DOI: 10.48011/asba.v2i1.1233

Abordagem Espacial via Quadtree para Detecção de Mudança de Conceito em Fluxos de Dados Contínuos

Rodrigo A. Coelho * **Cristiano L. de Castro** *

* *Programa de Pós-Graduação em Engenharia Elétrica - Universidade Federal de Minas Gerais - Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brasil, (e-mail: toluenotnt@gmail.com; crislcastro@ufmg.br).*

Abstract: During online learning, distribution that generates data stream can change over time. It has been a challenge in the literature to monitor and detect these changes in order to preserve the learning algorithm's performance. This work presents a drift detection method based on the quadtree recursive division structure. In contrast to other methods based on statistical techniques, the proposed algorithm uses the data spatial arrangement to detect concept drift of data. To evaluate the proposed algorithm, experiments were conducted with real and synthetic datasets. The results are promising, pointing out that our model achieves good performance when compared with drift detection methods commonly adopted in literature.

Resumo: No aprendizado online nem sempre os dados se comportam de forma estacionária. Monitorar e detectar as mudanças no fluxo de dados, para preservar o desempenho do algoritmo de aprendizado é um desafio. Este trabalho apresenta um método de detecção de mudança de conceito baseado na estrutura de divisão recursiva do espaço da *quadtree*. Diferentemente dos métodos de detecção baseados em técnicas estatísticas, o modelo proposto utiliza da disposição espacial dos dados para detectar mudanças de conceito. Experimentos utilizando bases de dados reais e sintéticas foram realizados para confrontar nossa proposta com outros métodos do estado da arte, mostrando resultados promissores.

Keywords: Online learning; classification; data stream; concept drift; quadtree.

Palavras-chaves: Aprendizado online; classificação; fluxo de dados; mudança de conceito; quadtree.

1. INTRODUÇÃO

O aprendizado online se refere ao processo de extrair informação de um fluxo de dados possivelmente ilimitado em que as observações são apresentadas em função do tempo, de forma que não é possível revisitar os dados já apresentados. O fluxo de dados nem sempre apresenta um comportamento estacionário, mudando a distribuição dos dados com o tempo. Este comportamento é caracterizado pela mudança da função geradora dos dados, sendo conhecido na literatura como mudança de conceito dos dados. A mudança de conceito prejudica os algoritmos de aprendizado online, deteriorando a sua acurácia, já que não foram desenvolvidos para esta situação (Gama et al., 2014).

O detector de mudança de conceito é responsável por alertar o algoritmo de aprendizado online quando uma mudança acontece, para que uma estratégia de atualização ou reconstrução seja utilizada. Lidar com a mudança de conceito tem sido um problema multidisciplinar (Khamassi et al., 2018) e foco de diversos trabalhos (Gama et al., 2014; Khamassi et al., 2018; Barros and Santos, 2018).

Diferentes métodos têm sido aplicados para a implementação de um detector de mudança de conceito, como *Wilcoxon Rank Sum Test Drift Detector* (WSTD) (de Barros et al., 2018) que utiliza do teste de hipótese para comparação de duas distribuições, *Cosine Similarity Drift Detector* (CSDD) (Hidalgo et al., 2019) que usa o teste de similaridade de cossenos e, ExStream (Demšar and Bosnić, 2018) que utiliza do teste de Page–Hinkley combinado com algoritmos de controle estatístico de processo. Um ponto em comum nestes métodos de detecção de mudança de conceito é o fato de utilizarem técnicas estatísticas baseadas em análise de sequência. Até mesmo os métodos de detecção baseados em algoritmos para diferentes janelas, também utilizam de técnicas estatísticas como pode ser visto em *Adaptive WINdowing* (ADWIN) (Bifet and Gavalda, 2007) e *Error Distance Based Approach for drift Detection and Monitoring* (EDIST) (Khamassi and Sayed-Mouchaweh, 2014).

Métodos de detecção de mudanças de conceito baseados em técnicas estatísticas geralmente assumem que os dados são estacionários. Sejam os dados apresentados pelo fluxo ou as saídas (erros) do algoritmo de aprendizado, estes têm que apresentar a propriedade de que a estrutura de autocorrelação não mude com o decorrer do tempo. A mudança desta autocorrelação é interpretada como mudança no conceito dos dados.

Este trabalho aborda um novo ponto de vista sobre a detecção de mudanças de conceito, ao apresentar um detector baseado unicamente na disposição espacial dos dados por meio da estrutura de divisão recursiva do espaço da *quadtree*. Esse novo método assume que a relação entre dados e espaço é imutável, e que qualquer mudança nesse contexto pode ser considerada uma mudança de conceito dos dados. Experimentos utilizando de bases de dados reais e sintéticas foram realizados para confrontar o método proposto frente a outros, mostrando resultados promissores.

O presente trabalho está organizado da seguinte forma. A próxima seção apresenta a descrição do problema de pesquisa. A seção 3 apresenta a base teórica do método proposto. Na seção 4 é feita a avaliação do método proposto, usando de outros métodos de detecção de mudança de conceito em conjuntos de dados sintéticos e reais. A seção 5 conclui o trabalho e apresenta os trabalhos futuros.

2. DESCRIÇÃO DO PROBLEMA

O fluxo de dados pode ser definido da seguinte forma: seja um fluxo de dados $DS_t \in \mathcal{R}^d$ uma sequência de dados de d dimensões ordenados sobre o tempo $DS_t = (x_1, x_2, \ldots, x_t, \ldots)$, em que x_t é o mais novo exemplo de dado a ser apresentado. Cada exemplo x_t está associado a um rótulo $y_t \in Y$ em que $Y = \{0, 1\}$.

A mudança no conceito dos dados ocorre quando a relação entre os dados de entrada e a variável de saída muda ao longo do tempo. Seja DS_t um fluxo de dados com a seguinte função de distribuição de probabilidade conjunta $\prod_{f1}$, onde $\hat{y}_t = f_t(x_t)$ é o modelo atual capaz de atribuir corretamente o exemplo de entrada ao rótulo de saída ($\hat{y}_t = y_t$) até o instante t. A mudança de conceito acontece quando a atual função geradora dos dados muda para uma função de distribuição de probabilidade conjunta distinta $\prod_{f2}$, de forma que DS_{t+1} representa o fluxo de dados após a mudança de conceito. É provável que o classificador atual tenha seu desempenho afetado, de forma que ele não consiga classificar corretamente a amostra x_{t+1}, ou seja, $\hat{y}_{t+1} \neq y_{t+1}$.

De acordo com Gama et al. (2014), existem dois tipos de mudança de conceito, o virtual e o real, conforme ilustrado na Figura 1. A mudança de conceito virtual não exige mudança na margem de separação entre as classes, já que o classificador mantém sua acurácia. Já a mudança de conceito real exige uma mudança na margem de separação entre as classes para que a acurácia do classificador seja mantida, ação que deve ser desencadeada pelo detector de mudança de conceito.

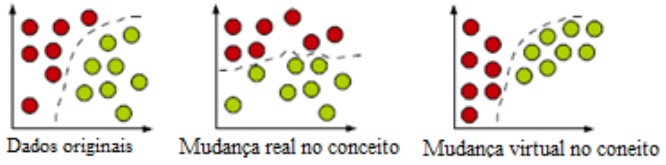

Figura 1. Mudança de conceito real e virtual (Gama et al., 2014)

A mudança de conceito real dos dados pode ocorrer das seguintes formas: abrupta, incremental, gradual e recorrente. O módulo de detecção de mudanças deve ser capaz de identificar e indicar a melhor ação a ser tomada. A Figura 2 ilustra os principais tipos de mudança de conceito.

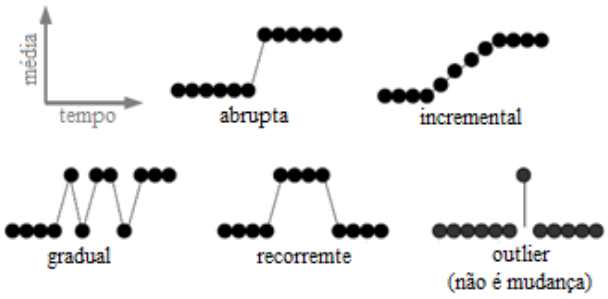

Figura 2. Tipos de mudança de conceito (Gama et al., 2014)

Na literatura, os métodos de detecção de mudança de conceito dos dados têm sido desenvolvidos para atuar de forma dependente ou independente do algoritmo de aprendizado online. Os métodos que utilizam do algoritmo de aprendizado online monitoram o seu desempenho, e ao identificar uma mudança significativa em seu desempenho acusam a mudança no conceito. Por meio de modelos estatísticos, esses métodos de detecção de mudança assumem que os resultados apresentados pelo algoritmo de aprendizado online possuem as características de um processo estacionário e os dados do fluxo são independentes e distribuídos de forma idêntica. Embora essas hipóteses não sejam válidas no caso de um fluxo de dados, vários trabalhos têm a utilizado e apresentado bons resultados.

Dentre os trabalhos cujos métodos de detecção de mudança dependem do algoritmo de aprendizado online, alguns merecem destaque por serem considerados estados da arte, como o *Drift Detection Method* (DDM) (Gama et al., 2004) e o *Early Drift Detection Method* (EDDM) (Baena-Garcıa et al., 2006) que utilizam de técnicas estatísticas baseadas na distribuição de Bernoulli em seus detectores.

Os métodos de detecção de mudança que não dependem do algoritmo de aprendizado online utilizam de duas janelas de dados distintas e uma técnica estatística para comprará-las. Estes métodos de detecção geralmente são paramétricos e precisam saber a priori o tipo de alteração a ser detectada, como por exemplo: alteração na média, na variância, nos quantis, entre outras ou a combinação destas. Como exemplo, os trabalhos de Gama (2010); Mahdi et al. (2020), que utilizam de um teste de comparação mais elaborado, como o teste de Page-Hinkley.

Uma estratégia comumente aplicada que favorece o uso de técnicas estatísticas para comparação de distribuições é o uso de janelas deslizantes de tamanhos variáveis. A janela pode ser definida como $W[n] = (x_{t-n}, x_{t-(n-1)}, x_{t-(n-2)} \ldots, x_t)$ em que n é a quantidade de dados que podem ser armazenados. Tanto métodos de detecção que dependem ou independem do algoritmo de aprendizado online costumam utilizar desta estrutura. Como trabalhos bem conhecidos na literatura que utilizam de técnicas baseadas na diferença estatística entre duas janelas deslizantes de tamanhos diferentes tem-se o *ADaptive WINdowing*

(ADWIN) (Bifet and Gavalda, 2007) e *Error Distance Based Approach for drift Detection and Monitoring* (EDIST) (Khamassi and Sayed-Mouchaweh, 2014).

3. O MÉTODO PROPOSTO

O método de detecção de mudança de conceito proposto é baseado na ocupação espacial dos dados e na divisão recursiva do espaço por meio da *quadtree*. Através de uma estratégia de verificação dos espaços delimitados pela *quadtree*, é possível identificar uma mudança de conceito sem o uso de técnicas estatísticas.

3.1 Quadtree

A *quadtree* é uma estrutura de dados do tipo árvore, que possui arranjos hierárquicos espaciais que se baseiam no princípio da decomposição recursiva do espaço. Por meio da decomposição recursiva do espaço $\mathcal{R}^2$, usando de separadores paralelos aos eixos das coordenadas e abscissas, tem-se como resultado quatro regiões (quadrantes) de igual dimensão. O termo "*quad*" surge da divisão resultante do espaço do nó raiz, que resulta em quatro nós folha com regiões congruentes. A *quadtree* admite apenas 1 dado por região, ou seja, somente os nós folha possuem dados. Ao inserir um segundo dado, o nó é dividido recursivamente em 4 nós filhos, e cada dado deve ser mapeado para o correto nó folha, conforme visto na Figura 3 (Mehta and Sahni, 2004).

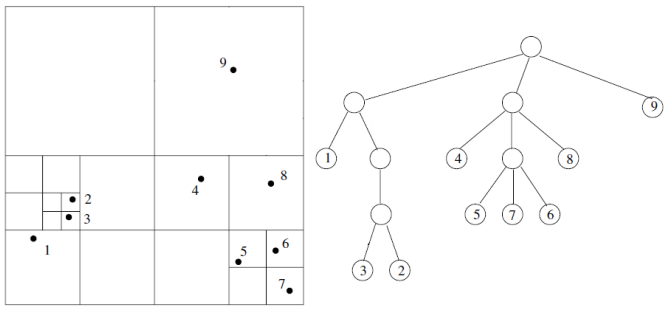

Figura 3. Estrutura de dados de uma *quadtree*

Para uma estrutura de dados com mesmo funcionamento em $\mathcal{R}^3$, dá-se o nome de *octree*. Nesta árvore, a decomposição recursiva do espaço do nó raiz resulta em oito hipercubos congruentes. À medida em que a dimensionalidade do espaço $\mathcal{R}^d$ aumenta, para um $d > 3$, cada nova recursão da estrutura de dados resulta em um aumento de 2^d espaços congruentes. É prática usual a aplicação do nome *quadtree* independentemente da dimensionalidade da estrutura, com complexidade de tempo para construção da árvore de $O(dn \log n)$.

3.2 Quadtree para $d > 3$

A construção de uma *quadtree* para 2 dimensões consiste na divisão recursiva de uma região em quatro regiões correspondentes aos quadrantes sudoeste, noroeste, sudeste e nordeste, os quais são mapeados para cada um dos nós filhos. A construção de uma *quadtree* para mais alta dimensão, $d > 3$, exige um mapeamento mais complexo para os nós filhos. Para contornar este problema, este trabalho propõe o mapeamento binário do espaço a partir de uma função *hash* para mapear cada um dos 2^d diferentes espaços congruentes.

Com função *hash* é feito um cálculo a partir do ponto médio do espaço a ser dividido, atribuindo o valor binário "0" e "1" para compor a chave que vai endereçar o novo espaço. São comparados os valores de mesma posição dos vetores do ponto médio do espaço e a posição do vetor de dados. É atribuído o valor binário "0" se o valor da observação for menor que o ponto médio, e o valor binário "1" se esta for maior, dessa forma criando uma chave binária com tamanho igual a dimensão do espaço de características dos dados. Por exemplo, para um espaço $d = 4$ a sua divisão recursiva resulta em $2^4 = 16$ novos espaços. Cada um desses espaços é endereçado de forma binária pela composição de um número de *d bits*, ou seja, composto de uma *string* de 4 *bits* capaz de endereçar de "0000" a "1111". Para um ponto médio do espaço igual a $[0.5, 0.5, 0.5, 0.5]$ e um dado $[0.3, 0.6, 0.8, 0.1]$ temos como chave a *string* "0110". O Algoritmo 1 formaliza o processo acima descrito.

Algoritmo 1 Pseudocódigo para construção das chaves que mapeiam os dados em uma *quadtree* de d dimensões.

```
função CHAVE HASH(centro[ ], dado[ ])
    chave ← " "
    para i em length(centro) faça
        se dado[i] < centro[i] então
            chave + "0"
        senão
            chave + "1"
    retorne chave
```

3.3 Detector de mudança

O detector de mudança proposto segue uma abordagem a partir da ocupação espacial dos dados em uma estrutura de *quadtree*. A *quadtree* é capaz de manter uma certa quantidade de dados dependendo de sua altura h e dimensão d, agindo como uma estrutura de memória. Cada recursão da *quadtree* é capaz de armazenar 2^d dados por região congruente existente. Para uma *quadtree* de $d = 2$ dimensões e altura $h = 3$, temos que com uma $h = 0$ a sua raiz é capaz de armazenar apenas 1 dado, com $h = 1$ é possível armazenar até 4 dados, com $h = 2$ até 16 dados e com $h = 3$ até 64 dados. Desta forma tem-se, que o aumento na altura da *quadtree*, é proporcional a quantidade de memória gasta e a qualidade (resolução) com que o espaço pode ser dividido e representado.

A fim de limitar a qualidade com que o espaço pode ser representado e a quantidade máxima de memória que pode ser gasta por esta estrutura, uma altura h máxima deve ser definida. Ao determinar uma altura máxima, dados que estiverem em um nó folha, que não pode ser recursivamente dividido, serão reduzidos a um só ponto com o valor do ponto médio de seus dados, como ilustrado na Figura 4.

O princípio básico por traz da utilização da estrutura de *quadtree* como detector de mudança de conceito consiste na ideia de que dados de classes opostas não possam ocupar o mesmo espaço, ou seja, não possam ocupar o mesmo nó folha. Caso os dados presentes na *quadtree* se

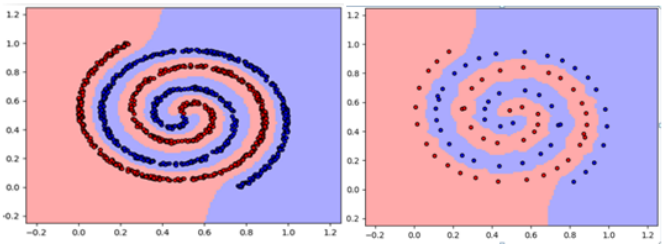

Figura 4. Dados brutos à esquerda e a sua representação via *quadtree* à direita.

sobreponham, significa que o conceito dos dados mudou a ponto de uma classe passar a ocupar o mesmo espaço da outra classe.

O funcionamento deste método de detecção de mudança de conceito consiste em armazenar na *quadtree* os mesmos dados utilizados para o pré-treino ou treino incremental do classificador. Desta forma, assim que inicializa o processo de classificação dos dados apresentados pelo fluxo, o método de detecção de mudança está pronto para o uso.

O detector de mudança de conceito é acionado somente em caso de erro do classificador. Seja DS_t um fluxo de dados no instante de tempo t, em que a função de classificação $\hat{y}_t = f_t(x_t)$ foi capaz de classificar corretamente os dados até o instante t. No instante de tempo $t+1$, o classificador apresentou um erro $\hat{y}_{t+1} \neq f_t(x_{t+1})$, acionando assim o detector de mudança de conceito. O dado que causou o erro é inserido na *quadtree* e então uma ação é proposta, dependendo do local em que o dado ficou na árvore. As ações implementadas foram duas:

- A primeira ação consiste em uma advertência, a qual sugere que o atual classificador deva ser melhorado incrementalmente, o dado inserido na árvore ocupou um lugar vazio, um nó folha vazio. Esta situação pode indicar uma possível mudança de conceito gradual, ou pendência no treinamento, no caso de um treinamento incremental, de forma que o atual classificador tem capacidades de aprender com esse novo dado.
- A segunda ação implementada consiste em uma situação real de mudança de conceito, quando o dado inserido na árvore foi alocado em um nó folha que não suporta divisões recursivas e já existe um dado de outra classe. Nesta situação, o atual classificador é incapaz de manter a qualidade da classificação, e então o treinamento de um novo classificador é recomendado a partir dos dados mais atuais.

O Algoritmo 2 formaliza o processo de detecção de mudança de conceito acima descrito.

Algoritmo 2 Detector de mudança de conceito

```
função DETECTOR(arvore, dado[ ])
    \\ valor da folha em que dado vai ser inserido
    _folha ← arvore.folha(dado)
    se _folha == vazia então
        retorne atualiza classificador
    senão
        retorne novo classificador
```

4. EXPERIMENTOS

Esta seção descreve toda a configuração experimental, incluindo os detectores utilizados, algoritmos de aprendizado, bases de dados e a forma de avaliação utilizada.

4.1 Detectores de Mudança de Conceito

Para a comparação do método proposto, foram utilizados os seguintes métodos de detecção de mudança: *Drift Detection Method* (DDM) (Gama et al., 2004), *Early Drift Detection Method* (EDDM) (Baena-Garcıa et al., 2006), *ADaptive WINdowing* (ADWIN) (Bifet and Gavalda, 2007) e Page-Hinkley (PH) (Gama, 2010; Mahdi et al., 2020) bem conhecidos na literatura. Estes métodos fazem parte do *framework scikit-multiflow* [1] .

Os parâmetros utilizados foram os mesmos recomendados pelos autores: os métodos DDM e EDDM com ($min\_num\_instances = 30$, $warning\_level = 2.0$, $out\_control\_level = 3.0$), ADWIN ($delta = 0.002$) e PH ($min\_instances = 30$, $delta = 0.005$, $threshold = 50$, $alpha = 0.9999$). O método proposto (QT) tem como parâmetro a altura h da *quadtree*, que representa a quantidade máxima de possíveis divisões recursivas para o espaço, ajustado com os valores QT $h = 4$ e QT $h = 6$.

4.2 Algoritmos de Aprendizado

Foram utilizados dois algoritmos de aprendizado distintos junto com os detectores de mudança: *Support Vector Machine* (SVM) e *Random Forest* (RF), que fazem parte do *framework scikit-learn* [2] . Os parâmetros utilizados foram: Para o SVM com ($C = 1.0$, $kernel = 'linear'$) onde C é o fator de regularização. O RF utilizou ($n\_estimators = 20, max\_depth = 2, random\_state = 0$) em que $n\_estimators$ é o número de árvores na floresta, max_depth a profundidade das árvores e $random\_state$ controla a aleatoriedade do *bootstrapping* das amostras usadas na construção de árvores.

4.3 Bases de Dados

Para realização dos experimentos foram utilizadas 4 bases de dados, sendo uma real. A base de dados *Electricity* (Zliobaite, 2013) é formada a partir de dados coletados no mercado de eletricidade da Austrália. Neste mercado, os preços flutuam de acordo com a oferta e demanda do mercado definidos a cada cinco minutos. Esta base possui 45312 instâncias de 8. Não foi informado o tipo de mudança de conceito que a base apresenta. A base de dados SEA (Street and Kim, 2001) possui 60000 instâncias de 3 dimensões. Ela contém duas classes desbalanceadas com 22384 e 37616 instâncias cada. A base apresenta quatro conceitos diferentes, com 15000 instâncias cada e 10% dos dados são ruídos.

A base de dados SINE1 possui um total de 10000 instâncias, sem ruído, de 2 dimensões, dois conceitos diferentes com 5000 instâncias cada e uma mudança abrupta de conceito. No primeiro conceito, todos os dados abaixo da curva $y = sin(x)$ são classificados como positivos, após a

[1] Disponível em: `https://scikit-multiflow.github.io/` .
[2] Disponível em: `https://scikit-learn.org/` .

mudança de conceito, a classificação é invertida. A base de dados SINE2 possui um total de 10000 instâncias, com ruído, de 4 dimensões, sendo 2 dimensões com valores de ruído, dois conceitos diferentes de 5000 instâncias cada e mudança gradual. A mudança de conceito gradual foi realizada por meio de uma função sigmoidal $f(t) = 1/(1 + e^{-4(t-p)/w})$ onde $p = 5000$ indica a posição em que acontece a mudança e $w = 500$ a largura da transição (Gama et al., 2004).

A base de dados *Toy* possui um total de 20000 instâncias de 2 dimensões e 3 mudanças de conceito abruptas. Construída a partir de quatro diferente funções geradoras (Duas Meia Luas, Círculos, Espirais e Duas Normais) com 5000 amostras cada, como ilustrado na Figura 5.

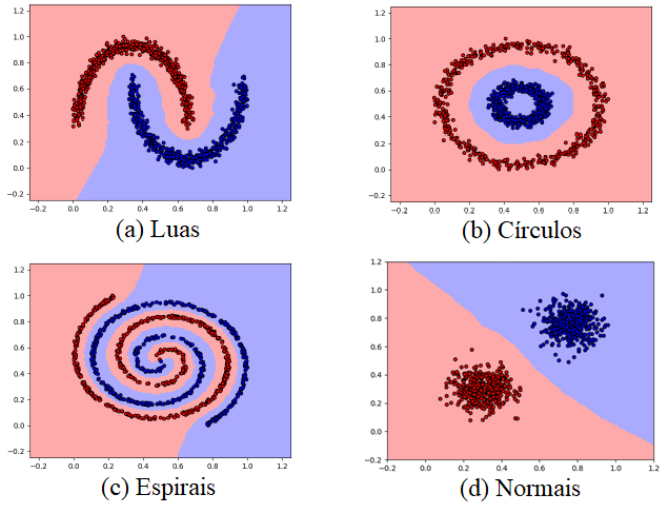

Figura 5. Funções que compõe a bases de dados *Toy*

4.4 Quantificação do Desempenho

As comparações de desempenho foram feitas avaliando a acurácia do classificador e a qualidade da detecção frente as mudanças de conceito. Foi definido como um Verdadeiro Positivo (TP) uma detecção dentro de um intervalo de atraso fixo (tamanho do *mini batch* ou janela), após a ocorrência de uma mudança de conceito, um Falso Negativo (FN) como falta de detecção dentro do intervalo de atraso e um Falso Positivo (FP) como uma detecção fora desse intervalo. Para cada detector, sua qualidade de detecção foi então avaliada pelo $Recall = TP/(TP + FN)$ e $Precision = TP/(TP + FP)$.

4.5 Resultados

Para alcançar os resultados, foram combinados os detectores de mudança de conceito com cada algoritmo de aprendizado para a construção de classificadores online. O método de avaliação adotado foi o *prequential* e a janela de dados utilizada é de tamanho fixo $j = 100$. As Tabelas 1 e 2 apresentam a acurácia com o uso dos classificadores *Random Forest* (RF) e *Support Vector Machine* (SVM) respectivamente. As tabelas mostram duas versões do método proposto baseado em *quadtree*, QT $h4$ e QT $h6$.

Os diferentes classificadores, respeitando suas peculiaridades, mostraram resultados próximos. Apesar das diferentes

Tabela 1. RF Acurácia

Detectores	Electricity	SEA	Toy	SINE1	SINE2
QT $h4$	88.40	**81.45**	93.76	93.22	89.38
QT $h6$	89.35	77.54	93.01	**95.05**	87.50
ADWIN	88.11	77.00	92.67	94.87	**89.51**
DDM	**89.51**	77.52	93.38	94.46	89.24
EDDM	88.74	81.18	**93.99**	94.12	88.81
PH	88.92	77.06	92.90	94.73	89.18

Tabela 2. SVM Acurácia

Detectores	Electricity	SEA	Toy	SINE1	SINE2
QT $h4$	76.64	84.83	75.35	96.44	74.42
QT $h6$	82.66	77.72	75.22	**97.20**	73.61
ADWIN	84.30	75.66	**75.48**	**97.20**	74.91
DDM	**87.53**	76.66	73.93	95.73	74.84
EDDM	85.54	**84.95**	75.29	97.10	**76.48**
PH	86.49	80.06	74.06	96.49	74.55

combinações entre classificadores, detectores e bases de dados, é possível observar um equilíbrio nos resultados entre os métodos de detecção de mudança, mostrando que método proposto QT é competitivo. Cabe ressaltar a importância do ajuste do parâmetro altura h para a adequação do mapeamento do espaço em relação a base de dados. O aumento do $h = 4$ para o $h = 6$ na base de dados SEA culmina na diminuição da acurácia, já para a base de dados *Electricity* tem efeito contrário.

As Tabelas 3 e 4 apresentam os valores de *Precision* e *Recall* dos detectores para as bases de dados *Toy* e SINE1. Neste contexto, o valor da *Recall* se refere ao quão responsivo é o detector frente a uma mudança de conceito real. O valor de *Precision* diz respeito ao quão acurado o detector é ao identificar uma possível mudança de conceito. Os detectores que não foram capazes de identificar a mudança de conceito, após esta ter acontecido dentro do intervalo de tamanho igual a janela de dados, tiveram seus valores zerados.

Tabela 3. *Precision* e *Recall* com uso do RF

	Toy		SINE1	
Detectores	Recall	Precision	Recall	Precision
QT $h4$	0.9566	0.9930	0.94	0.6143
QT $h6$	0.6433	0.9948	0.69	0.9452
ADWIN	0.4	0.9756	0.35	0.9722
DDM	0.3866	1.0	0.67	1.0
EDDM	0.39	1.0	0.59	0.9672
PH	0.0166	1.0	0.44	1.0

Tabela 4. *Precision* e *Recall* com uso do SVM

	Toy		SINE1	
Detectores	Recall	Precision	Recall	Precision
QT $h4$	0.65	0.9898	0.96	0.65
QT $h6$	0.3533	0.9814	0.68	0.9189
ADWIN	0.0	0.0	0.35	1.0
DDM	0.2233	1.0	0.82	1.0
EDDM	0.0	0.0	0.9	0.9574
PH	0.0866	0.9629	0.46	1.0

O valores de *Recall* e *Precision* alcançados foram promissores. QT demonstrou qualidade na detecção de mudança de conceito. Obteve os melhores valores de *Recall*, mostrando destreza frente a mudanças, e valores de *Precision* competitivos. A mudança no parâmetro de altura h tem efeito no compromisso entre *Recall* e *Precision*. Assim como visto nos resultados de acurácia, o aumento de h nem sempre

resulta no aumento de *Precision*, dependendo da base de dados e do algoritmo de aprendizado utilizado.

5. CONCLUSÃO

Este trabalho apresentou uma nova técnica para detecção de mudanças de conceito, um detector baseado unicamente na disposição espacial dos dados por meio da estrutura de divisão recursiva do espaço da *quadtree*. Seguindo o princípio de que a relação entre dados e espaço é imutável, qualquer mudança nesta relação pode ser considerada uma mudança de conceito.

O método está desacoplado do algoritmo de aprendizado, permitindo que diferentes algoritmos sejam utilizados, mas depende dos erros do algoritmo de aprendizado para que atue. Os resultados experimentais demonstraram que o método proposto é capaz de detectar mudanças de conceito tão bem quanto outros métodos comumente usados na literatura. O destaque pode ser visto nos valores de *Recall* e *Precision*, os quais apontam qualidade na detecção das mudanças.

Os resultados apresentados neste estudo ainda são preliminares. Uma maior quantidade de algoritmos de aprendizado será utilizada junto com o novo detector. Mais bases de dados, com diferentes situações de mudança de conceito, serão testadas a fim de delimitar com maior clareza os pontos fortes e fracos do método proposto. Por se tratar de uma análise espacial dos dados, pretende-se estender a técnica para que esta não precise ser acionada a partir dos erros do classificador, atuando de forma independente do algoritmo de aprendizado.

AGRADECIMENTOS

Os autores gostariam de agradecer o apoio do CNPQ, da CAPES-Brasil e da FAPEMIG.

REFERÊNCIAS

Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R., and Morales-Bueno, R. (2006). Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, 77–86.

Barros, R.S.M. and Santos, S.G.T.C. (2018). A large-scale comparison of concept drift detectors. *Information Sciences*, 451, 348–370.

Bifet, A. and Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, 443–448. SIAM.

de Barros, R.S.M., Hidalgo, J.I.G., and de Lima Cabral, D.R. (2018). Wilcoxon rank sum test drift detector. *Neurocomputing*, 275, 1954–1963.

Demšar, J. and Bosnić, Z. (2018). Detecting concept drift in data streams using model explanation. *Expert Systems with Applications*, 92, 546–559.

Gama, J. (2010). *Knowledge discovery from data streams*. CRC Press.

Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In *Brazilian symposium on artificial intelligence*, 286–295. Springer.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4), 44.

Hidalgo, J.I.G., Mariño, L.M.P., and de Barros, R.S.M. (2019). Cosine similarity drift detector. In *International Conference on Artificial Neural Networks*, 669–685. Springer.

Khamassi, I. and Sayed-Mouchaweh, M. (2014). Drift detection and monitoring in non-stationary environments. In *2014 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, 1–6. IEEE.

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., and Ghédira, K. (2018). Discussion and review on evolving data streams and concept drift adapting. *Evolving systems*, 9(1), 1–23.

Mahdi, O.A., Pardede, E., Ali, N., and Cao, J. (2020). Diversity measure as a new drift detection method in data streaming. *Knowledge-Based Systems*, 191, 105227.

Mehta, D.P. and Sahni, S. (2004). *Handbook of data structures and applications*. Chapman and Hall/CRC.

Street, W.N. and Kim, Y. (2001). A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 377–382.

Zliobaite, I. (2013). How good is the electricity benchmark for evaluating concept drift adaptation. *arXiv preprint arXiv:1301.3524*.