Classificação de fluxos de dados com mudança de conceito e latência de verificação

Denis Moreira dos Reis

Denis Moreira dos Reis

Classificação de fluxos de dados com mudança de conceito e latência de verificação

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *EXEMPLAR DE DEFESA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Gustavo Enrique de Almeida Prado Alves Batista

USP – São Carlos
Agosto de 2016

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

R634c

Reis, Denis Moreira dos
Classificação de fluxos de dados com mudança de conceito e latência de verificação / Denis Moreira dos Reis; orientador Gustavo Enrique de Almeida Prado Alves Batista. - São Carlos - SP, 2016.
105 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2016.

1. Kolmogorov-Smirnov. 2. Mudança de conceito. 3. Fluxo de dados. 4. Aprendizado de máquina. 5. Árvore cartesiana. I. Batista, Gustavo Enrique de Almeida Prado Alves, orient. II. Título.

Denis Moreira dos Reis

Data stream classification with concept drift and verification latency

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *EXAMINATION BOARD PRESENTATION COPY*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Gustavo Enrique de Almeida Prado Alves Batista

USP – São Carlos
August 2016

Dedico este trabalho aos grãos de poeira que flutuam sem rumo e,
sem conhecimento, alteram para sempre o futuro do mundo ao seu redor.
Em especial, aos grandes grãos de poeira que chamo de amigos.

AGRADECIMENTOS

Os agradecimentos principais são direcionados à Gustavo Batista, Peter Flach, Vinícius Souza, Meelis Kull, Reem Al-Otaibi, cujas ideias propostas contribuíram à aquisição de antigos e e construção de novos conhecimentos, diretamente relacionados ao conteúdo deste trabalho.

Agradecimentos especiais são direcionados à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa concedida durante o início do curso de Mestrado (processo DS-6909543/M), e à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), pela bolsa concedida para o restante do curso no país e pela bolsa concedida para estágio de pesquisa no exterior (processos 14/12333-7 e 15/01701-8, respectivamente).

“Necessitamos sempre de ambicionar alguma coisa que,
alcançada, não nos torna sem ambição.”
(Santos Dumont)

RESUMO

DOS REIS, D. M.. **Classificação de fluxos de dados com mudança de conceito e latência de verificação**. 2016. 105 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Apesar do grau relativamente alto de maturidade existente na área de pesquisa de aprendizado supervisionado em lote, na qual são utilizados dados originários de problemas estacionários, muitas aplicações reais lidam com fluxos de dados cujas distribuições de probabilidade se alteram com o tempo, ocasionando mudanças de conceito. Diversas pesquisas vêm sendo realizadas nos últimos anos com o objetivo de criar modelos precisos mesmo na presença de mudanças de conceito. A maioria delas, no entanto, assume que tão logo um evento seja classificado pelo algoritmo de aprendizado, seu rótulo verdadeiro se torna conhecido. Este trabalho explora as situações complementares, com revisão dos trabalhos mais importantes publicados e análise do impacto de atraso na disponibilidade dos rótulos verdadeiros ou sua não disponibilização. Ainda, propõe um novo algoritmo que reduz drasticamente a complexidade de aplicação do teste de hipótese não-paramétrico Kolmogorov-Smirnov, tornado eficiente seu uso em algoritmos que analisem fluxos de dados. A exemplo, mostramos sua potencial aplicação em um método de detecção de mudança de conceito não-supervisionado que, em conjunto com técnicas de Aprendizado Ativo e Aprendizado por Transferência, reduz a necessidade de rótulos verdadeiros para manter boa performance de um classificador ao longo do tempo, mesmo com a ocorrência de mudanças de conceito.

Palavras-chave: Kolmogorov-Smirnov, Mudança de conceito, Fluxo de dados, Aprendizado de máquina, Árvore cartesiana.

ABSTRACT

DOS REIS, D. M.. **Classificação de fluxos de dados com mudança de conceito e latência de verificação**. 2016. 105 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Despite the relatively maturity of batch-mode supervised learning research, in which the data typifies stationary problems, many real world applications deal with data streams whose statistical distribution changes over time, causing what is known as concept drift. A large body of research has been done in the last years, with the objective of creating new models that are accurate even in the presence of concept drifts. However, most of them assume that, once the classification algorithm labels an event, its actual label become readily available. This work explores the complementary situations, with a review of the most important published works and an analysis over the impact of delayed true labeling, including no true label availability at all. Furthermore, this work proposes a new algorithm that heavily reduces the complexity of applying Kolmogorov-Smirnov non-parametric hypotheis test, turning it into an uselful tool for analysis on data streams. As an instantiation of its usefulness, we present an unsupervised drift-detection method that, along with Active Learning and Transfer Learning approaches, decreases the number of true labels that are required to keep good classification performance over time, even in the presence of concept drifts.

Key-words: Kolmogorov-Smirnov, Concept drift, Data stream, Machine learning, Cartesian tree.

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

Fluxo de dados, em computação, é uma sequência de registros de dados, aqui chamados eventos, que se tornam disponíveis ao longo do tempo, em uma determinada ordem (FEIGENBAUM *et al.*, 2002). No contexto de aprendizado de máquina, mineração em fluxos de dados é a extração de estruturas de conhecimento a partir dos eventos de um fluxo (GAMA *et al.*, 2010).

Por serem potencialmente rápidos e infinitos, fluxos de dados necessitam frequentemente serem tratados com severos limites de memória e processamento (BIFET, 2009). As características que definem os eventos de um fluxo se alteram ao longo do tempo (WIDMER; KUBAT, 1996). Tais alterações são chamadas *mudanças de conceito*.

Na última década, houve um grande aumento de interesse em algoritmos que sejam capazes de aprender a partir de fluxos de dados. O aumento de interesse levou à proposição de um grande número de algoritmos de aprendizado para diversas tarefas, como classificação, agrupamento e detecção de anomalias (GAMA *et al.*, 2004; GUHA *et al.*, 2000; HILL; MINSKER, 2010).

Em particular, classificação em fluxo de dados é a rotulação de eventos em categorias pré-estabelecidas. Para classificação, as propostas encontradas na literatura têm, em grande número, sido avaliadas e comparadas assumindo a instantânea disponibilidade dos rótulos verdadeiros, tão logo as predições são realizadas (GAMA *et al.*, 2004; BIFET; HOLMES; PFAHRINGER, 2010; OZA, 2005; BIFET; GAVALDÀ, 2007; BIFET; GAVALDÀ, 2009).

O tempo entre a classificação e a disponibilidade do rótulo correspondente é denotada latência de verificação (MARRS; HICKEY; BLACK, 2010). Quando esse tempo aproxima o infinito, pode-se dizer que há um cenário de *latência de verificação extrema*, no qual nenhum dado rotulado é obtido após a indução do modelo de classificação. Por sua vez, quando esse tempo aproxima zero, se diz que há um cenário de *latência de verificação nula*, no qual os rótulos verdadeiros são obtidos instantaneamente após a classificação de seus respectivos eventos.

A premissa de latência nula nunca é inteiramente satisfeita em cenários práticos. Caso o rótulo verdadeiro se faça disponível instantaneamente, não há razão para a tarefa de classificação existir. Por outro lado, é plausível a existência de uma latência não extrema e bem conhecida. Por exemplo, considere a tarefa de predizer se o preço de ações do mercado irá subir ou descer na próxima hora. É possível obter todos os rótulos corretos com exatamente uma hora de diferença da predição. Outro exemplo é a predição do consumo de energia. Em particular, tarefas de classificação derivadas de problemas de regressão geralmente cumprem o requerimento de disponibilidade dos rótulos corretos após a classificação.

Por outro lado, para a maior parte das aplicações, a latência de verificação pode estar sob influência de fatores não controláveis, sendo apenas parcialmente conhecida ou desconhecida. Além disso, a latência está relacionada à anotação dos rótulos corretos, que pode envolver custos financeiros, como a contratação de especialistas de domínio, para classificação manual, e obrigatoriamente um custo de tempo, como o necessário para que o especialista conclua a classificação. A obtenção instantânea dos rótulos de todos os exemplos classificados provê ao classificador uma posição vantajosa e usualmente irrealista para se adaptar, pois permite a detecção de mudanças de conceito antes do que seria possível na prática, principalmente em métodos de detecção que dependam de análise *online* de medidas de performance.

Para problemas de classificação *benchmark*, os resultados reportados nos trabalhos da literatura que possuem a premissa de latência nula são claramente otimistas, quando comparados ao uso prático. Entretanto, é possível argumentar que esse não é um problema real, uma vez que o interesse dos experimentos é a obtenção de performance relativa: o intuito é identificar qual o melhor algoritmo para uma determinada tarefa, independentemente de sua performance em termos escalares. Entretanto, essa consideração contém a premissa implícita de que a latência de verificação não altera a performance relativa dos classificadores.

Este projeto de pesquisa possui dois objetivos principais:

1. Mostrar, empiricamente, que diferentes latências de verificação afetam diferentes algoritmos diferentemente e, além disso, a inserção de latência de verificação pode alterar a performance relativa de classificadores;

2. Oferecer métodos eficientes para a detecção de mudanças de conceito, quando estas são possíveis de serem detectadas sem a informação de rótulos verdadeiros, e

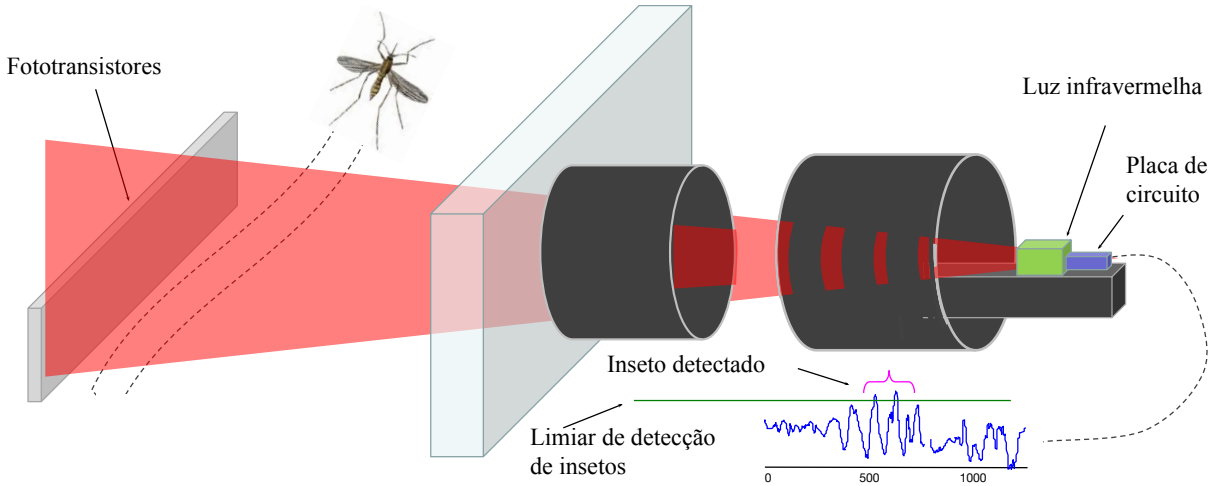
    a) quando possível e sob determinadas circunstâncias, adaptar o modelo de classificação sem o uso de rótulos corretos adicionais;

    b) caso contrário, usar os rótulos verdadeiros de apenas uma porção dos dados, para efetuar a adaptação do modelo.

1.1 Justificativa e motivação

Como exemplo concreto de aplicação na qual há latência de verificação extrema em um ambiente não estacionário, considere a classe de problemas envolvendo sensores inteligentes, como o proposto por Batista *et al.* (2011b), como ilustrado pela **??**. Esse sensor em particular captura informações sobre o voo de insetos, utilizando uma fonte de luz infravermelha e fototransistores, e os classifica de acordo com suas espécies. Tais sensores são a chave para uma estimativa em tempo real das distribuições espaço-temporal de insetos importantes como pragas agrícolas e vetores de doença. A estimativa precisa destas distribuições de insetos possibilitará a criação de sistemas de alarme efetivos para surtos de insetos, o uso inteligente de técnicas de controle, como inseticidas, e estará presente no núcleo da próxima geração das armadilhas, que será capaz de capturar apenas espécies de interesse.

Figura 1 – Esquema de armadilha inteligente.



Fonte: Batista *et al.* (2011a)

Sensores inteligentes monitoram ambientes externos, estando tipicamente sujeitos a lidar com fluxos potencialmente infinitos de dados. Esses dados podem apresentar mudanças de conceito ao longo do tempo, causadas por alterações nas condições externas, como temperatura e umidade. No caso do sensor de insetos, estudos em entomologia mostram que tais condições influenciam o metabolismo de insetos e, consequentemente, as medições realizadas (TAYLOR, 1963; MELLANBY, 1936). Mais importante, como a maior parte dos sensores funciona sem intervenção humana por longos períodos de tempo (dias ou meses), não há maneiras de obter dados rotulados, a baixo custo, durante o período de operação do sensor. Entretanto, é possível obter dados rotulados a baixo custo em condições especiais. Batista *et al.* (2011b) coletou uma quantidade considerável de dados rotulados em laboratório, utilizando insetários separados por espécies. Tais dados podem ser utilizados para induzir modelos de classificação anteriormente à aplicação em condições de campo.

Outra abordagem poderia ser a existência de um especialista da área, que coletaria e

classificaria os dados por um tempo limitado, no local de instalação do sensor. Esses dados poderiam ser utilizados para induzir o modelo antes do processo de classificação não supervisionada. Entretanto, não importando qual abordagem é utilizada para coletar os dados iniciais, frequentemente é muito difícil coletar dados que cubram todas as possíveis condições que ocasionam mudança de conceito. Assim, há a necessidade de métodos capazes de se adaptar às mudanças ao longo do tempo sem supervisão.

1.2 Objetivos, hipóteses e premissas

A hipótese central deste trabalho é a de que é possível identificar, de maneira não supervisionada, a ocorrência de mudanças de conceito – quando detectáveis sem rótulos verdadeiros (ŽLIOBAITĖ, 2010) – e consequentemente diminuir a necessidade de rótulos verdadeiros para manter um modelo de classificação com alta performance de predição, ao longo de um fluxo de dados. Além disso, foi levantada a hipótese de que, quando as mudanças de conceito ocorrentes têm natureza conhecida e reconhecível, é possível, por meio de Aprendizado por Transferência, atualizar o modelo de classificação sem a necessidade de rótulos verdadeiros adicionais. As hipóteses levam a objetivar:

- A elaboração de uma técnica não-supervisionada para a detecção de mudança de conceitos;
- A elaboração de reações supervisionadas e não-supervisionadas às detecções realizadas.

A seção seguinte discute trabalhos originados a partir da pesquisa realizada pelos objetivos propostos.

1.2.1 Trabalhos originados

Parte da produção científica deste trabalho, referente aos Capítulos 6 e 7, originou o artigo *Fast Unsupervised Online Drift Detection Using Incremental Kolmogorov-Smirnov Test* na *22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (REIS *et al.*, 2016).

Para os cumprir os objetivos propostos por este trabalho, a pesquisa foi conduzida de acordo com a elaboração descrita na seção que segue.

1.3 Organização do trabalho

Este trabalho está organizado da seguinte forma:

- O Capítulo 2 formaliza o que é fluxo de dados e os conceitos relevantes para o restante deste trabalho, e sumariza as publicações mais relevantes e diretamente relacionadas às propostas do presente trabalho;
- o Capítulo 3 descreve o ferramental frequentemente empregado, na literatura, para avaliar e comparar estatisticamente os resultados obtidos por trabalhos relacionados à classificação em fluxos de dados;
- o Capítulo 4 propõe diversas técnicas exploratórias para identificação e alteração de características existentes em bases de dados. O foco deste capítulo é fornecer ferramental para análises preliminares sobre mudanças de conceito e dependência temporal;
- o Capítulo 5 analisa o impacto da variação da latência de verificação sobre a performance relativa de diferentes classificadores;
- o Capítulo 6 propõe um algoritmo incremental para realização do teste de hipótese não-paramétrico Kolmogorov-Smirnov. O algoritmo proposto possibilita a rápida modificação das amostras envolvidas e a realização instantânea do teste após cada alteração, sendo ideal para o contexto de fluxo de dados;
- o Capítulo 7 propõe um método não-supervisionado para detecção de mudanças de conceito, e reações de Aprendizado Ativo e de Aprendizado por Transferência às detecções realizadas;
- por fim, o Capítulo 8 apresenta as nossas conclusões e sugestões para trabalhos futuros.

CAPÍTULO

2

FLUXO DE DADOS E MUDANÇA DE CONCEITO

Este capítulo introduz as principais concepções presentes na área de aprendizado de máquina em fluxo de dados, dentre as quais destaca-se a definição de mudança de conceito, um dos, senão o principal problema recorrente em aplicações na área. Pela sua importância, a mudança de conceito é devidamente detalhada na Seção 2.2.

2.1 Fluxo de dados

Fluxo de dados pode ser definido como uma sequência de exemplos, contextualmente chamados de eventos, que ocorrem no tempo em uma taxa que não permite o seu armazenamento permanente em memória. Devido ao tamanho potencialmente ilimitado, métodos tradicionais de aprendizado em lote não são aplicáveis, pois existem severas restrições de memoria e tempo de processamento (GAMA; GABER, 2007).

As principais características dos fluxos de dados implicam nas seguintes restrições aos algoritmos que os processam (BIFET, 2009):

1. É impossível armazenar todos os dados do fluxo. Somente uma pequena parcela pode ser processada e armazenada, enquanto o restante é descartado;
2. A velocidade de chegada dos eventos no fluxo exige que cada elemento em particular seja processado em tempo real, e, em seguida, descartado na maior parte das vezes;
3. A distribuição dos dados pode mudar com o tempo. Assim, os dados do passado podem se tornar irrelevantes ou mesmo prejudiciais para a descrição dos conceitos atuais.

A primeira restrição limita a quantidade de memória que os algoritmos que lidam com fluxo de dados podem utilizar, enquanto a segunda restrição limita o tempo para o processamento

de cada evento. Assim, as duas primeiras restrições levam ao desenvolvimento de técnicas que reduzem as informações contidas no fluxo de dados, tais como as que fazem uso de amostragem (CHAUDHURI; MOTWANI; NARASAYYA, 1999), *sketching* (ALON; MATIAS; SZEGEDY, 1999), histogramas (GILBERT *et al.*, 2002), *wavelets* (MATIAS; VITTER; WANG, 2000) e janela deslizante (DATAR *et al.*, 2002). A terceira restrição faz com que os algoritmos lidem com técnicas de detecção de mudanças e atualização dos dados. Tais mudanças são melhor abordadas na próxima seção.

2.2 Mudança de conceito

Em diversas aplicações do mundo real, os dados que descrevem um problema podem passar por modificações ao longo do tempo devido a sua natureza não estacionária (WIDMER; KUBAT, 1996). Mais especificamente, em um problema de classificação, as descrições das classes podem variar ao longo do tempo, de modo que os atributos usados na discriminação das classes no tempo t podem não ser mais úteis no tempo $t+k$. Assim, um classificador induzido a partir de um conjunto de treinamento inicial logo será inutilizado devido a alterações no problema. Desse modo, sendo que o termo conceito refere-se à distribuição dos dados em um período de tempo, se dá o nome de mudança de conceito, ou *concept drift*, alterações nas seguintes probabilidades (KELLY; HAND; ADAMS, 1999):

***a priori* das classes** $P(c)$ onde c é uma classe dada: a probabilidade de ocorrência de um evento que pertence à classe c se altera com o tempo;

condicional $P(x|c)$ onde x é o descritor de um evento: a distribuição de probabilidade que define a classe c se altera com o tempo, seja pelos seus parâmetros ou pela função da distribuição em si;

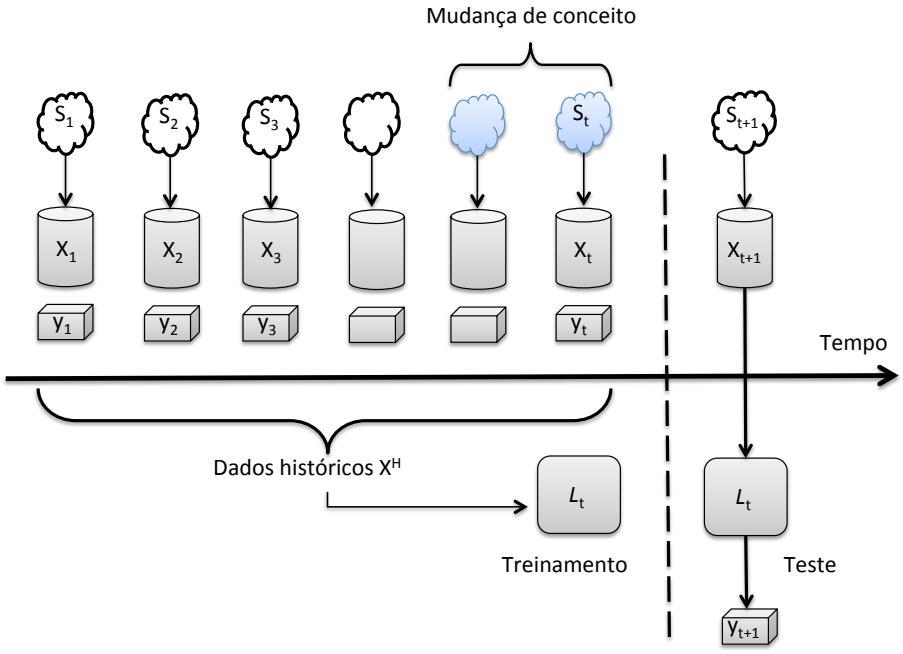
condicional *a posteriori* $P(c|x)$**:** uma mudança nessa probabilidade é devida a mudança em uma das ou ambas as probabilidade anteriores. Essa mudança explicita a discrepância entre um modelo de predição arcaico e os dados atuais.

Apesar de mudanças de conceito serem essencialmente causadas por alteração em ao menos uma das três probabilidades listadas, é comum encontrar, na literatura, a diferenciação de mudanças *virtuais* (WIDMER; KUBAT, 1993), nas quais $P(x)$ se altera sem a alteração de $P(c|x)$. Essas mudanças dizem respeito apenas à quantidade relativa dos eventos da classe. Mudanças opostas, nas quais $P(c|x)$ se altera sem a alteração de $P(c)$, revelam a movimentação do conceito no espaço de atributos. Kull e Flach (2014) propõe uma extensa lista de outras nomenclaturas para diversas especificações sobre quais e em que condições cada probabilidade se altera. O artigo se refere especificamente ao aprendizado em lote, no qual há um ambiente de treino e um ambiente de aplicação, que seguem uma única distribuição de probabilidade estacionária cada

um. Porém, os tipos de mudança de conceito levantados se mantém válidos em fluxos de dados, onde há um conceito não estacionário que evolui com o tempo. Entretanto, a especificação das mudanças nas três probabilidades são suficientes para os objetivos propostos neste trabalho, de forma que não será feita maior distinção em como as probabilidades se alteram.

Considere $X_i \in \Re^p$, uma variável aleatória de p dimensões no espaço de características gerada por uma distribuição D_i, o i-ésimo evento ocorrido no fluxo de dados, cujo rótulo correto é notado por y_i. Considere que t seja o índice do último evento ocorrido, e que não há, necessariamente, um intervalo regular de tempo entre as ocorrências dos eventos. Ao conjunto $X^H = (X_1, \ldots, X_t)$ dá-se o nome de *dados históricos*. Formalmente, o problema de classificação em fluxo de dados corresponde, na ocorrência do evento X_{t+1}, chamado *evento de teste*, à indução de um classificador $\mathscr{L}_t$, utilizando neste processo total ou parcialmente os dados históricos, de forma a obter o rótulo predito $\mathscr{L}_t(X_{t+1}) = \hat{y}_{t+1}$, o qual se deseja aproximar y_{t+1}. Este processo é ilustrado pela Figura 2. Se todos os eventos forem gerados por uma mesma distribuição, de modo que $D_1 = D_2 = \ldots = D_{t+1} = S$, o conceito é estável. Se em qualquer dois pontos i e j, $D_i \neq D_j$, diz-se que houve mudança de conceito..

Figura 2 – Representação da classificação de um evento X_{t+1} em um fluxo de dados com mudança de conceito.



Fonte: Adaptada de Žliobaitė (2009).

Tão logo X_{t+1} seja classificado, este evento passa a compor os dados históricos. Isto não significa, porém, que necessariamente será utilizado para a atualização do modelo em um momento anterior à classificação do evento seguinte, X_{t+2}. Há dois importantes fatores que devem ser considerados nessa etapa: a disponibilidade do rótulo correto e a política de adaptação adotada pelo algoritmo de classificação.

A disponibilidade do rótulo é definida pela latência de verificação do problema. No caso de latência de verificação nula, o algoritmo de classificação dispõe da classe verdadeira de X_{t+1} em um tempo anterior a necessidade de classificação de X_{t+2}, podendo contar com uma política que utilize essa informação de forma segura. Latências de verificação mais elevadas, ainda que não extremas, implicam na necessidade de classificar eventos sem a posse da classe correta de uma quantidade dos últimos eventos. O algoritmo pode utilizar a classe predita ou nem mesmo utilizar tais eventos até que suas classes corretas estejam disponíveis.

De modo geral, classificadores que processam fluxos de dados com mudanças de conceito devem lidar com quatro sub-problemas (ŽLIOBAITĖ, 2009):

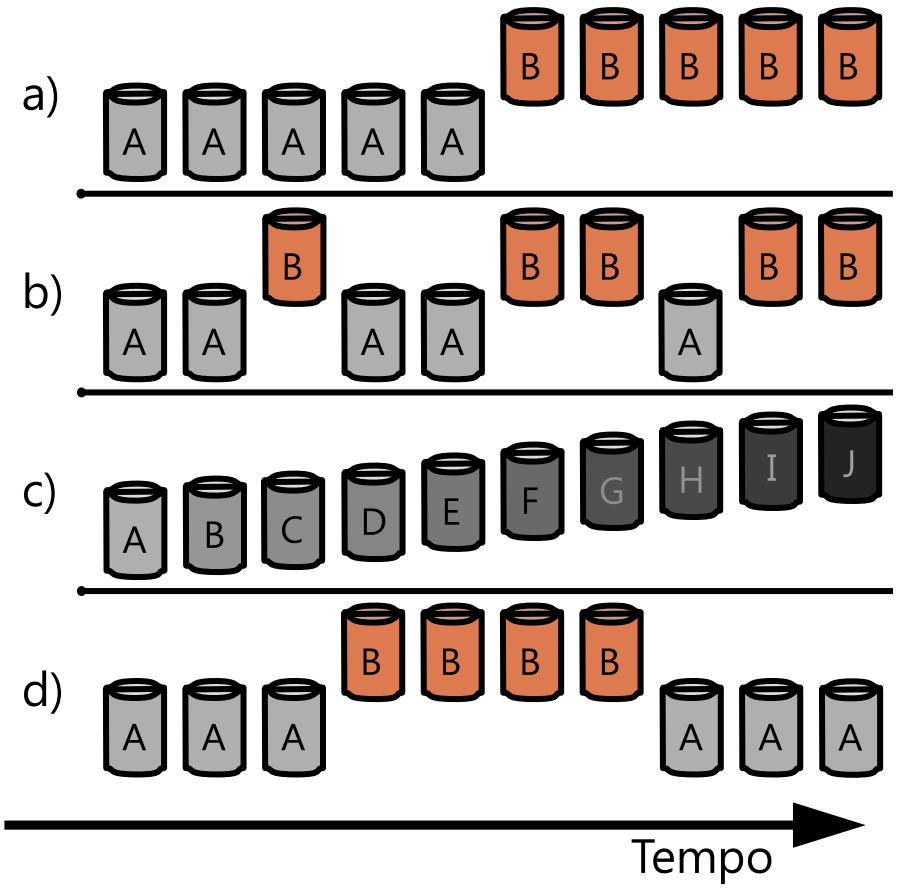
Suposição futura: A principal consideração que se deve fazer quando se lida com o problema de mudança de conceito é a incerteza sobre o futuro. Assim, pode-se estimar ou predizer o rótulo do evento de teste X_{t+1}, mas assume-se que sua distribuição não é totalmente conhecida. No exemplo da Figura 2, assume-se que o conceito D_{t+1} é desconhecido. Para identificar a mudança de conceito, pode-se estimar a distribuição D_{t+1} a partir de medidas de comparações entre X_{t+1} e os eventos do conjunto de dados históricos X^H. Esse tipo de suposição foi utilizada nos trabalhos de Delany *et al.* (2005), Pourkashani e Kangavari (2008) e Tsymbal *et al.* (2008). Outra alternativa é o uso de métodos que treinam regras de predição para estimar o estado futuro e então incorporar essa informação no processo de aprendizado incremental. Trabalhos como os de Yang, Wu e Zhu (2006), Bottcher, Spott e Kruse (2008) e Brown e Steyvers (2009) utilizaram predições futuras.

Tipo de mudança: Existem diferentes possibilidades de mudanças de conceito e sua identificação é importante para que o classificador possa se adaptar à mudança. De modo simplificado, considerando-se apenas duas distribuições D_I e D_{II} que representam os conceitos A e B, há ao menos quatro diferentes tipos de mudanças, conforme apresentado na Figura 3, no qual o conceito ocorrente é disposto ao longo do tempo. Assim, as mudanças representadas são: *a)* Abrupta; *b)* Gradual; *c)* Incremental e *d)* Recorrente. Entretanto, os tipos de mudanças não se limitam a este pequeno conjunto. O trabalho de Minku, White e Yao (2010) aborda outras possibilidades de mudanças levando em consideração características como a previsibilidade, gravidade, velocidade, frequência e recorrência.

Na mudança abrupta (*sudden drift*), um conceito A é repentinamente substituído por um conceito B.

Na mudança gradual (*gradual drift*), eventos pertencentes ao conceito B são incrementalmente mais frequentes, enquanto que eventos pertencentes ao conceito A são incrementalmente menos frequentes. Em outras palavras, em um primeiro momento há apenas eventos de A. Após um tempo, eventos de A e B coexistem. Com o passar do tempo, o número de ocorrências de eventos do conceito A diminui até que restem eventos apenas pertencentes ao conceito B.

Figura 3 – Representação de quatro tipos de mudanças de conceitos: (a) Mudança abrupta (*sudden drift*); (b) Mudança gradual (*gradual drift*); (c) Mudança incremental (*incremental drift*) e (d) Mudança recorrente (*reoccurring drift*).



Fonte: Adaptada de Žliobaitė (2009).

A mudança incremental (*incremental drift*) descreve a existência de um único conceito que evolui com o tempo. Entretanto, esse conceito pode ser discretizado em uma sequência de conceitos consecutivos. Cada um deles difere apenas um pouco de seu antecessor imediato e de seu sucessor imediato. Mudanças são notáveis, portanto, apenas à longo prazo.

A mudança recorrente (*reoccurring drift*) ocorre quando um conceito anteriormente ativo reaparece após um determinado tempo. Esse tipo de mudança é diferente de uma sazonalidade periódica, pois não é evidente o momento no qual o conceito voltará a ser ativo.

Adaptatividade: Devido às alterações nos conceitos, o classificador deve adotar estratégias adaptativas baseadas no tipo de mudança identificada e na suposição futura que está sendo assumida pelo algoritmo. Assim, tanto a sua parametrização deve ser passível de adaptação como a formação do conjunto de treinamento. Vale lembrar que devido ao grande volume de dados presente no fluxo, é impossível o armazenamento constante em memória para a formação de um grande conjunto de treinamento. Além disso, as mudanças de conceito

exigem que o conjunto de treinamento seja atualizado ou que o algoritmo faça uma seleção de eventos de modo a se adaptar.

Avaliação e seleção do modelo: Espera-se que um classificador tenha capacidade de generalizar os conceitos aprendidos. A generalização de um classificador está relacionada a sua capacidade de predição em dados não vistos, ou seja, que não estão presentes no conjunto de treinamento. Determinar o erro de generalização de um classificador é de extrema importância, uma vez que este erro está relacionado com seu desempenho e pode ser utilizado para a escolha de diferentes algoritmos e na seleção de modelos. A escolha do algoritmo pode depender de outros fatores além do erro de generalização, tais como tempo de resposta, uso de memória, tipo e quantidade dos dados. Entretanto, escolhido o algoritmo de aprendizado mais adequado é necessário ajustar as suas opções paramétricas. Cada possível variação de parâmetros para o treinamento de um determinado algoritmo constitui uma diferente instância do classificador denominado de *modelo*. Assim, o problema de seleção de modelo está relacionado à determinação do melhor modelo para um problema de classificação.

Virtualmente todos os trabalhos publicados que lidam com mudança de conceito, e usam bases reais, fazem premissas sobre a natureza das mudanças de conceito existentes a partir de argumentação informal. Até onde vai nosso conhecimento, Sarnelle *et al.* (2015) publicou a primeira tentativa de formalizar e quantificar tais premissas. O trabalho lida com mudanças de conceito que se dão pela movimentação dos eventos de uma classe no espaço de atributos, com alguma direção e velocidade. Foram propostas duas medidas. Considerando um problema de classificação binária, no qual o fluxo é dividido em janelas consecutivas, X_t^c é o conjunto de exemplos que pertencem à classe c dentro da t-ésima janela e $D(A,B)$ é uma distância métrica entre conjuntos, como a *Distância Média Pareada*, então $\Delta(t)$ é calculado de acordo com a Equação 2.1. $\Delta(t)$ é definido como a mudança que ocorreu entre o tempo de duas janelas e, quanto mais baixo, mais fácil é o rastreamento da mudança de conceito.

$$\Delta(t) = \frac{\max\left(D(X_t^1, X_{t-1}^1), D(X_t^2, X_{t-1}^2)\right)}{D(X_{t-1}^1, X_{t-1}^2)} \tag{2.1}$$

Entretanto, $\Delta(t)$ não faz considerações sobre a direção da mudança de conceito. Com relação ao caso mais abrangente da classificação de duas ou mais classes, para fazer uso a direção da mudança, considere $D(X_{j,t+1}^j, C(X_{i,t}^j))$ a distância entre os dados da classe j na janela $t+1$ e os suportes centrais da classe i que pertencem à janela t, sendo os suportes centrais os eventos mais internos e importantes dentro da distribuição. *DCR*, definida de acordo com Equação 2.2, mede a direção e velocidade da mudança e, quando menor for, mais fácil é rastrear uma mudança

de conceito. Particularmente, um valor 1 indica uma incerteza de 50%.

$$DCR = \max_{j} \left(\frac{D(X^j_{j,t+1}, C(X^j_{j,t}))}{\min_{i \neq j} D(X^j_{j,t+1}, C(X^j_{i,t}))} \right) \tag{2.2}$$

2.3 Latência de verificação

Para a tarefa de classificação, pode-se ter o interesse em obter o rótulo verdadeiro de um evento após sua classificação. De fato, muitos algoritmos de classificação em fluxo de dados (HULTEN; SPENCER; DOMINGOS, 2001; KOLTER; MALOOF, 2005; ZHU; WU; YANG, 2006) necessitam dessa informação, a incorporando ao classificador – incrementalmente ou por meio de um sistema de detecção de mudança de conceito. Entretanto, é frequente o caso em que não é possível obter o rótulo verdadeiro imediatamente após a classificação. Formalmente, consideramos três casos:

Latência de verificação nula: nesse caso, é possível obter o rótulo verdadeiro de um evento imediatamente após prover sua classificação, para, ao menos, uma porção dos dados. Possibilita o uso de sistemas de detecção de mudança de conceito que são baseados na coleta e avaliação de medidas de desempenho, como acurácia, ao longo do tempo;

Latência de verificação extrema: nesse caso, os rótulos verdadeiros nunca são obtidos. Algoritmos que lidam com esse problema podem simplesmente assumir como ocorrem as mudanças de conceito ou rastreá-las, quando incrementais, ao longo do tempo. Até o presente momento, pelo melhor do nosso conhecimento, nenhum classificador que lide com essa situação e tente rastrear as mudanças é capaz de lidar com mudanças não incrementais, sem solicitar rótulos verdadeiros (SOUZA *et al.*, 2014);

Latência de verificação não-nula e não-extrema: nesse caso, um classificador, após classificar um determinado evento, necessita classificar outros eventos antes de receber o rótulo do primeiro. O tempo entre a classificação de um evento e a obtenção de seu rótulo é chamado *latência*, e pode ser variável ou fixo. Usualmente, mede-se a latência em quantidade de eventos ou tempo real de relógio (em segundos, minutos, etc).

2.4 Dependência temporal

Dependência temporal é a existência de uma relação entre os rótulos de eventos consecutivos. A dependência temporal pode ser positiva ou negativa. Se positiva, há uma maior probabilidade de eventos consecutivos pertencerem a mesma classe. Se negativa, eventos consecutivos provavelmente possuem classes diferentes.

Virtualmente, nenhum dos algoritmos estado-da-arte, para classificação em fluxo de dados, incorporam mecanismos para lidar diretamente com dependência temporal. Entretanto, Žliobaitė *et al.* (2014) apresenta dois arcabouços que podem adicionar a capacidade de lidar com essa situação a qualquer classificador: *Temporal Correction Classifier* e *Temporally Augmented Classifier*. Para algoritmos de classificação que produzem como saída uma estimativa da probabilidade de um evento pertencer a uma dada classe, o primeiro método proposto efetua uma correção da probabilidade estimada, considerando dependência temporal de primeira ordem. O segundo método torna possível a consideração de dependência temporal de ordem N à qualquer classificador, por meio da introdução dos últimos N rótulos vistos como atributos adicionais de um evento a ser classificado. Ambos métodos requerem latência de verificação nula.

2.5 Trabalhos relacionados

Latência de verificação, ou *delay*, é o período entre a ocorrência de um evento não rotulado e a disponibilidade de seu rótulo verdadeiro. Esse período de tempo depende do domínio de aplicação. Por exemplo, em aplicações de predizer a tendência de uma ação na bolsa de valores ou a demanda de eletricidade de uma região, a latência de verificação é a janela de predição, *i.e.*, a quantidade de tempo no futuro para a qual se deseja fazer a predição. Nesse caso, a latência de verificação é fixa para todas as predições. Em outras aplicações, a latência de verificação pode ser variável, como no caso de sensores que se mantém em ambientes externos, com pouca ou nenhuma interação humana por longos períodos de tempo (BATISTA *et al.*, 2011b).

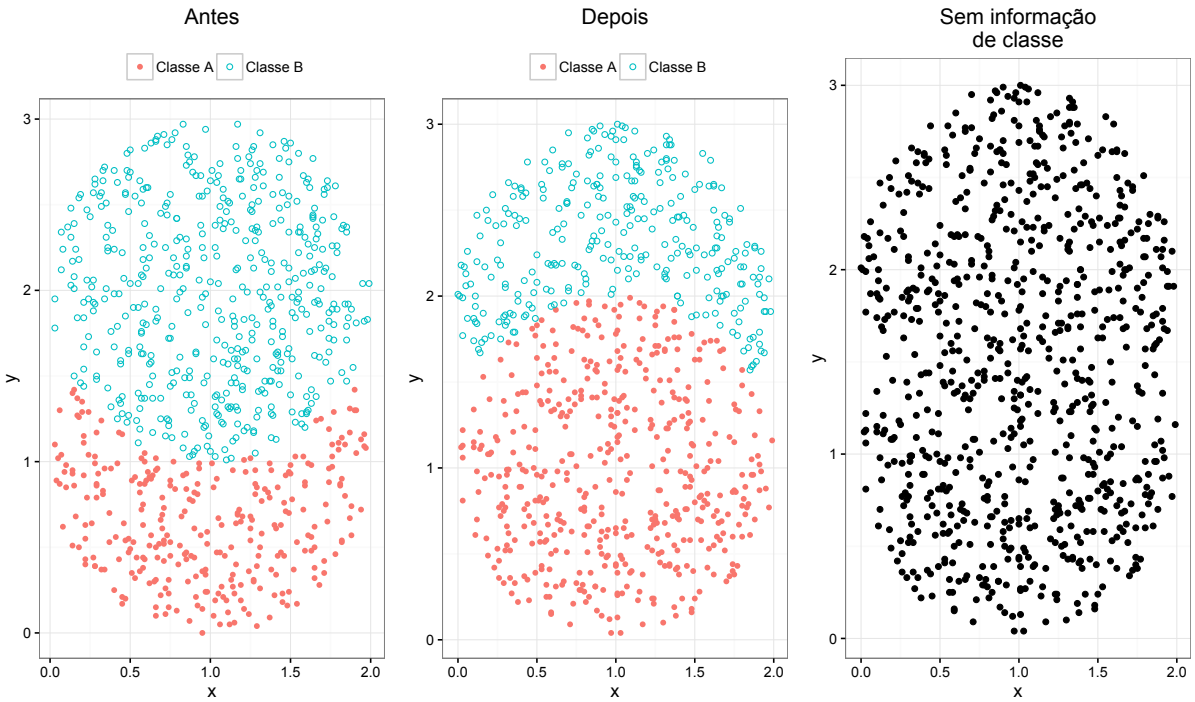
Cenários de latência nula são raros na prática, porém são comuns em configurações experimentais nos trabalhos científicos. Eles são raros em aplicações do mundo real pois ou *(a)* necessitam um certo período de tempo entre a predição e a concretização do evento no mundo real, quando se faz predições para o futuro, ou *(b)* porque necessitam de um tempo para que o rótulo verdadeiro seja fornecido por um oráculo externo. Em ambos os casos, o tempo entre a predição e a obtenção do rótulo verdadeiro é aquele disponível para se planejar e executar uma ação. Por exemplo, em uma aplicação de predição da demanda elétrica, na qual se planeja predizer a demanda com 30 minutos de antecedência, há 30 minutos para planejar e executar ações que possibilitem a geração da energia de acordo com a demanda prevista.

Diferentes artigos lidam com o problema de disponibilidade de rótulo considerando diferentes configurações experimentais. Em uma delas, se considera que nenhum rótulo verdadeiro é fornecido após determinado e curto período de tempo (SOUZA *et al.*, 2015; DYER; CAPO; POLIKAR, 2013). Outra configuração assume que apenas uma porção dos rótulos verdadeiros são disponibilizados ao longo do tempo. Em particular, para aprendizado semi-supervisionado, é comum assumir que uma parte dos rótulos se tornam disponíveis com latência nula, enquanto a parte restante nunca é disponibilizada (MASUD *et al.*, 2011).

Similarmente à configuração de aprendizado semi-supervisionado, aprendizado ativo também assume que apenas parte dos dados são rotulados. A diferença crítica entre os dois casos é a de que, em aprendizado ativo, o algoritmo pode escolher quais exemplos serão rotulados por um oráculo. Uma de nossas propostas é, em essência, um algoritmo de aprendizado ativo. O fluxo é monitorado de forma não supervisionada e nenhum rótulo é requerido caso não haja mudança de conceito. Caso uma mudança de conceito seja detectada, uma das soluções propostas é a requisição dos rótulos dos últimos eventos, para re-induzir o modelo de predição.

Na literatura, o trabalho mais relacionado ao nosso foi proposto por Žliobaitė (2010). Ela mostrou que alguns tipos de mudança de conceito se tornam indetectáveis quando rótulos verdadeiros não são providos. No geral, mudanças indetectáveis são aquelas nas quais nem a densidade relativa, nem o formato das distribuições no espaço de atributos se alteram. Um exemplo simples é, em um problema de classificação binária com classes balanceadas, a troca das distribuições que geram os exemplos das duas classes, de forma que uma delas passe a ter características da outra e vice-versa. Figura 4 ilustra esse fenômeno.

Figura 4 – Ilustração de uma mudança de conceito indetectável sem informação de classe.



Fonte: Elaborada pelo autor.

Zliobaite propôs um método para identificar mudanças detectáveis sem necessidade de rótulos verdadeiros. Duas janelas deslizantes consecutivas, sem interseção e de igual tamanho, acompanham o fluxo de dados. As janelas podem conter informações do espaço de atributos ou informações da saída do classificador. A cada instante, *i.e.*, a cada ocorrência de um evento, um teste de hipótese é aplicado para verificar se as janelas deslizantes contém observações geradas por uma mesma distribuição. Caso negativo, uma mudança de conceito é detectada.

No caso do uso de informação do espaço de atributos, Zliobaite sugere utilizar todo o vetor descritivo dos eventos como variáveis multidimensionais, ou mapear tais vetores para variáveis unidimensionais. Entretanto, nenhum experimento foi executado para nenhuma das duas configurações. No caso do uso de informações da saída do classificador, Zliobaite sugere utilizar estimativas de probabilidades para os rótulos preditos, caso o classificador ofereça essa informação, ou apenas os rótulos preditos, caso contrário. Porém, no último caso é apenas possível detectar mudanças que ocasionem alterações na proporção das classes preditas.

Três testes de hipóteses foram aplicados na avaliação experimental. Entre eles, a utilização da implementação padrão do teste Kolmogorov-Smirnov foi defendida por ser não paramétrico e computacionamente eficiente – quando comparado às outras opções. Testes não paramétricos se adequam melhor às misturas de distribuições intrínsecas ao fluxo de dados, que possui diferentes classes e mudanças de conceito. Enquanto Žliobaitė (2010) apresenta resultados relacionados à detecção de mudança de conceito, nenhum resultado é reportado referente ao impacto prático da detecção em um problema de classificação, seja em termos de acurácia ou em termos de quantidade de rótulos verdadeiros requeridos.

Haque, Khan e Baron (2015) propõe um algoritmo que não faz uso de rótulos para identificar o tamanho ótimo de uma janela deslizante de tamanho variável, que supostamente contém eventos pertencentes a um mesmo conceito. Uma vez que uma mudança é detectada, a janela é encolhida para manter apenas os eventos que pertencem ao mais novo conceito. O algoritmo assume a disponibilidade instantânea de 99.7% dos rótulos corretos.

Masud *et al.* (2011) apresentam um algoritmo que é capaz tanto de lidar com atrasos na disponibilidade de rótulos quanto identificar novidades, *i.e.*, novas classes durante o fluxo de dados. Entretanto, além das premissas específicas relativas ao espaço de atributos, há também a premissa de que, uma vez que um evento é apresentado, o classificador pode atrasar sua classificação por um período de tempo, em um cenário levemente diferente daquele a qual este trabalho se propõe a resolver.

Apesar de Kuncheva *et al.* (2008) e Amir e Toshniwal (2010) proporem métodos para lidar diretamente com atrasos na disponibilidade de rótulos verdadeiros, eles estão limitados unicamente a problemas com distribuições estacionárias, *i.e.*, problemas sem mudanças de conceito. Kuncheva propõe diferentes variações para o *Nearest Neighbor Classifier* (NNC) para aprendizado *online*. Em todas elas, o conjunto de referência do algoritmo cresce com o tempo, com rótulos não rotulados sendo adicionados quando suas confianças de classificação são baixas. Nesse caso, os rótulos preditos são assumidos corretos até serem futuramente revistos, quando rótulos corretos são fornecidos. A proposta de Amir é uma extensão do trabalho de Kuncheva. Faz uso de *padrões emergentes* para selecionar quais eventos devem ser inseridos no conjunto de referência do NNC.

Outros trabalhos lidam com o problema de latência de verificação extrema, *i.e.*, não disponibilidade dos rótulos durante toda a fase de classificação do fluxo de dados. Os métodos

propostos requerem apenas uma porção inicial dos dados rotulados, que geralmente necessita ser disponibilizada em um período imediatamente anterior ao restante do fluxo de dados. Nenhum outro rótulo é requisitado, depois deste ponto.

Dyer, Capo e Polikar (2013) e Souza *et al.* (2015) lidam exclusivamente com o problema de latência extrema. Ambos os trabalhos fazem premissas referentes ao formato dos dados no espaço de atributos e assumem uma natureza incremental às mudanças de conceito. A proposta de Dyer aplica uma técnica de classificação semi-supervisionada em janelas consecutivas de dados não rotulados. Para cada janela, é considerado que os rótulos preditos para os eventos da janela anterior estão corretos. A abordagem de Souza utiliza agrupamento de dados periodicamente e verifica a similaridade espacial entre os grupos para assumir o movimento dos dados, no espaço de atributos.

A literatura também tratou do problema de escassez e atraso de rótulos verdadeiros com técnicas de aprendizado semi-supervisionado.

Pozzolo *et al.* (2015) propõe um método para tratar diretamente de atrasos na disponibilidade de dados rotulados na detecção de fraudes de cartão de crédito. A proposta assume que, enquanto uma porção maior dos dados é afetada por atraso, uma porção menor é disponível instantaneamente. Os resultados sugerem que combinar um modelo induzidos sobre os dados rotulados instantaneamente e um modelo diferente, induzido sobre os dados que foram afetados por atraso, provê melhores resultados do que usando apenas um modelo induzido sobre todos os dados.

Masud *et al.* (2012) apresentam um método para o problema de escassez de rótulos, no qual apenas uma porção dos dados é rotulada. Os eventos rotulados, entre uma quantidade muito maior de eventos não rotulados, são usados em uma abordagem semi-supervisionada para classificar futuros eventos não rotulados.

Wu, Li e Hu (2012) introduzem outra abordagem semi-supervisionada que lida com escassez de dados rotulados, em fluxos de dados. É baseada em uma árvore de decisão que cresce incrementalmente e mantém grupos em suas folhas, para detectar mudanças de conceito.

2.6 Considerações finais

Boa parte das técnicas de classificação de fluxo de dados são baseadas em algoritmos de aprendizado supervisionado (HULTEN; SPENCER; DOMINGOS, 2001; KOLTER; MALOOF, 2005; ZHU; WU; YANG, 2006). Desse modo, a qualidade e a quantidade de dados rotulados influenciam diretamente no desempenho do classificador induzido, uma vez que assim que as previsões de exemplos de teste são realizadas, os rótulos verdadeiros são fornecidos para a avaliação do modelo e sua consequente adaptação ou escolha no caso de um comitê de modelos de classificação. Algumas abordagens consideram um atraso no recebimento dos rótulos. Outras,

que tipicamente utilizam aprendizado semi-supervisionado ou ativo, ainda conseguem funcionar em um ambiente em que apenas parte dos rótulos verdadeiros são fornecidos. Alguns poucos métodos existentes na literatura lidam com a situação na qual nenhum rótulo verdadeiro é fornecido. De modo geral, trabalhos que, de alguma forma, introduzem atraso ou escassez de rótulos, os fazem com premissas específicas, geralmente atreladas à algum domínio de aplicação, e que não são necessariamente são válidas à maioria dos outros trabalhos. Apesar da necessidade de métodos de avaliação específicos à cada caso, a maior parte dos trabalhos deriva sua avaliação de métodos já difundidos na área de fluxo de dados. Formas de avaliação e comparação de algoritmos comuns na literatura são apresentadas no próximo capítulo.

CAPÍTULO

3

AVALIAÇÃO EM FLUXO DE DADOS

A quantidade emergente de trabalhos relacionados à classificação em fluxo de dados evidencia a necessidade de formas de avaliação e comparação que possibilitem uma arguição estatisticamente válida sobre a superioridade de um algoritmo frente a outros. Devido principalmente ao fator não estacionário, muitos dos métodos de avaliação já consagrados na classificação estática não podem ser diretamente aplicados. Este capítulo introduz os métodos que hoje são utilizados para avaliar e comparar algoritmos que lidem com fluxos de dados.

3.1 Motivação

Apesar do grande e crescente número de trabalhos relacionados ao aprendizado em fluxo de dados, medidas e metodologias experimentais para estimar e comparar a qualidade de modelos de aprendizado ainda são um problema em aberto. As principais dificuldades existentes são (GAMA *et al.*, 2010):

- Há um fluxo contínuo de dados ao invés de uma amostra de exemplos independentes e identicamente distribuídos;
- Modelos de decisão evoluem com o tempo, ao contrário de modelos estáticos;
- Dados são gerados em um ambiente dinâmico, por distribuições não estacionárias, ao contrário de uma amostragem estacionária. Assim, a ordem dos exemplos é decisiva para compreensão das mudanças existentes, de forma que técnicas conhecidas como *k-fold cross validation* ou *holdout* - no qual os exemplos são dispostos em ordem aleatória, antes do particionamento em treino e teste - não podem ser diretamente aplicadas.

Dietterich (1998) propõe uma técnica direta para avaliar algoritmos de aprendizado quando dados são abundantes: induzir um modelo a partir de uma grande quantidade de dados e

testá-lo em uma grande quantidade de dados.

A técnica aparenta ser viável para um cenário de fluxo de dados, uma vez que em muitos casos há um fluxo contínuo e potencialmente infinito, do qual pode-se extrair quantos exemplos forem necessários. Entretanto, apesar de um fluxo de dados ser potencialmente infinito, dificilmente há a possibilidade de obter a quantidade desejada de exemplos de forma instantânea, fazendo-se necessário aguardar sua ocorrência, o que pode demandar uma quantidade de tempo inaceitável. Mesmo que seja realizada a coleta dos exemplos, ainda é necessária sua classificação, que pode ser manual e implicar em custos financeiros relacionados a contratação de um especialista na área. Por fim, um grande conjunto de dados para o treino não necessariamente implica em um melhor aprendizado sobre determinada classe, quando os exemplos são provenientes de distribuições não estacionárias: a coleta de dados por um período prolongado de tempo favorece a existência de múltiplos conceitos para uma mesma classe dentro do conjunto de treino inicial, devido às mudanças de conceito que possam ter ocorrido.

3.2 Medidas de avaliação

Para avaliar um modelo de aprendizado em um contexto de fluxo de dados, Dawid (1984) apresenta o método *Prequential*, no qual o erro de um modelo é computado a partir de uma sequência de exemplos. Para cada evento do fluxo, o modelo avaliado efetua a classificação conhecendo apenas seus atributos, *i.e.* ignora sua classe correta. O *prequential error* S_i sobre i eventos passados é computado com base em uma soma acumulada de uma função de perda L entre a predição $\hat{y}_j$ e o a classe correta y_j, $1 \leq j \leq i$, apresentada na Equação 3.1. A perda média - ou erro médio - E_i é apresentada na Equação 3.2.

$$S_i = \sum_{j=1}^{i} L(y_j, \hat{y}_j) \tag{3.1}$$

$$E_i = \frac{1}{i} \times S_i \tag{3.2}$$

Observa-se que é possível utilizar o método mesmo quando não se tem conhecimento de y_j para todos os valores de j, pelo cálculo da função de perda e S_j correspondente apenas para os exemplos cujas classes são conhecidas.

Se for disponível uma base com uma quantidade n de eventos de teste grande o suficiente, pode-se dividí-la em Q fluxos, considerar cada um como uma base independente e utilizar as abordagens exibidas nas Seções 3.3.2 e 3.3.3 para comparação de algoritmos com múltiplas bases de dados. Não se deve calcular uma média dos erros médios como medida segura pois, devido à mudança de conceito, as amostras não provém de dados pertencentes às mesmas distribuições. Ademais, no caso específico de aprendizado com latência de verificação extrema, pode ser interessante utilizar a base tão grande quanto possível, sem dividí-la, para averiguar a capacidade

do algoritmo em rastrear de maneira correta as mudanças de conceito em um período prolongado. A situação de latência de verificação extrema favorece esse cenário - quanto maior o fluxo, em termos de eventos de teste, menor é a quantidade proporcional de rótulos corretos que estão disponíveis.

Para qualquer função de perda, é possível estimar um intervalo de confiança para a probabilidade de erro, $E_i \pm \varepsilon$, por meio dos Limites de Chernoff (CHERNOFF, 1952), exibido na Equação 3.3, na qual $\delta \in (0,1)$ é um nível de confiança definido pelo usuário, n é a quantidade de observações, e $\bar{\mu}$ é um estimador para a perda média, por exemplo, pode-se considerar $\bar{\mu} = E_i$.

$$\varepsilon_c = \sqrt{\frac{3 \times \bar{\mu}}{n} \ln(2/\delta)} \tag{3.3}$$

Caso esteja sendo utilizada uma função perda com valores discretos e finitos, como a função $0-1$, que possui valor 0 quando ambas as classes predita e correta são iguais e 1 caso contrário, pode-se fazer uso dos Limites de Hoeffding (HOEFFDING, 1963), exibido na Equação 3.4, na qual R é o número de valores possíveis da função de perda menos um. A função de perda $0-1$ tem $R = 1$, por exemplo.

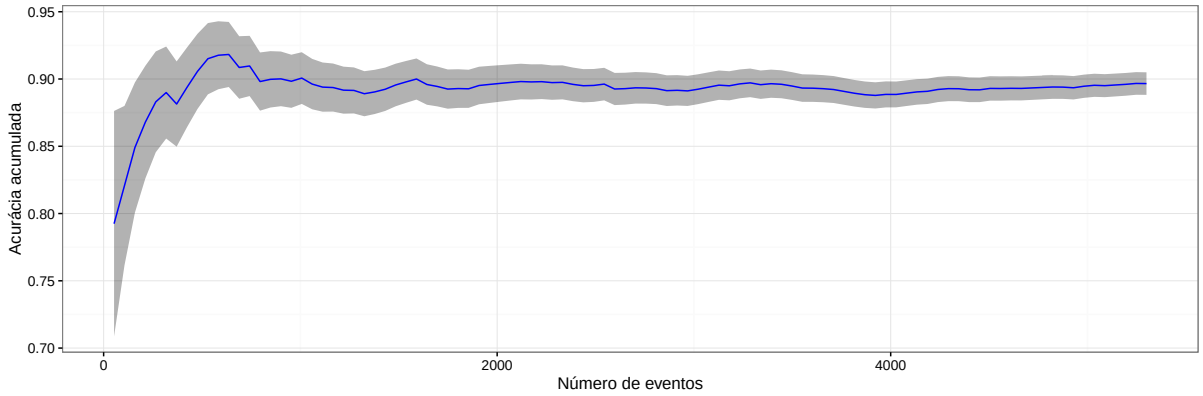
$$\varepsilon_h = \sqrt{\frac{R}{2n} \ln(2/\delta)} \tag{3.4}$$

Há um efeito colateral para a avaliação de *prequential-error* como apresentada. Quanto maior o valor de n, menor a influência de cada perda instantânea L_j no cálculo de S_i, de forma que S_i converge para um valor específico ao longo do tempo. Simultaneamente, os intervalos de confiança se estreitam e, quando n tende ao infinito, os intervalos se aproximam de ± 0. Essa situação seria muito desejável em classificação com distribuições estacionárias. Nesse cenário, qualquer parte do fluxo apresentaria resultados referentes a capacidade do algoritmo em classificar o mesmo conceito, e uma medida média convergente faz sentido. Porém, quando se lida com mudança de conceito, o gráfico obtido torna muito difícil, senão impossível, a avaliação de partes específicas do fluxo, para as quais se tem conhecimento dos conceitos que definem as classes e que são diferentes dos conceitos de outras partes do fluxo. Além disso, uma medida média convergente para todo o fluxo pode não ser interessante, visto que o fluxo apresenta diferentes características em seus dados, ao longo do tempo. A Figura 5 apresenta a acurácia, ou seja, o equivalente ao erro médio utilizando como função de perda definida na Equação 3.5, a fim de ilustrar os problemas mencionados.

$$L(y,\hat{y}) = \begin{cases} 1 & \text{se } y = \hat{y} \\ 0 & \text{caso contrário} \end{cases} \tag{3.5}$$

Para contornar este problema, é possível utilizar mecanismos de esquecimento, como janelas deslizantes. Com este mecanismo, escolhido um tamanho de janela w, o erro médio E_i^{sw}

Figura 5 – Acurácia do algoritmo Vizinho Mais Próximo com janela deslizante de 100 eventos sobre a base Insects (SOUZA *et al.*, 2013). A área sombreada corresponde ao intervalo de confiança de 95% segundo os limites de Hoeffding.



Fonte: Elaborada pelo autor.

é redefinido pela Equação 3.6, na qual S_i^{sw} é redefinido pela Equação 3.7.

$$E_i^{sw} = \frac{1}{w} \times S_i^{sw} \tag{3.6}$$

$$S_i^{sw} = \sum_{j=i-w+1}^{i} L_j \tag{3.7}$$

Para a confecção de um gráfico, é necessário escolher um valor $s \geq 1$ correspondente ao espaçamento entre as janelas, *i.e.*, a diferença dos índices de quaisquer dois exemplos que assumam a mesma posição relativa em janelas diferentes consecutivas. Por exemplo, caso $s = 1$, têm-se a maior sobreposição possível, com uma janela diferindo apenas em um exemplo da janela anterior: o evento mais antigo da janela anterior deixa de ser utilizado e o exemplo mais recente ocorrido é incorporado na nova janela. Habitualmente, $1 \leq s \leq w$, pois caso $s > w$, há exemplos que são descartados sem serem empregados em janela alguma, causando perda de informação útil. De modo geral, quando s se aproxima de 1 há mais sobreposição e, quando s se aproxima de w, há menos sobreposição, não havendo sobreposição algum caso $s = w$. Quanto maior a sobreposição, mais *suave* será a transição entre dois pontos consecutivos em um gráfico. Quanto menor a sobreposição, menor a suavidade.

Outro método para contornar a convergência do erro médio é a utilização de um fator de desvanecimento $\alpha \in (0,1)$ (GAMA *et al.*, 2010). Neste caso, o erro médio E_i^{ff} é redefinido segundo a Equação 3.8, enquanto o *prequential error* S_i^{ff} é redefinido de acordo com a Equação 3.9.

$$E_i^{ff} = \frac{1}{n} \times S_i^{ff} \tag{3.8}$$

$$S_i^{ff} = L_i + \alpha \times S_{i-1}^{ff} \qquad , \qquad 0 < \alpha < 1 \tag{3.9}$$

Nota-se que E_i^{ff} converge para 0 quando i tende a $+\infty$. Utilizando um fator de correção, obtém-se a Equação 3.10, na qual n_i é o número de exemplos usados para calcular L_i. Quando L_i é calculado para todo exemplo, $n_i = 1$.

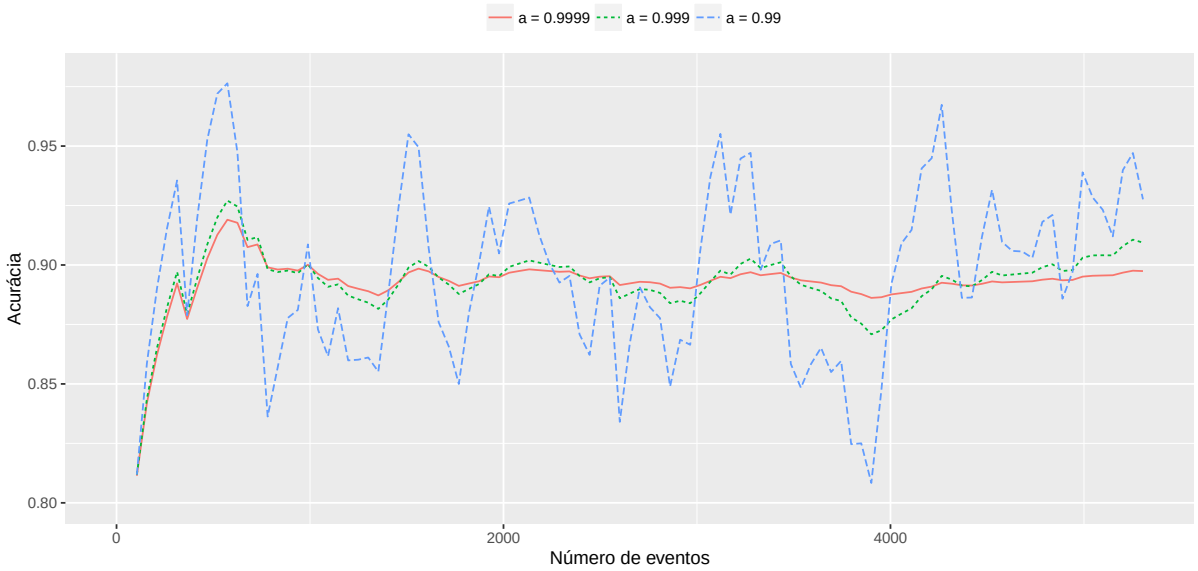
$$E_i^{ff} = \frac{S_i^{ff}}{B_i} = \frac{L_i + \alpha \times S_{i-1}^{ff}}{n_i + \alpha \times B_{i-1}} \tag{3.10}$$

O uso de tais artifícios não é desacompanhado de novas dificuldades. Tanto o uso de fatores de desvanescimento quanto janelas deslizantes inserem novas variáveis que devem ser consideradas, para reproduzir seus resultados. No caso dos fatores de desvanescimento, o valor de α precisa ser escolhido. No caso das janelas deslizantes, precisam ser escolhidos tanto n quanto a taxa de sobreposição das janelas, *i.e.*, se de uma janela para a seguinte só haverá apenas um exemplo diferente (troca do mais antigo para ou mais novo), ou mais exemplos diferentes, até a possível escolha de não haver qualquer sobreposição. A Figura 6 exibe o resultado da execução de um algoritmo de classificação arbitrário, em termos de acurácia, com diferentes fatores de desvanescimento. Quanto menor o valor de α, maior o fator de esquecimento e mais o gráfico se aproxima da exibição instantânea da função perda. Por fim, o uso de fatores de desvanescimento torna os intervalos de confiança apresentados não aplicáveis. Já no caso de janelas deslizantes, porém, é possível aplicar tais limites sem qualquer modificação, um para cada janela calculada. A Figura 7 ilustra a curva de acurácia obtida, aplicada a técnica de janelas deslizantes, com Limites de Hoeffding para um intervalo de confiança de 95%.

Os fatores de desvanecimento são multiplicativos, o que implica em um esquecimento exponencial: para o i-ésimo exemplo, o exemplo $i-k$ tem peso α^k. Assim, se é desejado ignorar exemplos com um peso inferior a ε, um limite superior para k pode ser dado por $\log(\varepsilon)/\log(\alpha)$. Entretanto, em um uso iterativo do método apresentado, para avaliar a classificação em um fluxo de dados, tal consideração é desnecessária. Um elemento da sequência depende diretamente apenas do anterior, que já deve ter sido calculado quando foi efetuada a classificação do evento anterior, no fluxo. Essa é uma grande vantagem no uso dos fatores de desvanecimento quando comparado com o uso de janelas deslizantes: não é necessário armazenar muitos valores da sequência. De fato, apenas o último valor obtido para cada algoritmo precisa ser armazenado. O esquecimento exponencial implica, no entanto, em um parâmetro α mais sensível e difícil de ser escolhido do que o tamanho da janela, no caso do uso de janelas deslizantes.

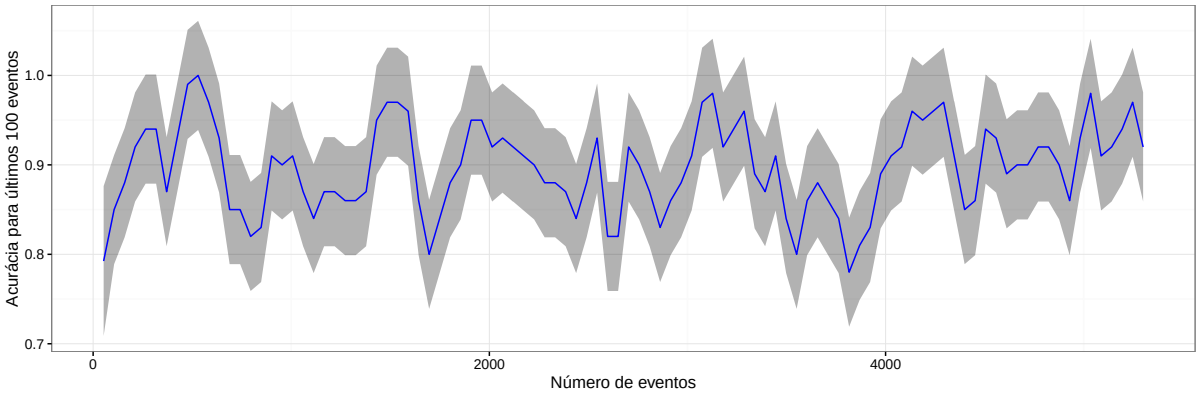
Mais recentemente, Bifet *et al.* (2015) sugere a aplicação da técnica ADWIN (BIFET; GAVALDÀ, 2007) como alternativa ao uso típico de janelas deslizantes de tamanho fixo. ADWIN é um algoritmo para adaptar o tamanho de uma janela deslizante automaticamente. A janela ADWIN se expande até os dados nela contidos passem a formar duas distribuições de probabilidade diferentes e consecutivas. Quando o algoritmo identifica essa situação, a janela é encolhida

Figura 6 – Acurácia com diferentes fatores de desvanescimento do algoritmo Vizinho Mais Próximo com janela deslizante de tamanho 100 sobre a base Insects.



Fonte: Elaborada pelo autor.

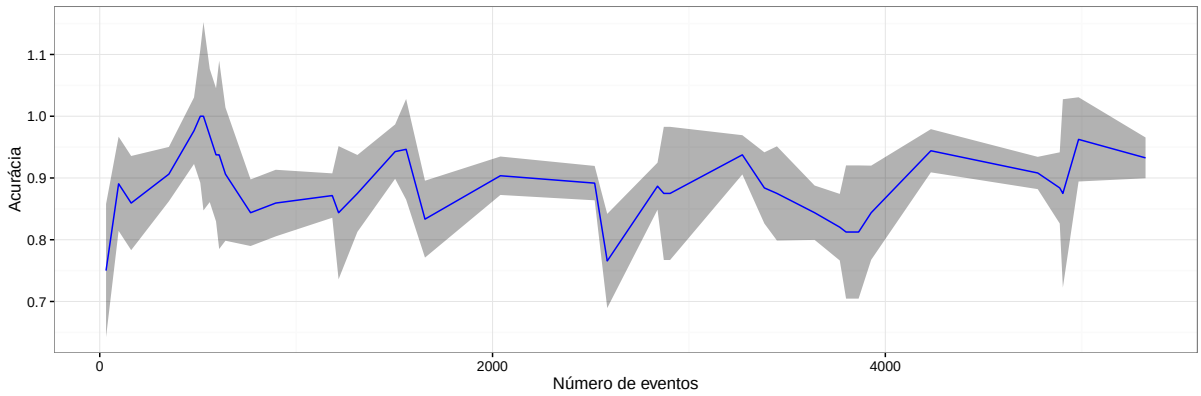
Figura 7 – Acurácia com janela deslizante sem sobreposição. A curva é formada por 101 pontos que exibem a média da função de perda para os 100 eventos anteriores. As perdas foram obtidas pelo algoritmo Vizinho Mais Próximo com janela deslizante de tamanho 100 sobre a base Insects. A área sombreada corresponde ao intervalo de confiança de 95% segundo os limites de Hoeffding.



Fonte: Elaborada pelo autor.

e as observações mais antigas, pertencentes à primeira distribuição, são descartadas da janela. No caso de seu uso em substituição à aplicação de janelas deslizantes para graficação do *prequential*, os valores da função de perda são inseridos à janela ADWIN. Figura 8 ilustra a curva obtida pela técnica. É notável sua semelhança à curva apresentada pela Figura 7. Apesar de conter menos detalhes, o formato obtido é grosseiramente o mesmo. Apesar da facilidade aparente, deve-se notar que, em vias práticas, a aplicação do método ADWIN inclui a escolha de parâmetro de sensibilidade, que influencia diretamente na quantidade de janelas detectadas.

Figura 8 – Acurácia com janela deslizante sem sobreposição. Os valores de perda foram obtidos pelo algoritmo Vizinho Mais Próximo com janela deslizante de tamanho 100 sobre a base Insects. Enquanto o tamanho de janela utilizada pelo classificador é fixo, Os tamanhos das janelas para cálculo e graficação das acurácias são variáveis e escolhidos automaticamente pelo algoritmo ADWIN. A área sombreada corresponde ao intervalo de confiança de 95% segundo os limites de Hoeffding.



Fonte: Elaborada pelo autor.

3.3 Comparação de algoritmos

A seção anterior apresenta ferramentas para quantificar o desempenho de um único classificador em um fluxo de dados, o que pode ser útil para verificar, intuitivamente, se o algoritmo utilizado é suficientemente bom para suprir as necessidades de um dado problema. Entretanto, é natural o interesse em comparar diferentes algoritmos, para averiguar qual deles possui o melhor desempenho, seja de acordo com menor taxa de erro ou outra medida desejada, como acurácia. Esta seção discute separadamente a comparação entre algoritmos em casos específicos. São os que seguem: comparação entre dois algoritmos em um único fluxo de dados; comparação entre dois algoritmos em vários fluxos de dados; e comparação entre múltiplos algoritmos em vários fluxos de dados.

3.3.1 Comparação entre dois algoritmos em um fluxo de dados

Sejam S_i^A e S_i^B as somas acumuladas de perdas de dois diferentes algoritmos (A e B), em um fluxo de dados. O objetivo é o de mostrar que os algoritmos têm desempenhos diferentes. Assim, a hipótese nula é a de que ambos são iguais. Uma estatística útil que pode ser utilizada com praticamente qualquer função de perda é exibida na Equação 3.11 (GAMA *et al.*, 2010).

$$Q_i(A,B) = \log\left(\frac{S_i^A}{S_i^B}\right) \tag{3.11}$$

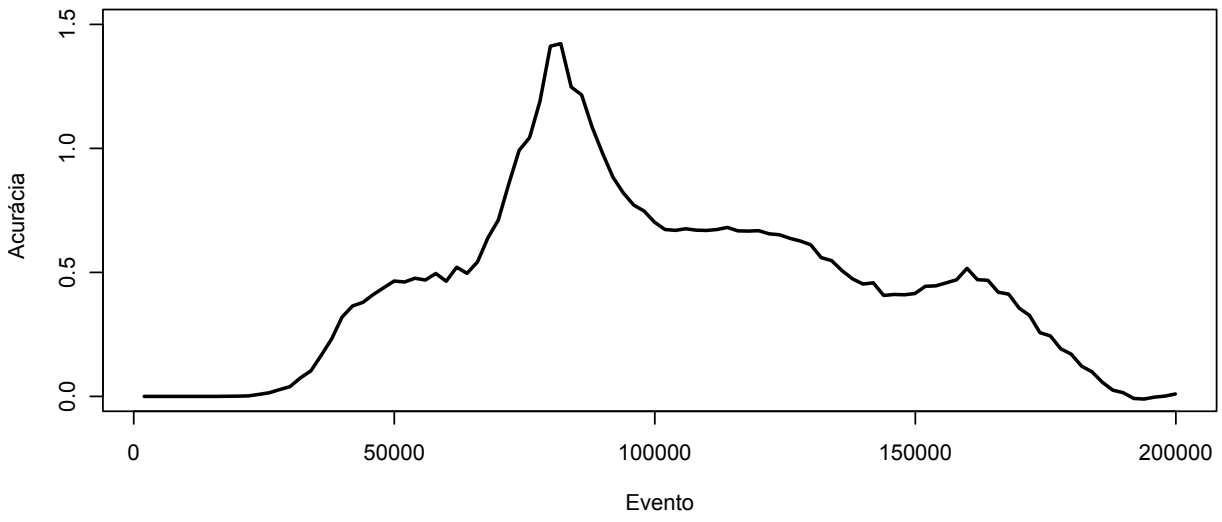
O sinal de Q_i é informativo quanto a performance relativa dos dois modelos, enquanto o valor mostra a força da diferença. Aplicando fatores de desvanecimento à estatística, obtém-se

como resultado a Equação 3.12.

$$Q_i(A,B) = \log\left(\frac{L_i^A + \alpha \times S_{i-1}^A}{L_i^B + \alpha \times S_{i-1}^B}\right) \tag{3.12}$$

Também é possível utilizar a técnica com janelas deslizantes, ao invés de fatores de desvanescimento, como ilustrado pela Figura 9, que exibe uma comparação entre dois algoritmos arbitrários em uma base qualquer. Como a curva permanece sempre acima de 0, o algoritmo A foi melhor que B ao longo de todo o fluxo. Além disso, é possível observar que a maior perda de desempenho relativo se deu no meio do fluxo. A perda de desempenho relativo pode ter ocorrido pela perda individual de desempenho de A, pelo aumento de desempenho individual de B, ou uma combinação de ambos. Conhecendo as características da base, é possível inferir os motivos dessa diferença de desempenho nessa parte do fluxo, por exemplo.

Figura 9 – Comparação entre dois algoritmos arbitrários na base UG_UC_3D (SOUZA *et al.*, 2014), utilizando estatística Q com janela deslizante sem sobreposição com tamanho de 1% do fluxo total.



Fonte: Elaborada pelo autor.

Essa técnica pouco difere de exibir, em um mesmo gráfico, uma sobreposição do *prequential* de cada algoritmo e avaliar visualmente a diferença de performance entre os eles. Apesar da simplicidade, que é de fato um ponto positivo, carece de garantias estatísticas.

A fim de obter tais garantias, um dos testes mais usados é o *McNemar* (MCNEMAR, 1947). Para possibilitar seu cálculo, são necessários dois valores: $n_{0,1}$ e $n_{1,0}$. O primeiro denota o número de classificações incorretas efetuadas por A e não efetuadas por B. O segundo denota o número de classificações incorretas efetuadas por B e não efetuadas por A. Obtendo estes valores, calcula-se a estatística M segundo a Equação 3.13, que possui distribuição χ^2 com 1 grau de liberdade. Para um nível de confiança de 99%, a hipótese nula é rejeitada se a estatística for

superior a 6.635 (DIETTERICH, 1998).

$$M = \text{sign}(n_{0,1} - n_{1,0}) \times \frac{(n_{0,1} - n_{1,0})^2}{n_{0,1} + n_{1,0}} \tag{3.13}$$

Entretanto, esse método geralmente não se faz adequado para a aplicação em fluxo de dados, pois mesmo que os eventos sejam independentes, as medidas de avaliação, que são de fato as variáveis aleatórias do problema, não são independentes, garantia exigida pelo método. A medida de avaliação é uma função das classificações, pois utiliza a contagem de acertos em sua formulação. Logo, se a medida de avaliação for utilizada como parâmetro para decidir quando adaptar o modelo de classificação, as classificações seguintes são influenciadas diretamente pelas anteriores, não sendo, portanto, independentes. Por fim, há ainda a possibilidade de adaptar o modelo com base nas classes preditas, o que leva ao mesmo problema de dependência das variáveis. Porém, se nunca houver adaptação do modelo, a medida é apropriada caso exista garantia de que as observações do fluxo sejam independentes. Entretanto, esta é uma garantia difícil de ser obtida em aplicações reais com fluxos de dados.

Para eliminar este problema, pode-se efetuar a comparação entre os dois algoritmos utilizando vários fluxos de dados, ao invés de apenas um. Se há um valor confiável para determinada medida de avaliação obtido em cada fluxo de dados, o uso de múltiplas bases de dados naturalmente gera uma amostra que possui valores independentes (DEMŠAR, 2006).

As seções seguintes apresentam os testes mais indicados para efetuar tais comparações, utilizando um único valor de medida de desempenho para cada par algoritmo-base, por exemplo o último erro médio E_n, obtido para todo o fluxo, sem uso de janelas deslizantes ou fatores de desvanescimento.

3.3.2 Comparação entre dois algoritmos em múltiplos fluxos de dados

Um teste amplamente empregado para a comparação de múltiplos tratamentos em múltiplas amostras é o Teste T pareado. Uma dificuldade em sua utilização é a de que, para amostras pequenas (menores do que cerca de 30 valores), a diferença entre as medidas obtidas pelos algoritmos comparados precisa constituir uma variável aleatória de distribuição normal. Como, a partir desta seção, os valores formados são cada um uma medida extraída a partir da execução de um algoritmo sobre um fluxo de dados completo, para utilizar o Teste T ou é necessário dispor de uma grande quantidade de bases de teste, ou garantir a normalidade das diferenças dos resultados. Infelizmente, testes que verificam a normalidade de uma amostra requerem grandes quantidades de dados (DEMŠAR, 2006), e possuir inúmeros fluxos pode se mostrar custoso e inviável. Uma alternativa é o emprego do Teste Sinal-Ranqueado de Wilcoxon (WILCOXON, 1945), um teste não paramétrico que ignora grandezas e se preocupa apenas com a posição dos valores em um *rank* para diferenças positivas e negativas.

Seja d_i a diferença entre a medida de performance entre os dois algoritmos comparados, para o i-ésimo fluxo de dados (assumindo que não haja $d_i = 0$, para qualquer i), de um total de N bases. A diferença é ranqueada de acordo com seus valores absolutos, sendo que a maior diferença absoluta possui *rank* 1, e a menor diferença absoluta possui *rank* N. Sejam R^+ a soma dos *ranks* para bases nas quais o segundo algoritmo obteve melhores resultados que o primeiro, e R^- a soma dos *ranks* para o caso oposto. As Equações 3.14 e 3.15 apresentam os cálculos de R^+ e R^- formalmente.

$$R^+ = \sum_{d_i>0} rank(d_i) \tag{3.14}$$

$$R^- = \sum_{d_i<0} rank(d_i) \tag{3.15}$$

Seja, por fim, $T = \min(R^+, R^-)$. Enquanto muitos livros de estatística incluem o valor crítico para valores de T quando N vai até 25, para uma quantidade ainda maior de bases de dados, a estatística z pode ser calculada de acordo com a Equação 3.16 e é distribuída aproximadamente de forma normal. Com uma confiança de 95%, a hipótese-nula, que dita a igualdade de performance dos algoritmos comparados, pode ser rejeitada caso z seja menor que -1.96 (DEMŠAR, 2006).

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{3.16}$$

O teste de Wilcoxon assume comensurabilidade qualitativa das diferenças. Apesar de diferenças maiores influenciarem mais, o quão grande são tais diferenças é irrelevante, o que pode ser considerado conservador. Entretanto, isso o torna menos suscetível à *outliers*. De um ponto de vista estatístico, é um teste mais seguro que o Teste T pareado, por não assumir normalidade dos dados.

3.3.3 Comparação entre múltiplos algoritmos em múltiplos fluxos de dados

De maneira semelhante à presente na decisão entre adotar o Teste T pareado ou o Teste Sinal-Ranqueado de Wilcoxon, quando comparando dois algoritmos em múltiplas bases, o método estatístico ANOVA (FISHER, 1956), que poderia ser utilizado para comparar múltiplos algoritmos em múltiplas bases, necessita de garantias que dificilmente podem ser cumpridas em aplicações de aprendizado de máquina (DEMŠAR, 2006), favorecendo a adoção de outro método, não paramétrico, como o Teste de Friedman (FRIEDMAN, 1937; FRIEDMAN, 1940).

Seja r_i^j o *rank* do j-ésimo algoritmo, dentre k comparados, para o i-ésimo fluxo utilizado, dentre N. O ranqueamento acontece de forma separada para cada base. Dado o i-ésimo fluxo, o

algoritmo que obteve melhor resultado de classificação segundo a medida utilizada recebe *rank* 1, o segundo melhor recebe *rank* 2 e assim por diante. O Teste de Friedman compara a média dos *ranks* dos algoritmos, R_j, definida segundo a Equação 3.17.

$$R_j = \frac{1}{N}\sum_i r_i^j \tag{3.17}$$

Sob a hipótese nula, que declara a equivalência dos algoritmos, a estatística χ_F^2, definida na Equação 3.18, é distribuída de acordo com a distribuição X^2 com $k-1$ graus de liberdade quando N e k são grandes o suficiente - nominalmente, $N > 10$ e $k > 5$ (DEMŠAR, 2006). Para um número pequeno de algoritmos e bases de dados, valores críticos foram computados (ZAR *et al.*, 1999; SHESKIN, 2000).

$$\chi_F^2 = \frac{12N}{k(K+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{3.18}$$

Iman e Davenport (1980) mostraram que a estatística χ_F^2 é indesejavelmente conservadora e derivaram uma melhor estatística, F_F, exibida pela Equação 3.19.

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} \tag{3.19}$$

F_F é segue a distribuição F com graus de liberdade $k-1$ e $(k-1)(N-1)$. Tabelas com valores críticos podem ser encontrados em livros de estatística.

Uma vez que a hipótese nula é rejeitada, considera-se que os algoritmos não são equivalentes e é possível então proceder com um teste *post-hoc*. Caso o objetivo seja comparar todos os algoritmos entre eles, pode-se utilizar o Teste de Nemenyi (NEMENYI, 1962), no qual dois classificadores são significantemente diferentes se a diferença das médias de seus *ranks* correspondentes for maior ou igual à diferença crítica *DC*, apresentada pela Equação 3.20, que possui valores $q_{0.05}$ e $q_{0.10}$, para confianças de 95% e 90%, respectivamente, apresentados na Tabela 1, quando o número de classificadores vai até 10. q_α são valores baseados na estatística *Studentized range*, divididos por $\sqrt{2}$.

$$DC = q_\alpha\sqrt{\frac{k(k+1}{6N}} \tag{3.20}$$

3.4 Comparação com *baselines*

Apesar de frequentemente negligenciada na literatura, a comparação de uma proposta com *baselines* pode revelar sua inutilidade, mesmo antes de confrontá-lo com os métodos estado-da-arte.

Tabela 1 – Valores críticos para o Teste de Nemenyi de duas caudas.

# classificadores	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.343	2.569	2.728	2.850	2.949	3.031	3.102	3.164
$q_{0.10}$	1.645	2.052	2.291	2.459	2.589	2.693	2.780	2.855	2.920

Fonte: Demšar (2006).

Caso os classificadores estiverem sendo comparados com um classificador de controle, pode-se utilizar o Teste de Bonferroni-Dunn (DUNN, 1961), que é bem mais poderoso para este caso específico (DEMŠAR, 2006). Uma forma de utilizar este teste é considerar a diferença crítica *DC* do teste de Nemenyi, mas utilizar um $\alpha' = \alpha/(k-1)$. A Tabela 2 apresenta valores $q'_{0.05}$ e $q'_{0.10}$, para confianças de 95% e 90%, respectivamente.

Tabela 2 – Valores críticos para o Teste de Bonferroni-Dunn de duas caudas.

# classificadores	2	3	4	5	6	7	8	9	10
$q'_{0.05}$	1.960	2.241	2.394	2.498	2.576	2.638	2.690	2.724	2.773
$q'_{0.10}$	1.645	1.960	2.218	2.241	2.326	2.394	2.450	2.498	2.539

Fonte: Demšar (2006).

Um classificador *baseline* consiste de um algoritmo extremamente simples e viciado por algum motivo, que leva um mínimo de informação em consideração. *Baselines* são utilizados para verificar se um algoritmo avaliado é minimamente bom, *i.e.*, se é capaz de classificar melhor que algoritmos não-inteligentes. O uso de *baselines* se dá pela simples comparação direta: exceto em casos especiais, o algoritmo avaliado deve possuir melhor performance que *todos* os *baselines* conhecidos para o problema.

Žliobaitė *et al.* (2014) apresentam dois *baselines*: *Majority classifier* e *Persistent classifier*.

O *Persistent classifier* classifica todos os eventos com a mesma classe de seus predecessores imediatos. Provê alta acurácia em fluxos de dados com dependência temporal positiva, sendo ao menos tão performático quanto *Majority classifier*, nesses casos.

O *Majority classifier* classifica cada evento com a classe mais recorrente até então. Esse algoritmo classifica, no mínimo, tão bem quanto um classificador aleatório, e classifica tão bem ou melhor que o *Persistent classifier* caso não haja dependência temporal (ŽLIOBAITĖ *et al.*, 2014). Adicionalmente, *Majority classifier* provê alta acurácia em dados desbalanceados.

Derivam dos *baselines* apresentados duas medidas de avaliação: estatística kappa e estatística kappa temporal (ŽLIOBAITĖ *et al.*, 2014).

A estatística kappa é uma medida de performance de classificação relativa à performance obtida pelo *Majority classifier*. O intervalo da medida é $(-\infty, 1]$. Um valor positivo é obtido se, e somente se, o algoritmo avaliado obtiver melhor acurácia que o *Majority classifier*. A estatística é dada segundo Equação 3.21, na qual A_Q é a acurácia obtida pelo algoritmo avaliado e A_M é a

acurácia obtida pelo *Majority classifier*.

$$\kappa = \frac{A_Q - A_M}{1 - A_M} \tag{3.21}$$

Similarmente, a estatística kappa temporal é uma medida que compara a performance de um classificador à performance obtida pelo *Persistent classifier*. Seu contra-domínio e significado são análogos aos da estatística kappa. A estatística é dada segundo a Equação 3.22, na qual A_Q é a acurácia obtida pelo algoritmo avaliado e A_P é a acurácia obtida pelo *Persistent classifier*.

$$\kappa_T = \frac{A_Q - A_P}{1 - A_P} \tag{3.22}$$

Os *baselines* apresentados, assim como as medidas derivadas, não foram originalmente propostos para serem aplicados em situações com latência de verificação superior a nula, e as garantias de performance são válidas apenas neste contexto.

3.5 Considerações finais

Esta seção apresentou formas de medir o desempenho de um algoritmo de classificação em fluxo de dados ao longo do tempo com margens de segurança estatisticamente válidas, formas de compará-lo com outro algoritmo, ainda ao longo do tempo, porém sem garantias estatísticas e, por fim, compará-lo com outro ou outros algoritmos com garantias estatísticas, mas apenas utilizando uma medida que sumarize o desempenho ao longo de todo o fluxo. De um modo prático, quando se utiliza bases de dados do tipo *benchmark*, para as quais são conhecidas as mudanças de conceito, a comparação entre algoritmos ao longo do fluxo, de forma contínua, pode se mostrar muito mais útil, evidenciando pontos fracos e fortes de cada algoritmo, mesmo que não se conte com garantias estatísticas das diferenças entre eles.

No entanto, para uso em aplicações reais, quando não se tem conhecimento de como e onde as mudanças de conceito realmente acontecem ao longo do tempo, uma medida que dita o quão bem o algoritmo se comportou no geral pode ser suficiente e permite o uso de ferramentas estatísticas para observar com determinado grau de certeza qual algoritmo tem melhor desempenho médio. Observa-se, porém, o risco do uso de uma medida sumarizadora para o desempenho em um fluxo: medidas médias podem indicar desempenho razoável, omitindo resultados pífios que são, porventura, compensados com momentos de excelentes resultados. A distinção desses casos pode ser facilmente observada pelas graficações ao longo do tempo, com uso de técnicas como fatores de desvanescimento ou janelas deslizantes.

CAPÍTULO

4

TÉCNICAS EXPLORATÓRIAS

Este capítulo apresenta ideias úteis para análises preliminares de bases de dados reais. Essencialmente, técnicas para coleta de evidência referentes à existência ou não de mudança de conceito e suas características, e manipulação de dependência temporal.

4.1 Distinção de problemas e definições comuns

Como mostrado na Seção 2.2, mudanças podem ocorrer em $P(c)$ ou em $P(x|c)$, resultando em mudanças em $P(c|x)$. Mudanças em cada uma destas duas primeiras probabilidades forma um problema distinto que pode ser mensurado independentemente. Entretanto, ambos dependem da definição de *tempo* e de uma definição formal para *conceito*.

Neste capítulo, um conceito específico para uma dada classe será definido como $S_t^c = \{P(c), P(x|c)\}$, para um dado instante t. Além disso, uma classe possui exatamente um conceito para qualquer t, apesar de $P(x|c)$ poder ser formado por uma mistura de diferentes distribuições de probabilidade.

Em fluxo de dados, é assumido que um conceito é capaz de mudar ao longo do tempo. Entretanto, há duas formas aceitáveis de definir tempo, no contexto de fluxo de dados. O primeiro, mais óbvio, é considerar o tempo de relógio, anotando o instante real em que cada evento foi registrado. Neste capítulo, esta forma de considerar o tempo será chamada *análise de tempo de relógio*. A segunda forma é considerar apenas a ordem relativa dos eventos, chamada *análise de tempo relativo*.

A *análise de tempo de relógio*, por ser um superconjunto da *análise de tempo relativo*, provê mais informações. Com ela é possível, por exemplo, verificar a variação de $P(c)$ para uma dada classe c independentemente de outras classes. Opostamente, na *análise de tempo relativo*, é apenas possível medir $P(c)$ ao longo do tempo de forma relativa às outras classes. *Análise de tempo de relógio* também pode ser útil para descoberta de padrões, *i.e.*, mudanças recorrentes.

Por exemplo, é possível observar a frequência de mudanças recorrentes de $P(x|c)$ sem que essas mudanças sejam obscurecidas por $P(c)$. Entretanto, muitas bases de dados disponíveis pela comunidade não anotam o instante no qual os eventos foram observados. Adicionalmente, a informação de tempo real não é necessariamente relevante para todas as aplicações reais. Um exemplo é a medição de características de seções consecutivas do asfalto de uma rua interditada ao longo de um percurso, para verificar se cada seção é considerada danificada ou não. O tempo não influencia as características medidas, mesmo que a sequência dos eventos se mantenha relevante. De forma geral, apesar de sequências poderem apresentar potenciais relações de similaridade entre elementos consecutivos, essas relações não estão obrigatoriamente vinculadas a um fator temporal. Esse capítulo lida apenas com *análise de tempo relativo*.

4.2 Evidências de mudanças em $P(x|c)$

Seja X^c uma sequência de eventos que pertencem à mesma classe c, em um determinado cojunto de dados. Formalmente, $y(X_i^c) = c \quad \forall X_i^c \in X^c, 1 \leq i \leq |X^c|$. Ademais, $i < j$ implica em X_i^c ter sido observado antes de X_j^c. Seja $D : X^c \times X^c \to \Re$ a dissimilaridade entre dois eventos. Para todos os propósitos, considere D uma métrica, o que impõe as restrições que seguem:

Não-negatividade: $D(x,y) \geq 0$

Simetria: $D(x,y) = D(y,x)$

Nulidade restrita a pontos coincidentes: $D(x,y) = 0 \leftrightarrow x = y$

Desigualdade triangular: $D(x,z) \leq D(x,y) + D(y,z)$

Adicionalmente, considere $p : \mathbb{N}^\star_{\leq |X^c|} \to \mathbb{N}^\star_{\leq |X^c|}$ uma função bijetiva que permuta uniformemente os valores no intervalo $[1, |X^c|]$.

Podemos definir $\bar{D}_{i,j}$ a média das distâncias entre $j-i$ pares aleatórios de eventos consecutivos.

$$\bar{D}_{i,j} \triangleq (j-i)^{-1} \sum_{k=i+1}^{j} D\left(X^c_{p(k)}, X^c_{p(k)-1}\right) \tag{4.1}$$

Similarmente, podemos definir $\bar{D}_{p(i,j)}$ como a média das distâncias entre $j-1$ pares aleatórios de eventos não necessariamente consecutivos.

$$\bar{D}_{p(i,j)} \triangleq (j-i)^{-1} \sum_{k=i+1}^{j} D\left(X^c_{p(k)}, X^c_{p(k-1)}\right) \tag{4.2}$$

$\bar{D}_{i,j}$ e $\bar{D}_{p(i,j)}$ consistem em duas variáveis aleatórias. Podemos amostrar r observações para cada uma delas, sendo r um parâmetro inteiro, e estimar suas médias e variâncias. A média

estimada são $\bar{D}$ e $\bar{D}_p$, dadas pelas Equações 4.3 e 4.4, respectivamente. As variâncias estimadas são V e V_p, dadas pelas Equações 4.5 e 4.6, respectivamente.

$$\bar{D} \triangleq \left\lfloor \frac{|X^c|}{r} \right\rfloor^{-1} \sum_{\forall k \in \mathbb{N}_{>0} | (k+1)r < |X|^c} \bar{D}_{kr,(k+1)r} \tag{4.3}$$

$$\bar{D}_p \triangleq \left\lfloor \frac{|X^c|}{r} \right\rfloor^{-1} \sum_{\forall k \in \mathbb{N}_{>0} | (k+1)r < |X|^c} \bar{D}_{p(kr,(k+1)r)} \tag{4.4}$$

$$V \triangleq \left\lfloor \frac{|X^c|}{r} \right\rfloor^{-1} \sum_{\forall k \in \mathbb{N}_{>0} | (k+1)r < |X|^c} \left(\bar{D}_{kr,(k+1)r} - \bar{D}\right)^2 \tag{4.5}$$

$$V_p \triangleq \left\lfloor \frac{|X^c|}{r} \right\rfloor^{-1} \sum_{\forall k \in \mathbb{N}_{>0} | (k+1)r < |X|^c} \left(\bar{D}_{p(kr,(k+1)r)} - \bar{D}_p\right)^2 \tag{4.6}$$

Sabe-se, por meio do Teorema do Limite Central, que $\bar{D}_{kr,(k+1)r} \forall k \in \mathbb{N}_{>0} \mid (k+1)r < |X|^c$ é aproximadamente normalmente distribuído com média estimada $\mu \approx \bar{D}$ e variância estimada $\sigma^2 \approx V$ e, analogamente, $\bar{D}_{p(kr,(k+1)r)} \forall k \in \mathbb{N}_{>0} \mid (k+1)p < |X|^c$ é aproximadamente normalmente distribuído coim média estimada $\mu_p \approx \bar{D}_p$ e variância estimada $\sigma_p^2 \approx V_p$. A partir disto, pode-se formular a seguinte hipótese nula:

Hipótese nula: $\mu = \mu_p$

Hipótese alternativa: $\mu \neq \mu_p$

Com o fim de verificar se se deve rejeitar a hipótese nula, pode-se calcular z de acordo com a Equação 4.7. Caso $|Z| > 1.96$, pode-se rejeitar a hipótese nula com significância de 0.05.

$$Z = (\mu - \mu_p) \left(\frac{\sigma + \sigma_p}{\left\lfloor \frac{|X^c|}{p} \right\rfloor} \right)^{-\frac{1}{2}} \sim \mathcal{N}(0,1) \tag{4.7}$$

Na prática, rejeitar a hipótese nula oferece uma evidência de que $P(X|c)$ se altera com o tempo. Entretanto, caso a hipótese nula não seja rejeitada, há apenas uma *fraca* evidência de que $P(X|c)$ não se altera com o tempo. A intuição por trás do teste vem do Problema do Caixeiro Viajante (TSP). Precisamente, se estamos interessados em resolver o problema usando todos os eventos da base de dados como se fossem pontos a serem visitados em um hiperespaço, então podemos dizer que $\sum_{i=2}^{|X^c|} D\left(X_i^c, X_{i-1}^c\right)$, a soma das distâncias entre todos os eventos consecutivos, dois a dois, é uma possível solução para o problema, chamada *solução causal*. Esta solução equivale a caminhar pelos pontos da forma como foram apresentados na base. Outra solução, entretanto, poderia ser $\sum_{i=2}^{|X^c|} D\left(X_{p(i)}^c, X_{p(i-1)}^c\right)$, chamada *solução ocasional*, que equivale a um

percurso aleatório pelos pontos disponíveis. A solução causal pode ser tão boa quanto uma solução ocasional – situação expressada pela hipótese nula –, melhor que a solução ocasional, o que se dá por uma menor soma de dissimilaridades, ou pior do que a solução ocasional, o que se dá por uma soma maior de dissimilaridades – os dois últimos casos representados pela hipótese alternativa.

A solução causal para o TSP ser melhor do que a solução ocasional significa que o problema possui um percurso lógico que diminui as distâncias de eventos consecutivos em relação a eventos mais distantes – como sub-soluções ótimas do TSP. Intuitivamente, representa o movimento da classe pelo espaço de atributos ao longo do tempo.

Apesar do uso de uma função distância (respeitando, portanto, desigualdade triangular) ser essencial para a intuição geométrica do problema do caixeiro viajante, na prática, para os propósitos apresentados, D pode ser qualquer função de dissimilaridade.

Outras ferramentas úteis provindas da analogia são as graficações de $f_{\text{TSP}}(i) = D\left(X_i^c, X_{i-1}^c\right)$ e $f_{\text{RTSP}}(i) = D\left(X_i^c, X_{p(i)}^c\right)$, que provém evidências com relação ao tipo de mudança de conceito – abrupta, incremental ou gradual – e com relação ao formato da classe no espaço de atributos – se a quantidade de grupos e se seus formatos e densidades se alteram com o tempo – com a análise de diferentes padrões. Observamos que $f_{\text{RTSP}}(i) = D\left(X_i^c, X_{p(i)}^c\right) \neq D\left(X_{p(i)}^c, X_{p(i-1)}^c\right)$. Em outras palavras, apesar de ambas $D\left(X_i^c, X_{p(i)}^c\right)$ e $D\left(X_{p(i)}^c, X_{p(i-1)}^c\right)$ serem essencialmente distâncias entre pares de eventos aleatórios, a graficação da primeira oferece padrões mais interessantes por fixar o primeiro evento da função de distância de acordo com a sequência original da base de dados.

A Figura 10 ilustra o padrão de mudanças abruptas em um conjunto de dados artificial. Enquanto Figura 10a apresenta uma faixa, a Figura 10b apresenta duas faixas separadas. Na Figura 10a, o ponto isolado acima, após cerca de 100 eventos, representa a distância entre o último evento de um conceito e o primeiro do conceito seguinte. A diferença dos padrões das duas figuras se dá pelas altas distâncias entre eventos que possuem diferentes conceitos e se tornam consecutivos apenas após a permutação da base.

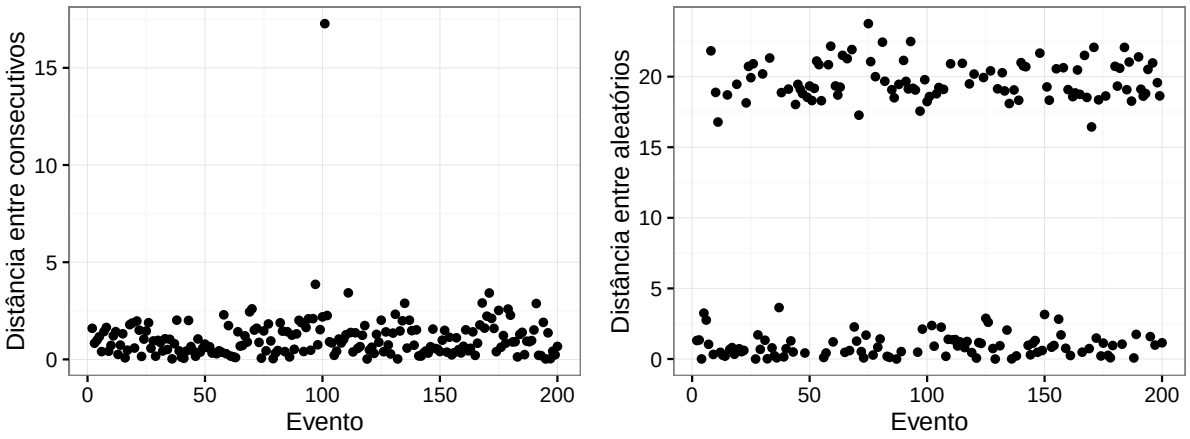
A Figura 11 apresenta o padrão de uma mudança semi-abrupta. É similar ao caso anterior, entretanto é possível observar um período de tempo no qual há uma segunda faixa, na Figura 11a. Esse é o período no qual ambos os contextos coexistem.

A Figura 12 apresenta uma mudança gradual. Em outras palavras, a probabilidade de ocorrência de um conceito cresce gradativamente, enquanto a do outro decresce. O padrão se apresenta mostra na forma de uma intensificação do caso anterior.

A Figura 13 apresenta o padrão para um grupo se movendo lentamente no espaço de atributos, *i.e.*, mudança de conceito incremental.

A Figura 14 ilustra um grupo que se torna dois e, após certo tempo, retorna a ser um.

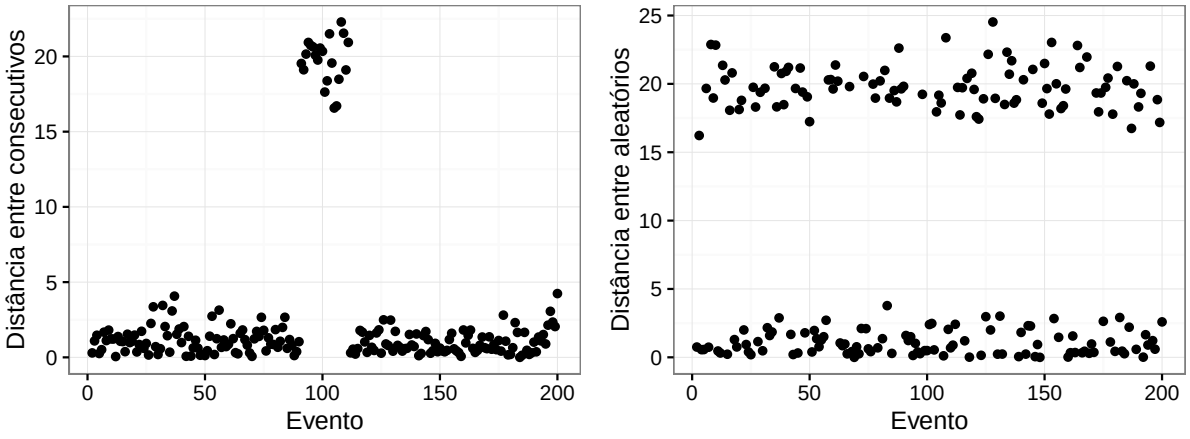
Figura 10 – Padrão de uma mudança abrupta.



(a) $f_{\mathrm{TSP}}(i)$ (b) $f_{\mathrm{RTSP}}(i)$

Fonte: Elaborada pelo autor.

Figura 11 – Padrão de uma mudança semi-abrupta.



(a) $f_{\mathrm{TSP}}(i)$ (b) $f_{\mathrm{RTSP}}(i)$

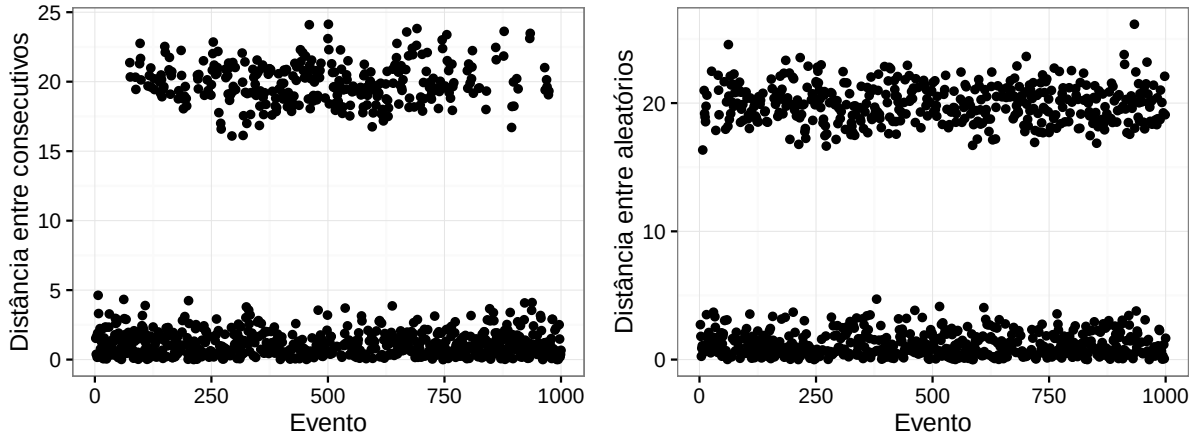
Fonte: Elaborada pelo autor.

Uma análise ingênua sugere que o número de faixas possui ligação direta com o número de grupos. Essa análise é enganosa, como visto a seguir.

A Figura 15 ilustra o caso no qual há três grupos em um espaço unidimensional, sem mudança de conceitos. As médias de cada grupo são coprimas duas a duas. O número de faixas, no gráfico, é quatro, e a altura média de cada faixa é a distância média entre exemplos de cada par de grupos.

Essa não foi uma demonstração exaustiva de padrões. Outros padrões interessantes podem ser produzidos, como mudança de volume em um grupo. Entretanto, os casos apresentados são suficientes para mostrar o potencial de análise da ferramenta apresentada.

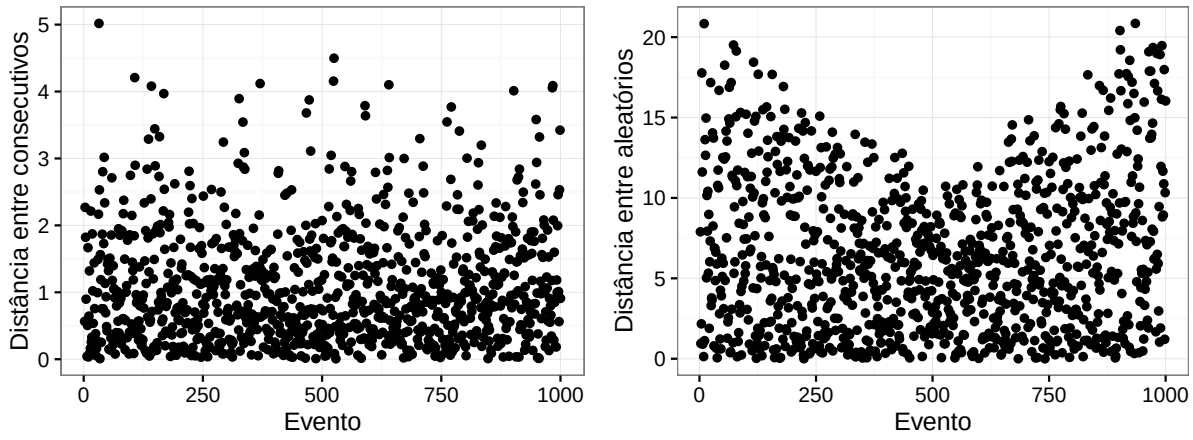
Figura 12 – Padrão de uma mudança gradual.



(a) $f_{\text{TSP}}(i)$ (b) $f_{\text{RTSP}}(i)$

Fonte: Elaborada pelo autor.

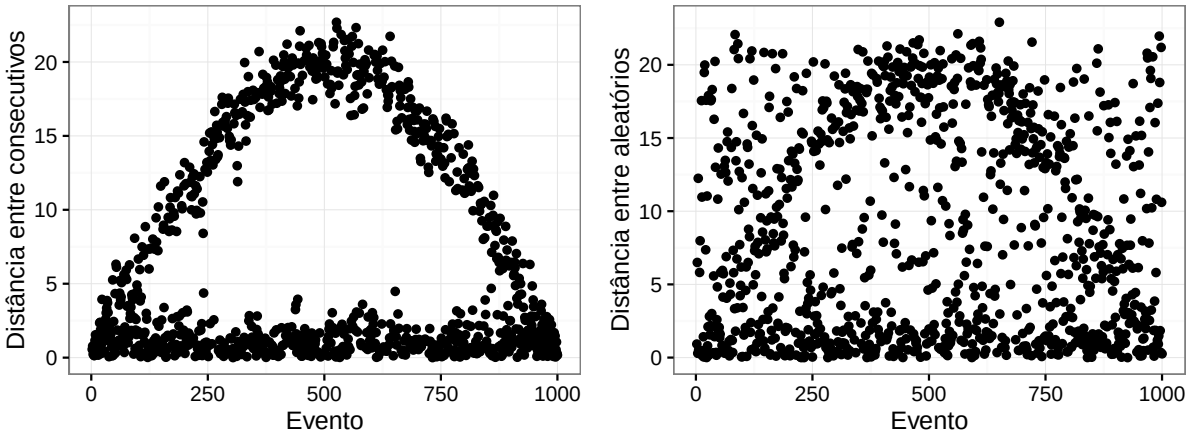
Figura 13 – Padrão de uma mudança de conceito incremental.



(a) $f_{\text{TSP}}(i)$ (b) $f_{\text{RTSP}}(i)$

Fonte: Elaborada pelo autor.

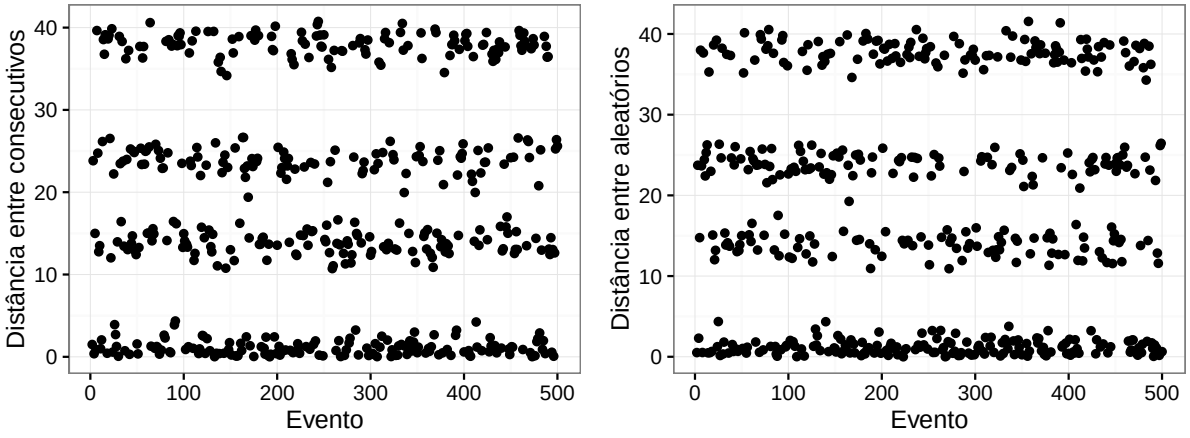
Figura 14 – Padrão de um grupo que se torna dois e retorna a ser um.



(a) $f_{\mathrm{TSP}}(i)$ (b) $f_{\mathrm{RTSP}}(i)$

Fonte: Elaborada pelo autor.

Figura 15 – Padrão para grupos em posições coprimas no espaço.



(a) $f_{\mathrm{TSP}}(i)$ (b) $f_{\mathrm{RTSP}}(i)$

Fonte: Elaborada pelo autor.

4.3 Evidências de mudanças em $P(c)$

Em *análise de tempo relativo*, é apenas possível mensurar $P(c)$ – e, consequentemente, mudanças em $P(c)$ – relativamente a outras classes. De agora em diante, $P(c)$ é definida como a probabilidade a priori de c considerando toda extensão do fluxo de dados. $P_t(c)$ é a probabilidade do t-ésimo evento possuir o rótulo c e $P_{[i,j]}(c)$ é a probabilidade da ocorrência de um evento c no intervalo de eventos $[i, j]$. É possível observar que $P(c) = P_{[1,|X|]}(c)$, onde X é a sequência que constitui o fluxo de dados. Ademais, $\sum_{c \in C} P(c) = \sum_{c \in C} P_{[i,j]}(c) = \sum_{c \in C} P(c) = 1$.

Uma possível abordagem para analisar a variação da probabilidade a priori ao longo do tempo é a graficação de $f_r(t) = \hat{P}_t(c)$ ou $f_p(t) = \hat{P}_{[1,t]}(c)$, onde $\hat{P}$ é uma estimativa para probabilidade, estimada a partir dos eventos ocorridos no fluxo de dados, como mostra a Equação 4.8, na qual z é definido segundo a Equação 4.9.

$$\hat{P}_{[i,j]}(c) = \frac{\sum_{k=i}^{j} z(y(X_i), c)}{j - i + 1} \tag{4.8}$$

$$z(a,b) = \begin{cases} 1, & \text{se } a = b \\ 0, & \text{caso contrário} \end{cases} \tag{4.9}$$

Entretanto, essa abordagem tem os mesmos problemas relacionados ao erro *prequential*: o gráfico tende a um valor médio, com o tempo, e a interpretação se torna difícil. Do mesmo modo, as mesmas soluções se aplicam, como o uso de janelas deslizantes, *i.e.*, graficação de $f_w(t) = \hat{P}_{[t,t+w]}(c)$, sendo w o tamanho da janela. Nessa situação, é possível e aconselhável o uso de intervalos de confiança. Particularmente, os limites de Hoeffding para a equação apresentada são dados segundo a Equação 4.10, na qual $\delta \in [0,1]$ é o nível de confiança.

$$\varepsilon_w^h = \sqrt{\frac{1}{2w} \ln(2/\delta)} \tag{4.10}$$

4.4 Dependência temporal

Dependência temporal é o fenômeno no qual a classe do evento atual influencia a probabilidade das classes dos eventos seguintes. Esse tema já foi amplamente discutido na comunidade (ŽLIOBAITĖ *et al.*, 2014; BIFET *et al.*, 2013b; ŽLIOBAITĖ, 2013).

A ordem de uma dependência temporal diz respeito a quantos eventos no passado são considerados para o cálculo da probabilidade da classe do evento atual. A forma mais simples de dependência temporal é a dependência temporal positiva de primeira ordem, que implica na probabilidade de repetição de uma mesma classe por vezes seguidas, durante o fluxo de dados, sendo um evento influenciado apenas de forma direta pelo evento anterior. A Equação 4.11 apresenta a probabilidade da classe de um evento, considerando dependência temporal de

primeira ordem, e exibe a probabilidade resultante no caso da ausência de dependência temporal. Observe que $P(y(X_{t-1}) = c) = P_{t-1}(c)$.

$$P\left(y(X_t) = c | y(X_{t-1}) = c\right) = \frac{P\left(y(X_t) = c, y(X_{t-1}) = c\right)}{P(y(X_{t-1}) = c)} \overset{\text{Se variáveis independentes}}{=} P\left(y(X_t) = c\right) \quad (4.11)$$

Para estimar a dependência temporal de primeira ordem, verifica-se a probabilidade de ocorrência de eventos consecutivos apresentarem uma mesma classe c. A Equação 4.12 formaliza a estimativa.

$$\hat{P}(y(X_t) = c | y(X_{t-1}) = c) = \frac{\hat{P}(y(X_t) = c, y(X_{t-1}) = c)}{\hat{P}(y(X_{t-1}) = c)} = \frac{\hat{P}(y(X_t) = c, y(X_{t-1}) = c)}{\hat{P}_{t-1}(c)} \approx \frac{\sum_{i=2}^{|X|} z(y(X_i), c) z(y(X_{i-1}), c)}{\sum_{i=1}^{|X|} z(y(X_i), c)} \quad (4.12)$$

Se $P(y(X_t) = c | y(X_{t-1}) = c) \gg P_{t-1}(c)$, há dependência temporal positiva para a classe c, *i.e.*, há maior probabilidade de ocorrência da classe c em sequências consecutivas de eventos do que seria esperado caso a ocorrência de cada evento fosse independente. Similarmente, caso $P(y(X_t) = c | y(X_{t-1}) = c) \ll P_{t-1}(c)$, há dependência temporal negativa, indicando uma menor probabilidade de sequências consecutivas de c do que seria esperado caso a ocorrência de cada evento fosse independente.

Alternativamente, pode haver maior interesse em saber da ocorrência de dependência temporal, independentemente das classes envolvidas. Neste caso, pode-se estimar $P(y(X_t) = y(X_{t-1}))$ pela proporção de elementos consecutivos que possuem a mesma classe, como formaliza a Equação 4.13.

$$\hat{P}\left(y(X_t) = y(X_{t-1})\right) = \frac{\sum_{i=2}^{|X|} z\left(y(X_i), y(X_{i-1})\right)}{|X| - 1} \quad (4.13)$$

Apesar de $\hat{P}\left(y(X_t) = y(X_{t-1})\right)$ fornecer a probabilidade de eventos consecutivos da mesma classe, a interpretação prática desse valor do ponto de vista de dependência temporal é mais difícil, uma vez que depende do número de classes e suas proporções.

Por meio da Equação 4.14, nota-se que, em *análise de tempo relativo*, mudanças na probabilidade a priori de uma classe em uma porção do fluxo é compensada por mudanças em outra porções, *i.e.*, se a $P_a(c) > P(c)$ em um instante a, há um b tal que $P_b(c) < P(c)$. Adicionalmente, como as probabilidades, em qualquer instante, somam 1, a mudança da probabilidade em uma classe obrigatoriamente implica na mudança da probabilidade em pelo menos uma outra classe.

$$\hat{P}(c) = \frac{1}{|X|}\sum_{i=1}^{|X|} z(X_i,c) = \frac{\frac{t}{t}\sum_{i=1}^{t} z(X_i,c) + \frac{|X|-t}{|X|-t}\sum_{i=t+1}^{|X|} z(X_i,c)}{|X|-t+t}$$
$$= \frac{1}{|X|}\left(t\hat{P}_{[1,t]}(c) + (|X|-t)\hat{P}_{t+1}(c)\right) = \frac{t}{|X|}\hat{P}_{[1,t]}(c) + \frac{(|X|-t)}{|X|}\hat{P}_{t+1}(c)$$
$$= \alpha\hat{P}_{[1,t]}(c) + (1-\alpha)\hat{P}_{t+1}(c) \quad (4.14)$$

Uma medida que pode ser extraída de uma base de dados é o número médio de exemplos consecutivos que pertençam a uma dada classe c, além da correspondente variância e desvio padrão. Por fim, também é possível graficar a auto correlação estimada para as classes de interesse (ŽLIOBAITĖ, 2013).

Por um número de razões, pode ser útil testar um algoritmo em uma base real sem dependência temporal, ou com uma diferente daquela já existente. As próximas duas seções apresentam técnicas para manipular artificialmente esse fator em bases já existentes.

4.4.1 Removendo dependência temporal

Remover dependência temporal pode ser feito de duas formas diferentes. Uma delas é a remoção de eventos excedentes, na qual eventos de uma classe com alta dependência temporal são removidos estrategicamente, para diminuir a probabilidade de ocorrência de eventos consecutivos da classe em questão. Essa abordagem se aplica bem a *análise de tempo de relógio*, pois preserva a ordem de todos os exemplos. Entretanto, além de alterar a probabilidade a priori das classes, a técnica pode eventualmente levar à remoção de eventos importantes para a evolução do modelo de classificação.

Outra abordagem é a reordenação dos eventos, mantendo a mesma ordem relativa dentre o que pertençam a uma mesma classe, enquanto não preservando a ordem relativa de eventos de classes diferentes.

O algoritmo consiste em retirar *bilhetes* a partir de um *balde*, sem reposição. O balde contém, inicialmente, $|X^c|$ bilhetes para cada classe $c \in C$. A cada turno, um bilhete é retirado do balde. Para a retirada do k-ésimo bilhete pertencente à uma classe c, o k-ésimo evento de classe c da sequência original é adicionado ao final do novo fluxo de dados. O processo se repete até o balde estar vazio.

Apesar de se adequar à *análise de tempo de relógio*, essa abordagem preserva todos os exemplos e as probabilidades a priori das classes, considerando toda a extensão do fluxo de dados. Observe que, como um efeito colateral, o algoritmo que segue também remove, além da dependência temporal, quaisquer variações de $P(c)$ ao longo do tempo.

4.4.2 Inserção de dependência temporal

Similarmente ao processo de remoção, é possível inserir dependência temporal pela exclusão estratégica de eventos do fluxo de dados. Essa estratégia se mostra adequada à *análise de tempo de relógio*, uma vez que não altera a ordem dos eventos. Entretanto, também é possível inserir dependência temporal reordenando os eventos, mantendo a ordem relativa dentro de uma classe, mas não assegurando a ordem relativa inter-classe. Nesta seção, são sugeridos diferentes algoritmos para esse propósito.

Inserir dependência temporal é mais complicado do que remover, uma vez que a quantidade de eventos disponíveis é limitada, de forma que certas restrições devem ser mantidas para evitar o esgotamento precoce de eventos de alguma classe.

O primeiro método é escolher, indiretamente, a quantidade média de eventos consecutivos para cada classe. Seja $\hat{P}(c)$ a probabilidade a priori estimada para a classe c, então o número médio de eventos consecutivos – ou *comprimento* – é $k\hat{P}(c)$, sendo k um parâmetro tal que $(k|C|)^{-1}\sum_{c\in C} k\hat{P}(c) = 1$. Por consequência, se obtém R tal que $R\sum_{c\in C} k\hat{P}(c) = |X|$. R é a quantidade estimada do número de vezes que sequências consecutivas de exemplos pertencentes a cada classe irão aparecer, durante o fluxo.

Escolhido k e, consequentemente, o comprimento médio de cada classe, um processo iterativo tem início. Em cada etapa, uma classe é uniformemente e aleatoriamente escolhida – excluindo a classe que foi escolhida na última iteração. Supondo que a classe escolhida seja c, então a variável aleatória $v \sim \mathcal{N}(k\hat{P}(c), \sigma_c)$ é amostrada, e os próximos $\lfloor v \rceil$ eventos pertencentes a c são adicionados ao final do novo fluxo de dados.

Alternativamente, pode-se ter interesse em escolher a probabilidade de repetição ao invés do comprimento de uma classe. Nesse caso, é possível a conversão da primeira escolha para a segunda. Uma vez que β_c, a probabilidade de repetição de c, seja escolhida, o comprimento esperado μ_c é dado por $\mu_c = \beta_c(\beta_c - 1)^{-2}$, com correspondente variância $\sigma_c^2 = (-\beta_c^4 - 2\beta_c^3 + 3\beta_c^2 - \beta_c)(\beta_c - 1)^{-5}$. A partir daí, obtém-se $k = \mu_c/\hat{P}(c)$, com o qual se deriva os comprimentos das demais classes.

4.5 Sobreposição de classes

Sobreposição de classes constitui um problema bem discutido em Aprendizado de Máquina (GARCÍA *et al.*, 2006; PRATI; BATISTA; MONARD, 2004), incluindo trabalhos que apresentam evidências que uma relação entre medidas de avaliação e o grau de sobreposição pode ser mais forte do que a relação entre as mesmas medidas de avaliação e desbalanceamento dos dados (PRATI; BATISTA; MONARD, 2004). Particularmente no contexto de fluxo de dados, caso se considere toda a extensão do fluxo como uma base de dados sem informação temporal, a não existência de sobreposição provê duas informações. A primeira delas é a de que quaisquer

mudanças de conceito existentes são detectáveis (ŽLIOBAITĖ, 2010). A segunda é o forte indício de que um mecanismo de esquecimento, amplamente pregado como essencial por trabalhos relacionados à área, não se faz necessário, uma vez que o conhecimento espacial passado de uma certa classe não sobrepõe o conhecimento espacial futuro de uma classe diferente.

Entretanto, obter evidências sobre a não existência de sobreposição é uma tarefa desafiadora. Nessa seção, é proposto um método simples que pode ser usado para esse propósito, e que foi fortemente inspirado pelos algoritmos Edição de Wilson (WILSON, 1972) e DBScan (ESTER *et al.*, 1996).

O método consiste no cálculo da média das proporções, para cada evento x, de quantos eventos, dentre os K mais próximos, possuem a classe de x. A ideia é formalizada pela Equação 4.15, na qual X é o conjunto de dados e $C_K(x)$ é a quantidade de exemplos, entre os K vizinhos mais próximos de x, que possuem a mesma classe de x.

$$O_L(X) = \frac{1}{|X|} \sum_{x \in X} \frac{C_K(x)}{K} \tag{4.15}$$

O_L varia entre 0 e 1. Quanto mais alto seu valor, maior a evidência de que a base de dados não apresenta sobreposição das classes no espaço de atributos. É possível notar que calcular essa medida é, muitas vezes, impraticável devido ao alto custo computacional. Particularmente, se a busca pelos K vizinhos mais próximos for efetuada com uma busca linear, com uma árvore *heap* para manter as distâncias dos K vizinhos mais próximos, a complexidade do cálculo da medida seria $O(|X|^2 log_2 K)$. Alternativamente, é possível utilizar uma estrutura de índice espacial, como Arvore *k-d* (BENTLEY, 1975), para reduzir a complexidade, ou simplesmente amostrar uniformemente a base de dados.

4.6 Considerações finais

Este capítulo apresentou algumas técnicas exploratórias que podem ser utilizadas para melhor compreender as características de dados reais. Nomeadamente, técnicas para evidenciar presença de mudanças de conceito em $P(x|c)$ e $P(c)$, identificar e manipular dependência temporal e, por fim, verificar a ocorrência de sobreposição das classes no espaço de atributos. As ferramentas propostas foram úteis para análises preliminares realizadas no decorrer da pesquisa.

CAPÍTULO

5

IMPACTO DA LATÊNCIA DE VERIFICAÇÃO

Latência de verificação, *delay* ou atraso é o período entre o instante em que um evento não rotulado é registrado o instante em que seu rótulo verdadeiro se torna disponível. Esse período depende intrinsecamente do domínio da aplicação. Por exemplo, em aplicações de predição de tendência do mercado de ações ou demanda elétrica, a latência de verificação é exatamente o tempo no futuro para o qual se deseja realizar uma predição. Nesse caso, a latência de verificação é fixa para todas as predições. Em outras aplicações, a latência de verificação pode ser variável, como o caso de sensores que classificam eventos em ambientes externos e para os quais se deseja pouca ou nenhuma intervenção humana (BATISTA *et al.*, 2011b).

Como apresentado na Seção 2.5, entre as inúmeras variações de cenários que relacionam latência de verificação, poucos trabalhos lidaram diretamente com a ideia de aprender e avaliar modelos com rótulos atrasados. Até onde vai o nosso conhecimento, esse é o primeiro trabalho que extensivamente avalia o impacto da introdução de latência de verificação em classificadores estado-da-arte já existentes, no contexto de fluxo de dados.

Este capítulo fornece, como contribuição à comunidade, uma ampla avaliação experimental para entender o impacto do atraso de rótulos verdadeiros em classificadores de fluxo de dados. O objetivo é empiricamente quantificar a perda de performance entre diferentes classificadores, à medida em que a latência de verificação é incrementada.

5.1 Configuração experimental

Foram avaliados 25 classificadores estado-da-arte disponíveis no ambiente MOA (BIFET *et al.*, 2010). Os classificadores foram aplicados em 7 bases de *benchmark* reais e 2 bases sintéticas, por meio do procedimento padrão *teste-então-treine*. Neste procedimento, cada evento observado é utilizado para testar o modelo atual e, uma vez que seu rótulo verdadeiro se torna disponível, é utilizado para atualizar o modelo. Como anotações de tempo de relógio não são

disponíveis em todos as bases, o atraso foi quantificado em número de instâncias que devem ser observadas após uma classificação ser emitida, antes de obter o rótulo verdadeiro referente a esta classificação. Em outras palavras, um atraso d significa que, após um evento x ser classificado, outros d eventos serão classificados para que seja obtido o rótulo verdadeiro de x.

Durante os experimentos, o atraso variou de 0 a 9 com passo 1, de 10 a 90 com passo 10, de 100 a 1.000 com passo 100 e incluímos, adicionalmente, o atraso 2.000, totalizando 30 valores de atraso. Os resultados foram aferidos por meio de acurácia, como medida de avaliação. Também foram calculadas as estatística kappa e kappa temporal(ŽLIOBAITĖ *et al.*, 2014). Entretanto, como essas duas medidas são transformações monotônicas da acurácia com respeito ao classificador de referência, elas não afetam a ordem relativa dos rankings dos métodos avaliados. Portanto, os resultados obtidos para estas medidas foram reservados ao Material Suplementar Online (REIS, 2015).

Os algoritmos empregados foram 1-NN (1 Vizinho Mais Próximo) com janelas deslizantes de tamanho 50, 200 e 1000, 10-NN com janela deslizante de tamanho 1000, 10-NN com PAW (BIFET *et al.*, 2013a) – um método para selecionar quais elementos em uma janela deslizante serão substituídos, de acordo com sua probabilidade de ser útil no processo de classificação – e janela deslizante de tamanho 1000, k-NN com PAW e ADWIN (BIFET; GAVALDÀ, 2007), Naive Bayes, Hoeffding Tree (DOMINGOS; HULTEN, 2000), Hoeffding Adaptive Tree (BIFET; GAVALDÀ, 2009), Random Hoeffding Tree, Leverage Bagging (BIFET; HOLMES; PFAHRINGER, 2010), versões temporalmente aumentadas (ŽLIOBAITĖ *et al.*, 2014) dos algoritmos Hoeffding Tree e Leverage Bagging, e todas as 10 combinações dos seguintes métodos de detecção de mudança de conceito: DDM (GAMA *et al.*, 2004), EDDM (BAENA-GARCÍA *et al.*, 2006), CUSUM (PAGE, 1954), ADWIN e Page Hinkley (PAGE, 1954) com os classificadores base Naive Bayes e Hoeffding Tree. Todos os classificadores estão disponíveis no ambiente MOA (BIFET *et al.*, 2010). Os parâmetros omitidos são configurados para seus valores padrões, no ambiente MOA.

As bases de dados de benchmark aplicadas nesse experimento são Electricity (HARRIES; WALES, 1999) (27.549 eventos, 2 classes), Cover Type (normalizada 581.012 eventos, 7 classes), Poker Hand e Sorted Poker Hand (1M eventos, 10 classes), Gas Sensor (LICHMAN, 2013) (13.910 eventos, 6 classes), Sensor (2.219.803 eventos, 54 classes) e Power Supply Array (ZHU, 2010) (29.928 eventos, 24 classes). As bases sintéticas aplicadas foram 2CHT e 2CDT (SOUZA *et al.*, 2015) (amas com 16.000 eventos, 2 classes), nas quais dois grupos circulares se movem pelo espaço de atributos. Nenhum pré-processamento foi realizado nas bases, com exceção da Electricity, da qual foram descartados os eventos com valores faltantes de atributos. Na prática, 30% dos eventos iniciais foram removidos.

O objetivo é mensurar a discrepância entre os *rankings* dos classificadores com o aumento da latência, de acordo com a acurácia. Para isso, foi calculada a correlação de *rank* de Spearman, para cada base de dados. Essa estatística quantifica a concordância entre dois *rankings*. Valores

1 e −1 indicam perfeita positiva e perfeita negativa correlações, respectivamente, enquanto 0 indica a inexistência de correlação. Foi calculada a correlação entre o *ranking* dos classificadores obtido no caso de **(a)** latência nula e os rankings obtidos para latências **(b)** 5, 10, 20, 100, 1000 e 2000. A Tabela 3 sumariza os resultados obtidos, indicando que para algumas bases de dados, há uma expressiva discordância entre os rankings, mesmo quando se insere atrasos pequenos.

Tabela 3 – Correlação de *rank* de Spearman entre *ranking* obtido com latência nula e *rankings* obtidos com latências 5, 10, 20, 100, 1000 e 2000.

	Atraso					
Base de dados	5	10	20	100	1000	2000
Electricity	0.29	0.05	-0.20	-0.29	-0.06	-0.23
Cover Type	0.58	0.30	0.18	-0.12	-0.02	-0.05
Power supply	0.56	0.55	0.52	0.50	0.09	-0.02
S. Poker Hand	0.92	0.86	0.87	0.68	0.33	0.13
Poker Hand	0.89	0.85	0.84	0.41	0.00	-0.03
Gas Sensor	0.90	0.98	0.95	0.66	0.03	0.07
Sensor	0.99	0.98	0.97	0.98	0.96	0.93
2CHT	0.99	0.99	0.99	0.98	0.94	0.49
2CDT	0.99	0.99	0.99	0.99	0.95	0.40

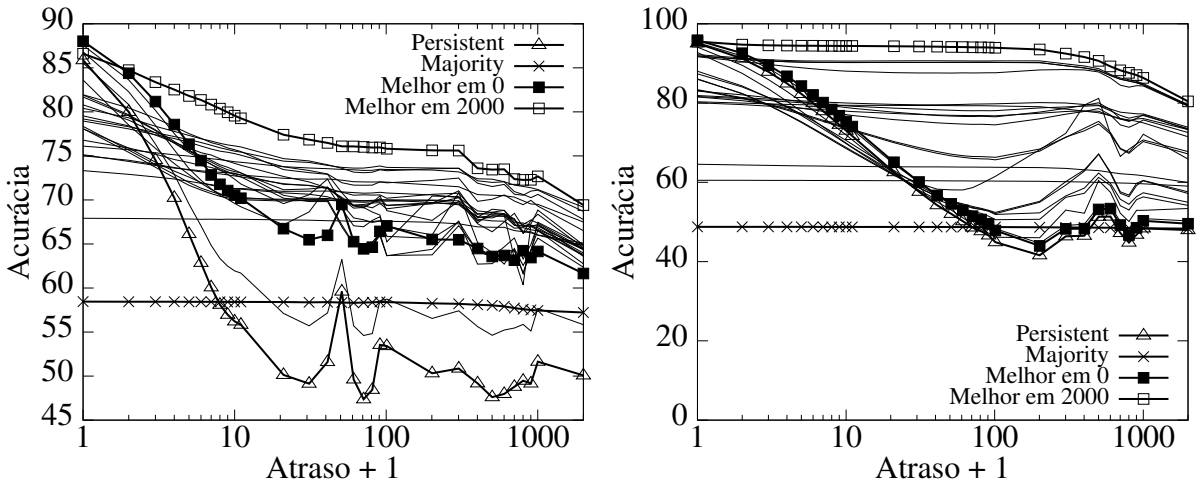
Fonte: Dados da pesquisa.

Na Figura 16, destacam-se alguns classificadores para referência nas bases Electricity e Cover Type. Essas bases foram escolhidas pois produziram os menores valores absolutos de correlações para a maioria dos atrasos. O melhor classificador em ambas as bases, quando a latência é nula, é o *Temporally Augmented Classifier* com classificador base *Leverage Bagging*. Os melhores classificadores quando atrasode2000 são *Leverage Bagging* para a base Electricity e 1-*Nearest Neighbor* com janela deslizante de tamanho 1000 para a base Cover Type. Pode-se notar que o aumento dos atrasos pode provocar severas quedas de desempenho nos classificadores de fluxo de dados avaliados. Por exemplo, os melhores classificadores apresentaram performances de 88.0% e 95.9% para atraso nulo nas bases Electricity e Cover Type, respectivamente. Quando o atraso corresponde a 10, as performances para os mesmos classificadores foram $70,2\%$ e $74,1\%$. Quando o atraso é 100, as performances foram $67,1\%$ e $47,9\%$. Em geral, todos os classificadores são afetados de alguma maneira. Por exemplo, os melhores classificadores considerando unicamente atraso de 2000 apresentam performances de $61,7\%$ e $49,6\%$, para as bases Electricity e Cover Type, respectivamente.

Apesar do impacto do atraso sobre a performance dos classificadores ser claro, é desejável confirmar estatisticamente. Como o experimental envolve mudanças ao longo do tempo, foi utilizado um teste estatístico para estudos longitudinais, com um *design* F1-LD-F1 (NOGUCHI *et al.*, 2012).

No *design* F1-LD-F1, os algoritmos são grupos de tratamento, atraso representa o tempo e o par base, algoritmo é o sujeito de análise. Usando a estatística não paramétrica Wald-Type

Figura 16 – Acurácia relativa à latência de verificação para bases Electricity e Cover Type. Os melhores classificador para os atrasos 0 e 2000, assim como os baselines *Majority classifier* e *Persistent classifier* estão em destaque.



(a) Electricity (b) Cover Type

Fonte: Dados da pesquisa.

Statistic (WTS), rejeita-se com 99% de confiança a hipótese nula dos classificadores sendo igualmente afetados pela introdução de diferentes valores de atraso.

5.2 Considerações finais

Diferentes configurações de atraso na disponibilidade dos rótulos verdadeiros implicam em possíveis diferentes resultados relativos, em uma comparação de classificadores. Deste modo, a necessária a especificação das configurações utilizadas em toda e qualquer análise comparativa. Mais importante, as análises realizadas devem ser compatíveis com o cenário real no qual se deseja aplicar o modelo de classificação.

A premissa de latência nula, presente em considerável parte da literatura, não é razoável para medir o desempenho médio relativo de diferentes algoritmos para uso na prática. O algoritmo de melhor desempenho médio relativo, na a ausência de latência, pode ter desempenho inferior a outros algoritmos, em uma situação mais fiel à realidade em termos de latência de verificação.

CAPÍTULO

6

TESTE INCREMENTAL DE KOLMOGOROV-SMIRNOV

O número de pesquisas na área de fluxo de dados tem crescido rapidamente nas últimas décadas. Dois aspectos importantes distinguem aprendizado em fluxo de dados de aprendizado em lote: eventos são gerados ao longo do tempo e a distribuição dos dados pode ser não estacionária, levando ao fenômeno da mudança de conceito. Consequentemente, a maior parte dos trabalhos de classificação em fluxo de dados se concentra na proposição de modelos eficientes que possam se adaptar às mudanças de conceito e manter uma performance estável ao longo do tempo.

Entretanto, especificamente para a tarefa de classificação, a maior parte dos métodos propostos se apoia na premissa de que, eventualmente, todos os rótulos verdadeiros dos eventos já classificados estarão disponíveis. Essa é uma premissa muito forte, raramente factível em aplicações práticas. Logo, há a clara necessidade de métodos eficientes que sejam capazes de detectar mudanças de conceito de forma não-supervisionada. Uma possibilidade é o emprego do teste de hipótese não-paramétrico Kolmogorov-Smirnov (KS), utilizado para verificar se duas amostras seguem uma mesma distribuição.

Este capítulo introduz um novo algoritmo, Incremental Kolmogorov-Smirnov (IKS) (REIS *et al.*, 2016), capaz de executar o teste (KS) em $O(1)$ para um par de amostras que se altera com o tempo. As alterações possíveis são a inclusão de uma nova observação e a remoção de uma observação existente, de qualquer uma das amostras. Ambas operações são executadas em $O(\log N)$, sendo N o número total de observações consideradas. O algoritmo representa uma significante melhoria se comparado ao uso de uma implementação padrão do KS, que possui custo $O(N \log N)$.

A próxima seção deste capítulo introduz o teste de KS e como ele é calculado.

6.1 Kolmogorov-Smirnov

A e B são duas amostras contendo observações unidimensionais. É desejado saber, a um nível de significância α, se é possível rejeitar a hipótese nula de que as observações de A e de B foram originadas por uma mesma distribuição de probabilidade.

Se nenhuma informação é fornecida referente à distribuição dos dados, mas é seguro assumir que as observações são I.I.D., pode-se utilizar o teste não paramétrico, baseado em *ranking*, Kolmogorov-Smirnov (KS) para verificar a hipótese proposta. De acordo com o teste, rejeita-se a hipótese nula ao nível de significância α se a seguinte inequação for satisfeita:

$$D \overset{?}{>} c(\alpha)\sqrt{\frac{n+m}{nm}}$$

na qual o valor de $c(\alpha)$ pode ser resgatado de uma tabela conhecida, n é o número de observações em A e m é o número de observações em B. O lado direito da inequação é o valor *p objetivo*. D é a estatística Kolmogorov-Smirnov, *i.e.*, o valor *p obtido*, e é definido como segue:

$$D = \sup_x |F_A(x) - F_B(x)|$$

onde

$$F_C(x) = \frac{1}{|C|}\sum_{c \in C, c \leq x} 1$$

Observa-se que D pode, por vias práticas, ser computado da maneira que segue:

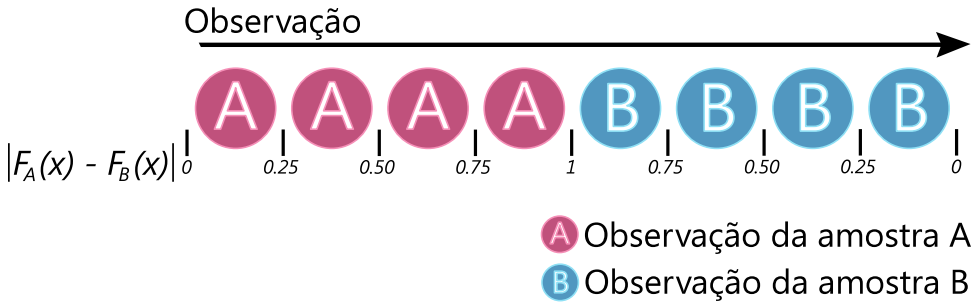
$$D = \max_{x \in A \cup B} |F_A(x) - F_B(x)|$$

A intuição por trás do teste de Kolmogorov-Smirnov é a que segue: observações aleatórias de uma mesma distribuição tendem a ficar *misturadas*, enquanto que observações de amostras diferentes se agrupam. As Figuras 17 e 18 ilustram duas situações. A primeira delas apresenta o caso no qual as amostras A e B têm alta probabilidade de possuírem distribuições diferentes. A segunda apresenta o caso no qual ambas as amostras provavelmente seguem a mesma distribuição.

A variante incremental do algoritmo assume que A e B podem se alterar com o tempo. É assumido um tipo abstrato de dados (TAD) com as seguintes operações:

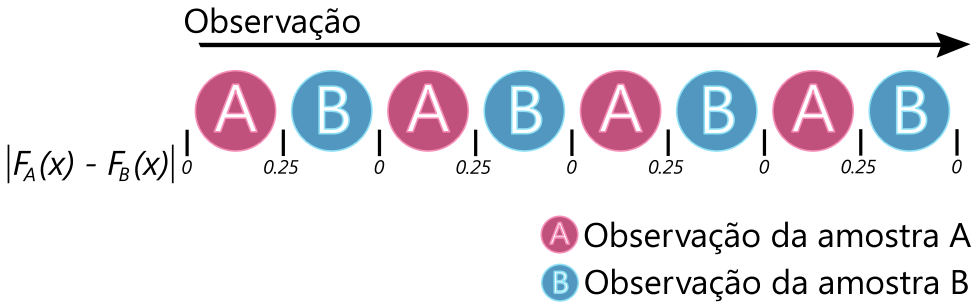
- Inserir nova observação em A;
- Inserir nova observação em B;
- Remover observação de A;

Figura 17 – Ilustração do teste de Kolmogorov-Smirnov para amostras de distribuições diferentes.



Fonte: Elaborada pelo autor.

Figura 18 – Ilustração do teste de Kolmogorov-Smirnov para amostras de mesma distribuição.



Fonte: Elaborada pelo autor.

- Remover observação de B;
- Aplicar $KS(A,B)$.

Na próxima seção, é introduzido um algoritmo que possibilita o cálculo das quatro primeira operações em tempo logarítmico de acordo com o número total de observações, com alta probabilidade, e a última operação em tempo constante.

6.2 Variante incremental

Esta seção introduz um algoritmo para rápida computação das operações da versão incremental do teste KS. Antes disso, se faz necessária a clarificação de dois pontos referentes à abordagem:

1. Na abordagem proposta, há uma distinção entre $|A|$ e n e, equivalentemente, $|B|$ e m. $|A|$ e $|B|$ são os números de observações em A e B, respectivamente, que foram inseridas em uma estrutura de dados, enquanto n e m correspondem ao real número de observações. Em princípio, $|A|$ e n e $|B|$ e m devem ser iguais. Entretanto, pode-se manipular o número de observações inseridas na estrutura de dados para superar uma limitação da implementação, explicada a seguir;

2. A proposta se limita à computação de D quando $|A| = r|B|$, sendo $r \in \mathbb{R}$ um parâmetro que permanece inalterado ao longo de todo o fluxo de dados. A princípio, essa parece ser uma limitação muito restritiva. Porém, observa-se que um caso comum do uso do KS incremental em um fluxo de dados é a comparação de duas janelas deslizantes de tamanhos fixos e iguais, tal qual $r = 1$. Pode-se ainda lidar com outros casos com a replicação de elementos inseridos na estrutura de dados, como melhor discutido na Seção 6.5.

Com a restrição de que $|A| = r|B|$, D pode ser reescrito como

$$D = \frac{1}{|A|} \max_{x \in A \cup B} |F'_A(x) - F'_B(x)|$$

onde F'_A é definido como uma soma de 1's

$$F'_A(x) = \sum_{a \in A, c \leq x} 1$$

e F'_B é definido como uma soma de r's

$$F'_B(x) = \sum_{b \in B, c \leq x} r$$

Além disso, com $G(x) = F'_A(x) - F'_B(x)$, tem-se

$$D = \frac{1}{|A|} \max \left\{ \left(\max_{x \in A \cup B} G(x) \right), - \left(\min_{x \in A \cup B} G(x) \right) \right\} \tag{6.1}$$

Considera-se a existência de uma lista na qual estão inseridas todas as observações $o_i \in A \cup B$, ordenadas de maneira que $o_i \leq o_{i+1}$ e, para cada observação, há também um valor correspondente $g_i = G(o_i)$. A Tabela 4 ilustra a estrutura de dados.

Tabela 4 – $G(x)$ para cada observação $x \in A \cup B$. A tabela representa uma lista ordenada tal que $o_i \leq o_{i+1}$.

Índice	1	2	...	$\|A\|+\|B\|-1$	$\|A\|+\|B\|$
o_i	o_1	o_2	...	$o_{\|A\|+\|B\|-1}$	$o_{\|A\|+\|B\|}$
$G(o_i)$	g_1	g_2	...	$g_{\|A\|+\|B\|-1}$	$g_{\|A\|+\|B\|}$

Fonte: Elaborada pelo autor.

Ao inserir uma nova observação o_j na estrutura, é obedecida a restrição $o_{j-1} < o_j \leq o_{j+1}$. Em outras palavras, todas as observações mais antigas e que possuem o mesmo valor ou valores mais altos são mantidas ao lado direito da nova observação e, consequentemente, possuem maiores índices. Como resultado, $g_j = g_{j-1} + v$, sendo $v = 1$ se $o_j \in A$ ou $v = -r$ se $o_j \in B$. Adicionalmente, após a inserção, todos g_i tais que $i < j$ se mantém inalterados, enquanto todos g_i tais que $i > j$ são acrescidos de v. Similarmente, a remoção da observação descresce todos g_i tais que $i > j$ em v.

Com o auxílio de uma estrutura de dados chamada *Treap* (ou *Árvore Cartesiana*) (ARAGON; SEIDEL, 1989; SEIDEL; ARAGON, 1996) com *propagação preguiçosa*, é possível inserir/remover observações na/da estrutura, e adicionar um valor constante a todos os elementos g_i tais que $i > j$ em $O(\log|A| + |B|)$, com alta probabilidade. Também é possível computar os valores $\min_i\{g_i\}$ e $\max_i\{g_i\}$ em $O(1)$. Portanto, é possível calcular D de forma exata sempre que $|A| = r|B|$, em $O(1)$, e incrementalmente modificar $|A|$ e $|B|$, com complexidade esperada de $O(\log|A| + |B|)$.

A próxima seção se dedica a explicar a Treap com propagação preguiçosa. Em seguida, a solução final na forma de algoritmo é apresentada.

6.3 Treap com propagação preguiçosa

Uma árvore cartesiana é definida como uma árvore binária que segue as seguintes propriedades:

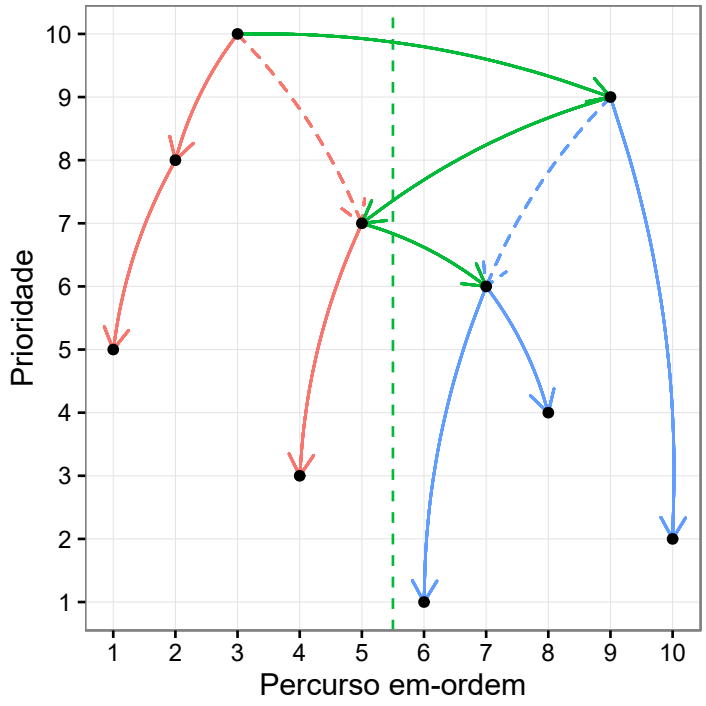
1. Possui a propriedade de uma ordenação *heap*, *i.e.*, um nó não-folha possui maior prioridade que ambos seus filhos;
2. É construída sobre uma sequência de valores de forma que uma travessia em-ordem na árvore resulta na sequência original. Em outras palavras, os elementos na sub-árvore esquerda de um nó são os valores que aparecem antes, na sequência original, que o valor do nó. Os valores da sub-árvore direita, por sua vez, aparecem depois, na sequência original.

Define-se uma operação de junção (*merge*) entre duas árvores cartesianas já existentes. Uma delas chamada de árvore *esquerda* e a outra chamada de árvore *direita*. Após a junção, as seguintes propriedades devem ser mantidas: *(a)* a árvore resultante ainda mantém uma ordenação *heap*, e *(b)* um percurso em-ordem na árvore resultante deve equivaler a um percurso em-ordem na árvore esquerda, seguido de um percurso em-ordem na árvore direita. Tal junção pode ser utilizada para construir uma árvore cartesiana a partir de árvores triviais, com apenas um nó cada e, por consequência, construir uma árvore a partir de uma sequência de valores.

A operação de junção é obtida pela aplicação da definição, recursivamente. Seja L a árvore esquerda e R a árvore direita. Se a raiz de L possuir maior prioridade que a raiz de R, então a raiz de L é a raiz Z da árvore resultante. A sub-árvore esquerda de Z é a sub-árvore esquerda de L, de forma a preservar o percurso em-ordem à esquerda e a ordenação *heap*. A sub-árvore direita de Z é o fruto da operação recursiva de junção entre a sub-árvore direta de L – na posição de árvore esquerda – e R – na posição de árvore direita. Caso a prioridade da raiz de R fosse maior do que a prioridade da raiz de L, a solução seria análoga. O processo é sumarizado pelo Algoritmo 1 e ilustrado pela Figura 19. As setas correspondem relações pai–filho. As setas vermelhas pertencem originalmente à árvore esquerda. As setas azuis pertencem originalmente à

árvore direita. Setas verdes são criadas pelo processo de junção, enquanto setas tracejadas são removidas.

Figura 19 – Operação de junção entre duas árvores cartesianas.



Fonte: Elaborada pelo autor.

Algoritmo 1: Junção de duas árvores cartesianas.

```
1  Function Junção(NIL, NIL)
3      return NIL
1  Function Junção(ArvoreEsquerda, NIL)
3      return ArvoreEsquerda
1  Function Junção(NIL, ArvoreDireita)
3      return ArvoreDireita
1  Function Junção(ArvoreEsquerda, ArvoreDireita)
2      if ArvoreEsquerda.prioridade > ArvoreDireita.prioridade then
4          ArvoreEsquerda.SubarvoreDireita ← Junção(ArvoreEsquerda.SubarvoreDireita,
           ArvoreDireita)
6          return ArvoreEsquerda
7      else
9          ArvoreDireita.SubarvoreEsquerda ← Junção(ArvoreEsquerda,
           ArvoreDireita.SubarvoreEsquerda)
11         return ArvoreDireita
12     end
```

Treap é um caso especial de árvore cartesiana na qual cada nó possui, além de um valor de prioridade *heap*, uma chave adicional. Da perspectiva da prioridade, a árvore é uma

heap. Da perspectiva da chave adicional (chave BST), é uma árvore binária de busca (BST). É possível construir uma árvore que respeite ambos os critérios para qualquer conjunto de pares (prioridade, chave BST). O nó com maior prioridade é a raiz da árvore. A subárvore esquerda é composta apenas pelos nós com chaves BST menores que a chave da raiz, enquanto a subárvore direita é composta por nós com chaves BST maiores que a chave da raiz. Os formatos das subárvores esquerda e direita, incluindo suas raízes, respeitam os mesmo critérios, recursivamente.

Treaps possuem uma importante propriedade. Se as prioridades forem números aleatórios, o formato da árvore é uma variável aleatória com a mesma distribuição de probabilidade de uma árvore binária de busca aleatória; particularmente, sua altura é proporcional ao logaritmo do número de nós, com alta probabilidade.

A Treap é facilmente obtida por construção, por meio da função de junção definida anteriormente. Em uma operação de junção, se as árvores esquerda e direita foram ambas Treaps e todas as chaves da árvore esquerda forem inferiores às chaves da direita, então a árvore resultante da junção é uma Treap.

Inserir um novo par (prioridade, chave BST) a uma Treap já existente requer uma operação de separação. O Algoritmo 2 descreve como essa operação pode ser realizada. A entrada é uma Treap e uma chave BST. A saída é um par de Treaps. A primeira (árvore esquerda) contém todos os elementos da Treap original que possuem chaves inferiores à fornecida. A segunda árvore (árvore direita) contém os elementos restantes.

Algoritmo 2: Separação de duas árvores cartesianas.

```
1  Function Separação(NIL, ?)
3      return NIL, NIL
1  Function Separação(Arvore, ChaveBST)
2      if Key ≤ Tree.BSTKey then
4          Esquerda, Arvore.SubarvoreDireita ← Separação(Arvore.SubarvoreEsquerda,
           ChaveBST)
6          Direita ← Arvore
7      else
9          Arvore.SubarvoreDireita, Direita ← Separação(Arvore.SubarvoreDireita,
           ChaveBST)
11         Direita ← Arvore
12     end
14     return Esquerda, Direita
```

Por fim, com as operações de junção e separação, é possível inserir novos elementos em uma Treap já existente, separando-a e rejuntando-a adequadamente. Algoritmo 3 transcreve o processo.

A complexidade de tempo para ambas as operações de junção e separação dependem linearmente da profundidade da árvore. Para uma lista de pares de prioridades únicas e chaves

Algoritmo 3: Inserção de um elemento em uma Treap.

```
1 Function Inserção(Arvore, Prioridade, ChaveBST)
3     ArvoreTrivial ← NovoNó(Prioridade, ChaveBST)
5     Esquerda, Direita ← Separação(Arvore, ChaveBST)
7     return Junção(Junção(Esquerda, ArvoreTrivial), Direita)
```

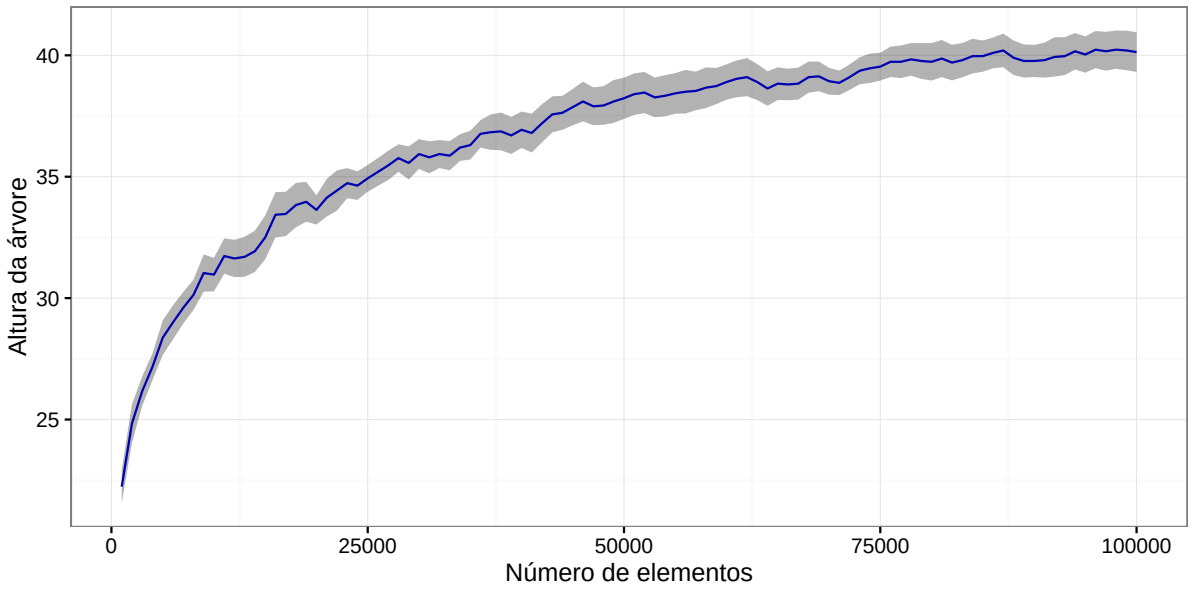
BST únicas, há apenas uma Treap possível, pois a travessia em-ordem é única pelas chaves BST e o balanço – *i.e.* as relações pai–filho – é único pela ordem das prioridades. Em outras palavras, a Treap resultante após a inserção de todos os elementos da lista, em qualquer ordem, é sempre a mesma. A distribuição de probabilidade das prioridades não é relevante, provido que sejam únicas, pois a Treap utiliza apenas sua ordem relativa. Especificamente, a ordem das prioridades é análoga a ordem de inserção dos elementos de uma árvore binária de busca aleatória, construída de uma só vez. A Figura 19 ilustra este fato ao exibir uma árvore cartesiana em um plano cartesiano, no qual as prioridades correspondem ao eixo-y e as chaves BST correspondem ao eixo-x. Cada nó é pai de ambos nó mais alto à esquerda e nó mais alto à direita. Se forem inseridas todas as chaves BST, de cima para baixo, em uma árvore binária de busca não balanceada, utilizando apenas a inserção tradicional, obtém-se exatamente a mesma árvore. Portanto, se as prioridades forem escolhidas aleatoriamente, a Treap herda uma propriedade comum às árvores binárias de busca aleatórias: a altura da árvore cresce logaritmicamente com alta probabilidade (DEVROYE, 1986).

Portanto, junção e separação e, consequentemente, inserção e remoção de elementos são operações $O(\log N)$ com alta probabilidade, sendo N o número de elementos. A Figura 20 exibe empiricamente o crescimento logaritmico da Treap, na qual a curva é uma média das alturas de 30 árvores cartesianas e a sombra indica um intervalo de confiança de 95%.

Graças a inexistência de rotação, pois prioridades não se alteram com o tempo, se, em um dado momento, um nó é predecessor de outro, a situação oposta nunca ocorrerá. Essa propriedade pode ser explorada para que seja possível inserir informações sumarizadoras acerca de uma subárvore em sua raiz, e manter essa informação atualizada, sem alterar a complexidade das operações apresentadas até agora. Como exemplo, caso cada nó possua um *valor* arbitrário, além da prioridade e da chave BST. O Algoritmo 4 altera as operações apresentadas, de forma que se torna possível computar, eficientemente, mínimo e máximo desse valor, para uma subárvore completa.

Similarmente, é possível efetuar operações em massa que modificam todos os valores da árvore, eficientemente, se essas operações forem *compatíveis*. Uma operação é dita *compatível* se, para modificar todos os valores de uma árvore, necessita apenas alterar o sumário de sua raiz, sem acessar suas subárvores. O procedimento de propagar uma operação dos sumários da raiz para os sumários das raízes de subárvores, estritamente quando necessário, é chamado *propagação preguiçosa*. Precisa ser efetuado com a mesma complexidade da aplicação da operação em massa,

Figura 20 – Crescimento da altura de uma treap de acordo com a inserção de elementos.



Fonte: Dados da pesquisa.

usualmente $O(1)$, modificando apenas os sumários das raízes das subárvores esquerda e direita. O Algoritmo 5 apresenta modificações nas operações até então apresentadas, para tornar possível a adição de uma nova operação, *Incremento*, que soma uma constante ao valor de todos os nós da árvore. A complexidade da nova operação é $O(1)$ e as modificações realizadas não alteram a complexidade das demais operações.

Uma nova versão da operação de separação é necessária para remover um elemento da Treap, dada uma chave BST correspondente. O Algoritmo 6 apresenta três novas operações: *SepMenor*, que separa a árvore em duas, sendo a esquerda uma árvore trivial com o nó de menor chave BST, da árvore original, e a direita uma árvore contendo os nós remanescentes; *SepMaior*, operação análoga a anterior, mas que separa o nó de maior chave BST; e *Remoção*, que remove um nó que possua a chave BST fornecida.

6.4 Teste IKS com Treaps

Considere a estrutura apresentada pela Tabela 4 implementada na forma de uma Treap com propagação preguiçosa: as chaves BST são as observações o_i, enquanto que os os valores de cada nó correspondem aos valores g_i. A solução final segue da Equação 6.1. O Algoritmo 7 descreve a implementação das cinco operações necessárias para o teste Kolmogorov-Smirnov Incremental: inserção de observação em A, inserção de observação em B, remoção de observação em A, remoção de observação em B e aplicação do teste KS. A última operação assume $|A| = r|B|$.

Algoritmo 4: Modificações necessárias à Treap para armazenar informações sumarizadoras.

```
1  Function Atualização(Arvore)
3      Arvore.Maximo ←
       max{Arvore.Valor, Arvore.SubarvoreEsquerda.Valor, Arvore.SubarvoreDireita.Valor}
5      Arvore.Minimo ←
       min{Arvore.Valor, Arvore.SubarvoreEsquerda.Valor, Arvore.SubarvoreDireita.Valor}
1  Function Junção(ArvoreEsquerda, ArvoreDireita)
2      if ArvoreEsquerda.Prioridade > ArvoreDireita.Prioridade then
4          ArvoreEsquerda.SubarvoreDireita ← Junção(ArvoreEsquerda.SubarvoreDireita,
           ArvoreDireita)
6          Arvore ← ArvoreEsquerda
7      else
9          ArvoreDireita.SubarvoreEsquerda ← Junção(ArvoreEsquerda,
           ArvoreDireita.SubarvoreEsquerda)
11         Arvore ← ArvoreDireita
12     end
14     Atualização(Arvore)
16     return Arvore
1  Function Separação(Arvore, ChaveBST)
2      if ChaveBST ≤ Arvore.ChaveBST then
4          Esquerda, Arvore.SubarvoreDireita ← Separação(Arvore.SubarvoreEsquerda,
           ChaveBST)
6          Direita ← Arvore
7      else
9          Arvore.SubarvoreDireita, Direita ← Separação(Arvore.SubarvoreDireita,
           ChaveBST)
11         Direita ← Arvore
12     end
14     Atualização(Esquerda)
16     Atualização(Direita)
18     return Esquerda, Direita
```

6.5 Limitações e soluções alternativas

A única limitação no Kolmogorov-Smirnov Incremental, em relação à versão original, é a de que só pode ser executado quando $|A| = r|B|$, sendo r um parâmetro que se mantém constante ao longo do fluxo de dados. Apesar dessa limitação parecer restritiva, na prática isso não o é.

Se ambas as amostras possuírem tamanho fixo e as mudanças forem apenas substituição de observações – pares de inserções/remoções –, a condição restritiva é validada, não importando o tamanho das amostras. Adicionalmente, se ambas as amostras sempre possuírem o mesmo tamanho, $r = 1$ e a condição é validada, mesmo que o tamanho das amostras se altere com o tempo.

Para os demais casos, é sugerido que r seja escolhido tal que $r = 1$ e, caso haja disparidade

entre $|A|$ e $|B|$, haja reamostragem. Nesses casos, m e n – as quantidades reais de observações únicas em A e B, respectivamente – são utilizados para computar o *valor-p objetivo*, enquanto $|A|$ e $|B|$ são utilizados para computar o *valor-p obtido*.

Observa-se que, por exemplo, se $m = 2n$, então todas as observações de A são inseridas duas vezes, de forma que $|A| = |B|$. Ambos *valor-p objetivo* e *valor-p obtido* serão idênticos aos obtidos pela versão padrão do Kolmogorov-Smirnov. De fato, se A possui um tamanho fixo enquanto B cresce indefinidamente, é possível manter equivalência exata entre o KS Incremental e o KS padrão sempre que $m = kn, k \in \mathbb{N}$. Diferenças entre KS Incremental e KS padrão entre valores consecutivos de k também se tornam incrementalmente menores.

6.6 Performance

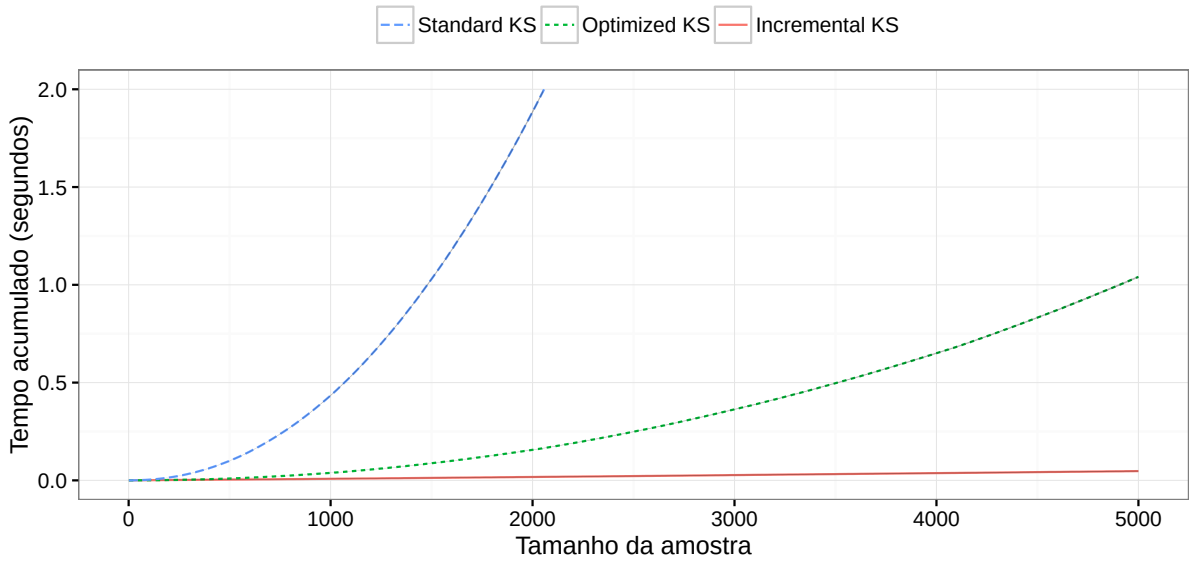
A performance do IKS foi avaliada em dois cenários distintos, comparando três versões do Kolmogorv Smirnov: *Incremental Kolmogorov-Smirnov*, ou IKS, a versão proposta neste capítulo; *Standard Kolmogorov-Smirnov*, a versão padrão e sem otimizações; *Optimized Kolmogorov-Smirnov*, uma implementação da versão padrão que sofreu melhoramentos para o cenário de fluxos de dados, executando o teste em tempo linear entre uma modificação e outra das amostras Os melhoramentos consistem na inserção e remoção lineares de observações em vetores já ordenados para as amostras, e na computação linear de F_A e F_B. Todos os testes foram executados em um i7 4790k @3.6Ghz, 16GB DDR3 @1600Mhz.

No primeiro cenário de avaliação, foram consideradas duas amostras sempre crescentes. Inicialmente, possuem 1 observação cada e, a cada instante, incrementam seus tamanhos em 1 observação. O experimento foi conduzido até a inclusão de 5.000 observações. Os resultados reportados são uma média de 30 repetições do experimento, e as observações inseridas são sempre números aleatórios.

Figura 21 apresenta o tempo acumulado pelo tamanho das amostras. Apesar dos intervalos de confiança terem sido graficados, sua espessura os tornou imperceptíveis. Por exemplo, para uma observação que cresce até 5.000 observações, Incremental KS levou, em média, $0,0491$ segundos para completar o experimento. Optimized KS levou $1,0297$ segundos, nas mesmas condições, enquanto o Standard KS levou $12,7645$ segundos.

No segundo cenário de avaliação, mais realista e significativo para a proposta desta pesquisa, foi analisado o desempenho dos algoritmos para percorrer um fluxo com janelas deslizantes de tamanho fixo. Figura 22 apresenta o tempo necessário para escanear todo o fluxo pelo tamanho da janela deslizante que foi utilizada. Novamente, intervalos de confiança foram graficados mas se tornaram imperceptíveis. Por exemplo, para uma janela deslizante contendo 1.000 observações, IKS levou em média, para 30 execuções do experimento, $0,0410$ segundos. Optimized KS levou $0,9605$ segundos, e Standard KS levou $5,326$ segundos. Novamente, IKS foi uma ordem de magnitude mais rápido do que OKS e duas ordens de magnitude mais rápido

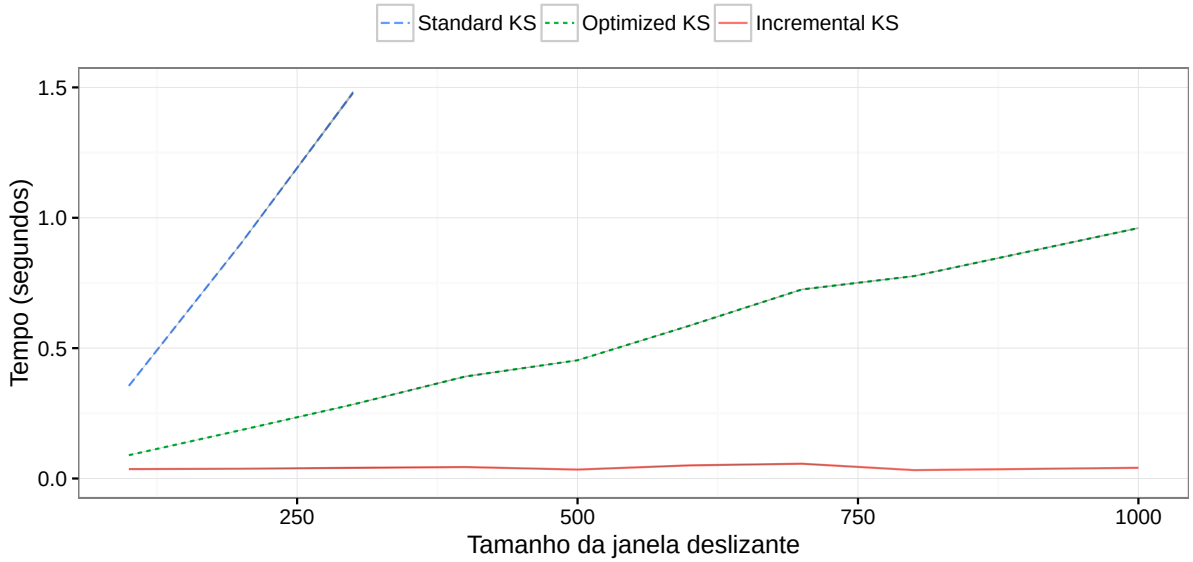
Figura 21 – Tempo acumulado para amostra sempre crescente.



Fonte: Dados da pesquisa.

do que o Standard KS.

Figura 22 – Tempo acumulado para percorrer um fluxo de dados com diferentes tamanhos de janela deslizante.



Fonte: Dados da pesquisa.

6.7 Considerações finais

Os resultados obtidos revelam a superioridade do Incremental Kolmogorov-Smirnov para os cenários propostos, quando comparado a versão padrão do algoritmo. Apesar da sofisticada

implementação, IKS calcula a estatística Kolmogorov-Smirnov de forma exata perante uma restrição razoável – $|A| = r|B|$, sendo $r \in \mathbb{R}$ um parâmetro. Ainda que a condição não possa ser satisfeita, soluções alternativas de uso foram apresentadas.

A existência de um método eficiente e não paramétrico para verificar se duas amostras possuem a mesma distribuição, em cenários de fluxo de dados, abre margem para algoritmos novos e inovativos. Como demonstrativo dessa afirmação, os próximos capítulos apresentam tarefas específicas de aprendizado de máquina, nas quais o algoritmo proposto foi utilizado.

Uma implementação completa do IKS, e consequentemente de árvore Treap com propagação preguiçosa, está disponível como material suplementar (REIS, 2016).

Algoritmo 5: Modificações necessárias à Treap para propagação preguiçosa.

```
1  Function Incremento(Arvore, Constante)
3      Arvore.Valor ← Arvore.Valor + Constante
5      Arvore.Maximo ← Arvore.Maximo + Constante
7      Arvore.Minimo ← Arvore.Minimo + Constante
9      Arvore.Preguica ← Arvore.Preguica + Constante
1  Function Espreguiço(Arvore)
3      Incremento(Arvore.SubarvoreEsquerda, Arvore.Preguica)
5      Incremento(Arvore.SubarvoreDireita, Arvore.Preguica)
7      Arvore.Preguica ← 0
1  Function Atualização(Arvore)
3      Espreguiço(Arvore)
5      Arvore.Maximo ←
       max{Arvore.Valor, Arvore.SubarvoreEsquerda.Valor, Arvore.SubarvoreDireita.Valor}
7      Arvore.Minimo ←
       min{Arvore.Valor, Arvore.SubarvoreEsquerda.Valor, Arvore.SubarvoreDireita.Valor}
1  Function Junção(ArvoreEsquerda, ArvoreDireita)
2      if ArvoreEsquerda.Prioridade > ArvoreDireita.Prioridade then
4          Espreguiço(ArvoreEsquerda)
6          ArvoreEsquerda.SubarvoreDireita ← Junção(ArvoreEsquerda.SubarvoreDireita,
           ArvoreDireita)
8          Arvore ← ArvoreEsquerda
9      else
11         Espreguiço(ArvoreDireita)
13         ArvoreDireita.SubarvoreEsquerda ← Junção(ArvoreEsquerda,
           ArvoreDireita.SubarvoreEsquerda)
15         Arvore ← ArvoreDireita
16     end
18     Atualização(Arvore)
20     return Arvore
1  Function Separação(Arvore, ChaveBST)
3      Espreguiço(Arvore)
4      if ChaveBST ≤ Arvore.ChaveBST then
6          Esquerda, Arvore.SubarvoreDireita ← Separação(Arvore.SubarvoreEsquerda,
           ChaveBST)
8          Direita ← Arvore
9      else
11         Arvore.SubarvoreDireita, Direita ← Separação(Arvore.SubarvoreDireita,
           ChaveBST)
13         Direita ← Arvore
14     end
16     Atualização(Esquerda)
18     Atualização(Direita)
20     return Esquerda, Direita
```

Algoritmo 6: Modificações necessárias à Treap para propagação preguiçosa.

```
1  Function SepMenor(Arvore)
2      if Arvore é NIL then
4          return NIL, NIL
5      end
7      Espreguiço(Arvore)
8      if Arvore.SubarvoreEsquerda não é NIL then
10         Esquerda, Arvore.SubarvoreEsquerda ← SepMenor(Arvore.SubarvoreEsquerda)
12         Direita ← Arvore
13     else
15         Direita ← Arvore.SubarvoreDireita
17         Arvore.SubarvoreDireita ← NIL
19         Esquerda ← Arvore
20     end
22     Atualização(Esquerda)
24     Atualização(Direita)
26     return Esquerda, Direita
1  Function SepMaior(Arvore)
2      if Arvore é NIL then
3          return NIL, NIL
4      end
6      Espreguiço(Arvore) if Arvore.SubarvoreDireita não é NIL then
8          Arvore.SubarvoreDireita, Direita ← SepMaior(Arvore.SubarvoreEsquerda)
9      else
11         Esquerda ← Arvore.SubarvoreEsquerda
13         Arvore.SubarvoreEsquerda ← NIL
15         Direita ← Arvore
16     end
18     Atualização(Esquerda)
20     Atualização(Direita)
22     return Esquerda, Direita
1  Function Remoção(Arvore, ChaveBST)
3      Esquerda, Direita ← Separação(Arvore, ChaveBST)
5      Ignorar, Direita ← SepMenor(Direita)
7      return Junção(Esquerda, Direita)
```

Algoritmo 7: Kolmogorov-Smirnov Incremental.

```
1  Function InsereObservaçãoEmA(Treap, Observacao)
3      Esquerda, Direita ← Separação(Treap, Observacao)
5      Esquerda, Temp ← SepMaior(Esquerda)
6      if Temp é NIL then
8          ValorInicial ← 0
9      else
11         ValorInicial ← Temp.Valor
12     end
14     Esquerda ← Junção(Esquerda, Temp)
16     Prioridade ← GeraNumeroPseudoaleatorio()
18     NovoNo ← NovoNó(Prioridade, Observacao, ValorInicial)
20     Incremento(Direita, 1)
22     return Junção(Esquerda, Direita)
1  Function InsereObservaçãoEmB(Treap, Observacao)
3      Mesmos passos iniciais de InsereObservaçãoEmA(Treap, Observacao)
5      Incremento(Direita, −r)
7      return Junção(Esquerda, Direita)
1  Function RemoveObservaçãoDeA(Treap, Observacao)
3      Esquerda, Direita ← Separação(Treap)
5      Ignorar, Direita ← SepMenor(Direita)
7      Incremento(Direita, r)
9      return Junção(Esquerda, Direita)
1  Function RemoveObservaçãoDeB(Arvore, Observacao)
3      Mesmos passos iniciais de RemoveObservaçãoDeA(Treap, Observacao)
5      Incremento(Direita, r)
7      return Junção(Esquerda, Direita)
1  Function AplicaTesteKS(Arvore, |A|, |B|, n, m, α)
3      D ← max{Treap.Maximo, −Treap.Minimo}/|A|
5      p-value ← c(α)√((n+m)/(n∗m))
6      if D > p-value then
8          Rejeita hipótese nula
9      else
11         Não rejeita hipótese nula
12     end
```

CAPÍTULO

7

APRENDIZADO ATIVO E POR TRANSFERÊNCIA

Neste capítulo, é considerado o cenário de classificação em fluxo de dados com as seguintes características: **(a)** há conhecimento dos rótulos verdadeiros de uma porção inicial dos dados, suficiente para indução de um classificador; **(b)** é possível solicitar o rótulo correto de qualquer evento e obtê-lo instantaneamente; e **(c)** o objetivo é obter alta acurácia do sistema de classificação, enquanto requisitando o menor número possível de rótulos corretos. Para este cenário, o capítulo apresenta uma solução baseada em detecção não-supervisionada de mudança de conceito com o auxílio do algoritmo Kolmogorov-Smirnov Incremental. As próximas seções detalham o funcionamento do algoritmo e análise experimental.

7.1 Proposta

Esta seção detalha como o algoritmo IKS pode ser utilizado detectar mudanças de conceito sem informações de rótulo verdadeiro e as possíveis ações a serem realizadas, após uma detecção.

É considerado, a princípio, um modelo de classificação induzido com os W primeiros eventos do fluxo de dados. Há, para cada atributo do espaço de atributos, duas amostras, também de tamanho W. A primeira delas, chamada de *amostra de referência*, contém os eventos utilizados para induzir o modelo de classificação. A segunda, chamada *amostra corrente*, é composta de uma janela deslizante com os últimos W eventos registrados no fluxo de dados.

Durante a análise do fluxo de dados, após cada evento ter sido classificado, um teste de Kolmogorov-Smirnov é realizado entre a amostra de referência e a amostra corrente. Caso a execução do teste indique a rejeição da hipótese nula de que ambas amostras possuem mesma distribuição de probabilidade, é assumida a ocorrência de mudança de conceito.

A proposta apresentada difere do trabalho de Žliobaitė (2010), descrito na Seção 2.5, em três quesitos fundamentais. Primeiro, na proposta aqui descrita, a janela de referência é fixa, ao contrário de duas janelas deslizantes consecutivas. O raciocínio por trás da decisão é consequência direta do cenário proposto: a necessidade de detectar mudanças de conceito existe como maneira de reconhecer que o modelo de classificação está defasado. Dessa forma, é mais prático e faz mais sentido utilizar os dados que foram aplicados no processo de indução do modelo do que assumir seu defasamento de forma indireta.

A segunda diferença é a aplicação de um teste KS para cada atributo individualmente, ao invés de utilizar um teste multidimensional ou uma função de mapeamento que transforma cada evento em um único valor unidimensional. Ambas as opções alternativas requiririam amostras maiores e a mudança em um único atributo já significa uma mudança de conceito. O ponto negativo da de testes individuais é o de que, se uma mudança de conceito em um atributo específico for não detectável (ŽLIOBAITĖ, 2010), a mudança poderia ser eventualmente detectada com o uso de um teste multivariável.

Quando um algoritmo oferece como saída, além do rótulo dado a um evento, a probabilidade estimada para este rótulo, é possível realizar o teste sobre as probabilidades estimadas, ao invés de valores do espaço de atributos. A amostra corrente deve conter as probabilidades estimadas (atuando como níveis de confiança), enquanto que a amostra de referência contém as probabilidades estimadas obtidas pela execução do conhecido procedimento *leave one out*. Neste procedimento, dado um conjunto de treino com N eventos, N classificadores são induzidos. Cada classificador utiliza como subconjunto de treino todos os eventos, exceto um, que é então classificado. A probabilidade estimada do rótulo dado é adicionada à amostra de referência.

O teste de Kolmogorov-Smirnov foi escolhido por Žliobaitė (2010), dentre as opções consideradas, por ser não-paramétrico e, principalmente, por ser computacionalmente mais eficiente do que as outras opções. A proposta descrita nesta seção, assim como a proposta de Žliobaitė (2010), são implementáveis com o uso do algoritmo IKS, tendo performance $O(nmlogn + n * C(n,m))$, sendo m o número de atributos, n a quantidade de eventos do fluxo de dados e C o custo de classificação de 1 evento, dependente exclusivamente de qual modelo é utilizado.

São propostas três ações que podem ser realizadas, uma vez que uma mudança de conceito é detectada. Elas seguem:

- **Substituição de modelo (MR):** uma vez que uma mudança é detectada, o sistema de classificação requisita os rótulos verdadeiros de todos os eventos da amostra corrente. Estes eventos então são utilizados para induzir um novo modelo de classificação, que substitui o que estiver sendo utilizado no momento. A amostra de referência é atualizada apropriadamente. A requisição de rótulos corretos caracteriza esta proposta como **aprendizado ativo**;

- **Transformação α/β (AB):** para o atributo que causou a detecção de mudança de conceito, o sistema translada e estica os dados da amostra de referência, de forma que a média e o desvio padrão daquele atributo sejam compatíveis com os encontrados na amostra corrente. A Equação 7.1 explicita a transformação f, sendo μ_{ref} e σ_{ref} a média e o desvio padrão da amostra de referência, respectivamente, e μ_{cor} e σ_{cor} a média e o desvio padrão da amostra corrente. Após a transformação, se a amostra de referência transformada e a amostra corrente forem distribuídas segundo a mesma distribuição de probabilidade, de acordo com o teste de Kolmogorov-Smirnov, a amostra de referência transformada é utilizada para induzir um novo modelo de classificação sem a necessidade de solicitar rótulos verdadeiros adicionais. Caso contrário, o algoritmo aplica a substituição de modelo (MR) descrita no item anterior. A Transformação α/β configura esta proposta como um método de **aprendizado por transferência**;

$$f(v) = \frac{v - \mu_{\text{ref}}}{\sigma_{\text{ref}}}\sigma_{\text{cor}} + \mu_{\text{cor}} \tag{7.1}$$

- **Adaptree (AT):** é uma versão modificada de árvores de decisão. Uma vez que uma mudança de conceito é detectada, o sistema solicita apenas os rótulos verdadeiros dos eventos que alcançam os nós da árvore que utilizam os atributos que sofreram mudança. Tais nós e, consequentemente, seus descendentes são substituídos pela indução de uma nova ramificação da árvore, induzida com os novos dados rotulados. O conjunto de referência é atualizado de forma a substituir os valores dos atributos que sofreram mudança de conceito e aqueles pertencentes aos nós descendentes. Os atributos utilizados por nós primos e antecessores são mantidos inalterados, no conjunto de referência, para evitar o mascaramento de mudanças de conceito.

A próxima seção explica os procedimentos de avaliação adotados pelo trabalho, para averiguar a eficiência das propostas apresentadas.

7.2 Configuração experimental

Para avaliar as propostas de reações à detecção de mudança de conceito não-supervisionada, foi mensurada a acurácia e o número de rótulos requeridos em diferentes bases de dados. Como as bases são suficientemente balanceadas, a acurácia é um bom indicador de qualidade de predição. Os resultados obtidos foram comparados com dois *baselines* e com um *topline*. O primeiro *baseline* (BL1) é um algoritmo treinado ao início do fluxo de dados, com os primeiros W eventos, e que nunca tem seu modelo de classificação alterado. O segundo *baseline* (BL2) retreina seu modelo aleatoriamente durante o fluxo. A quantidade de vezes em que esse retreino acontece é a mesma quantidade de retreinos realizados pelo procedimento MR. Devido à natureza aleatória deste *baseline*, os valores apresentados são as médias de 100 execuções do experimento. O

topline (TL) é um classificador que retreina seu modelo com os últimos W eventos após cada classificação. Adicionalmente, foi as propostas foram comparadas a um classificador que detecta mudança de conceito com um dos métodos propostos por Žliobaitė (2010) – especificamente, duas janelas deslizantes consecutivas de tamanho W, que armazenam as estimativas de probabilidade para a classe mais provável, de acordo com um modelo de classificação que seja capaz de prover esses dados. De agora em diante, a Substituição de Modelo com o método descrito será referida como *Detecção com Janelas Consecutivas* (CD).

Os classificadores utilizados durante o procedimento experimental foram Vizinho Mais Próximo (1NN), Árvore de Decisão (DT) e Naïve Bayes (NB). *Baselines* 1 e 2 (BL1 e BL2), *topline* (TL) e Substituição de Modelo foram aplicados com todos os algoritmos. Particularmente, MR foi aplicado com Naïve Bayes para probabilidades estimadas (MRP) e para espaço de atributos (MR), pois Naïve Bayes é o único algoritmo, dentre os utilizados, que provê estimativa para probabilidades. 1NN também foi testado com Transformação α/β (AB), DT foi também testado com Adaptree (AT) e Naïve Bayes foi também testado com Detecção com Janelas Consecutivas (CD). Para os métodos que usam árvores de decisão, foram consideradas apenas as detecções de mudança de conceito de atributos para os quais há correspondente nó de decisão.

Todas as comparações de pares mencionadas na seção são baseadas no Teste de Ranking com Sinal de Wilcoxon para validade estatística com nível de significância de $0,05$. A seção seguinte lista as bases de dado utilizadas nos experimentos.

7.2.1 Bases de dados

Foram utilizadas seis bases reais, nos experimentos. Em três delas, mudanças de conceito foram artificialmente introduzidas. Todas as bases estão disponíveis como material suplementar (REIS, 2016). Seguem descrições das bases:

(A) **Arabic** (HAMMAMI; BEDDA, 2010) contém atributos extraídos do áudio de 88 pessoas pronunciando dígitos arábicos, entre 0 e 9. 44 pessoas são do sexo feminino e 44 são do sexo masculino. A tarefa é identificar qual dígito foi pronunciado. Originalmente, os eventos da base de dados são independentes e identicamente distribuídos (i.i.d.). Para introduzir mudança de conceito, artificialmente, o fluxo de dados foi separado em 4 partes de tamanhos iguais. As partes se alternam entre vozes masculinas e femininas. O fluxo contém 8.800 eventos;

(B) **Posture** (KALUŽA *et al.*, 2010) contém dados de um sensor que é carregado por 5 pessoas diferentes. A tarefa é detectar qual movimento está sendo realizado, entre 11 possibilidades. Este é a única base de dados considerada que não é bem balanceada entre as classes, de forma que faz-se relevante a observação de que a proporção da classe majoritária corresponde a 33% dos dados. Há 164.860 eventos. O fluxo de dados é originalmente

I.I.D.. Para introduzir mudança de conceito, artificialmente, o fluxo possui segmentos contínuos de eventos com dados produzidos por uma mesma pessoa;

(C) **Bike** (FANAEE-T; GAMA, 2013) contém contagem por hora de bicicletas alugadas entre os anos 2011 e 2012, de um sistema de empréstimo de bicicleta. Os dados são acompanhados de informações meteorológicas. A tarefa é predizer a demanda, dentre *baixa* e *alta*. Espera-se a existência de mudança de conceito devido à sazonalidade. O fluxo contém 17.379 eventos;

(D) **Keystroke** (SOUZA *et al.*, 2015) contém dados referentes a cinco pessoas digitando uma mesma senha diversas vezes. A tarefa é identificar quem está digitando a senha. Espera-se existência de mudança de conceito devido à prática adquirida pelo digitador em reproduzir a senha com maior velocidade, ao longo do tempo. Há 1.600 eventos;

(E) **Insects** (SOUZA *et al.*, 2013) contém atributos de um sensor inteligente. A tarefa é identificar a espécie de um inseto voador que atravessa um feixe laser planar em um ambiente controlado. Há 5 espécies possíveis. Análise preliminar indicou que não há mudanças de conceito no espaço de atributos, embora haja mudanças graduais nas proporções das classes ao longo do tempo. O fluxo possui 5.325 eventos;

(F) **Abrupt Insects** é uma versão modificada da base anterior. Os eventos foram embaralhados de forma a eliminar mudanças nas probabilidades das classes. Em seguida, o fluxo foi dividido em três segmentos. Para o segmento do meio, os atributos foram embaralhados para introduzir mudança no espaço de atributos, sem inserir artefatos adicionais aos dados.

Todos os algoritmos aplicaram teste IKS com nível de significância de $0,001$ para detectar mudanças de conceito. O raciocínio por trás da decisão foi que, com um maior nível de significância, é possível evitar detecção de mudanças durante períodos nos quais a janela deslizante ainda transita entre dois conceitos. O tamanho da janela é de 100 eventos para todas as bases de dados, com exceção das bases Arabic e Posture. Como essas bases possuem maior número de classes, foram utilizadas janelas de 500 eventos para elas. A próxima seção apresenta os resultados obtidos e sua análise.

7.3 Resultados experimentais

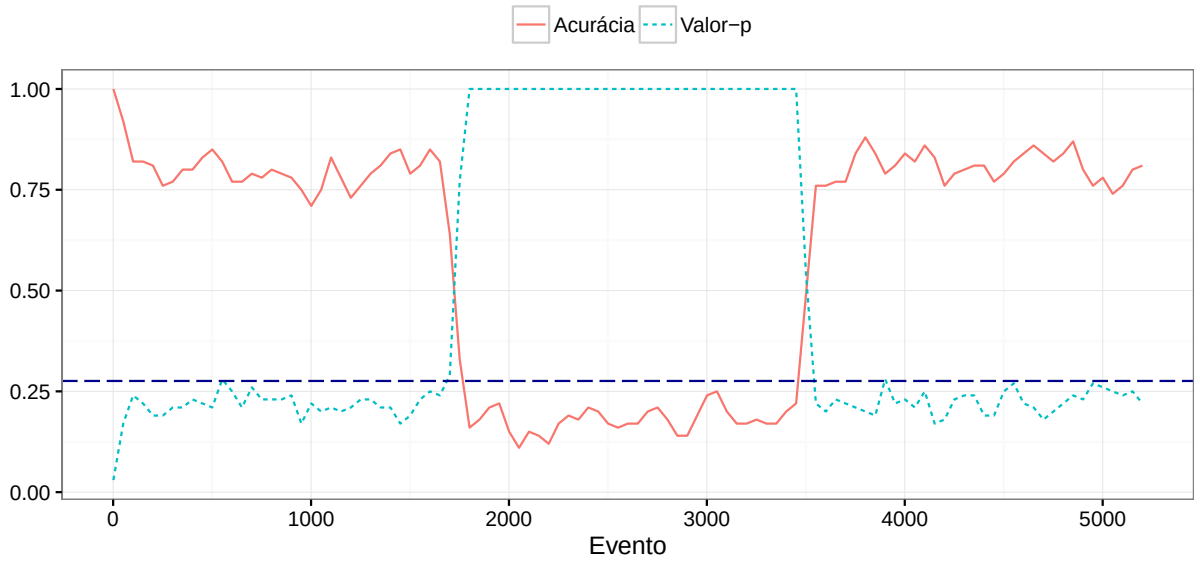
Nesta seção, inicialmente são apresentados resultados de um experimento adicional, no qual o teste de Kolmogorov-Smirnov é utilizado para prever o mal desempenho de um classificador. Para o fim proposto, foi utilizado o detector de mudança de conceito descrito neste capítulo em conjunto ao modelo de classificação *baseline* Vizinho Mais Próximo, e calculado o coeficiente de correlação de Pearson entre o valor-p produzido pelo detector e a acurácia obtida pelo modelo de classificação, na mesma janela. A Tabela 5 apresenta os resultados obtidos.

Tabela 5 – Coeficientes de correlação de Pearson entre acurácia obtida e o valor-p produzido pelo algoritmo incremental de Kolmogorov-Smirnov.

Base de dados	Coeficiente
Arabic	$-0,7285$
Posture	$-0,7658$
Bike	$-0,7208$
Keystroke	$-0,8872$
Insects	$-0,8160$
Abrupt Insects	$-0,9908$

Nota-se alta correlação absoluta entre os valores de acurácia e os valores-p produzidos pelo teste, indicando alto potencial do emprego do teste incremental de Kolmogorov-Smirnov para avaliação não-supervisionada. Em particular, a base Abrupt Insects obteve o maior valor absoluto, o que era esperado, devido à inserção artificial de mudança de conceito abrupta. A Figura 23 ilustra os valores de acurácia e de valor-p ao longo do tempo. A linha tracejada horizontal, em azul escuro, é o limiar de corte aplicado valor-p para reconhecimento de mudança de conceito, com nível de significância de $0,001$.

Figura 23 – Acurácia do *baseline* Vizinho Mais Próximo e valor-p ao longo do tempo, para base de dados Abrupt Insects.



Fonte: Dados da pesquisa.

O restante da seção é dedicado à análise dos resultados dos experimentos principais, que foram previamente descritos. As Tabelas 6 e 7 sumarizam os resultados experimentais para acurácia e porcentagem de rótulos verdadeiros solicitados, respectivamente. Para todos os classificadores, Substituição de Modelo produziu estatisticamente maior acurácia que ambos os correspondentes *baselines* e necessitou menos rótulos verdadeiros do que o correspondente *topline*. Embora a acurácia da Substituição de Modelo seja estatisticamente inferior à obtida

pelo *topline*, para todos os classificadores, Substituição de Modelo perde por uma margem muito pequena. Em média, sua acurácia foi de $99,39\%$, $94,31\%$ e $93,69\%$ das acurácias do *topline* para os classificadores Vizinho Mais Próximo, Naïve Bayes e Árvore de Decisão, respectivamente. Substituição de Modelo requiriu $47,59\%$, para Vizinho Mais Próximo e Naïve Bayes, e $42,17\%$ para Árvore de Decisão, da quantidade de rótulos verdadeiros.

Tabela 6 – Acurácia para cada classificador / base de dados (%).

Base	Vizinho Mais Próximo					Naïve Bayes					
	BL1	BL2	MR	AB	TL	BL1	BL2	MR	MRP	CD	TL
A	29,39	29,59	37,18	29,31	38,24	17,28	17,98	20,11	17,28	18,15	22,08
B	40,84	50,25	53,95	47,52	53,89	41,78	44,96	46,17	45,62	43,06	45,74
C	49,36	67,04	75,40	75,87	75,41	48,33	70,71	70,60	70,93	69,75	74,76
D	59,75	77,11	84,56	84,13	84,88	48,31	71,22	74,31	71,81	48,31	79,94
E	58,14	71,67	87,81	82,01	88,13	54,01	70,43	77,99	76,53	68,38	86,61
F	57,86	69,75	79,87	79,17	80,06	58,97	69,23	76,54	77,22	66,63	79,42

Base	Árvore de Decisão				
	BL1	BL2	MR	AB	TL
A	19,95	18,98	22,83	20,85	23,47
B	38,73	45,60	47,61	48,02	48,65
C	58,21	64,83	65,42	65,90	72,88
D	39,75	70,79	74,56	76,25	82,94
E	49,93	65,29	71,06	71,81	79,91
F	52,17	59,61	65,31	65,30	66,35

Fonte: Dados da pesquisa.

Tabela 7 – Porção dos rótulos verdadeiros requisitada para cada classificador / base de dados (%).

Base	Vizinho Mais Próximo					Naïve Bayes					
	BL1	BL2	MR	AB	TL	BL1	BL2	MR	MRP	CD	TL
A	5,68	25,40	28,41	5,68	100,00	5,68	25,67	28,41	5,67	11,36	100,00
B	0,30	5,91	6,07	0,91	100,00	0,30	5,90	6,07	2,73	0,61	100,00
C	0,58	62,25	98,39	32,22	100,00	0,58	62,54	98,39	74,80	44,31	100,00
D	6,25	12,15	43,75	43,75	100,00	6,25	12,05	43,75	25,00	6,25	100,00
E	1,88	57,56	84,51	9,39	100,00	1,88	57,40	84,51	39,44	30,05	100,00
F	1,88	21,84	24,41	9,39	100,00	1,88	21,93	24,41	9,39	5,63	100,00

Base	Árvore de Decisão				
	BL1	BL2	MR	AB	TL
A	5,68	25,51	28,41	14,91	100,00
B	0,30	5,90	6,07	5,91	100,00
C	0,58	62,22	98,39	92,13	100,00
D	6,25	12,36	37,50	24,13	100,00
E	1,88	57,64	69,48	60,90	100,00
F	1,88	22,11	13,15	15,74	100,00

Fonte: Dados da pesquisa.

Todos os métodos restantes levaram a quantidades significativamente menores do número de rótulos verdadeiros requisitados. Entretanto, a performance de classificação usualmente também foi reduzida. Uma notável exceção foi Adaptree, que obteve performance similar à Substituição de Modelo para árvores de decisões, enquanto utilizando significativamente menos rótulos ($42,17\%$ para MR e $35,62\%$ para Adaptree, em média). Infelizmente, árvores de decisão não tiveram performance similar à obtida por Vizinho Mais Próximo.

Transformação α/β reduziu estatisticamente a porção de rótulos verdadeiros que foram requisitados se comparada à Substituição de Modelo ($16,89$, em média, pra α/β). Entretanto, essa transformação em particular é sensível à natureza da mudança de conceito, uma vez que

ela assume monotonicidade da transformação (AL-OTAIBI *et al.*, 2015). Se a premissa não for satisfeita, mesmo que a mudança de conceito seja detectável, a transformação pode levar a uma acurácia mais baixa. A acurácia produzida pelo método foi estatisticamente inferior à Substituição de Modelo sozinha. Em média, α/β obteve $92,75\%$ da acurácia do Topline.

Substituição de Modelo obteve consistentemente mais acurácia do que Detecção com Janelas Consecutivas ($82,15\%$ da acurácia do *topline* com CD), enquanto que CD consumiu consistentemente uma menor quantidade do número de rótulos verdadeiros ($16,37\%$ em média).

Por último, utilizar uma amostra fixa (amostra de referência) consistentemente produziu maior acurácia que Detecção com Janelas Consecutivas ($82,15\%$ da acurácia do *topline* para CD), enquanto CD consistentemente consumiu uma menor quantidade de rótulos verdadeiros (16,37% em média). Nota-se também que a detecção de conceito baseada em probabilidades estimadas não produziu acurácias significativamente diferentes das produzidas pela detecção baseada no espaço de atributos, ainda que consistentemente necessitando menos rótulos verdadeiros. Assim, o uso de probabilidades estimadas se mostrou uma abordagem promissora para dados de alta dimensionalidade, caso o classificador utilizado se mostrar compatível. Adicionalmente, observa-se que misturar as abordagens (valores do espaço de atributos, probabilidades estimadas e rótulos preditos) se apresenta como uma possível abordagem alternativa.

7.4 Considerações finais

Este capítulo propôs métodos para aprendizado ativo e por transferência que são ponderados pela velocidade do algoritmo IKS, apresentado no capítulo anterior. Os resultados obtidos pela análise experimental mostraram uma alta redução do número de rótulos verdadeiros necessários para se obter uma acurácia pouco inferior a obtida as algoritmos equivalentes, mas com acesso irrestrito aos rótulos. Nota-se que conhecimento de domínio, sobre o tipo de mudanças de conceito que possam ocorrer, possibilitam o desenvolvimento de reações à mudança de conceito capazes de economizar ainda mais rótulos corretos. Um exemplo é a Transformação α/β que, dada a validade de suas premissas, reduz drasticamente a quantidade de rótulos requiridos para atualizar o modelo de classificação. Particularmente, tanto a Transformação α/β quanto uma transformação por percentis (AL-OTAIBI *et al.*, 2015) têm potencial para classificação não supervisionada em fluxo de dados, ou latência de verificação extrema, caso seja confirmável que as únicas mudanças de conceito ocorrentes sejam no espaço de atributos e, ainda, sejam monotônicas.

CAPÍTULO

8

CONCLUSÕES

Em Aprendizado de Máquina, o estudo de fluxos de dados objetiva a capacidade de *a*) análise dos dados que são observados continuamente, potencialmente de forma ininterrupta e por tempo indeterminado; e *b*) inferência de informação relevante sobre os dados analisados.

A diminuição ou eliminação de intervenção externa, que possa prover informações adicionais aos mecanismos de análise e inferência, por exemplo um *oráculo*, é benéfica por diminuir custos de tempo, financeiro ou por simplesmente não ser aplicável à situações específicas. Portanto, a diminuição ou eliminação do uso de mecanismos externos caracteriza um problema a ser estudado.

Nossa revisão bibliográfica identificou diversos trabalhos que objetivam diminuir ou eliminar tal intervenção. Devido à dificuldade do problema, os trabalhos revisados apresentam escopos distintos, habitualmente não comparáveis, específicos às características das aplicações estudadas, com premissas que possibilitam o desenvolvimento das propostas realizadas.

Outros trabalhos, no entanto, não realizam qualquer restrição sobre consultas a fontes externas para obtenção de informação, apresentando resultados otimistas quando comparados à aplicação prática.

Nesta dissertação de mestrado, nosso maior interesse está na tarefa de classificação em fluxo de dados, na qual é desejada a inferência da classe de um evento do fluxo, dentre um conjunto limitado e conhecido de classes possíveis. Para esta tarefa, estudamos o impacto do atraso na disponibilidade de rótulos verdadeiros, caracteristicamente oferecidos por uma fonte externa, ou *oráculo*, e oferecemos métodos para mitigar a necessidade de tais rótulos.

Ao longo da pesquisa, diferentes contribuições se agregaram ao presente trabalho, sendo apresentadas na seção subsequente.

8.1 Contribuições

Esta seção apresenta as diferentes contribuições realizadas pela presente dissertação de mestrado.

8.1.1 Avaliação em fluxo de dados

Neste trabalho, apresentamos uma sólida revisão sobre diferentes técnicas utilizadas para avaliar e comparar estatísticas provindas da execução de algoritmos de classificação em fluxos de dados. A revisão pode ser empregada como base para a elaboração de análises experimentais em trabalhos futuros.

Para maior compreensão das bases de dados utilizadas nos experimentos, sugerimos técnicas de análise para identificação de padrões de mudança de conceito, e técnicas de manipulação de dependência temporal.

Como alerta à comunidade científica, apresentamos evidências empíricas de que a diferença de atrasos na disponibilidade de rótulos verdadeiros afeta o desempenho relativo de algoritmos de classificação que dependam dessa informação, *i.e.*, se um algoritmo é mais performático que outro para uma determinada configuração de atraso, o inverso pode ocorrer em uma configuração diferente de atraso. Portanto, o reconhecimento do atraso como uma variável é importante em avaliações experimentais.

Por fim, um breve experimento evidenciou o uso da estatística de Kolmogorov-Smirnov como relevante indicativo de performance de classificação em fluxos de dados, obtenível de maneira não-supervisionada.

8.1.2 Detecção não-supervisionada de mudanças de conceito

Durante a pesquisa, introduzimos um novo algoritmo para eficiente computação da estatística de Kolmogorov-Smirnov em fluxos de dados. O teste de hipótese não-paramétrico de Kolmogorov-Smirnov objetiva oferecer garantias estatísticas para rejeição ou não-rejeição da hipótese das amostras serem originadas pela mesma distribuição de probabilidade. O algoritmo introduzido por este trabalho considera duas amostras mutáveis, A e B, e é capaz de calcular a estatística após uma modificação sobre essas amostras com complexidade de tempo logarítmica, em oposição à complexidade linearítmica da reaplicação da implementação padrão do teste KS. As modificações possíveis são inserção e remoção de observações. Os experimentos realizados mostram a clara superioridade da nossa proposta para amostras sempre crescentes e janelas deslizantes em fluxos de dados.

O algoritmo proposto, chamado *Incremental Kolmogorov-Smirnov* (IKS), foi utilizado em nossa proposta de detecção não-supervisionada de mudanças de conceito. Analisamos o desempenho da detecção em conjunto ao emprego de mecanismos de reação às mudanças.

Os mecanismos propostos, no entanto, são semi-supervisionados e melhor descritos na seção seguinte.

As contribuições desta seção, e da seguinte, originaram o artigo *Fast Unsupervised Online Drift Detection Using Incremental Kolmogorov-Smirnov Test* na *22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (REIS *et al.*, 2016).

8.1.3 Adaptação de classificadores com quantidades limitadas de rótulos verdadeiros

O objetivo da detecção de mudança de conceito é a identificação inteligente dos instantes nos quais alterações no modelo de classificação são estritamente requiridas para manter boa performance. Em particular, neste trabalho, o uso de uma técnica não-supervisionada para a detecção tem o objetivo de reduzir a quantidade de informação externa, na forma de rótulos verdadeiros, necessária para a adaptação do modelo.

Propusemos três métodos de adaptação diferentes. O primeiro deles, e também o mais conservador, é a substituição completa do modelo antigo por um novo, induzido sobre os últimos eventos ocorrentes. Este método, chamado Substituição de Modelo, requer os rótulos verdadeiros dos eventos utilizados para indução do novo modelo.

O segundo método, Transformação α/β, tenta aplicar uma transformação afim aos dados que originaram o modelo de classificação em uso, para se adaptar ao novo conceito sem o uso de informação externa. Como mecanismo de falha, recorre à substituição de modelo.

O terceiro e último método, Adaptree, é o uso de uma árvore de decisão especial, capaz de filtrar quais eventos são necessários para adaptar os ramos afetados pela mudança de conceito e, assim, requisitar os rótulos corretos apenas para estes eventos em particular.

Em nossos experimentos, os três métodos apresentaram maior acurácia do que dois *baselines*, necessitando menor número de rótulos verdadeiros do um *topline*. Observamos que árvores de decisão consistentemente apresentaram pior performance quando comparadas aos outros algoritmos de de classificação empregados: vizinho mais próximo e Naïve Bayes.

Adicionalmente, realizamos os experimentos com a detecção baseada em duas fontes de observações para o teste de Kolmogorov-Smirnov: valores do espaço de atributos e estimativas das probabilidades das classes (quando aplicável). A performance da detecção baseada em estimativa de probabilidade das classes se mostrou promissora, o que pode ser decisivo para a aplicação das técnica em bases que venham a sofrer com alta dimensionalidade.

8.2 Limitações

Como limitação do método IKS está a restrição de que ambas as amostras tenham, virtualmente, as mesmas dimensões, *i.e.*, $|A| = |B|$. Entretanto, observamos que a restrição não oferece implicações práticas no uso da técnica.

Apesar das técnicas de adaptação de modelo apresentadas diminuírem a frequência com que rótulos verdadeiros são requeridos, eles ainda são necessários. A necessidade de informação externa oferecida por um oráculo característica a maior limitação atual das propostas realizadas.

Adicionalmente, o emprego da detecção de mudanças em bases com muitas dimensões pode comprometer a performance, uma vez que a aplicação do teste de Kolmogorov-Smirnov é efetuada para cada atributo. Uma maneira de contornar este problema é o uso de estimativas das probabilidades das classes para detecção de mudanças, em oposição aos valores do espaço de atributos. Entretanto, essa fonte de observações para o teste restringe quais classificadores podem ser utilizados, pois nem todos os classificadores oferecem essa informação.

8.3 Perspectivas futuras

O método de Substituição de Modelo para adaptação de modelo de classificação é muito simples e conservador. Técnicas mais elaboradas, como Transformação α/β, diminuem drasticamente a quantidade de informação externa requerida para adaptação. Como contraponto, realizam premissas específicas sobre os dados utilizados. A perspectiva de trabalhos futuros, em continuidade ao apresentado, está no desenvolvimento de técnicas de adaptação mais elaboradas, para diferentes tipos de dados, que possam eventualmente adaptar o modelo com pouca ou nenhuma interferência externa.

REFERÊNCIAS

AL-OTAIBI, R.; PRUDÊNCIO, R. B.; KULL, M.; FLACH, P. Versatile decision trees for learning over multiple contexts. In: **ECML/PKDD**. [S.l.: s.n.], 2015. p. 184–199. Citado na página 94.

ALON, N.; MATIAS, Y.; SZEGEDY, M. The space complexity of approximating the frequency moments. **Journal of Computer and System Sciences**, v. 58, n. 1, p. 137–147, 1999. Citado na página 30.

AMIR, M.; TOSHNIWAL, D. Instance-based classification of streaming data using emerging patterns. In: **ICT**. [S.l.: s.n.], 2010. p. 228–236. Citado na página 38.

ARAGON, C. R.; SEIDEL, R. G. Randomized search trees. In: IEEE. **FOCS**. [S.l.], 1989. p. 540–545. Citado na página 75.

BAENA-GARCÍA, M.; CAMPO-ÁVILA, J. del; FIDALGO, R.; BIFET, A.; GAVALDÀ, R.; MORALES-BUENO, R. Early drift detection method. In: **Fourth International Workshop on Knowledge Discovery from Data Streams**. [S.l.: s.n.], 2006. p. 1–10. Citado na página 68.

BATISTA, G. E. A. P. A.; HAO, Y.; KEOGH, E.; NETO, A. M. Towards automatic classification on flying insects using inexpensive sensors. In: **Proceedings of Tenth International Conference on Machine Learning and Applications**. Honolu, Hawaii, USA: IEEE Computer Society, 2011. v. 1, p. 364–369. ISBN 978-0-7695-4607-0. Citado na página 25.

BATISTA, G. E. A. P. A.; KEOGH, E. J.; NETO, A. M.; ROWTON, E. Sensors and software to allow computational entomology, an emerging application of data mining. In: **KDD**. [S.l.: s.n.], 2011. p. 761–764. Citado 3 vezes nas páginas 25, 36 e 67.

BENTLEY, J. L. Multidimensional binary search trees used for associative searching. **Communications of the ACM**, ACM, v. 18, n. 9, p. 509–517, 1975. Citado na página 66.

BIFET, A. Adaptive learning and mining for data streams and frequent patterns. **SIGKDD Explorations Newsletter**, ACM, New York, NY, USA, v. 11, n. 1, p. 55–56, nov. 2009. ISSN 1931-0145. Disponível em: <http://doi.acm.org/10.1145/1656274.1656287>. Citado 2 vezes nas páginas 23 e 29.

BIFET, A.; GAVALDÀ, R. Learning from time-changing data with adaptive windowing. In: **SDM**. [S.l.: s.n.], 2007. p. 443–448. Citado 3 vezes nas páginas 23, 45 e 68.

____. Adaptive learning from evolving data streams. In: **IDA**. [S.l.: s.n.], 2009. p. 249–260. Citado 2 vezes nas páginas 23 e 68.

BIFET, A.; HOLMES, G.; KIRKBY, R.; PFAHRINGER, B. Moa: Massive online analysis. **Journal of Machine Learning Research**, JMLR.org, v. 11, p. 1601–1604, ago. 2010. ISSN 1532-4435. Disponível em: <http://dl.acm.org/citation.cfm?id=1756006.1859903>. Citado 2 vezes nas páginas 67 e 68.

BIFET, A.; HOLMES, G.; PFAHRINGER, B. Leveraging bagging for evolving data streams. In: **ECML/PKDD**. [S.l.: s.n.], 2010. p. 135–150. Citado 2 vezes nas páginas 23 e 68.

BIFET, A.; MORALES, G. de F.; READ, J.; HOLMES, G.; PFAHRINGER, B. Efficient online evaluation of big data stream classifiers. In: ACM. **Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [S.l.], 2015. p. 59–68. Citado na página 45.

BIFET, A.; PFAHRINGER, B.; READ, J.; HOLMES, G. Efficient data stream classification via probabilistic adaptive windows. In: ACM. **ACM SAC**. [S.l.], 2013. p. 801–806. Citado na página 68.

BIFET, A.; READ, J.; ŽLIOBAITĖ, I.; PFAHRINGER, B.; HOLMES, G. Pitfalls in benchmarking data stream classification and how to avoid them. In: **Machine Learning and Knowledge Discovery in Databases**. [S.l.]: Springer, 2013. p. 465–479. Citado na página 62.

BOTTCHER, M.; SPOTT, M.; KRUSE, R. Predicting future decision trees from evolving data. In: IEEE. **Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on**. [S.l.], 2008. p. 33–42. Citado na página 32.

BROWN, S.; STEYVERS, M. Detecting and predicting changes. **Cognitive Psychology**, Elsevier, v. 58, n. 1, p. 49–67, 2009. Citado na página 32.

CHAUDHURI, S.; MOTWANI, R.; NARASAYYA, V. On random sampling over joins. **ACM SIGMOD Record**, ACM, v. 28, n. 2, p. 263–274, 1999. Citado na página 30.

CHERNOFF, H. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. **The Annals of Mathematical Statistics**, JSTOR, p. 493–507, 1952. Citado na página 43.

DATAR, M.; GIONIS, A.; INDYK, P.; MOTWANI, R. Maintaining stream statistics over sliding windows. In: SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS. **Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms**. [S.l.], 2002. p. 635–644. Citado na página 30.

DAWID, A. P. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. **Journal of the Royal Statistical Society. Series A (General)**, JSTOR, p. 278–292, 1984. Citado na página 42.

DELANY, S.; CUNNINGHAM, P.; TSYMBAL, A.; COYLE, L. A case-based technique for tracking concept drift in spam filtering. **Knowledge-Based Systems**, Elsevier, v. 18, n. 4, p. 187–195, 2005. Citado na página 32.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **The Journal of Machine Learning Research**, JMLR. org, v. 7, p. 1–30, 2006. Citado 4 vezes nas páginas 49, 50, 51 e 52.

DEVROYE, L. A note on the height of binary search trees. **JACM**, ACM, v. 33, n. 3, p. 489–498, 1986. Citado na página 78.

DIETTERICH, T. G. Approximate statistical tests for comparing supervised classification learning algorithms. **Neural computation**, MIT Press, v. 10, n. 7, p. 1895–1923, 1998. Citado 2 vezes nas páginas 41 e 49.

DOMINGOS, P.; HULTEN, G. Mining high-speed data streams. In: **Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining**. New York, NY, USA: ACM, 2000. p. 71–80. ISBN 1-58113-233-6. Disponível em: <http://doi.acm.org/10.1145/347090.347107>. Citado na página 68.

DUNN, O. J. Multiple comparisons among means. **Journal of the American Statistical Association**, Taylor & Francis Group, v. 56, n. 293, p. 52–64, 1961. Citado na página 52.

DYER, K. B.; CAPO, R.; POLIKAR, R. Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. **TNNLS**, 2013. Citado 2 vezes nas páginas 36 e 39.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: **Kdd**. [S.l.: s.n.], 1996. v. 96, n. 34, p. 226–231. Citado na página 66.

FANAEE-T, H.; GAMA, J. a. Event labeling combining ensemble detectors and background knowledge. **Progress in Artificial Intelligence**, Springer Berlin Heidelberg, p. 113–127, 2013. ISSN 2192-6352. Citado na página 91.

FEIGENBAUM, J.; KANNAN, S.; STRAUSS, M. J.; VISWANATHAN, M. An approximate l 1-difference algorithm for massive data streams. **SIAM Journal on Computing**, SIAM, v. 32, n. 1, p. 131–151, 2002. Citado na página 23.

FISHER, R. A. Statistical methods and scientific inference. Hafner Publishing Co., 1956. Citado na página 50.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. **Journal of the American Statistical Association**, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937. Citado na página 50.

____. A comparison of alternative tests of significance for the problem of m rankings. **The Annals of Mathematical Statistics**, JSTOR, v. 11, n. 1, p. 86–92, 1940. Citado na página 50.

GAMA, J.; GABER, M. **Learning from data streams: processing techniques in sensor networks**. [S.l.]: Springer-Verlag Berlin Heidelberg, 2007. 244 p. Citado na página 29.

GAMA, J.; MEDAS, P.; CASTILLO, G.; RODRIGUES, P. Learning with drift detection. In: **SBIA**. [S.l.: s.n.], 2004. p. 286–295. Citado 2 vezes nas páginas 23 e 68.

GAMA, J.; RODRIGUES, P. P.; SPINOSA, E. J.; CARVALHO, A. C. P. L. F. de. **Knowledge discovery from data streams**. [S.l.]: Chapman & Hall/CRC Boca Raton, 2010. Citado 4 vezes nas páginas 23, 41, 44 e 47.

GARCÍA, V.; ALEJO, R.; SÁNCHEZ, J. S.; SOTOCA, J. M.; MOLLINEDA, R. A. Combined effects of class imbalance and class overlap on instance-based classification. In: **Intelligent Data Engineering and Automated Learning–IDEAL 2006**. [S.l.]: Springer, 2006. p. 371–378. Citado na página 65.

GILBERT, A. C.; GUHA, S.; INDYK, P.; KOTIDIS, Y.; MUTHUKRISHNAN, S.; STRAUSS, M. J. Fast, small-space algorithms for approximate histogram maintenance. In: **Proceedings of the thiry-fourth annual ACM symposium on Theory of computing**. New York, NY, USA: ACM, 2002. (STOC 02), p. 389–398. ISBN 1-58113-495-9. Disponível em: <http://doi.acm.org/10.1145/509907.509966>. Citado na página 30.

GUHA, S.; MISHRA, N.; MOTWANI, R.; O'CALLAGHAN, L. Clustering data streams. In: **FOCS**. [S.l.: s.n.], 2000. p. 359–366. Citado na página 23.

HAMMAMI, N.; BEDDA, M. Improved tree model for arabic speech recognition. In: **ICCSIT**. [S.l.: s.n.], 2010. v. 5, p. 521–526. Citado na página 90.

HAQUE, A.; KHAN, L.; BARON, M. Semi supervised adaptive framework for classifying evolving data stream. In: **PAKDD**. [S.l.: s.n.], 2015. p. 383–394. Citado na página 38.

HARRIES, M.; WALES, N. **Splice-2 comparative evaluation: Electricity pricing**. [S.l.], 1999. Citado na página 68.

HILL, D. J.; MINSKER, B. S. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. **Environmental Modelling & Software**, Elsevier, v. 25, n. 9, p. 1014–1022, 2010. Citado na página 23.

HOEFFDING, W. Probability inequalities for sums of bounded random variables. **Journal of the American statistical association**, Taylor & Francis Group, v. 58, n. 301, p. 13–30, 1963. Citado na página 43.

HULTEN, G.; SPENCER, L.; DOMINGOS, P. Mining time-changing data streams. In: **Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining**. New York, NY, USA: ACM, 2001. (KDD '01), p. 97–106. ISBN 1-58113-391-X. Disponível em: <http://doi.acm.org/10.1145/502512.502529>. Citado 2 vezes nas páginas 35 e 39.

IMAN, R. L.; DAVENPORT, J. M. Approximations of the critical region of the fbietkan statistic. **Communications in Statistics-Theory and Methods**, Taylor & Francis, v. 9, n. 6, p. 571–595, 1980. Citado na página 51.

KALUŽA, B.; MIRCHEVSKA, V.; DOVGAN, E.; LUŠTREK, M.; GAMS, M. An agent-based approach to care in independent living. In: **AmI**. [S.l.: s.n.], 2010. p. 177–186. Citado na página 90.

KELLY, M. G.; HAND, D. J.; ADAMS, N. M. The impact of changing populations on classifier performance. In: **Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining**. New York, NY, USA: ACM, 1999. (KDD '99), p. 367–371. ISBN 1-58113-143-7. Disponível em: <http://doi.acm.org/10.1145/312129.312285>. Citado na página 30.

KOLTER, J.; MALOOF, M. Using additive expert ensembles to cope with concept drift. In: ACM. **Proceedings of the 22nd international conference on Machine learning**. [S.l.], 2005. p. 449–456. Citado 2 vezes nas páginas 35 e 39.

KULL, M.; FLACH, P. Patterns of dataset shift. ECML PKDD LCME Workshop, 2014. Citado na página 30.

KUNCHEVA, L. I. *et al.* Nearest neighbour classifiers for streaming data with delayed labelling. In: IEEE. **ICDM**. [S.l.], 2008. p. 869–874. Citado na página 38.

LICHMAN, M. **UCI Machine Learning Repository**. 2013. <http://archive.ics.uci.edu/ml>. Citado na página 68.

MARRS, G. R.; HICKEY, R. J.; BLACK, M. M. The impact of latency on online classification learning with concept drift. In: **Knowledge Science, Engineering and Management**. [S.l.]: Springer, 2010. p. 459–469. Citado na página 23.

MASUD, M. M.; GAO, J.; KHAN, L.; HAN, J.; THURAISINGHAM, B. Classification and novel class detection in concept-drifting data streams under time constraints. **TKDE**, IEEE, v. 23, n. 6, p. 859–874, 2011. Citado 2 vezes nas páginas 36 e 38.

MASUD, M. M.; WOOLAM, C.; GAO, J.; KHAN, L.; HAN, J.; HAMLEN, K. W.; OZA, N. C. Facing the reality of data stream classification: coping with scarcity of labeled data. **KAIS**, v. 33, n. 1, p. 213–244, 2012. Citado na página 39.

MATIAS, Y.; VITTER, J.; WANG, M. Dynamic maintenance of wavelet-based histograms. In: **Proceedings of the 26th International Conference on Very Large Data Bases**. [S.l.: s.n.], 2000. p. 101–110. Citado na página 30.

MCNEMAR, Q. Note on the sampling error of the difference between correlated proportions or percentages. **Psychometrika**, Springer, v. 12, n. 2, p. 153–157, 1947. Citado na página 48.

MELLANBY, K. Humidity and insect metabolism. **Nature**, v. 138, p. 124–125, 1936. Citado na página 25.

MINKU, L.; WHITE, A.; YAO, X. The impact of diversity on online ensemble learning in the presence of concept drift. **Knowledge and Data Engineering, IEEE Transactions on**, IEEE, v. 22, n. 5, p. 730–742, 2010. Citado na página 32.

NEMENYI, P. Distribution-free multiple comparisons. In: INTERNATIONAL BIOMETRIC SOC 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210. **BIOMETRICS**. [S.l.], 1962. v. 18, n. 2, p. 263. Citado na página 51.

NOGUCHI, K.; GEL, Y. R.; BRUNNER, E.; KONIETSCHKE, F. nparld: an r software package for the nonparametric analysis of longitudinal data in factorial experiments. 2012. Citado na página 69.

OZA, N. C. Online bagging and boosting. In: IEEE. **SMC**. [S.l.], 2005. v. 3, p. 2340–2345. Citado na página 23.

PAGE, E. Continuous inspection schemes. **Biometrika**, JSTOR, v. 41, n. 1/2, p. 100–115, 1954. Citado na página 68.

POURKASHANI, M.; KANGAVARI, M. A cellular automata approach to detecting concept drift and dealing with noise. In: IEEE COMPUTER SOCIETY. **Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications**. [S.l.], 2008. p. 142–148. Citado na página 32.

POZZOLO, A. D.; BORACCHI, G.; CAELEN, O.; ALIPPI, C.; BONTEMPI, G. Credit card fraud detection and concept-drift adaptation with delayed supervised information. In: **IJCNN**. [S.l.: s.n.], 2015. Citado na página 39.

PRATI, R. C.; BATISTA, G. E.; MONARD, M. C. Class imbalances versus class overlapping: an analysis of a learning system behavior. In: **MICAI 2004: Advances in Artificial Intelligence**. [S.l.]: Springer, 2004. p. 312–321. Citado na página 65.

REIS, D.; FLACH, P.; MATWIN, S.; BATISTA, G. Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In: **Proceedings of the 22th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.]: ACM, 2016. v. 22. Citado 3 vezes nas páginas 26, 71 e 97.

REIS, D. dos. **Data Stream Classification Under Non-Null Verification Latency**. 2015. <http://sites.labic.icmc.usp.br/denismr/SDM16/>. Citado na página 68.

____. **Fast Unsupervised Online Drift Detection, Online Supplementary Material**. 2016. <https://github.com/denismr/incremental-ks>. Citado 2 vezes nas páginas 83 e 90.

SARNELLE, J.; SANCHEZ, A.; CAPO, R.; HAAS, J.; POLIKAR, R. Quantifying the limited and gradual concept drift assumption. In: **Proceedings of the International Joint Conference on Neural Networks**. [S.l.: s.n.], 2015. Citado na página 34.

SEIDEL, R.; ARAGON, C. R. Randomized search trees. **Algorithmica**, Springer, v. 16, n. 4-5, p. 464–497, 1996. Citado na página 75.

SHESKIN, D. J. Parametric and nonparametric statistical procedures. **Boca Raton: CRC**, 2000. Citado na página 51.

SOUZA, V.; SILVA, D.; GAMA, J.; BATISTA, G. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: **SDM**. [S.l.]: SIAM, 2014. p. 1–9. Citado 2 vezes nas páginas 35 e 48.

SOUZA, V. M.; SILVA, D. F.; GAMA, J. a.; BATISTA, G. E. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: **SDM**. [S.l.]: SIAM, 2015. p. 873–881. Citado 4 vezes nas páginas 36, 39, 68 e 91.

SOUZA, V. M. D.; SILVA, D. F.; BATISTA, G. E. *et al.* Classification of data streams applied to insect recognition: Initial results. In: IEEE. **BRACIS**. [S.l.], 2013. p. 76–81. Citado 2 vezes nas páginas 44 e 91.

TAYLOR, L. Analysis of the effect of temperature on insects in flight. **Journal of Animal Ecology**, v. 32, n. 1, p. 99–117, 1963. Citado na página 25.

TSYMBAL, A.; PECHENIZKIY, M.; CUNNINGHAM, P.; PUURONEN, S. Dynamic integration of classifiers for handling concept drift. **Information Fusion**, Elsevier, v. 9, n. 1, p. 56–68, 2008. Citado na página 32.

WIDMER, G.; KUBAT, M. Effective learning in dynamic environments by explicit context tracking. In: SPRINGER. **Machine Learning: ECML-93**. [S.l.], 1993. p. 227–243. Citado na página 30.

____. Learning in the presence of concept drift and hidden contexts. **Machine learning**, Springer, v. 23, n. 1, p. 69–101, 1996. Citado 2 vezes nas páginas 23 e 30.

WILCOXON, F. Individual comparisons by ranking methods. **Biometrics bulletin**, JSTOR, p. 80–83, 1945. Citado na página 49.

WILSON, D. L. Asymptotic properties of nearest neighbor rules using edited data. **SMC**, IEEE, n. 3, p. 408–421, 1972. Citado na página 66.

WU, X.; LI, P.; HU, X. Learning from concept drifting data streams with unlabeled data. **Neurocomputing**, Elsevier, v. 92, p. 145–155, 2012. Citado na página 39.

YANG, Y.; WU, X.; ZHU, X. Mining in anticipation for concept change: Proactive-reactive prediction in data streams. **Data mining and knowledge discovery**, Springer, v. 13, n. 3, p. 261–289, 2006. Citado na página 32.

ZAR, J. H. *et al.* **Biostatistical analysis**. [S.l.]: Pearson Education India, 1999. Citado na página 51.

ZHU, X. **Stream Data Mining Repository**. 2010. <http://www.cse.fau.edu/~xqzhu/stream.html>. Citado na página 68.

ZHU, X.; WU, X.; YANG, Y. Effective classification of noisy data streams with attribute-oriented dynamic classifier selection. **Knowledge and Information Systems**, Springer, v. 9, n. 3, p. 339–363, 2006. Citado 2 vezes nas páginas 35 e 39.

ŽLIOBAITĖ, I. **Learning under concept drift: an overview**. Lithuania, 2009. Citado 3 vezes nas páginas 31, 32 e 33.

____. Change with delayed labeling: when is it detectable? In: IEEE. **ICDMW**. [S.l.], 2010. p. 843–850. Citado 6 vezes nas páginas 26, 37, 38, 66, 88 e 90.

____. How good is the electricity benchmark for evaluating concept drift adaptation. **arXiv preprint arXiv:1301.3524**, 2013. Citado 2 vezes nas páginas 62 e 64.

ŽLIOBAITĖ, I.; BIFET, A.; READ, J.; PFAHRINGER, B.; HOLMES, G. Evaluation methods and decision theory for classification of streaming data with temporal dependence. **Machine Learning**, Springer, p. 1–28, 2014. Citado 4 vezes nas páginas 36, 52, 62 e 68.