respectivamente). Já uma silhueta zero indica que o ponto está na fronteira de dois grupos. A situação ideal e realista é para valores próximos de 1.

2.5 Avaliação de Modelos

As métricas de avaliação de modelos de AM são utilizadas para mensurar o desempenho do modelo sob diferentes aspectos e permite realizar comparações entre modelos. É importante entender as várias métricas, pois cada uma tem um objetivo diferente.

Para isso, usa-se a matriz de confusão. Ela é composta por uma tabela (Tabela 1) que avalia o desempenho de um classificador comparando os valores alvo verdadeiros com os valores inferidos pelo modelo. Ela é composta pelos seguintes componentes:

- Verdadeiro Positivo (TP): instâncias positivas corretamente inferidas;
- Verdadeiro Negativo (TN): instâncias negativas corretamente inferidas;
- Falso Positivo (FP): instâncias positivas inferidas incorretamente;
- Falso Negativo (FN): instâncias negativas inferidas incorretamente.

	Valores Inferidos	
Valores Verdadeiros	Verdadeiro Positivo (TP)	False Negativo (FN)
	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Tabela 1 – Matriz de Confusão

A matriz é preenchida numericamente, com cada campo (TP, FP, FN e TN) indicando o número de ocorrências de acordo com o modelo e um conjunto de teste

rotulado. Com base nos elementos dessa matriz é possível calcular diversas métricas de desempenho do modelo sobre os dados do conjunto.

2.5.1 Acurácia

A Acurácia calcula o quanto as instâncias foram corretamente inferidas pelo número total de instâncias. Pode ser uma boa medida quando os dados são balanceados. Sua fórmula é:

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN}$$

2.5.2 Precisão

A Precisão calcula o quanto que as instâncias que foram inferidas com uma classe realmente pertencem a ela. Ela pode ser utilizada quando se pretende diminuir o número de falso positivos. Sua fórmula é:

$$\text{Precisão} = \frac{TP}{TP + FP}$$

2.5.3 Revocação

A Revocação calcula o quanto que as instâncias de uma classe foram corretamente inferidas como sendo pertencendo a ela. Ela pode ser utilizada quando se pretende diminuir o número de falso negativos. Sua fórmula é:

$$\text{Revocação} = \frac{TP}{TP + FN}$$

2.5.4 F1-Score

O F1-Score é uma medida que relaciona a precisão e a revocação tentando balancear as duas informações. Sua fórmula é:

$$\text{F1-Score} = 2 \times \frac{\text{Precisão} \times \text{Revocação}}{\text{Precisão} + \text{Revocação}}$$

2.6 Considerações Finais

Neste capítulo, foi apresentado o problema que este trabalho mitiga: desvio de conceito e suas características. Além disso, foram mostrados detectores de desvio e conceitos de aprendizado de máquina. Detectores podem ser usados para auxiliar um modelo de AM ou ele mesmo pode ser um algoritmo de AM. A fim de comparar os algoritmos no final deste trabalho, algumas métricas de avaliação de modelos foram descritas.

3

TRABALHOS RELACIONADOS

Este capítulo aborda diferentes trabalhos que ajudaram a construir a ideia central deste mestrado. São pesquisas referentes à detecção de desvio de conceito em tarefas supervisionadas e não supervisionadas, identificação das características do desvio, agrupamento e suas características, que podem ser utilizadas para detectar a mudança de conceito.

3.1 Desvio de Conceito

Há muitos algoritmos que detectam o desvio de conceito (Seção 2.1) e, que para tal atividade, utilizam dados rotulados. Entretanto, dados sem rótulos fazem parte do mundo real, especialmente em fluxos, nos quais a mudança de conceito é um problema importante, e rotular os exemplos pode ser muito caro ou impossível devido à velocidade com que os dados chegam no fluxo. Assim, algoritmos de detecção que não dependem de rótulos verdadeiros podem ser vantajosos. Algumas vantagens são: diminui o custo de uso do algoritmo; não requerem manutenção manual, podendo ser utilizados para realizar um monitoramento contínuo do modelo; privacidade, caso rotular envolva trabalhar com dados sensíveis.

Apesar de todas essas vantagens, existem alguns desafios. Sem o rótulo verdadeiro, detectar e confirmar o desvio de conceito fica mais difícil, principalmente quando o desvio acontece apenas no rótulo. Ruído, anomalias e *outliers* podem atrapalhar nesse processo.

Existem várias técnicas que utilizam agrupamento para detectar desvio de conceito, como: avaliar a distribuição dos dados, analisar o limite de decisão dos grupos, combinar com métricas estatísticas, realizar uma análise sequencial etc.

Um exemplo de trabalho baseado em distribuição de dados é apresentado em (PEÑA et al., 2021). Nessa abordagem o classificador algoritmo KNN é usado para para determinar se houve mudança de conceito; em particular, o algoritmo identifica novidades, isto é, se a mudança de conceito é o aparecimento de uma nova classe ou grupo. Esse método começa estabelecendo uma base, um conjunto de dados que não tem desvio e serve de referência ao comparar com os dados que irão chegar. Assim, as novas instâncias são comparadas com essa base utilizando o classificador KNN.

Na primeira etapa do algoritmo, uma janela com as novas instâncias é formada e cada exemplo é rotulado como "Novo", indicando que esse exemplo potencialmente pertence a um novo grupo. Na segunda etapa, o KNN infere a classe de uma única instância, que pode ser classificada como uma das classes ou grupos já existentes ou permanece como "Novo". Ao final dessa etapa, independentemente de como foi classificado, o ponto escolhido volta a ter a classe "Novo". Esse processo se repete para cada ponto na janela e o algoritmo anota quantas instâncias foram classificadas como uma das classes ou grupos já existentes e quantas permaneceram como "Novo". Na última etapa, depois de classificar toda a janela, é verificado se a proporção das instâncias que foram classificadas como "Novo", ao invés de uma das classes definidas no conjunto base, é maior que um limite definido pelo usuário. Se sim, então houve uma mudança de conceito.

Já o algoritmo CUSUM (Seção 2.1.2.4), utilizado em (NAMITHA; KUMAR, 2020), detecta mudança de conceito analisando sequencialmente os dados que chegam. Ele funciona acumulando a soma dos erros e, para isso, precisa dos rótulos. Assim, foi necessário fazer uma modificação. Desse modo, já com os grupos e seus centroides definidos, o erro considerado foi a distância entre o novo ponto e o centroide mais próximo.

Além disso, o usuário deve definir um valor para o raio do grupo. Ele serve para o cálculo do erro, pois se o ponto estiver dentro do grupo, então o erro é zero; se não, o

erro é o valor da distância. Desse modo, o acúmulo do erro é utilizado para detectar se houve ou não mudança de conceito.

Outro exemplo é o algoritmo apresentado em (NAMITHA et al., 2020) que usa o Clustream (Seção 2.4.2) para agrupar os dados e o PHT (Seção 2.1.2.5) para detectar mudança de conceito. Assim como o algoritmo anterior, ele faz uma mudança no detector para funcionar no ambiente não supervisionado. O PHT monitora a média das distâncias do ponto até o centroide. Se o centroide for maior que o limite definido, então o desvio é detectado.

Esses algoritmos não usam rótulos, mas há alguns que utilizam, como o "*Task-sensitive drift detection framework*" (CASTELLANI; SCHMITT; HAMMER, 2021) e CADE ("*Contrastive Autoencoder for Drifting detection and Explanation*") (YANG et al., 2021). Ambos utilizam um autocodificador contrastivo treinado para codificar a entrada para um espaço latente. O objetivo é, que nesse espaço, as instâncias possam ser melhor separadas. Assim, no treinamento, os dados são transformados para esse espaço e os centroides são calculados para cada grupo. Depois, na inferência, cada nova instância também vai ser transformada para esse espaço e a distância euclidiana é calculada do novo ponto para os centroides. A diferença entre eles ocorre na detecção de desvio de conceito, no qual cada um utiliza um método diferente.

Para detectar o desvio, o primeiro, depois de encontrar os centroides utilizando os dados de treinamento, usa uma janela de referência que é formada pelas distâncias entre os dados de treinamento e seus centroides. Assim, as estatísticas (média, variância, etc) dessa janela são utilizadas para comparar com as das novas janelas que chegam. O desvio é detectado quando essa comparação apresenta um valor maior que um limite definido pelo usuário.

Já o CADE usa outra abordagem. No treinamento, é definido um tamanho de raio a cada centroide baseada nas distâncias de cada ponto ao seu centroide. Desse modo, se um novo dado não está inserido dentro de algum grupo formado pelo centroide e seu raio, então houve desvio de conceito.

3.2 Detecção do Tipo de Desvio de Conceito

A maioria dos trabalhos foca apenas na detecção do desvio de conceito, mas existem alguns trabalhos que também identificam o tipo de desvio de conceito.

3.2.1 Meta Detector de Desvio Ativo

Nos poucos artigos no qual o experimento é realizado para detectar o tipo de desvio de conceito, tem-se o artigo (YU et al., 2021). Ele descreve um algoritmo, Meta Detector de Desvio Ativo (Meta-ADD), que é capaz de descobrir o tipo de desvio de conceito que ocorreu em um fluxo de dados baseado na taxa de erro do classificador.

A ideia é que uma mudança na distribuição dos dados, geralmente, acarreta em uma mudança também na taxa de erro, sendo que diferentes tipos de mudança de conceito têm diferentes impactos nessa taxa. Então, ela pode ser utilizada para decidir o tipo de desvio que ocorre. Assim, é esperado que a taxa de erro do classificador para um desvio abrupto aumente rápido em um curto período de tempo; para um desvio gradual, a taxa varie no tempo; e para o incremental, a taxa aumente incrementalmente sobre um período de tempo.

O artigo trata de três tipos de desvio: gradual, incremental e abrupto. Entretanto, também utiliza-se dados sem desvio para representar um fluxo sem mudança. Então, tem-se a vantagem de não só o desvio é detectado, mas também o seu tipo.

Assim, o algoritmo cria um meta-detector utilizando a rede prototípica (Seção 2.3.5). Primeiro, um classificador de interesse (*e.g.*, Naive Bayes) é treinado e empregado para classificar dados de um fluxo. Os erros do classificador são acumulados em uma janela denominada "janela interna". Quando n pontos são acumulados e preenchem totalmente a "janela interna", a média dos erros é calculada e começa-se a preencher uma "janela externa". Quando um número suficiente de pontos é alcançado e a "janela externa" está cheia, ela é mapeada para um espaço de *embedding* pela rede prototípica. Essa "janela externa" é calculada de acordo com o centroide mais próximo (protótipo) no espaço de *embedding* gerado pela rede prototípica.

Desse modo, os meta-atributos gerados são os erros cometidos pelo classificador

em diferentes "janelas internas" e eles são a entrada da rede prototípica. Ela irá achar um ponto em um novo espaço que melhor representa cada tipo de desvio. Na inferência, um novo ponto para consulta de classe irá passar pela rede e será transformado em um *embedding*. Ele, por sua vez, é utilizado para calcular a distância para cada protótipo de tipo de desvio. No final da detecção, o protótipo mais perto representará a classe desse ponto (Figura 3.2.1).

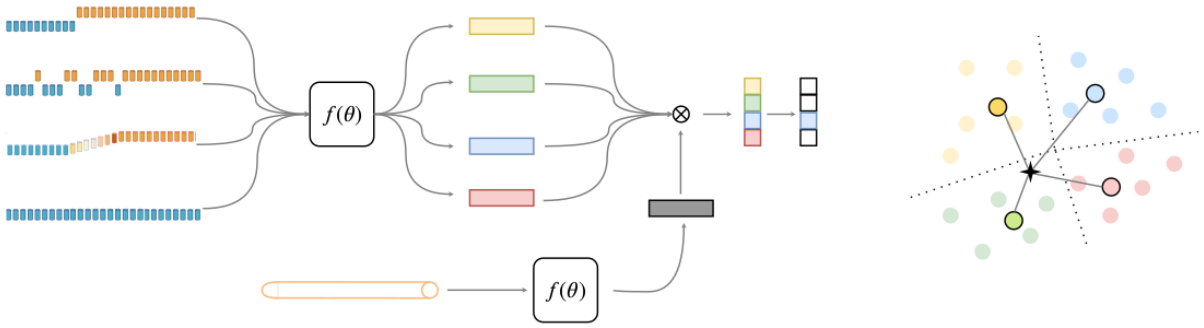

Figura 3.2.1 – Rede prototípica classificando o tipo de desvio de conceito—Fonte (YU et al., 2021).

3.2.2 Janela Deslizante

(GUO et al., 2022) é outro trabalho nessa área. Ele divide o desvio de conceito em duas categorias: desvio de conceito transitório (DCT) e desvio de conceito progressivo (DCP). Compõe o DCT os tipos abrupto, recorrente e blip. Já o DCP é composto por gradual e incremental. O algoritmo trabalha com duas janelas e consiste em três passos: detecção, crescimento e rastreamento.

Na etapa de detecção, duas janelas, sendo uma fixa (W_A) e uma deslizante (W_B), são utilizadas para detectar o desvio. Um modelo é treinado e infere a classe de N novos dados, e, assim, a acurácia das janelas é calculada. A cada novo ponto, a janela W_A não é atualizada, apenas W_B e sua nova acurácia é calculada. Assim, o desvio é detectado quando a acurácia dos dados de W_B sobre a acurácia dos dados de W_A é maior que um limite definido pelo usuário.

Na etapa de crescimento, uma janela deslizante W_R é utilizada. Ela começa a

partir do ponto que foi detectado o desvio e desliza com um passo s definido pelo usuário. A cada passo, a variância das acurácias dos dados dessa janela é calculada e comparada com um limite δ também definido pelo usuário. Quando a variância é menor que o limite, essa etapa para. Se apenas com um passo foi suficiente para chegar nesse limite, então o desvio é DCT, se não, ele é DCP.

Na última etapa, a de rastreamento, a janela W_A desliza até o final do desvio achado, ou seja, até o final da janela W_R. Isso é para comparar o desempenho do classificador antes de depois do desvio. Assim, ao analisar a taxa de acurácia da janela W_B sobre W_A a medida que ela desliza, o algoritmo identifica a qual subcategoria o desvio pertence: abrupto, blip, recorrente, incremental e gradual.

3.3 Considerações Finais

Este capítulo explicou trabalhos que retratam sobre a detecção de desvio de conceito, alguns utilizando rótulos e outros não; alguns são utilizados em classificação e outros em agrupamento. Para o último, pode-se detectar com o auxílio de métricas de agrupamento. Com essas informações, percebeu-se que há uma lacuna para trabalhos que façam mais do que apenas detectar o desvio de conceito, ou seja, que também identifique o tipo de desvio e a sua causa.

A Tabela 2 mostra as características dos detectores de desvios apresentados os comparando em relação a como detectam o desvio, se o identificam e se relata a sua causa. O Classificador de Desvio de Conceito (CDC), algoritmo desenvolvido neste trabalho, também foi incluído nessa tabela. Pode-se perceber que o CDC é o mais completo dos algoritmos visto que dá mais informações sobre o desvio de conceito ocorrido.

Diante disso, o CDC foi desenvolvido a partir da análise de métricas de agrupamento para escolher a melhor opção para obter informações mais completas do desvio, como o tipo e a causa.

	Como	Tipo	Causa
(PEÑA et al., 2021)	Distribuição dos Dados	x	x
(NAMITHA; KUMAR, 2020)	Distribuição dos Dados	x	x
(NAMITHA et al., 2020)	Distribuição dos Dados	x	x
(CASTELLANI; SCHMITT; HAMMER, 2021)	Distribuição dos Dados	x	x
(YANG et al., 2021)	Distribuição dos Dados	x	x
(YU et al., 2021)	Erro do Modelo	✓	x
(GUO et al., 2022)	Erro do Modelo	✓	x
CDC	Distribuição dos Dados	✓	✓

Tabela 2 – Comparação das características dos detectores de desvio

4

PROPOSTA DE TRABALHO

Este capítulo apresentará a solução desenvolvida para identificar o tipo de desvio de conceito, sua causa e sua generalização para outros domínios.

Embora o desvio de conceito seja uma área que vem sendo estudada há bastante tempo, ainda há espaço para estudos que abordam perspectivas pouco debatidas. Tendo isso em mente, a ideia apresentada a seguir busca analisar a mudança de conceito utilizando o método do agrupamento, explorando as suas características e obtendo informações que auxiliem o usuário a decidir a melhor maneira de adequar o modelo utilizado para atingir o seu objetivo.

4.1 Proposta

A ideia desenvolvida neste trabalho consiste em analisar métricas de agrupamento para detectar o desvio de conceito e classificá-lo de acordo com suas características. Além disso, será verificado se o modelo treinado em um domínio pode ser utilizado em outro.

Dentre os tipos de desvio comentados na Seção 2.1, foram considerados neste estudo o desvio gradual e o abrupto. Isso porque eles possuem características consideravelmente distintas.

Além do tipo de desvio, o algoritmo desenvolvido identifica o que aconteceu com o grupo, ou seja, se um novo grupo foi adicionado, se um grupo foi dividido ou se dois grupos foram mesclados. A remoção não foi incluída porque não pode ser identificada pela abordagem proposta, dado que o CDC utiliza as distâncias do ponto

para o centroide e essa métrica deixa de existir quando os pontos do grupo desaparecem.

Representando o que foi discutido na Seção 2.1 sobre o tipo e a causa do desvio de conceito, a saída do algoritmo poderá ser uma das 9 classes:

- sem mudança (SM);
- adição abrupto (AA) ou gradual (AG);
- separação abrupto (SA) ou gradual (SG);
- fusão abrupto (FA) ou gradual (FG);
- deslocamento (DA) abrupto ou gradual (DG).

As próximas seções vão detalhar os dados e o algoritmo.

4.2 Dados

4.2.1 Dados Sintéticos

Existem muitas bases sintéticas para treinar e testar modelos na tarefa de classificação. Entretanto, para a tarefa de agrupamento, o número de conjuntos sintéticos é reduzido e, por isso, bem como para ter-se um maior controle sobre os dados de treinamento, foi necessário gerar uma base própria, viabilizando o aprendizado do modelo criado para diferentes padrões.

Assim, para criar um banco de dados sintético foi utilizada a função `make_blobs` da biblioteca Sklearn (PEDREGOSA et al., 2011). Ela cria grupos diferentes no espaço de acordo com os parâmetros passados pelo usuário: quantidade de exemplos, de grupos e de atributos, dentre outros. Desse modo, foram formados vários fluxos de dados, cada um formando uma janela com 200 exemplos. Essa função foi executada várias vezes para formar o conjunto de dados. Cada execução foi realizada com uma semente aleatória diferente para produzir fluxos distintos. Além disso, utilizou-se a silhueta (Seção 2.4.3) para formar um conjunto de fluxos que possuem dois grupos coesos e separáveis. Para isso, após a formação do fluxo foi executado o K-Means com $k = 2$; se

o valor médio de silhueta fosse inferior a 0,85, o fluxo era desprezado e um novo fluxo era gerado.

A seguir, está a descrição de como foi formado cada conjunto de desvio de conceito:

- Normal: janela com 200 elementos contendo sempre o mesmo conceito;
- Abrupto: janela com 200 elementos contendo um novo conceito;
- Gradual: utiliza dois conceitos, um atual e um novo, que são misturados de modo que o novo vai aparecendo aos poucos e o atual vai diminuindo até restar apenas o novo. Neste contexto, é utilizada a distribuição binomial para formar a janela, que começa com a maior probabilidade de ter o conceito atual e vai diminuindo ao longo da janela, aumentando a probabilidade do novo conceito.

Além do tipo de desvio, também é analisado o que aconteceu com o grupo:

- Normal: a distribuição dos dados permanece a mesma;
- Adição: um novo grupo é adicionado;
- Separação: um grupo é separado, formando dois grupos menores;
- Fusão: dois grupos se juntam, formando um grande grupo;
- Deslocamento: um dos grupos se desloca no espaço, se afastando da posição inicial das instâncias desse grupo.

Assim, para gerar dados com esses movimentos de grupos, utilizou-se um conjunto base formado pela função make_blob com parâmetro de número de grupos igual a 2. Depois, esse conjunto foi modificado da seguinte maneira:

- Adição: execução do make_blob com parâmetro de número de grupos igual a 3 e mesma semente da base;
- Separação: um dos dois grupos é selecionado e os seus dados são deslocados em duas direções opostas no espaço, formando dois grupos;

- Fusão: os dois grupos se juntam no espaço ao mover seus dados para mais perto um do outro;
- Deslocamento: um dos dois grupos é escolhido e seus dados são deslocados no espaço ficando distante da sua posição inicial;

Neste contexto, o total de classes geradas é nove. Os diferentes tipos de fluxo também diferem no número total de grupos. Todos os fluxos começam com um conceito no qual os dados são mais naturalmente separados em dois grupos e essa quantidade pode variar, dependendo do tipo de mudança, conforme a Tabela 3.

Classe	Quantidade Final de Grupos
SM	2
AA	3
AG	3
SA	3
SG	3
FA	1
FG	1
DA	2
DG	2

Tabela 3 – Quantidade final de grupos em cada desvio de conceito

Para um melhor entendimento das classes, a Figura 4.2.1 mostra um exemplo, visual e em duas dimensões, de cada uma delas. Cada figura deve ser comparada com a primeira, na qual tem-se dados dispostos em dois grupos. Nas demais, pode-se observar divergências do primeiro caso. Nela, pode-se ver que nos casos de desvio abrupto a mudança ocorre mais rapidamente e no gradual mais lenta. Isso pode ser percebido com a diferença de densidade dos pontos entre os casos abrupto e gradual em um mesmo tipo.

Esses são os dados para treinar o modelo. Já para os testes, foram utilizados outros conjuntos sintéticos. Os conjuntos escolhidos podem ser encontrados em (SOUZA et al., 2015). Tratam-se de problemas de classificação binária, consequentemente, dados naturalmente distribuídos em dois grupos. Além disso, os fluxos têm anotações dos pontos de desvio. Eles são referenciados como 1CDT, 2CDT, 1CHT e 2CHT (Figura 4.2.2).

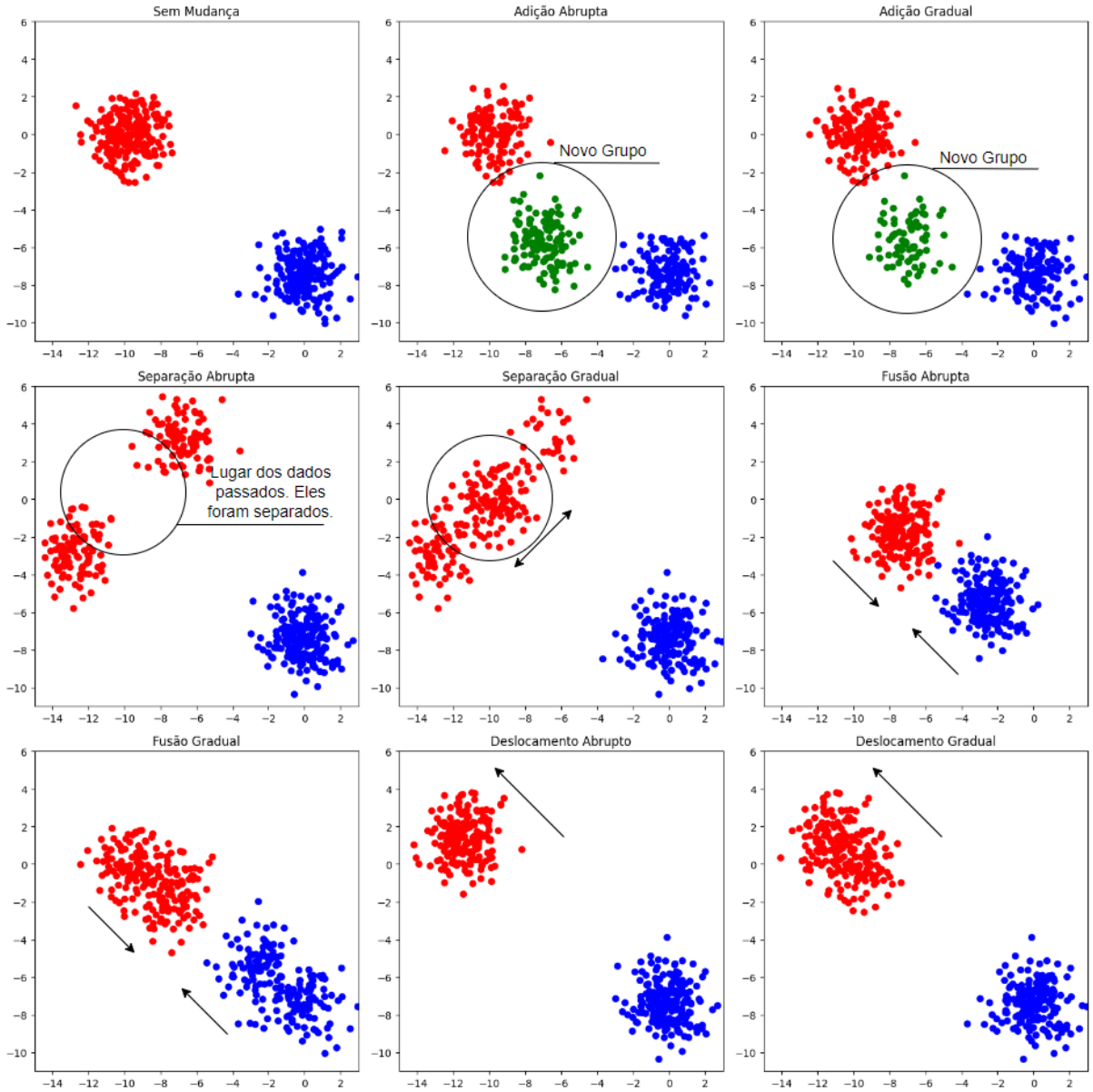

Figura 4.2.1 – Mudanças de conceito em agrupamento. A primeira figura mostra o conceito original. As demais mostram um novo conceito após t unidades de tempo (igual para todos os casos). Observe que as mudanças abruptas apresentam maior diferença para o conceito original, pois o novo conceito se estabelece mais rapidamente.

Nos dois primeiros, o movimento do grupo é na diagonal; a diferença é que em um conjunto só um grupo se desloca no espaço e, no outro, são os dois. Os dois últimos são similares aos dois primeiros; a diferença é no deslocamento que ocorre na horizontal. As 4 bases têm 16.000 exemplos cada uma e o desvio ocorre a cada 400 pontos.

A Figura 4.2.3 mostra em mais detalhes o desvio ocorrendo nos conjuntos: 1CDT, 2CDT, 1CHT e 2CHT. Nela, os centroides (estrela verde) fixos nas 4 janelas de tempo.

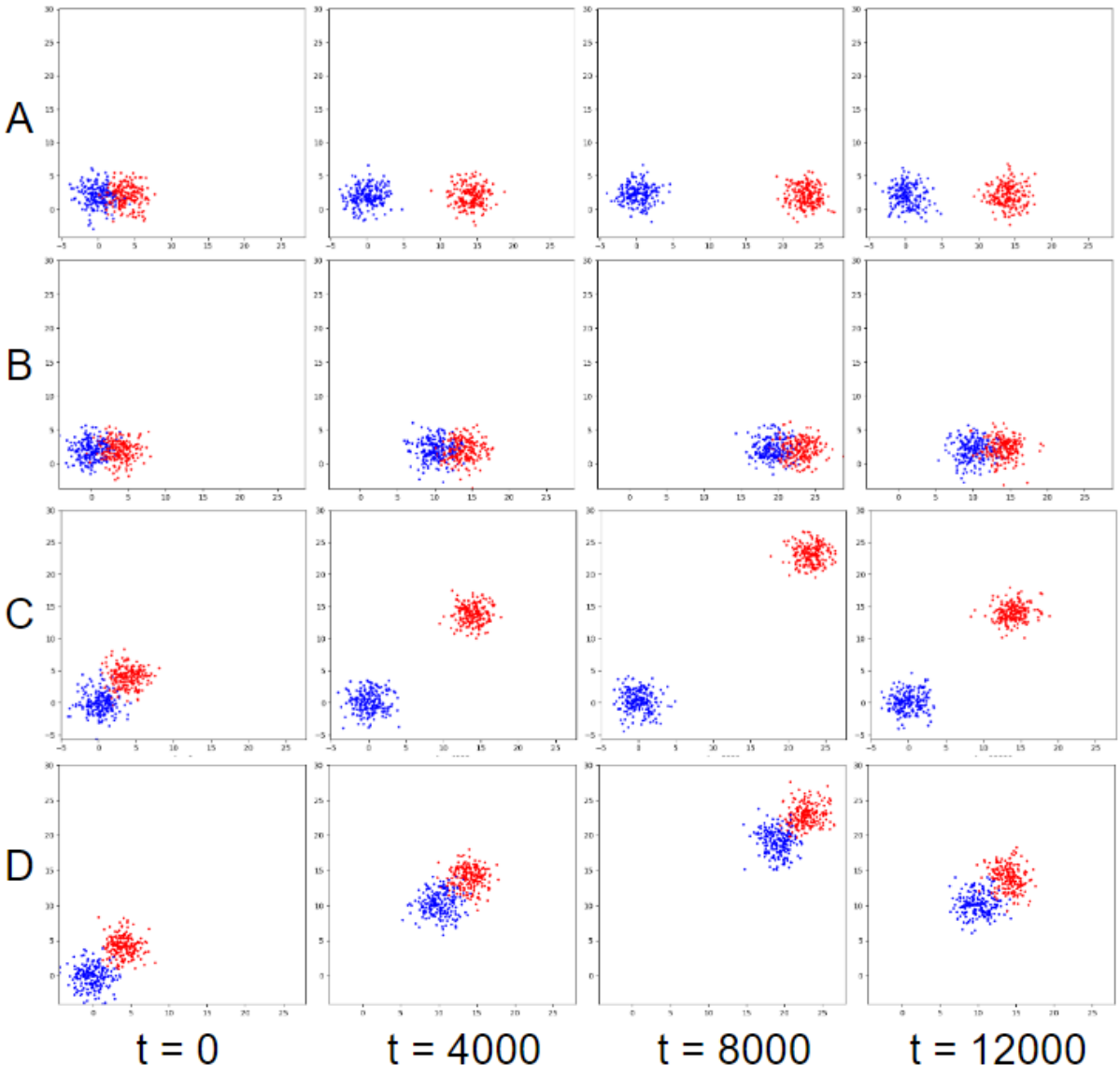

Figura 4.2.2 – Exemplos de mudanças de conceito nas bases 1CDT (A), 2CDT (B), 1CHT (C) e 2CHT (D).

Eles foram obtidos executando o K-Means com as 400 primeiras instâncias. Pode-se perceber que os novos dados se afastam do centroide de seu grupo, ocorrendo, assim, um desvio de conceito.

4.2.2 Dados Reais

Souza et al. (2020) apresentam vários conjuntos de dados do mundo real, sendo que dois deles foram utilizados neste trabalho: NOAA(DITZLER; POLIKAR, 2012)

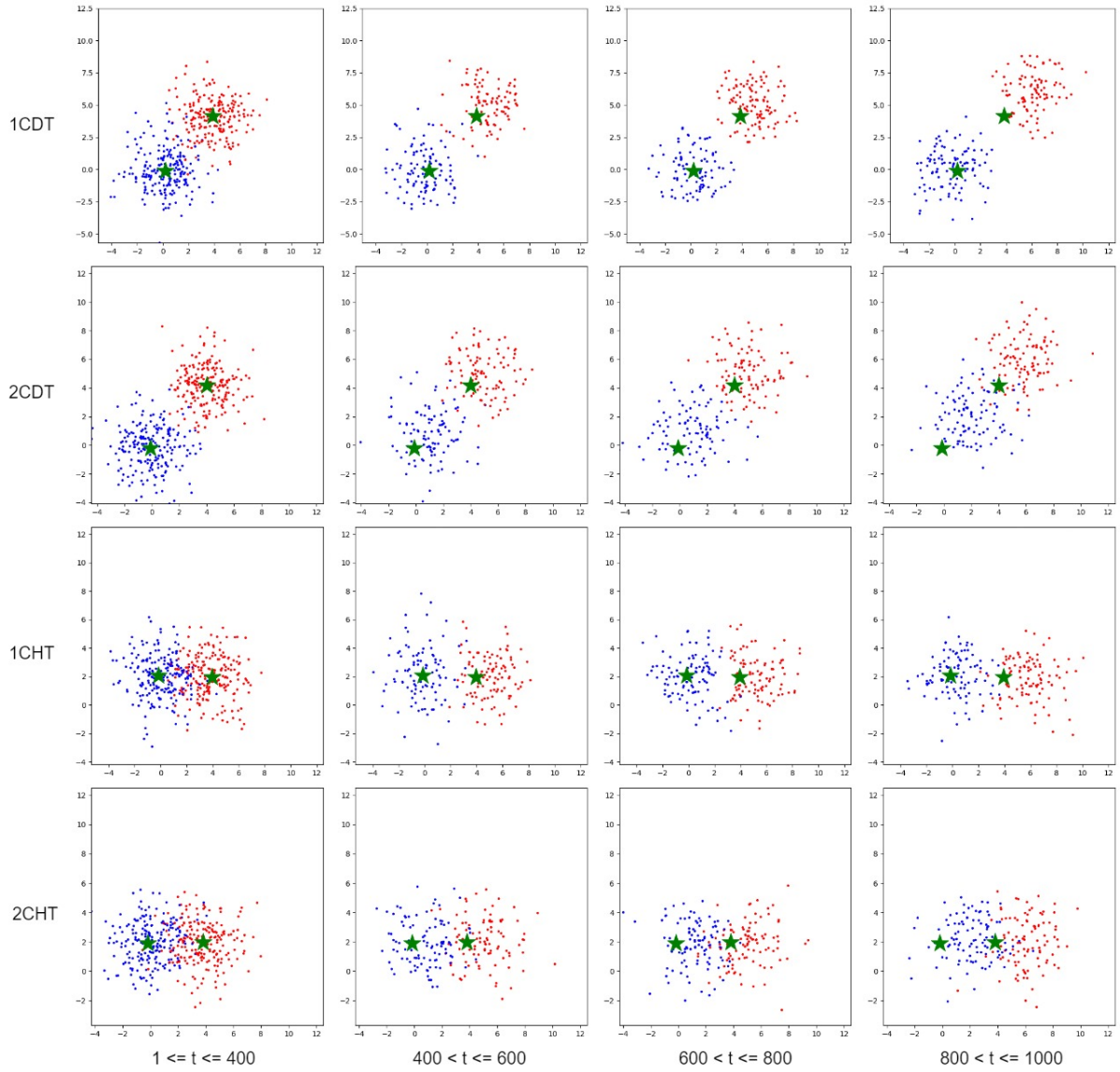

Figura 4.2.3 – Desvio de conceito ao longo do tempo.

e OZONE(DUA; GRAFF, 2017). Para uma comparação inicial, foram escolhidas as bases que mais se assemelham ao conjunto gerado na seção anterior, ou seja, que possuem dados divididos em dois grupos, balanceados e contendo apenas atributos numéricos. Os fatores determinantes para a escolha foram o número de grupos e a abrangência de todos os atributos numéricos, diminuindo o trabalho de pré-processamento dos dados para se adequarem ao uso do k-means, o qual será utilizado para formar os grupos.

Levando isso em consideração, foram selecionados os conjuntos:

- NOAA: conjunto de dados sobre medições meteorológicas coletadas pela Administração Oceânica e Atmosférica Nacional (NOAA) ao longo de 50 anos no

Nebraska; ele contém oito atributos numéricos e duas classes: dia de chuva, com 5698 amostras, e dia sem chuva, com 12461 amostras;

- Ozone: são dados de atmosfera coletados entre 1998 e 2004 em Houston, Galveston e Brazoria; possui duas classes (dia de ozônio e dia normal), 2534 amostras e 72 atributos numéricos.

4.3 Classificador de Desvio de Conceito

Depois de definido o escopo do trabalho, incluindo as classes e os conjuntos de dados que serão utilizados, esta seção apresentará a solução desenvolvida: Classificador de Desvio de Conceito (CDC). Ele tem como objetivo detectar o desvio de conceito, identificar qual é o tipo de desvio que ocorre e o que aconteceu com o grupo.

Foi realizado um estudo na literatura sobre detecção de desvio de conceito em agrupamento e concluiu-se que muitos trabalhos utilizam a distância para detectar o desvio (NAMITHA; KUMAR, 2020; NAMITHA et al., 2020; CASTELLANI; SCHMITT; HAMMER, 2021; YANG et al., 2021). Então, neste trabalho foi verificado se uma janela com as distâncias medidas a partir dos pontos ao centroide mais próximo é um bom caracterizador do tipo de mudança de conceito. A ideia é observar como os pontos em uma janela de tempo se aproximam ou se afastam de seus centroides quando a mudança ocorre. Também será analisada a silhueta, uma métrica de qualidade de agrupamentos que avalia se os grupos estão coesos, de modo que seu valor deve diminuir em uma mudança de conceito e na modificação da estrutura dos grupos.

Assim, a solução funciona junto com um algoritmo de agrupamento, K-Means (Seção 2.4.1), para poder coletar o metadado necessário, distância ou silhueta. Desse modo, o treinamento do CDC começa formando janelas de tamanho N com uma característica do agrupamento (a silhueta do dado ou a distância do centroide ao ponto) para treinar a rede prototípica (Seção 2.3.5). Foi utilizado o tamanho de 5 para o conjunto de suporte e de consulta, gerando, assim, um conjunto de 90 janelas.

O conjunto formado é a entrada da rede, que converterá cada janela de dados para um espaço latente. O algoritmo, então, aprenderá a separar as janelas de dados em

grupos diferentes, um para cada classe. O objetivo é fazer com o que o CDC aprenda, nesse novo espaço, como separar melhor as janelas de metadados de acordo com o desvio que ocorreu e o que aconteceu com os grupos. A Figura 4.3.1 complementa o resumo das etapas de treinamento a seguir:

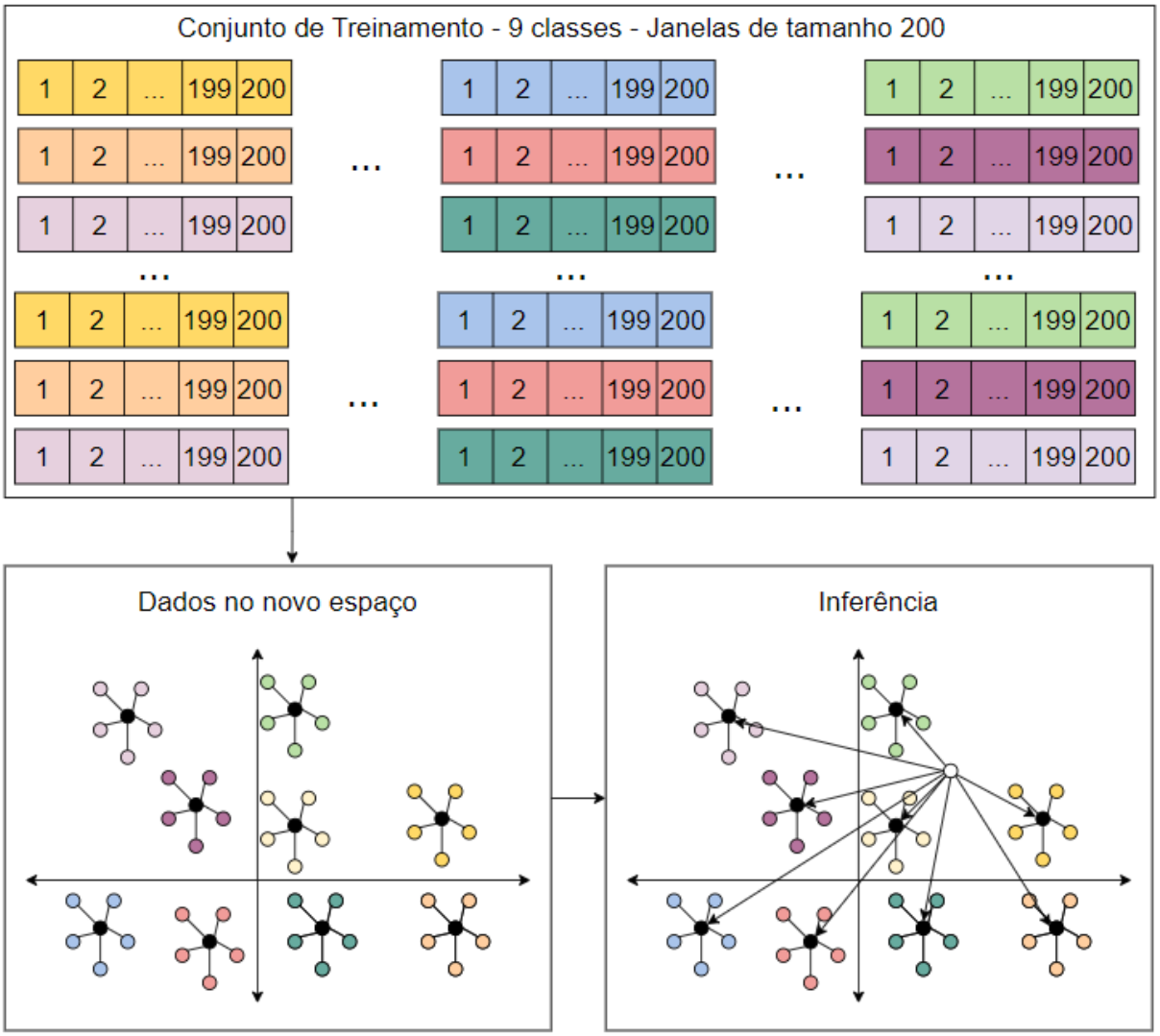

Figura 4.3.1 – Fluxo de treinamento do CDC. Na parte superior: janelas de cada classe (tipo de mudança) são selecionadas. Cada janela possui 200 valores de distância para o centroide ou de silhueta. Na parte inferior, à esquerda: pontos obtidos pela rede prototípica e os 9 centroides (um para cada classe). À direita: classificação de uma nova janela pela distância do centroide mais próximo no espaço de *embedding*.

1. Escolhe-se aleatoriamente 90 janelas (10 para cada classe) de fluxo de dados do conjunto que foi previamente gerado;

2. Desse conjunto, escolhe-se 45 janelas (5 de cada classe) para ser o conjunto de suporte;

3. As janelas são transformadas, gerando *embeddings*, para serem melhor agrupadas no espaço latente;

4. Nesse espaço, centroides para cada classe são calculados;

5. O restante das 45 janelas também são transformadas e suas classes são inferidas utilizando a classe do grupo no qual encontra-se o centroide mais perto do *embedding*;

6. O erro das classificações é calculado e os pesos são ajustados para começar o ciclo novamente.

A próxima etapa, depois do treinamento, é a inferência. A Figura 4.3.2 ilustra um fluxo de etapas necessárias para a detecção e identificação do desvio em uma janela de metadado. A diferença da etapa de treinamento para de inferência é que a primeira tem uma etapa a mais, no qual é calculado o erro do modelo e ele é atualizado. Os passos também são detalhados a seguir:

1. Calcula o metadado, ou seja, a distância do centroide ao ponto;

2. Forma a janela de tamanho N;

3. Passa a janela como entrada para o CDC;

4. CDC transforma a janela para outro espaço, convertendo em um *embedding*;

5. CDC calcula as distâncias do *embedding* para todos os grupos;

6. CDC infere a classe, a qual é escolhida como a classe do grupo que apresenta a menor distância de seu centroide até o *embedding*.

Com essas informações, o usuário pode compreender melhor o que acontece com o modelo treinado e analisar qual é a melhor solução para o seu caso. O modelo que sofreu desvio pode ser retreinado parcialmente ou substituído por um novo.

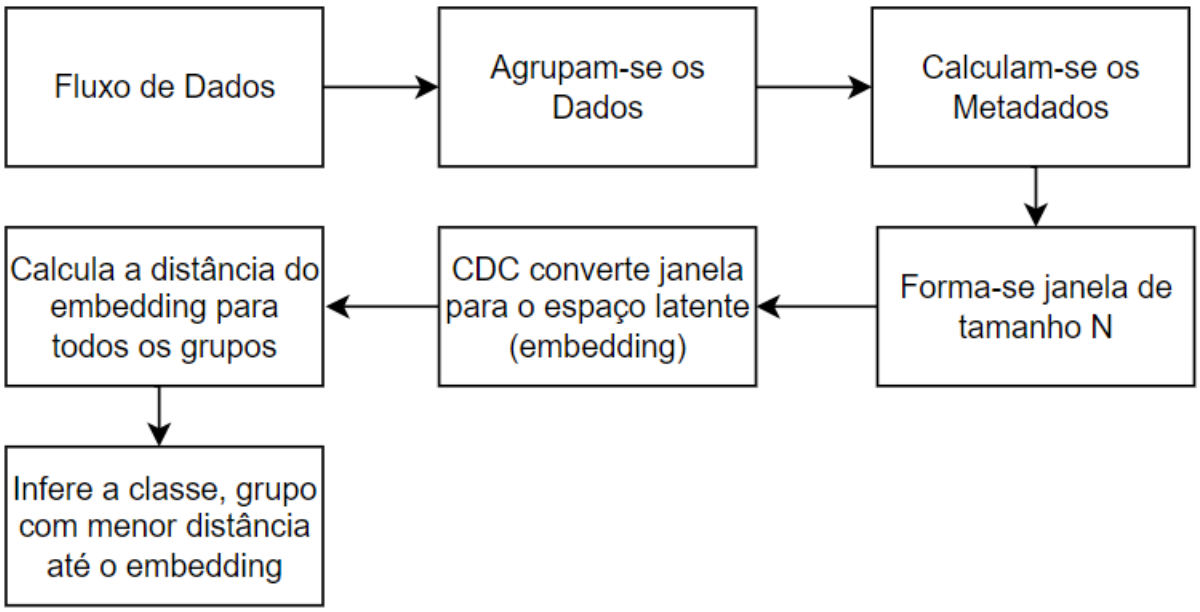

Figura 4.3.2 – Fluxo de funcionamento do CDC na inferência de uma nova janela de metadado.

Por utilizar um metadado, a ideia é que o CDC possa ser utilizado com a maior variedade possível de algoritmos de agrupamento e robusto o suficiente para ser utilizado em diferentes domínios além do qual ele foi treinado. Por isso, uma outra vantagem é que esse treinamento prévio permite detectar o desvio já no começo do fluxo, ou seja, nas primeiras instâncias.

4.3.1 Rede Neural

O CDC utiliza a rede prototípica (Seção 2.3.5). Elas têm algumas vantagens: são computacionalmente eficientes depois de treinadas; são explicáveis, visto que o processo de classificação é baseado na distância do *embedding* ao protótipo; podem ser usadas em outros domínios com pouco ou nenhum treinamento.

Na rede prototípica, cada classe é representada por um protótipo que é a média dos *embeddings* de exemplos das classes. Assim, o que essa rede faz é mapear as amostras de entradas para um espaço latente, no qual cada amostra é representada como um vetor (*embedding*). As instâncias usadas para calcular o protótipo são chamadas de conjunto de suporte e as que são utilizadas para testar os *embeddings* são conjunto de

busca. Seja $\phi(x_i)$ o vetor de uma instância x_i e p_c o protótipo da classe c, tem-se:

$$p_c = \frac{1}{N_c} \sum_{i \in C} \phi(x_i)$$

Depois de calcular os protótipos, o modelo usa uma função de distância (neste trabalho foi a Euclidiana) para comparar os *embeddings* do conjunto de busca com o protótipo de cada classe. Seja x_q uma instância do conjunto de busca, a distância é calculada como:

$$d(\phi(x_q), p_c) = \|\phi(x_q) - p_c\|_2$$

A distância verifica o quão similar as amostras são do protótipo de cada classe. Para classificar o dado de busca, o modelo acha o protótipo que está mais perto de seu *embedding*:

$$\hat{y} = arg\ min\ d(\phi(x_q), p_c)$$

Uma vez que as distâncias são calculadas, uma função Softmax é aplicada para converter as distâncias em probabilidades de classe. A probabilidade inferida de uma amostra x_q ser da classe c é:

$$P(\hat{y}_q = c \mid x_q) = \frac{\exp(-d(\phi(x_q), p_c))}{\sum_{c'} \exp(-d(\phi(x'_q), p_{c'}))}$$

A finalidade do modelo é aprender uma função de *embedding* para que protótipos de uma mesma classe fique próximos um do outro no espaço latente e distantes dos protótipos de classes diferentes. Desse modo, a função de perda tem o objetivo de minimizar o erro da classificação. É utilizada a entropia cruzada (*cross-entropy*) sobre as probabilidades inferidas e suas verdadeiras classes. Assim, a entropia cruzada para uma amostra x_q e seu rótulo verdadeiro é dado por:

$$\mathcal{L}_q = -\log P(\hat{y}_q = c \mid x_q)$$

A rede neural utilizada na rede prototípica consiste em 4 camadas, como mostra a Figura 4.3.3. Tem uma camada de entrada com N neurônios, de acordo com o tamanho

da janela escolhida, que se conecta com as camadas escondidas com, respectivamente, 128 e 32 neurônios. Por último, tem-se a camada final, com 9, mesmo número de classes. Vários números de camadas e tamanhos foram testados e essa combinação foi a que gerou o melhor resultado.

Além disso, a função de ativação utilizada é a ReLU, visto que ela permite um aprendizado mais rápido e é utilizado por vários outros trabalhos. O Dropout de 0,25 é utilizado depois da primeira função de ativação. Ele foi utilizado para prevenir *overfitting*, que estava ocorrendo sem o seu uso. O optimizador utilizado é o Adam com a taxa de aprendizado sendo 10^{-3}. Ambas as escolhas são amplamente utilizadas em vários trabalhos de aprendizado profundo. Para o conjunto de suporte e busca foi escolhido como o número de exemplos o valor 5, ou seja, cada conjunto apresenta 5 exemplos de cada classe.

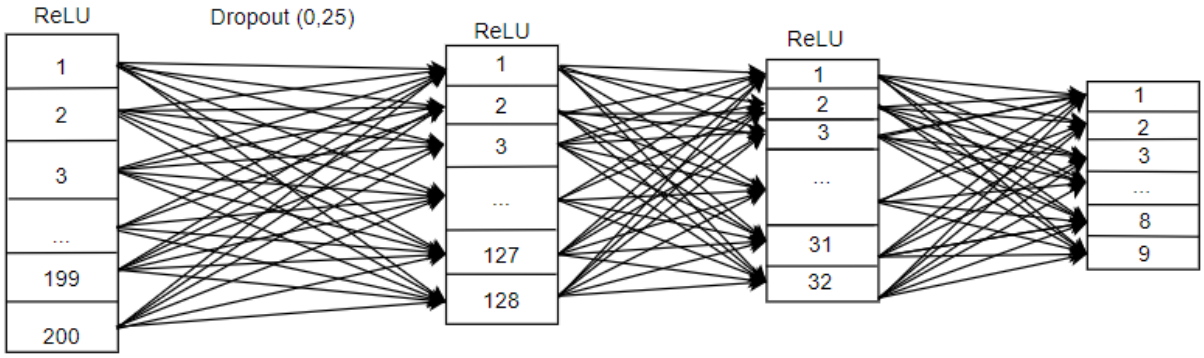

Figura 4.3.3 – Rede neural formada para a rede prototípica.

4.4 Considerações Finais

Este capítulo abordou toda a solução desenvolvida para detectar e identificar o tipo de desvio de conceito e sua causa. Ela é composta pelo classificador de desvio de conceito e pela geração do conjunto sintética que possibilitou o treinamento do CDC e que tem como saída classes que indicam o tipo de desvio (abrupto ou gradual) e o que está ocorrendo com os dados (adição, separação, fusão e separação de grupos).

Além disso, houve um detalhamento, não só do conjunto proposto por este

trabalho, mas também dos conjuntos sintéticos e reais utilizados para testar e comparar o CDC com outras alternativas. Ademais, apresentou-se o CDC, como é o seu treinamento, a sua inferência, como ele é construído e suas as características.

5

EXPERIMENTOS E RESULTADOS

Este capítulo descreve os experimentos realizados e seus resultados.

5.1 Protocolo Experimental

Nesta seção, serão abordados os experimentos: seus objetivos e protocolos.

5.1.1 Classificadores

Este experimento vai ocorrer junto de outros dois: "Características do Agrupamento" e "Tamanho das Janelas". Em todos eles o conjunto sintético criado (Seção 4.2.1) foi utilizado contendo janelas sem mudança (SM) e com as seguintes mudanças de conceito: adição (AA, AG), separação (SA, SG), fusão (FA, FG) e deslocamento (DA, DG), sendo que cada mudança tem um subtipo abrupto (segunda letra A) e um subtipo gradual (segunda letra G).

Junto com o CDC, nesses experimentos, o KNN (Seção 2.3.1) e o Adaboost (Seção 2.3.3) também serão utilizados com o objetivo de comparar seus desempenhos e analisar se o CDC é melhor do que eles.

Os classificadores foram treinados com metade dos dados apresentados na Seção 4.2.1 e a outra metade foi utilizada para teste. Foi escolhido o valor 1 para o parâmetro K do KNN, ou seja, a nova instância é classificada de acordo com a classe da instância que está mais perto no espaço. Para o Adaboost, o classificador selecionado é o Naive Bayes

(Seção 2.3.2) e foram utilizados 200 deles. Para o CDC foram utilizados 5 exemplos de cada classe tanto no treinamento quanto no teste, como o conjunto de suporte e de busca.

5.1.2 Características do Agrupamento

Esse experimento busca saber se uma característica do agrupamento é um bom indicador para detectar desvio e seu tipo. As características testadas foram silhueta e a distância da instância ao centroide mais próximo. Nesses dois casos, o algoritmo de agrupamento utilizado foi o k-means com o valor de k igual a 2, pois é o número de grupos criados no conjunto sintético.

A silhueta (Seção 2.4.3) é um dos indicadores de um agrupamento bem realizado, ou seja, se os dados semelhantes estão agrupados juntos e longe dos dados com características diferentes. O objetivo deste experimento é verificar se a variação da silhueta, conforme os grupos se deslocam, é uma boa ferramenta para identificar o tipo de mudança de conceito. A cada elemento que chega no fluxo, sua silhueta é calculada considerando todos os pontos e centroides anteriores e essa silhueta é inserida na janela.

A distância do ponto até o centroide pode indicar mudanças que estão ocorrendo nos grupos, visto que haverá uma variação das distâncias conforme a distribuição dos dados muda. Assim, cada janela será formada pela distância da instância até o seu centroide mais próximo.

No experimento da silhueta as janelas tiveram o tamanho de 200. Já no experimento da distância, o tamanho foi variado, pois foi realizado junto com outro: "Tamanho das Janelas". Essa diferença ocorre, pois durante os experimentos percebeu-se que a utilização da distância resultou em melhores métricas dos classificadores do que usando a silhueta.

5.1.3 Tamanhos de Janelas

Esse experimento testa diferentes tamanhos de janela (100, 200, 400) para identificar o melhor tamanho de janela a fim de facilitar a detecção e identificação do desvio. O conjunto criado na Seção 4.2.1 foi modificado para produzir as janelas de diversos tamanhos. O CDC também foi modificado, pois a camada de entrada da rede neural utilizada também muda de tamanho conforme o tamanho da janela de entrada.

5.1.4 Tarefa de Agrupamento

Nesta etapa de experimentos, foi comparado o CDC com outros detectores de desvio de conceito na tarefa de agrupamento em um ambiente não supervisionado com duas bases reais e 4 sintéticas. Foram utilizados os algoritmos: PHT (Seção 2.1.2.5) e Adwin (Seção 2.1.2.3). Eles foram comparados apenas na detecção de desvio, não sua classificação.

O k-means, com k igual a 2, foi utilizado como o algoritmo para agrupar os dados do conjunto de testes do CDC (Seção 4.2.1). A entrada dos detectores foi a distância da instância ao centroide e o tamanho da janela foi de 200.

Além do teste com o conjunto criado, foi realizado um com dois conjuntos de dados reais: o NOAA e o Ozone 4.2.2. Além disso, o modelo selecionado para agrupar os dados foi o k-means com k igual a 2 e a entrada dos detectores foi a distância da instância ao centroide. A métrica escolhida para analisar a qualidade do agrupamento foi a silhueta. Para comparar o efeito da detecção na silhueta, foi utilizada o k-means como base a fim de verificar se o valor de silhueta obtida sem detecção é maior ou menor do que quando se utiliza detectores.

Para esse experimento, os conjuntos reais foram normalizados com o *MinMaxScaler* (MÜLLER; GUIDO, 2016), pois as colunas têm faixas de valores muito diferentes. Depois, o k-means foi executado nas 400 primeiras instâncias. A cada novo dado, a distância para os centroides foi calculada e enviada para os detectores. Quando qualquer um dos algoritmos detectava o desvio, k-means é novamente utilizada nas 400 instâncias anteriores ao ponto de detecção do desvio. O valor da silhueta foi calculado a cada novo ponto.

No caso da CDC, que precisa de uma janela de tamanho 200, ela só começou a detectar depois dos 200 primeiros exemplos. A cada nova instância, a amostra mais antiga saía e entrava a mais nova.

5.1.5 Tarefa de Classificação

Nesta parte de experimentos, foi comparado o CDC com outros detectores de desvio de conceito na tarefa de classificação em um ambiente supervisionado. Os detectores de desvio foram: DDM, EDDM, ADWIN e PHT.

Os detectores utilizam como entrada o erro do classificador, enquanto o CDC continua utilizando a distância da instância ao centroide mais próximo. Os centroides foram obtidos com k-means e k igual a 2. O classificador utilizado na classificação é o Adaboost com Naive Bayes. Ele foi treinado com as 200 primeiras instâncias e, a cada desvio detectado, é treinado novamente com as 200 amostras anteriores ao desvio.

Foram testados 4 conjuntos sintéticos e 2 reais. O CDC utilizado foi treinado com o conjunto criado neste trabalho. Assim, é possível testar também a generalização do modelo ao treinar em um domínio e usá-lo em outro.

5.1.6 Identificação do Tipo de Desvio

O último experimento foi realizado com uma base sintética para testar a identificação e causa da mudança de desvio. Esse experimento foi realizado com 4 bases sintéticas onde a quantidade de desvio e sua causa são conhecidas. As bases reais não possuem essas informações e, por isso, não foram utilizadas.

5.2 Resultados

Nesta seção, serão abordados os resultados dos experimentos.

5.2.1 Silhueta

As tabelas 4 a 9 mostram os resultados obtidos pelos classificadores (1-NN, Adaboost e CDC) ao usarem a silhueta para compor as janelas de tamanho 200.

Tabela 4 – Métricas do 1-NN.

	Precisão	Revocação	F1-Score
SM	0.70	0.83	0.76
AA	0.34	0.70	0.46
AG	0.50	0.01	0.02
SA	0.34	0.94	0.50
SG	0.00	0.00	0.00
FA	0.90	0.99	0.94
FG	1.00	0.21	0.35
DA	0.47	0.79	0.59
DG	1.00	0.01	0.02
MÉDIA	0,58	0,50	0,40

Tabela 5 – Matriz de Confusão - 1-NN.

	SM	AA	AG	SA	SG	FA	FG	DA	DG
SM	73	0	0	0	0	0	0	15	0
AA	1	62	0	17	0	6	0	3	0
AG	1	31	1	53	0	0	0	3	0
SA	0	5	0	84	0	0	0	0	0
SG	0	2	0	87	0	0	0	0	0
FA	0	0	0	1	0	8 8	0	0	0
FG	0	63	1	2	0	4	1 9	0	0
DA	13	6	0	0	0	0	0	70	0
DG	17	14	0	0	0	0	0	57	1

Tabela 6 – Métricas do Adaboost.

	Precisão	Revocação	F1-Score
SM	0.38	0.16	0.22
AA	0.21	0.67	0.32
AG	0.33	0.29	0.31
SA	0.66	0.44	0.53
SG	0.82	0.10	0.18
FA	0.98	0.94	0.96
FG	0.00	0.00	0.00
DA	0.43	0.37	0.40
DG	0.36	0.65	0.47
MÉDIA	0,46	0,40	0,38

Além disso, a Tabela 10 compara a média das acurácias obtidas com a silhueta.

Tabela 7 – Matriz de Confusão - Adaboost.

	SM	AA	AG	SA	SG	FA	FG	DA	DG
SM	14	0	0	0	0	0	0	25	49
AA	1	60	0	18	0	2	0	2	6
AG	1	54	26	0	2	0	0	1	5
SA	0	46	3	39	0	0	0	0	1
SG	0	30	49	1	9	0	0	0	0
FA	0	4	0	1	0	84	0	0	0
FG	0	89	0	0	0	0	0	0	0
DA	9	6	0	0	0	0	0	33	41
DG	12	2	2	0	0	0	0	15	58

Tabela 8 – Métricas do CDC.

	Precisão	Revocação	F1-Score
SM	0.41	0.43	0.42
AA	0.44	0.40	0.42
AG	0.37	0.41	0.39
SA	0.38	0.32	0.35
SG	0.34	0.33	0.34
FA	0.39	0.38	0.38
FG	0.41	0.43	0.42
DA	0.40	0.41	0.40
DG	0.34	0.34	0.34
MÉDIA	0,39	0,39	0,38

Tabela 9 – Matriz de Confusão - CDC.

	SM	AA	AG	SA	SG	FA	FG	DA	DG
SM	39	4	7	2	5	10	7	5	11
AA	6	36	4	9	8	5	9	9	4
AG	5	3	37	4	8	7	6	12	8
SA	10	7	10	29	5	7	8	8	6
SG	8	5	15	9	30	5	6	5	7
FA	6	6	3	6	9	34	9	6	11
FG	6	4	10	5	10	4	39	5	7
DA	6	8	10	6	5	4	7	37	7
DG	10	9	4	6	7	12	5	6	31

Ao analisar essas tabelas pode-se concluir que:

- 1-NN apresenta a melhor acurácia, mas erra muito nos desvios graduais;
- Adaboost também erra muito nos desvios graduais;
- CDC acerta, em média, a mesma quantidade para todas as classes.

Tabela 10 – Média das acurácias obtidas com silhueta.

	1NN	Adaboost	CDC
Média Acurácia	50%	40%	39%

Tabela 11 – Acurácia com a distância do ponto ao centroide.

	KNN	Ada	RP
Acurácia	0,34	0,78	0,75

A Tabela 11 mostra a acurácia obtida utilizando a distância do ponto ao centroide como metadado e com o mesmo tamanho de janela e quantidade de fluxo de dados usados para silhueta. Comparando ela com as acurácias obtidas utilizando a silhueta, conclui-se que a distância pode ser melhor para detectar desvio de conceito e o que aconteceu com o agrupamento.

Assim, a próxima seção apresentará os resultados obtidos com o algoritmo treinado, utilizando a distância do ponto ao centroide, e com um conjunto maior de exemplos com a finalidade de intensificar a análise.

5.2.2 Distância para centroide com Diferentes Tamanhos de Janela

5.2.2.1 Tamanho 100

As tabelas 12 a 17 mostram os resultados obtidos pelos classificadores (1-NN, Adaboost e CDC) ao usarem a distância para compor as janelas de tamanho 100.

Tabela 12 – Métricas do 1-NN.

	Precisão	Revocação	F1-Score
SM	0,43	0,99	0,60
AA	0,16	0,02	0,03
AG	0,36	0,09	0,14
SA	0,18	0,02	0,04
SG	0,17	0,05	0,07
FA	0,44	1,00	0,62
FG	0,75	0,90	0,82
DA	0,23	0,29	0,25
DG	0,14	0,17	0,15
MÉDIA	0,32	0,39	0,30

Tabela 13 – Matriz de Confusão - 1NN.

	SM	AA	AG	SA	SG	FA	FG	DA	DG
SM	592	1	7	0	0	0	0	0	0
AA	45	10	1	50	5	298	22	131	38
AG	81	5	51	12	118	2	125	70	136
SA	0	5	0	14	5	407	21	145	3
SG	4	4	28	0	28	0	11	229	296
FA	0	0	0	0	0	600	0	0	0
FG	0	0	0	0	7	44	540	6	3
DA	204	28	25	0	0	0	0	173	170
DG	454	8	29	0	0	0	0	5	104

Tabela 14 – Métricas do Adaboost.

	Precisão	Revocação	F1-Score
SM	0.98	0.72	0.83
AA	0.79	0.52	0.62
AG	0.33	0.77	0.46
SA	0.67	1.00	0.80
SG	0.82	0.10	0.18
FA	1.00	0.98	0.99
FG	0.96	0.94	0.95
DA	0.75	0.98	0.85
DG	0.63	0.24	0.34
MÉDIA	0,77	0,69	0,67

Tabela 15 – Matriz de Confusão - Adaboost.

	SM	AA	AG	SA	SG	FA	FG	DA	DG
SM	434	1	109	0	0	0	0	2	54
AA	3	309	12	195	0	0	7	67	7
AG	6	70	460	14	6	0	12	11	21
SA	0	0	0	600	0	0	0	0	0
SG	0	4	476	53	61	0	6	0	0
FA	0	0	0	13	0	587	0	0	0
FG	0	0	0	26	7	0	567	0	0
DA	0	5	5	0	0	0	0	589	1
DG	2	0	343	0	0	0	0	113	142

Observando as tabelas, pode-se perceber que:

- O 1-NN obteve acurácia de 39%;. Apesar de ter a pior acurácia entre os três algoritmos, considerando apenas a detecção da existência ou não do desvio, o 1NN foi o melhor. O 1-NN confundiu a classe AA e SA com FA e DA e DG com SM;

Tabela 16 – Métricas do CDC.

	Precisão	Revocação	F1-Score
SM	0.76	0.76	0.76
AA	0.74	0.75	0.75
AG	0.75	0.73	0.74
SA	0.75	0.70	0.72
SG	0.75	0.75	0.75
FA	0.74	0.76	0.75
FG	0.74	0.73	0.73
DA	0.74	0.73	0.74
DG	0.75	0.80	0.77
MÉDIA	0,75	0,75	0,75

Tabela 17 – Matriz de Confusão - CDC.

	SM	AA	AG	SA	SG	FA	FG	DA	DG
SM	454	16	19	24	9	21	19	23	15
AA	17	451	17	14	18	22	19	20	22
AG	25	14	439	20	33	9	15	21	24
SA	17	23	22	422	31	23	20	23	19
SG	10	22	17	21	449	28	21	18	14
FA	24	23	19	15	13	454	14	18	20
FG	16	26	24	15	14	26	438	20	21
DA	13	22	13	16	23	17	31	439	26
DG	18	10	17	19	11	15	18	12	480

- O Adaboost obteve 69% de acurácia. Ele errou mais na separação e deslocamento gradual e confundiu SG com AG e DG com AG;
- O CDC obteve 75% de acurácia e valores de métricas parecidas para todas as classes;
- Para algumas classes, o Adaboost é melhor que a CDC, para outras, esse resultado é invertido.

5.2.2.2 Tamanho 400

As tabelas 18 a 23 mostram os resultados obtidos pelos classificadores (1-NN, Adaboost e CDC) ao usarem a distância para compor as janelas de tamanho 400.

Observando as tabelas, conclui-se que:

- O 1-NN obteve acurácia de 32%;. Apesar de ter a pior acurácia entre os três

Tabela 18 – Métricas do 1-NN.

	Precisão	Revocação	F1-Score
SM	0.34	1.00	0.51
AA	0.09	0.01	0.01
AG	0.10	0.01	0.01
SA	0.00	0.00	0.00
SG	0.00	0.00	0.00
FA	0.35	1.00	0.51
FG	0.61	0.78	0.68
DA	0.15	0.09	0.11
DG	0.01	0.01	0.01
MÉDIA	0,18	0,32	0,20

Tabela 19 – Matriz de Confusão - 1NN.

	SM	AA	AG	SA	SG	FA	FG	DA	DG
SM	600	0	0	0	0	0	0	0	0
AA	55	3	3	0	0	405	2	113	19
AG	111	1	4	0	7	10	273	43	151
SA	0	0	0	0	0	600	0	0	0
SG	0	0	4	0	0	0	29	146	421
FA	0	0	0	0	0	600	0	0	0
FG	0	0	0	0	0	124	466	4	6
DA	396	23	21	0	0	0	0	52	108
DG	579	6	10	0	0	0	0	0	5

Tabela 20 – Métricas do Adaboost.

	Precisão	Revocação	F1-Score
SM	0.00	0.00	0.00
AA	0.39	0.87	0.54
AG	0.26	0.73	0.38
SA	0.87	1.00	0.93
SG	0.90	0.84	0.87
FA	1.00	0.95	0.98
FG	1.00	0.93	0.96
DA	0.00	0.00	0.00
DG	0.00	0.00	0.00
MÉDIA	0,49	0,59	0,52

algoritmos, se for para detectar apenas se houve desvio ou não, o 1NN foi o melhor. Ele também é bom para detectar as classes de fusão. Ele classificou como sem desvio várias janelas das classes de adição e deslocamento e inferiu pouco nas classes AA, AG, SA e SG;

- O Adaboost obteve 59% de acurácia. Ele obteve um bom desempenho para as

Tabela 21 – Matriz de Confusão - Adaboost.

	SM	AA	AG	SA	SG	FA	FG	DA	DG
SM	0	2	598	0	0	0	0	0	0
AA	0	523	13	64	0	0	0	0	0
AG	0	140	436	0	24	0	0	0	0
SA	0	0	0	600	0	0	0	0	0
SG	0	5	92	0	503	0	0	0	0
FA	0	0	0	29	0	571	0	0	0
FG	0	11	2	0	29	0	558	0	0
DA	0	600	0	0	0	0	0	0	0
DG	0	73	527	0	0	0	0	0	0

Tabela 22 – Métricas do CDC.

	Precisão	Revocação	F1-Score
SM	0.75	0.75	0.75
AA	0.76	0.80	0.78
AG	0.76	0.77	0.77
SA	0.75	0.73	0.74
SG	0.78	0.77	0.77
FA	0.76	0.76	0.76
FG	0.75	0.72	0.74
DA	0.76	0.77	0.77
DG	0.78	0.79	0.79
MÉDIA	0,76	0,76	0,76

Tabela 23 – Matriz de Confusão - CDC.

	SM	AA	AG	SA	SG	FA	FG	DA	DG
SM	450	20	19	17	18	12	26	23	15
AA	16	481	17	19	14	15	12	20	6
AG	20	12	461	20	20	13	26	13	15
SA	24	22	17	440	18	28	18	17	16
SG	9	18	16	18	461	23	16	16	23
FA	17	19	15	18	12	458	17	16	28
FG	26	18	29	15	19	17	434	29	13
DA	16	24	17	19	13	14	19	463	15
DG	20	15	12	17	18	21	10	12	475

classes de adição, separação e fusão e errou tudo para as outras classes (SM, DA e DG);

- O CDC obteve 76% de acurácia e valores de métricas parecidas para todas as classes.

5.2.2.3 Tamanho 200

As tabelas 24 a 29 mostram os resultados obtidos pelos classificadores (1-NN, Adaboost e CDC) ao usarem a distância para compor as janelas de tamanho 200.

Tabela 24 – Métricas do 1-NN.

	Precisão	Revocação	F1-Score
SM	0.38	0.99	0.55
AA	0.09	0.01	0.01
AG	0.27	0.03	0.05
SA	0.00	0.00	0.00
SG	0.01	0.00	0.00
FA	0.37	1.00	0.53
FG	0.71	0.85	0.77
DA	0.19	0.18	0.19
DG	0.09	0.11	0.10
MÉDIA	0,23	0,35	0,25

Tabela 25 – Matriz de Confusão - 1NN.

	SM	AA	AG	SA	SG	FA	FG	DA	DG
SM	593	4	3	0	0	0	0	0	0
AA	50	4	2	1	0	385	2	136	20
AG	99	4	17	5	70	7	190	59	149
SA	0	0	0	0	0	570	1	29	0
SG	2	4	11	0	1	0	13	210	359
FA	0	0	0	0	0	600	0	0	0
FG	0	0	0	0	0	81	509	7	3
DA	294	27	21	0	0	0	0	107	151
DG	523	4	8	0	0	0	0	1	64

Tabela 26 – Métricas do Adaboost.

	Precisão	Revocação	F1-Score
SM	0.95	0.43	0.59
AA	0.41	0.79	0.54
AG	0.39	0.50	0.44
SA	0.80	1.00	0.89
SG	0.88	0.73	0.80
FA	1.00	0.95	0.98
FG	0.98	0.94	0.96
DA	0.86	0.28	0.42
DG	0.89	0.91	0.90
MÉDIA	0,80	0,73	0,72

Ao analisar as tabelas, pode-se verificar que:

Tabela 27 – Matriz de Confusão - Adaboost.

	SM	AA	AG	SA	SG	FA	FG	DA	DG
SM	259	26	314	0	0	0	0	0	1
AA	3	472	2	107	0	0	0	5	11
AG	10	174	302	10	30	0	13	7	54
SA	0	0	0	600	0	0	0	0	0
SG	0	14	142	5	439	0	0	0	0
FA	0	0	0	28	0	572	0	0	0
FG	0	2	0	2	31	0	565	0	0
DA	0	435	0	0	0	0	0	165	0
DG	0	27	14	0	0	0	0	14	545

Tabela 28 – Métricas do CDC.

	Precisão	Revocação	F1-Score
SM	0.74	0.77	0.76
AA	0.79	0.80	0.79
AG	0.79	0.79	0.79
SA	0.78	0.77	0.77
SG	0.77	0.78	0.77
FA	0.77	0.73	0.75
FG	0.78	0.80	0.79
DA	0.77	0.76	0.76
DG	0.75	0.74	0.74
MÉDIA	0,77	0,77	0,77

Tabela 29 – Matriz de Confusão - CDC.

	SM	AA	AG	SA	SG	FA	FG	DA	DG
SM	463	13	10	18	29	20	16	16	15
AA	20	480	18	12	12	21	8	14	15
AG	15	15	474	8	18	13	17	17	23
SA	20	16	10	463	13	21	15	20	22
SG	23	21	16	15	466	10	16	18	15
FA	22	17	21	26	13	439	20	22	20
FG	13	19	17	14	14	11	477	18	17
DA	24	14	14	16	27	13	17	456	19
DG	24	15	19	23	16	22	25	14	442

- O 1-NN obteve acurácia de 32%;. Apesar de ter a pior acurácia entre os três algoritmos, se for para detectar apenas se houve desvio ou não, o 1NN foi o melhor. Ele também é bom para detectar as classes de fusão. Além disso, classificou como sem desvio várias classes e inferiu pouco nas classes AA, AG, SA e SG;

- O Adaboost obteve 73% de acurácia e confundiu DA com AA, SM com AG;

- O CDC obteve 77% de acurácia e valores de métricas parecidas para todas as classes.

5.2.2.4 Considerações dos Hiperparâmetros

Analisando os testes realizados com as características do agrupamento e dos tamanhos de janelas, percebeu-se que os resultados indicam que o CDC obteve um desempenho melhor utilizando a distância do ponto ao centroide mais próximo como entrada dos algoritmos do que a silhueta. Ademais, o desempenho do CDC para um tamanho da janela de entrada de 200 superou o de tamanhos 100 e 400 e foi suficiente para que o modelo entendesse os padrões dos tipos de desvio.

Além disso, a Tabela 30 agrupa os resultados das métricas obtidas com a distância da instância ao centroide mais próximo e os diferentes tamanhos de janelas.

Modelo	Tamanho da Janela	Acurácia	Precisão	Revocação	F1-Score
1NN	100	0,39	0,32	0,39	0,30
1NN	200	0,35	0,23	0,35	0,25
1NN	400	0,32	0,18	0,32	0,20
Adaboost	100	0,69	0,77	0,69	0,67
Adaboost	200	0,73	0,80	0,73	0,72
Adaboost	400	0,59	0,49	0,59	0,52
CDC	100	0,75	0,75	0,75	0,75
CDC	200	0,77	0,77	0,77	0,77
CDC	400	0,76	0,76	0,76	0,76

Tabela 30 – Comparação dos modelos com diferentes tamanhos de janela.

Analisando a Tabela 30 e as de matrizes de confusão mostrada em cada janela, pode-se concluir:

- Nos três tamanhos, o 1-NN foi sempre o pior e o Adaboost e a CDC tiveram métricas parecidas;
- 1NN teve um bom desempenho para classificar as classes de fusão, mas errou a inferência de todas as outras classes. Ele presenta uma alta taxa de falso positivo para a classe "Sem Mudança";

- O tamanho da janela de 200 é o que apresenta as melhores métricas para todos os modelos;
- Adaboost e CDC apresentam valores de métricas parecidas se comparadas por tamanho de janela;
- Em cada tamanho de janela o Adaboost não obteve um bom desempenho em determinadas classes: na janela de 100, separação e deslocamento gradual; na janela de 200, sem mudança, abrupto gradual e deslocamento abrupto; na janela de 400, sem mudança e deslocamento.

Apesar do Adaboost e da CDC terem métricas parecidas, percebe-se uma diferença analisando as matrizes de confusão delas. Isto é, a CDC apresenta, aproximadamente, a mesma taxa de acerto para todas as classes, enquanto que o Adaboost tem taxas de acertos alta para certas classes e baixa para outras.

Por isso, a comparação com outros detectores foi feita utilizando a CDC e com um valor de tamanho de janela 200. Essa escolha se deve a análise realizada, na qual inferiu-se que os três modelos tiveram melhores métricas nessa escolha de tamanho.

5.2.3 Tarefa de Agrupamento

Na subseção 5.2.3.1 comparam-se os detectores na base sintética criada e que foi parte utilizada para treinar o modelo. Já na seção 5.2.3.2, eles são comparados com duas bases reais.

Os resultados indicam que o CDC, na base sintética, foi superior aos outros detectores ao encontrar desvios de conceito onde existiam. Já nas bases reais, o CDC detectou mais desvios que os outros, entretanto, analisando a silhueta, ele obteve o melhor valor para as duas bases.

5.2.3.1 Base sintética

As Tabelas 32 e 33 mostram os erros e acertos dos detectores. Analisando essas tabelas e a Tabela 31, percebe-se que, nesse conjunto, o CDC foi superior aos outros dois algoritmos em quase todas as detecções, errando, apenas, na detecção de desvio onde não existia.

	Sem Desvio	Com Desvio
Sem Desvio	926	274
Com Desvio	322	9278

Tabela 31 – CDC com Conjunto de Teste

	Sem Desvio	Com Desvio
Sem Desvio	1109	91
Com Desvio	9020	580

Tabela 32 – ADWIN com Conjunto de Teste

	Sem Desvio	Com Desvio
Sem Desvio	690	510
Com Desvio	5969	3631

Tabela 33 – PHT com Conjunto de Teste

5.2.3.2 Bases reais

As tabelas 34 e 35 comparam os algoritmos com os conjuntos calculando a média da silhueta obtida com cada ponto durante o fluxo de dados. Para o conjunto NOAA, o CDC detectou mais desvios que os outros, teve desempenho igual ao PHT e melhor que a base e ADWIN. Já para o Ozone, ele continuou detectando mais desvios de conceito, mas apenas um pouco mais. Além disso, ele obteve um desempenho igual ao ADWIN, que foi melhor que a base e o PHT.

5.2.4 Tarefa de Classificação

Os resultados indicam que o CDC, na base sintética, foi superior aos outros detectores ao encontrar desvios de conceito onde existiam e obtendo a melhor acurácia em cada

	Quantidade de Desvio	Silhueta
Base	Não se aplica	0,18
ADWIN	17	0,34
PHT	32	0,36
CDC	88	0,36

Tabela 34 – Teste com NOAA.

	Quantidade de Desvio	Silhueta
Base	Não se aplica	0,18
ADWIN	5	0,33
PHT	1	0,30
CDC	9	0,33

Tabela 35 – Teste com Ozone.

conjunto. Já nas bases reais, o CDC, geralmente, detectou mais desvios que os outros e obteve uma acurácia um pouco menor que os outros detectores.

5.2.4.1 Bases Sintéticas

A Tabela 36 mostra a quantidade de desvios encontrados por cada detector em cada conjunto de dados e média da acurácia ao longo do tempo. Em quase todos, o CDC obteve o melhor desempenho, mas, também, foi o que detectou mais desvios. O conjunto tem desvio a cada 400 pontos e são 16.000 pontos, o que resulta em 40 desvios. Nas bases 1CDT e 1CHT, todos os detectores, exceto o CDC, encontraram um número bem inferior de desvios comparando com o total esperado.

	1CDT Desvio	1CDT Acurácia	2CDT Desvio	2CDT Acurácia	1CHT Desvio	1CHT Acurácia	2CHT Desvio	2CHT Acurácia
ADWIN	3	98%	17	85%	2	95%	15	80%
PHT	3	98%	17	86%	2	96%	15	81%
DDM	6	99%	64	91%	9	97%	40	85%
EDDM	2	99%	39	92%	3	98%	55	88%
CDC	79	99%	79	94%	79	97%	78	88%

Tabela 36 – Comparação de desvios e acurácia utilizando detectores nos conjuntos sintéticos.

As figuras 5.2.1 a 5.2.4 são referentes aos conjuntos 1CDT, 2CDT, 1CHT e 2CHT, respectivamente, e mostram a acurácia ao longo do tempo.

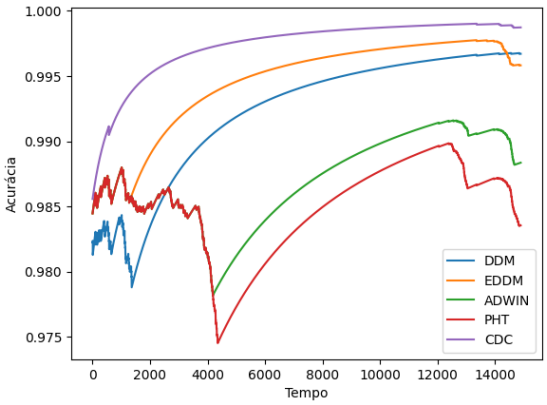

Figura 5.2.1 – Acurácia no conjunto 1CDT.

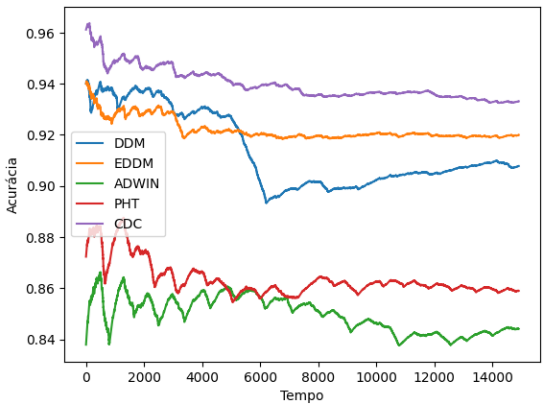

Figura 5.2.2 – Acurácia no conjunto 2CDT.

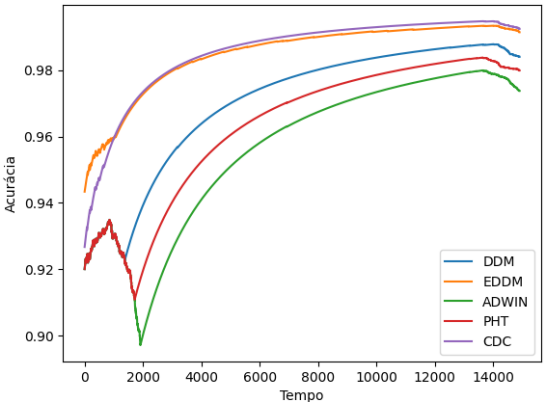

Figura 5.2.3 – Acurácia no conjunto 1CHT.

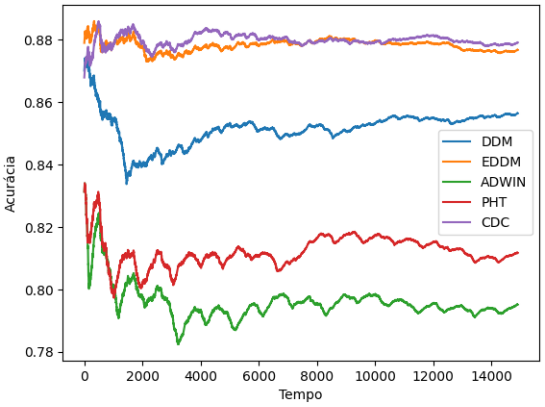

Figura 5.2.4 – Acurácia no conjunto 2CHT.

5.2.4.2 Bases Reais

As figuras 5.2.5 e 5.2.6 ilustram a acurácia obtida ao longo da classificação do conjunto Ozone e NOAA, respectivamente. No primeiro, a acurácia começa alta e vai diminuindo, mas, ao longo, ela aumenta gradualmente. No último, o CDC já começa com uma acurácia pior que as outras e aumenta.

As tabelas 37 e 38 mostram os desvios e a acurácia total. Nos dois conjuntos, o CDC foi o segundo que mais detectou desvio de conceito e não obteve o melhor resultado de acurácia, tendo ficado em último no conjunto NOAA. Nele, a maior quantidade de desvio detectado do CDC não o ajudou a superar a acurácia do PHT, que detectou quase 11 vezes menos.

	Desvio	Média Acurácia
DDM	8	92%
EDDM	4	91%
ADWIN	1	90%
PHT	0	89%
CDC	6	90%

Tabela 37 – Comparação com o conjunto Ozone

	Desvio	Média Acurácia
DDM	10	68%
EDDM	99	71%
ADWIN	7	72%
PHT	7	69%
CDC	88	67%

Tabela 38 – Comparação com o conjunto NOAA

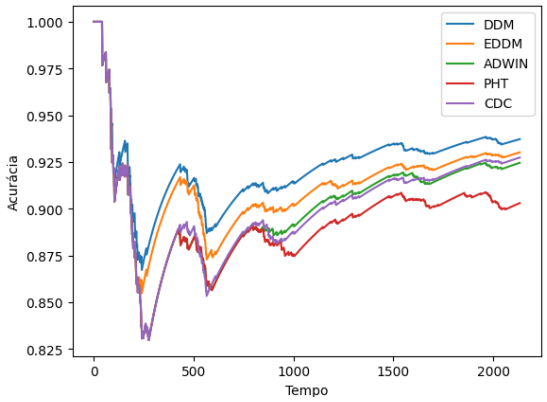

Figura 5.2.5 – Acurácia no conjunto Ozone.

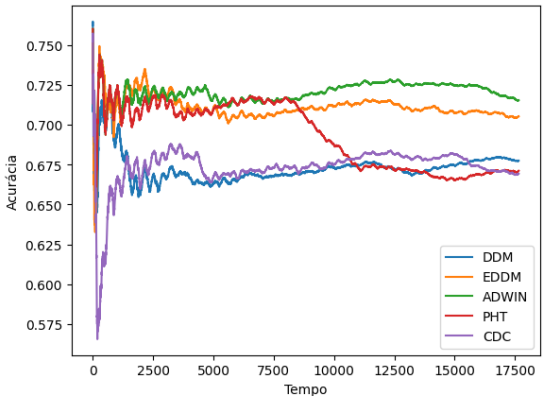

Figura 5.2.6 – Acurácia no conjunto NOAA.

5.2.5 Identificação do Tipo de Desvio

Na Tabela 39 estão presentes a quantidade e a classe dos desvios encontrados nas bases sintéticas: 1CDT, 2CDT, 1CHT e 2CHT. A maior dos desvios encontrados pelo CDC foi identificada como deslocamento abrupto, mas também houve janelas classificadas como adição e separação abrupta. A maior parte dos desvios foi classificada como abrupta.

	AA	AG	SA	SG	FA	FG	DA	DG
1CDT	15	0	14	0	0	0	46	4
2CDT	6	0	19	0	0	0	52	2
1CHT	9	0	11	0	0	0	57	2
2CHT	8	1	2	0	0	0	63	4

Tabela 39 – Classificação dos desvios encontrados pelo CDC nas bases sintéticas.

5.3 Considerações Finais

Ao longo deste capítulo foram mostrados diversos testes realizados para validar o algoritmo desenvolvido junto com o conjunto de dados criado. Embora haja espaço para melhoria, o CDC foi capaz de ter um desempenho equiparável com outros detectores tanto na tarefa de agrupamento quanto na de classificação.

6

Conclusão

Este trabalho apresentou um algoritmo capaz de detectar o desvio de conceito utilizando a tarefa de agrupamento, identificar o seu tipo e retornar o resultado do desvio nos grupos. Iniciou-se com uma revisão bibliográfica com os trabalhos relacionados, ou seja, que tratam de detecção de desvio e sua identificação, tanto na classificação como no agrupamento.

Na sequência, foi realizado um estudo com os diferentes tipos de desvio para analisá-los e encontrar padrões que os diferencie e os identifique no agrupamento. A partir disso, foram escolhidos dois tipos de desvio: o abrupto e o gradual. Além disso, durante os estudos, percebeu-se que poderia ser possível identificar o que ocorreu nos grupos em razão do desvio.

Para constatar que a solução pensada é viável, foi criado um conjunto de dados para treino e foram escolhidos conjuntos sintéticos e reais para testá-la. Ademais, os experimentos com o conjunto sintético criado foram realizados comparando a rede prototípica com 1-NN e Adaboost.

Outro ponto a ser observado, era se o algoritmo treinado no conjunto sintético teria um bom desempenho ao detectar desvio de conceito com dados reais no agrupamento. Essa comparação ocorreu junto com outros detectores: Adwin e PHT. O algoritmo proposto localizou mais desvios de conceitos e obteve o melhor valor de silhueta.

Além disso, o CDC foi testado também na tarefa de classificação com conjuntos sintéticos e reais. Nesse experimento, o CDC foi comparado com ADWIN, PHT, DDM e

EDDM. Nos conjuntos sintéticos, ele obteve uma boa acurácia, mas, nos reais, ele não foi bom.

Por fim, este trabalho trouxe uma nova visão para detecção de desvio utilizando a tarefa de agrupamento. Com todos os experimentos realizados, pode-se notar que o CDC consegue realizar o que se propõe, mas ainda há espaço para melhorias.

6.1 Dificuldades encontradas

Ao longo do experimento foram encontradas diversas dificuldades. Uma delas foi encontrar artigos que detectam o tipo de desvio de conceito e quais as métricas mais comuns para isso, visto que esse é o objetivo deste trabalho. Entretanto, existem poucos artigos sobre esse assunto.

Outra adversidade foi que a maioria dos conjuntos de dados reais que são utilizados para analisar o desempenho de um algoritmo em um ambiente com desvio de conceito não tem a informação de quantos desvios existem, dos instantes em que eles ocorrem, tampouco dos tipos de desvios. Além disso, a maioria dos conjuntos sintéticos são utilizados para classificação. Por isso, foi necessário gerar um conjunto sintético que tivesse as caraterísticas necessárias para este trabalho, possibilitando a identificação do local e o tipo do desvio em cada conjunto, colaborando no desenvolvimento do algoritmo.

Essa geração do conjunto de dados foi um desafio, visto que são 9 classes e o conjunto tem que ter exemplo de todos. Então, foi realizado um trabalho para fazer os fluxos terem desvio abrupto e gradual, além de ter diferentes mudanças no agrupamento (adição, separação, fusão e deslocamento de grupo).

6.2 Contribuições

Pelo experimento realizado empiricamente, o algoritmo desenvolvido conseguiu identificar se houve ou não desvio de conceito e, se sim, identificar se o desvio é abrupto ou gradual e o que aconteceu com o agrupamento: adição, separação, fusão ou desloca-

mento. Além disso, ele foi capaz de aprender os padrões dos tipos de desvios utilizando uma base sintética no treinamento e, depois, detectou desvio em outras bases. O CDC pode ser utilizado nas tarefas de agrupamento e classificação.

Mais estudos e experimentos precisam ser realizados para que se possa melhorar o desempenho do CDC e, talvez, detectar mais tipos de desvio e outras informações a fim de ajudar o usuário a melhorar a resposta ao desvio de conceito no seu fluxo de dados.

6.3 Trabalhos Futuros

Embora diversos experimentos tenham sido realizados, ainda existem cenários e modificações no algoritmo a serem testadas. Os conjuntos sintéticos utilizadas para treino e teste são balanceados, ou seja, as proporções das classes são parecidas. Outra característica em comum nesses conjuntos é que eles possuem apenas 2 classes. Então, experimentos que mexam nessas configurações podem ser realizados para testar o desempenho do modelo, tanto para conjuntos desbalanceados, quanto para conjuntos com mais de 2 classes. Para isso, uma ideia é utilizar uma janela de fluxo para cada centroide, o que poderia viabilizar a detecção de desvio quando o grupo desaparece.

Outro ponto a se considerar é o uso de outra métrica que não seja distância do ponto ao centroide ou silhueta. Por exemplo, pode-se utilizar a distância entre centroides realizada a partir de dois agrupamentos consecutivos, verificando o quanto que ele mudou ao agrupar novos dados. Além disso, só foi utilizado o k-means para obter os centroides que são utilizados no cálculo das distâncias. Assim, podem ser testados outros algoritmos de agrupamento. Outra ideia é utilizar um ensemble para melhorar o desempenho geral do algoritmo, juntando diferentes modelos para detectar o tipo de desvio.

Desse modo, o algoritmo desenvolvido pode melhorar na sua tarefa e generalizar ainda mais para ter um bom desempenho em diferentes cenários e dados. Assim, os usuários terão mais informações para reagir aos desvios de conceito que acontecem nos dados com que eles trabalham.

REFERÊNCIAS

AGGARWAL, C. C. et al. A framework for clustering evolving data streams. In: ELSEVIER. *Proceedings 2003 VLDB conference*. [S.l.], 2003. p. 81–92. 38

ARTHUR, D.; VASSILVITSKII, S. et al. k-means++: The advantages of careful seeding. In: *Soda*. [S.l.: s.n.], 2007. v. 7, p. 1027–1035. 38

BAENA-GARCIA, M. et al. Early drift detection method. In: *Fourth international workshop on knowledge discovery from data streams*. [S.l.: s.n.], 2006. v. 6, p. 77–86. 25

BIFET, A.; GAVALDA, R. Learning from time-changing data with adaptive windowing. In: SIAM. *Proceedings of the 2007 SIAM international conference on data mining*. [S.l.], 2007. p. 443–448. 26

CASTELLANI, A.; SCHMITT, S.; HAMMER, B. Task-sensitive concept drift detector with constraint embedding. In: IEEE. *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. [S.l.], 2021. p. 01–08. 17, 45, 49, 57

CUNNINGHAM, P.; DELANY, S. J. K-nearest neighbour classifiers-a tutorial. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 54, n. 6, p. 1–25, 2021. 29

DINO. *Cresce a procura por opções mais saudáveis na alimentação*. 2023. <https://valor.globo.com/patrocinado/dino/noticia/2023/03/31/cresce-a-procura-por-opcoes-mais-saudaveis-na-alimentacao.ghtml>. Accessed: 2024-08-15. 16

DITZLER, G.; POLIKAR, R. Incremental learning of concept drift from streaming imbalanced data. *IEEE transactions on knowledge and data engineering*, IEEE, v. 25, n. 10, p. 2283–2301, 2012. 55

DUA, D.; GRAFF, C. *UCI machine learning repository*. 2017. Disponível em: <http://archive.ics.uci.edu/ml>. 56

FREUND, Y.; SCHAPIRE, R. E. A desicion-theoretic generalization of on-line learning and an application to boosting. In: SPRINGER. *European conference on computational learning theory*. [S.l.], 1995. p. 23–37. 32

GAMA, J. et al. Learning with drift detection. In: SPRINGER. *Brazilian symposium on artificial intelligence*. [S.l.], 2004. p. 286–295. 24

GAMA, J.; SEBASTIAO, R.; RODRIGUES, P. P. On evaluating stream learning algorithms. *Machine learning*, Springer, v. 90, p. 317–346, 2013. 28

GUO, H. et al. Concept drift type identification based on multi-sliding windows. *Information Sciences*, Elsevier, v. 585, p. 1–23, 2022. 17, 47, 49

IWASHITA, A. S.; PAPA, J. P. An overview on concept drift learning. *IEEE access*, IEEE, v. 7, p. 1532–1547, 2018. 22

JR, P. M. G. et al. A comparative study on concept drift detectors. *Expert Systems with Applications*, Elsevier, v. 41, n. 18, p. 8144–8156, 2014. 16

LU, J. et al. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 31, n. 12, p. 2346–2363, 2018. 20, 23

MAHESH, B. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, v. 9, p. 381–386, 2020. 16

MITCHELL, T. *Machine learning*. [S.l.]: McGraw-hill New York, 1997. v. 1. 29

MÜLLER, A. C.; GUIDO, S. *Introduction to machine learning with Python: a guide for data scientists*. [S.l.]: "O'Reilly Media, Inc.", 2016. 66

NAMITHA, K.; KUMAR, G. S. Cusum based concept drift detector for data stream clustering. In: *BDIOT*. [S.l.: s.n.], 2020. p. 90–95. 17, 24, 44, 49, 57

NAMITHA, K. et al. Concept drift detection in data stream clustering and its application on weather data. *International Journal of Agricultural and Environmental Information Systems (IJAEIS)*, IGI Global, v. 11, n. 1, p. 67–85, 2020. 45, 49, 57

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, JMLR. org, v. 12, p. 2825–2830, 2011. 51

PEÑA, M. et al. Data-driven gearbox fault severity diagnosis based on concept drift. In: IEEE. *2021 ieee fifth ecuador technical chapters meeting (etcm)*. [S.l.], 2021. p. 1–6. 44, 49

RASCHKA, S.; MIRJALILI, V. *Python Machine Learning*. [S.l.]: Packt, 2017. 33

ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, Elsevier, v. 20, p. 53–65, 1987. 39

SAXENA, A. et al. A review of clustering techniques and developments. *Neurocomputing*, Elsevier, v. 267, p. 664–681, 2017. 36

SCHLIMMER, J. C.; GRANGER, R. H. Incremental learning from noisy data. *Machine learning*, Springer, v. 1, n. 3, p. 317–354, 1986. 17

SNELL, J.; SWERSKY, K.; ZEMEL, R. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, v. 30, 2017. 34, 35

SOUZA, V. M. et al. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, Springer, v. 34, n. 6, p. 1805–1858, 2020. 55

SOUZA, V. M. et al. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: SIAM. *Proceedings of the 2015 SIAM international conference on data mining*. [S.l.], 2015. p. 873–881. 53

WEBB, G. I.; KEOGH, E.; MIIKKULAINEN, R. Naïve bayes. *Encyclopedia of machine learning*, v. 15, n. 1, p. 713–714, 2010. 30

YANG, L. et al. {CADE}: Detecting and explaining concept drift samples for security applications. In: *30th USENIX Security Symposium (USENIX Security 21)*. [S.l.: s.n.], 2021. p. 2327–2344. 17, 45, 49, 57

YU, H. et al. Automatic learning to detect concept drift. *arXiv preprint arXiv:2105.01419*, 2021. 46, 47, 49

ZUBAROĞLU, A.; ATALAY, V. Data stream clustering: a review. *Artificial Intelligence Review*, Springer, v. 54, n. 2, p. 1201–1236, 2021. 36