Classificação de fluxos de dados não estacionários com algoritmos incrementais baseados no modelo de misturas gaussianas

Luan Soares Oliveira

Luan Soares Oliveira

Classificação de fluxos de dados não estacionários com algoritmos incrementais baseados no modelo de misturas gaussianas

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. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

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

USP – São Carlos
Setembro de 2015

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

Oliveira, Luan Soares
O48c Classificação de fluxos de dados não estacionários com algoritmos incrementais baseados no modelo de misturas gaussianas / Luan Soares Oliveira; orientador Gustavo Enrique de Almeida Prado Alves Batista. -- São Carlos, 2015.
84 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, 2015.

1. Aprendizado incremental. 2. Modelo de misturas gaussianas. 3. fluxo de dados. 4. Mudança de Conceito. I. Batista, Gustavo Enrique de Almeida Prado Alves, orient. II. Título.

Luan Soares Oliveira

Non-stationary data streams classification with incremental algorithms based on Gaussian mixture models

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. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

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

USP – São Carlos
September 2015

Este trabalho é dedicado principalmente à minha família, que sempre me apoiou e fez tudo que estava ao seu alcance para me ajudar quando necessário. Gostaria de dedicar especialmente à minha mãe, Silene, que é a pessoa que sempre busco orgulhar, para que eu possa devolvê-la pelo menos um pouco da felicidade que ela sempre me proporciona. Também quero dedicar à minha irmã, Raquel, cujos incentivos e elogios foram sempre importantes, sei que ela se orgulha muito de mim e quero que ela saiba que também me orgulho dela. Dedico também ao meu pai, Lourival, que sempre batalhou para que eu pudesse ter uma boa vida com condições de me dedicar aos estudos. Também gostaria de dedicar este trabalho aos meus tios, Gislene e Geraldo, que me receberam de braços abertos em sua casa durante minha graduação e possibilitaram que eu iniciasse um caminho que me levou até o presente momento.

AGRADECIMENTOS

Gostaria de agradecer à FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) pelo financiamento da pesquisa, à Universidade de São Paulo, por fornecer uma estrutura de excelência que com certeza fez com que este trabalho fosse desenvolvido da melhor forma possível e aos amigos que compartilharam experiências e conselhos. É muito importante também o agradecimento ao meu orientador, professor Gustavo E. A. P. A. Batista, pelo apoio durante toda a pesquisa, durante a qual ele esteve sempre disposto a ajudar, compartilhando seu conhecimento e apontando direções a serem seguidas, além de manter um caminho aberto para constante diálogo e troca de ideias.

“Qual o significado de algo perfeito? Nenhum! Eu tenho aversão a perfeição! Se algo for perfeito, não há nada acima disso. Não haveria espaço para a criação. Significa que não poderíamos usar a inteligência ou talento! Consegue entender? A perfeição é uma desgraça para todos nós.”

(涅 マユリ - Kurotsuchi Mayuri)

RESUMO

OLIVEIRA, L. S.. **Classificação de fluxos de dados não estacionários com algoritmos incrementais baseados no modelo de misturas gaussianas**. 2015. 84 f. Dissertação (Mestrado em em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos.

Aprender conceitos provenientes de fluxos de dados é uma tarefa significamente diferente do aprendizado tradicional em lote. No aprendizado em lote, existe uma premissa implicita que os conceitos a serem aprendidos são estáticos e não evoluem significamente com o tempo. Por outro lado, em fluxos de dados os conceitos a serem aprendidos podem evoluir ao longo do tempo. Esta evolução é chamada de mudança de conceito, e torna a criação de um conjunto fixo de treinamento inaplicável neste cenário. O aprendizado incremental é uma abordagem promissora para trabalhar com fluxos de dados. Contudo, na presença de mudanças de conceito, conceitos desatualizados podem causar erros na classificação de eventos. Apesar de alguns métodos incrementais baseados no modelo de misturas gaussianas terem sido propostos na literatura, nota-se que tais algoritmos não possuem uma política explicita de descarte de conceitos obsoletos. Nesse trabalho um novo algoritmo incremental para fluxos de dados com mudanças de conceito baseado no modelo de misturas gaussianas é proposto. O método proposto é comparado com vários algoritmos amplamente utilizados na literatura, e os resultados mostram que o algoritmo proposto é competitivo com os demais em vários cenários, superando-os em alguns casos.

Palavras-chave: Aprendizado incremental, Modelo de misturas gaussianas, fluxo de dados, Mudança de Conceito.

ABSTRACT

OLIVEIRA, L. S.. **Classificação de fluxos de dados não estacionários com algoritmos incrementais baseados no modelo de misturas gaussianas**. 2015. 84 f. Dissertação (Mestrado em em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos.

Learning concepts from data streams differs significantly from traditional batch learning. In batch learning there is an implicit assumption that the concept to be learned is static and does not evolve significantly over time. On the other hand, in data stream learning the concepts to be learned may evolve over time. This evolution is called concept drift, and makes the creation of a fixed training set be no longer applicable. Incremental learning paradigm is a promising approach for learning in a data stream setting. However, in the presence of concept drifts, outdated concepts can cause misclassifications. Several incremental Gaussian mixture models methods have been proposed in the literature, but these algorithms lack an explicit policy to discard outdated concepts. In this work, a new incremental algorithm for data stream with concept drifts based on Gaussian Mixture Models is proposed. The proposed method is compared to various algorithms widely used in the literature, and the results show that it is competitive with them in various scenarios, overcoming them in some cases.

Key-words: Incremental learning, Gaussian mixture model, Data stream, Concept drift.

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

Neste projeto de pesquisa está-se interessado em aprender conceitos provenientes de um fluxo de dados, como o descrito em Gama (2010)[1]. Essa forma de aprendizado diferencia-se significativamente do aprendizado de máquina tradicional, também conhecido como aprendizado em lote. No aprendizado em lote existe uma premissa implícita de que o conceito a ser aprendido é estático e, portanto, não evolui significativamente com o tempo. Dessa maneira é possível coletar um conjunto de dados e induzir um modelo que será aplicado em dados futuros, assumindo que os dados futuros possuem a mesma distribuição dos dados de treinamento.

Em aprendizado em fluxo de dados os conceitos a serem aprendidos podem evoluir com o tempo. Tal evolução é denominada *mudança de conceito* (ZLIOBAITE, 2009)[2]. Sendo assim, a criação de um conjunto de treinamento fixo não é mais aplicável. Na verdade, muitos fluxos de dados são potencialmente infinitos e, portanto, não se pode acumular todos os exemplos que ocorrem no fluxo em uma base de treinamento. Em geral, assume-se que em um fluxo de dados cada exemplo é visto uma única vez, e logo após descartado (GABER; ZASLAVSKY; KRISHNASWAMY, 2005).

O aprendizado incremental tem se mostrado uma abordagem promissora para o aprendizado em fluxo de dados (ZLIOBAITE, 2009; ENGEL; HEINEN, 2010; JÚNIOR, 2010; ELWELL; POLIKAR, 2011; BOUCHACHIA; VANARET, 2011). No aprendizado incremental, a hipótese de classificação é atualizada na presença de um novo evento, sem a necessidade de se rever todos os exemplos de treinamento anteriores. Dessa maneira, algoritmos incrementais são altamente eficientes em termos de uso de memória. Neste trabalho, é levantada a hipótese de que algoritmos incrementais também podem ser eficazes para aprender na presença de mudanças de conceitos. Intuitivamente, a hipótese é continuamente atualizada com novos exemplos que refletem o novo conceito a ser aprendido.

[1] Do inglês *data stream*.
[2] Do inglês *concept drift*.

Neste projeto de mestrado se está interessado em investigar métodos de classificação incrementais que sejam capazes de lidar com fluxo de dados na presença de mudanças de conceito. São investigadas abordagens incrementais como o modelo de misturas gaussianas incremental[3] (ENGEL; HEINEN, 2010), tendo como objetivo compará-las em termos de eficiência de atualização da hipótese e eficácia de classificação. Para isso são utilizados tanto bases de dados de *benchmark* reais quanto conjuntos de dados sintéticos que simulam uma evolução dos conceitos.

Existem diversas técnicas que visam atualizar o modelo de misturas gaussianas[4] de forma incremental com intuito de aplicá-lo em fluxos de dados. Contudo, boa parte dessas abordagens não levam em considerações cenários onde existem mudanças de conceito ao longo destes fluxos (SONG; WANG, 2005; ENGEL; HEINEN, 2010; KRISTAN; SKO ČAJ; LEONARDIS, 2010). Sendo assim, neste projeto são propostas alterações no modelo de misturas gaussianas incremental (ENGEL; HEINEN, 2010) com o intuito de permitir que este possa se adaptar às mudanças de conceito presentes ao longo do tempo em cenários representados por diversas bases de dados. Tais alterações consistem na aplicação de técnicas de esquecimento que permitem a exclusão de conceitos antigos e potencialmente desatualizados para a incorporação de conceitos novos e potencialmente mais relevantes.

O método proposto é comparado com diversos algoritmos amplamente utilizados no cenário de fluxos de dados e presentes na plataforma *Massive Online Analysis* (MOA)[5], tais como *Naive Bayes*, *Hoeffling Adaptive Tree* (HULTEN; SPENCER; DOMINGOS, 2001; BIFET; GAVALDÀ, 2009; BIFET *et al.*, 2011), Active Classifier (ŽLIOBAITĖ *et al.*, 2011; CESA-BIANCHI; GENTILE; ZANIBONI, 2006), *Drift Detection Method*, *Perceptron* e o *Accuracy Updated Ensemble* (BRZEZIŃSKI; STEFANOWSKI, 2011). Os resultados mostram que o algoritmo proposto é competitivo com os demais em diversos cenários, chegando a superar alguns de seus concorrentes em vários deles.

1.1 Hipóteses e objetivos

O objetivo do trabalho é propor e avaliar um algoritmo de aprendizado incremental baseado no modelo de misturas gaussianas, capaz de aprender na presença de mudanças de conceitos. Neste projeto, levanta-se a hipótese de que uma abordagem local para atualização das componentes e classificação dos exemplos neste cenário pode produzir melhores resultados em termos de acurácia. Tal hipótese se baseia na ideia de que exemplos que representam novos conceitos devem ter pequena ou nenhuma influência em componentes desatualizadas, agilizando a criação de novas componentes representando estes novos conceitos.

Além disso pretende-se fazer uma análise inicial do modelo de misturas gaussianas tradicional sob um cenário conhecido como latência de verificação extrema (KREMPL, 2011;

[3] do inglês IGMM - *Incremental Gaussian Mixture Model*
[4] do inglês GMM - *Gaussian Mixture Model*
[5] http://moa.cms.waikato.ac.nz/

DYER; CAPO; POLIKAR, 2014; SOUZA; GAMA; BATISTA, 2014), no qual os rótulos corretos dos eventos não estão disponíveis para os algoritmos após a fase de treinamento.

Espera-se que o método proposto seja competitivo com os algoritmos presentes na literatura e disponíveis na plataforma MOA tanto em termos de acurácia quanto em termos de custo computacional.

Já no cenário de latência de verificação extrema, pretende-se avaliar se o algoritmo GMM é capaz de manter uma boa acurácia mesmo sem possuir um retorno a respeito de suas classificações. Em outras palavras, pretende-se verificar se os rótulos inferidos pelo próprio algoritmo são suficientes para retreiná-lo e adaptá-lo às mudanças de conceito.

1.2 Propostas e contribuições

Como já mencionado, existem diversas abordagens que buscam atualizar os modelos de misturas gaussianas de forma incremental. Isso pode ser atribuido à adequabilidade das técnicas incrementais em cenários de fluxos contínuos de dados, visto que estas técnicas tem como característica intrínseca a atualização de seus modelos ao longo do tempo. Contudo, boa parte dos algoritmos incrementais derivados do GMM não são projetados para lidarem com mudanças de conceito. Sendo assim, o método proposto neste projeto busca adaptar uma versão incremental do GMM (ENGEL; HEINEN, 2010) para que esta possa lidar com cenários onde mudanças de conceitos estão presentes ao longo do fluxo de dados.

O método proposto difere-se do algoritmo proposto por Engel e Heinen (2010) devido a inserção de técnicas de esquecimento que visam manter no modelo do algoritmo apenas os conceitos mais relevantes, permitindo-o se adaptar a possíveis mudanças de conceito. Além disso, também existem diferenças no método proposto se comparado a outros (ARANDJELOVIC; CIPOLLA, 2006; KRISTAN; SKOČAJ; LEONARDIS, 2010), já que o modelo proposto usa uma atualização do modelo que pode ser chamada de local enquanto os demais modelos utilizam uma atualização global. De forma simplificada, a atualização local modifica apenas a componente gaussiana mais próxima daquela que recebeu o último evento observado pelo algoritmo, enquanto a atualização global atualiza todas as componentes do modelo. Mais detalhes sobre o método proposto são mostrados no Capítulo 4.

Em relação ao cenário de latência de verificação extrema, foi proposto o uso de uma estratégia em mini-lote para treinamento e atualização do modelo do GMM ao longo do tempo, juntamente com a fixação dos parâmetros da matriz de covariância e probabilidades a priori após a fase de treinamento.

Os resultados obtidos com os classificadores incrementais são comparados com outros classificadores no contexto de fluxo de dados. Os dados utilizados nas comparações são tanto bases de dados de *benchmark* reais quanto conjuntos de dados sintéticos que simulam uma

evolução dos conceitos. Nas avaliações comparativas é utilizado o sistema MOA, uma vez que esse sistema implementa um grande número de classificadores para fluxo de dados.

Os resultados obtidos mostram que existe um grande ganho, tanto em termos de tempo de execução quanto em termos de acurácia, ao se adicionar a capacidade de esquecimento ao algoritmo IGMM. Além disso, a abordagem de atualização local também mostrou-se promissora, permitindo que o algoritmo obtenha resultados bons em bases de dados onde o mesmo algoritmo com atualização global possui um desempenho fraco.

Em relação ao cenário de latência de verificação extrema, a abordagem proposta se mostrou eficaz nas bases de dados utilizadas para teste, exceto em casos onde as classes eram descritas por um número de componentes gaussianas maior do que o informado inicialmente para o algoritmo. Apesar de promissora, esta abordagem ainda necessita ser melhor estudada, visto que foi testada apenas em bases de dados sintéticas.

1.3 Organização do texto

Esta dissertação está organizado da seguinte maneira: No Capítulo 2 é apresentado o cenário de classificação de fluxos de dados não estacionários, mostrando algumas das abordagens existentes na literatura bem como características e dificuldades encontradas neste tipo de problema. No Capítulo 3 é mostrada a abordagem de aprendizado incremental, suas motivações e características. Já o Capítulo 4 apresenta uma descrição de algumas das técnicas que foram exploradas no decorrer da pesquisa com intuito de melhorar os resultados de classificação, evidenciando os algoritmos existentes e as alterações propostas. No Capítulo 5, são discutidas as formas de avaliação às quais os algoritmos foram submetidos, com o intuito de compará-los e avaliar a aplicabilidade das técnicas utilizadas. Além disso, são apresentados as bases de dados utilizadas durante o processo de avaliação. Também são mostrados os resultados obtidos pelas técnicas avaliadas no projeto, além de discussões a respeito destes. Por fim, o Capítulo 6 apresenta as conclusões do projeto e algumas possibilidades de continuação para este trabalho. Existe também o Apêndice A onde são apresentados os resultados da avaliação feita no cenário de latência de verificação extrema.

CAPÍTULO

2

CLASSIFICAÇÃO EM FLUXOS CONTÍNUOS DE DADOS NÃO ESTACIONÁRIOS

2.1 Considerações iniciais

Há muitos anos os bancos de dados tradicionais, tais como os relacionais, vêm sendo utilizados em aplicações que exigem uma forma de armazenamento de dados que permita, entre outras funções, a recuperação rápida de dados. Apesar dos ótimos resultados obtidos com este paradigma, na última década começaram a surgir necessidades de novas formas de processamento e armazenamento de dados, requeridas por novos tipos de aplicações. Isso se deve ao fato de que os dados de algumas aplicações são gerados naturalmente na forma de um fluxo de dados, ou seja, uma sequência (*stream*) de valores potencialmente infinita (GOLAB; OZSU, 2003). Fluxos de dados ocorrem desde aplicações que monitoram tráfego na internet até dados provenientes de sensores. Por tais motivos, análises em tempo real de fluxos de dados vem se tornando uma área chave de pesquisa em mineração de dados, assim como as aplicações que as utilizam (READ*et al.*, 2012; BIFET *et al.*, 2013). Tal perspectiva é uma das motivações da utilização de técnicas de classificação em fluxos de dados no presente projeto.

Um fluxo de dados é uma sequência contínua de itens, que são apresentados em tempo real segundo a ordem de chegada, sobre a qual não se tem controle (GOLAB; OZSU, 2003). Existem algumas diferenças fundamentais entre as aplicações cujo processamento se dá de forma *offline* e aquelas que trabalham com fluxo de dados. No primeiro caso, os dados ficam disponíveis integralmente para a aplicação, que pode acessá-los a qualquer momento. Vários algoritmos de classificação trabalham desta forma, tendo seus treinamentos realizados com toda a base de dados, sendo possível visitar várias vezes os mesmos dados. Além disso, esses classificadores geram um modelo estático, que geralmente não é mais revisado.

Já no segundo caso, cada exemplo é processado de forma individual (ou em pequenos

grupos), já que os dados chegam de forma contínua e não estão todos disponíveis logo de início. Além disso, tipicamente não é possível rever dados antigos após estes serem processados, já que são armazenados apenas sumários e/ou modelos dos dados já vistos, e não os dados em si. Sendo assim, é preciso que o modelo gerado seja atualizado de forma incremental a cada novo exemplo (ou grupo de exemplos) que lhe é apresentado (GOMES, 2012). Outra característica importante é a necessidade de estar sempre preparado para receber um novo exemplo, visto que o intervalo entre a chegada dos dados do fluxo pode não ser constante. É importante ressaltar que fluxos de dados são potencialmente infinitos, já que os exemplos alvo de análise tendem a ocorrer indefinidamente. Tal característica traz outra exigência para os algoritmos que trabalham sob essa abordagem: a necessidade de economia de recursos, tanto de memória quanto de processamento (BIFET *et al.*, 2013). Desta forma, uma aplicação que lide com fluxos de dados deve manter o custo de processamento de um exemplo aproximadamente constante, independente do número de exemplos que já tenham sido vistos anteriormente (GHOLIPOUR; HOSSEINI; BEIGY, 2013).

Neste capítulo são discutidos os conceitos referentes à classificação em fluxos de dados relevantes para a presente pesquisa. Inicialmente, na Seção 2.2 é feita uma discussão dos tipos de mudanças de conceito que ocorrem em fluxos de dados conhecidos como não estacionários. Já a Seção 2.3 apresenta uma descrição das abordagens existentes para se lidar com as mudanças de conceito, como o uso de janelas. Por fim, a Seção 2.4 apresenta as formas utilizadas na literatura para se avaliar problemas de classificação em fluxos de dados.

2.2 Mudanças de conceito

Ao se lidar com aplicações reais é provável que alguns comportamentos dos dados variem com o passar do tempo. O maior problema é que, muitas vezes, tais mudanças são provocadas por exemplos desconhecidos pelos algoritmos (*hidden context*) (TSYMBAL, 2004), o que dificulta a percepção e adequação dos mesmos. Pensando-se em classificadores, um conceito pode ser entendido como uma função que mapeia atributos para uma determinada classe (GOMES, 2012). Desta forma, pode-se citar como exemplos aplicações de previsão do tempo, cujas predições podem variar bastante de acordo com a estação do ano, ou modelos de preferência de compra de clientes, que variam de acordo com índices econômicos. Este tipo de mudança de comportamento, conhecida como mudança de conceito (*concept drift*), dificulta a tarefa de aprendizagem (TSYMBAL, 2004), tornando-se um dos maiores desafios quando se trata da classificação em fluxos de dados (GOMES, 2012; GHOLIPOUR; HOSSEINI; BEIGY, 2013).

Sendo todas as instâncias x_t de um fluxo de dados geradas por uma distribuição S_t, se todas as instâncias são geradas a partir da mesma distribuição ($S_1 = S_2 = ... = S_{t+1} = S$) diz-se que o conceito é estável. Contudo, se existe i e j tal que $S_i \neq S_j$ então diz-se que existe uma mudança de conceito. Segundo Zliobaite (2009), as mudanças de conceito podem acontecer

devido a três fatores. Para discutir as formas de mudança de conceito é necessário primeiro ter em mente uma definição formal do que é um problema de classificação. Segundo Narasimhamurthy e Kuncheva (2007), Zliobaite (2009) independentemente da presença ou ausência de mudanças de conceito, um problema de classificação pode ser definido da seguinte forma: Sendo $X \in \mathbb{R}^p$ uma instância em um espaço p-dimensional de atributos e $X \in c_i$ onde $c_1, c_2, ..., c_k$ é o conjunto de classes, o classificador ótimo para classificar $X \rightarrow c_i$ é determinado pelas probabilidades a priori das classes $P(c_i)$ e a função de densidade de probabilidade condicionada às classes $p(X|c_i), i = 1, ..., k$. Sendo assim, um **conceito**[1] pode ser definido como um conjunto de probabilidades a priori e condicionais das classes, como mostra a Equação 2.1.

$$S = \{(P(c_1), P(X|c_1)), (P(c_2), P(X|c_2)), ..., (P(c_k), P(X|c_k))\}. \tag{2.1}$$

Segundo a teoria bayesiana (DUDA; HART; STORK, 2000; ZLIOBAITE, 2009) a classificação de uma instância X baseada na máxima probabilidade a posteriori, que para uma dada classe c_i é representada pela Equação 2.2, onde $p(X)$ é constante para todas as classes c_i.

$$p(c_i|X) = \frac{p(c_i)p(X|c_i)}{p(X)}. \tag{2.2}$$

Sendo assim, segundo Kelly, Hand e Adams (1999), as mudanças de conceito podem ocorrer devido a:

1. Probabilidade a priori das classes $P(c)$ pode mudar ao longo do tempo.
2. A distribuição de uma ou mais classes $p(X|c)$ pode mudar.
3. As distribuições a posteriori das classes $p(c|X)$ pode mudar.

É possível que $p(X|c)$ se altere sem afetar $p(c|X)$, como no caso de uma movimentação simétrica para direções opostas (ZLIOBAITE, 2009). Um exemplo disso é visto na classificação de *spams* em *emails*. Mudanças ocorrem devido ao fato de que, apesar do conceito de mensagem indesejável se manter constante durante um longo período de tempo, a frequência de ocorrência de diferentes tipos de *spam* pode variar drasticamente com o passar do tempo (TSYMBAL, 2004). Mudanças em $p(X|c)$ são conhecidas como mudança de conceito virtual[2] enquanto mudanças em $p(c|X)$ são chamadas de mudanças reais[3] (WIDMER; KUBAT, 1993). Contudo saber se um problema possui mudanças reais ou virtuais não é de grande importância, visto que $p(c|X)$ depende de $p(X|c)$ (ZLIOBAITE, 2009).

Segundo Gomes (2012), devido a natureza desconhecida dos conceitos que geram as mudanças de conceito, suas causas não podem ser determinadas e/ou previstas pelos algoritmos

[1] do inglês *concept*, também chamado de *data source*
[2] Do inglês *virtual concept drift*
[3] Do inglês **real concept drift**

de aprendizagem. Sendo assim, um algoritmo de aprendizagem deve ser capaz de detectar quando alguma mudança ocorre e se adaptar a ela o mais rápido possível. Contudo, identificar a necessidade de uma adaptação do modelo não é uma tarefa trivial. Isso se deve ao fato de que, em muitos casos, é complicado distinguir um simples ruído de uma verdadeira mudança de conceito nos dados. Tal dificuldade acaba tornando algoritmos que se adequam rapidamente às mudanças muito sensíveis a ruídos e, na outra extremidade, algoritmos robustos a ruídos acabam sendo lentos para adaptar-se às mudanças (TSYMBAL, 2004). Tendo isso em vista, um algoritmo ideal deve ser robusto em relação aos ruídos e, simultaneamente, capaz de se adequar rapidamente às mudanças de conceito (WIDMER; KUBAT, 1996).

Na literatura, mudanças de conceito podem ocorrer de diversas formas em um fluxo de dados: de forma abrupta, gradual, incremental ou recorrente (GOMES, 2012; TSYMBAL, 2004; ZLIOBAITE, 2009; JÚNIOR, 2010). A Figura 1 apresenta uma representação gráfica dos tipos de mudança de conceito, evidenciando em (a) e (b) a diferença na quantidade de exemplos necessários até que as mudanças abrupta e gradual ocorram, respectivamente.

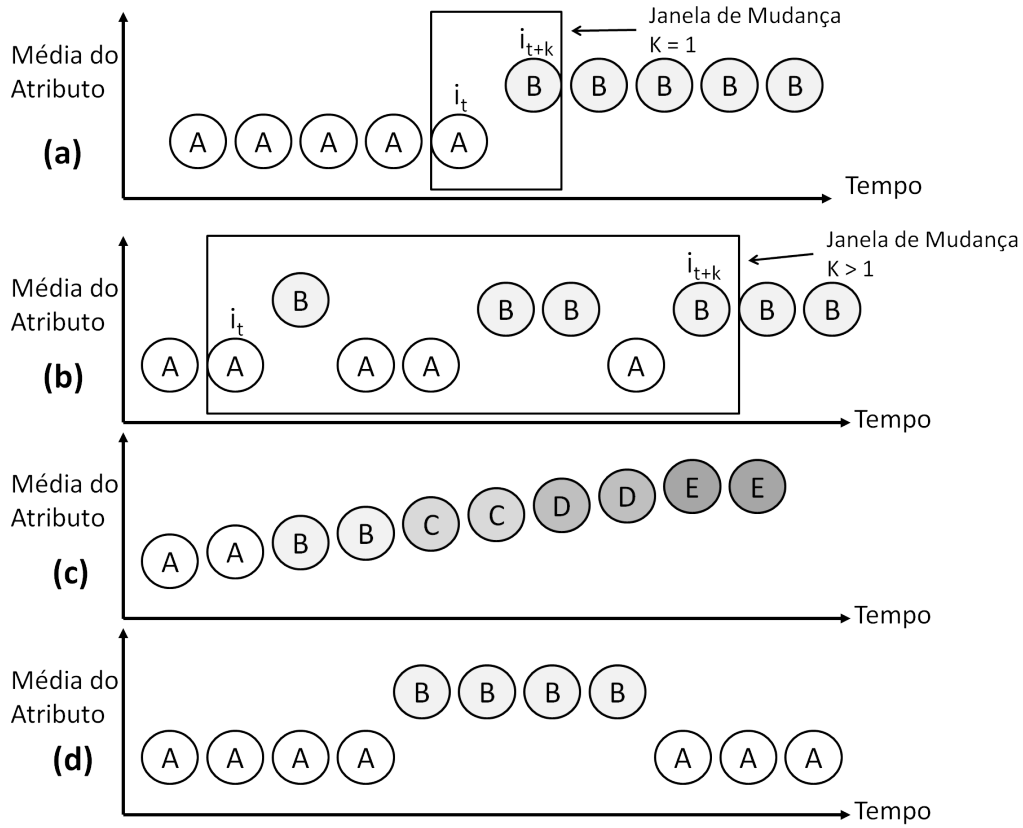


Figura 1 – Tipos de mudanças de conceito. (a) Mudança de conceito abrupta (b) Mudança de conceito gradual (c) Mudança de conceito incremental (d) Mudança de conceito recorrente (ZLIOBAITE, 2009)

A diferença entre os dois primeiros tipos de mudança de conceito está na velocidade em que um conceito deixa de ser válido, e dá lugar a um novo conceito estável. Se supormos uma mudança de conceito que se inicia na instância i_t e se torne estável na instância i_{t+k}, se $k \approx 1$ então a mudança é considerada abrupta. Caso $k \gg 1$, então diz-se que a mudança é gradual (GOMES,

2012). Em outras palavras, uma mudança abrupta ocorre quando um conceito conhecido deixa de ser válido assim que um novo surge, enquanto na mudança gradual a frequência de ocorrência do conceito antigo reduz gradativamente até que deixe de aparecer, dando lugar ao novo conceito. Já a chamada mudança de conceito incremental, pode ser entendida como um tipo de mudança gradual que envolve mais de dois conceitos, ocorrendo quando os conceitos que descrevem uma classe variam lentamente com o passar do tempo, de forma que a diferença no comportamento dos exemplos só é perceptível a longo prazo (ZLIOBAITE, 2009). Outra forma de ocorrência de mudanças, conhecida como recorrente, ocorre quando um conceito anteriormente ativo volta a ser válido depois de um determinado período de tempo (JÚNIOR, 2010). É importante ressaltar que as mudanças recorrentes são diferentes de mudanças periódicas, visto que no segundo caso os períodos em que os conceitos se tornam válidos são conhecidos.

2.3 Aprendizado com mudanças de conceito

De modo geral, existem duas formas de se lidar com as mudanças de conceito: *i*) abordagens que adaptam o modelo de classificação em intervalos regulares, sem considerar se a mudança realmente ocorreu e *ii*) abordagens que procuram detectar a ocorrência de uma mudança de conceito para só então adaptar o modelo (SEBASTIÃO; GAMA, 2009). Estas duas abordagens são discutidas mais detalhadamente nas seções a seguir.

2.3.1 Adaptação em intervalos regulares

A primeira abordagem para se lidar com mudanças de conceito busca retreinar o modelo dando prioridade àqueles exemplos considerados melhores sob algum critério (como o tempo de ocorrência), através da utilização de janelas de tempo ou aplicação de pesos sobre os dados. Ao se utilizar janelas, o modelo de classificação é induzido a cada determinado intervalo de tempo apenas com os exemplos contidos na janela. A motivação para essa técnica parte do pressuposto de que dados mais recentes são mais importantes, sendo suas informações mais representativas. Desta forma os dados mais antigos não precisam ser observados e, portanto, podem ser descartados, já que há a possibilidade destes terem se tornado desatualizados com o passar do tempo. Além disso, devido ao caráter ilimitado dos fluxos de dados, sua análise por completo é inviável, o que justifica a existência de critérios de descarte. Contudo, existem algumas dificuldades intrínsecas a essa abordagem. A principal delas é a escolha do tamanho ideal da janela, já que uma janela pequena pode agilizar a adaptação do classificador a fases mudanças, mas pode prejudicar seu desempenho em fases de maior estabilidade. Além disso, assim como já citado, existe a possibilidade da presença de ruídos nos dados, e uma janela pequena torna o classificador mais sensível a estes. Por outro lado, uma janela muito grande pode melhorar o desempenho do classificador em fases estáveis, mas acaba tornando a adaptação a mudanças de conceito mais lenta.

Diversos tipos de janelas podem ser encontrados na literatura, tais como as janelas deslizantes (*sliding window*) e as janelas por marcação (*landmark window*) (GAMA, 2010). O primeiro tipo, pode ser utilizado sob duas abordagens, sendo a mais simples definir-se um tamanho w fixo em termos de número de observações. Em outras palavras, essa estrutura acumula elementos até que o tamanho w seja atingido e, a partir dai, a cada novo exemplo armazenado descarta-se o dado mais antigo, de modo análogo ao que acontece na estrutura $FIFO$ (*First In First Out*). Outra possível abordagem é o armazenamento de observações baseado em tempo ao invés de quantidade. Nessa abordagem a janela não possui quantidade de elementos fixa, de forma que os exemplos inseridos nesta só são descartados após estarem fora do intervalo de tempo coberto pela janela. Desta forma, uma janela que cobre os exemplos das últimas duas horas, por exemplo, descartaria qualquer dado que tenha ocorrido a mais de duas horas.

Já nas janelas por marcação, um ponto considerado importante é marcado como referência (*landmark*), e todos os pontos do fluxo posteriores a este estarão contidos na janela, de forma que esta aumente de tamanho com o passar do tempo (GEHRKE; KORN; SRIVASTAVA, 2001). Exemplos das janelas deslizantes e de marcação são encontrados na Figura 2.

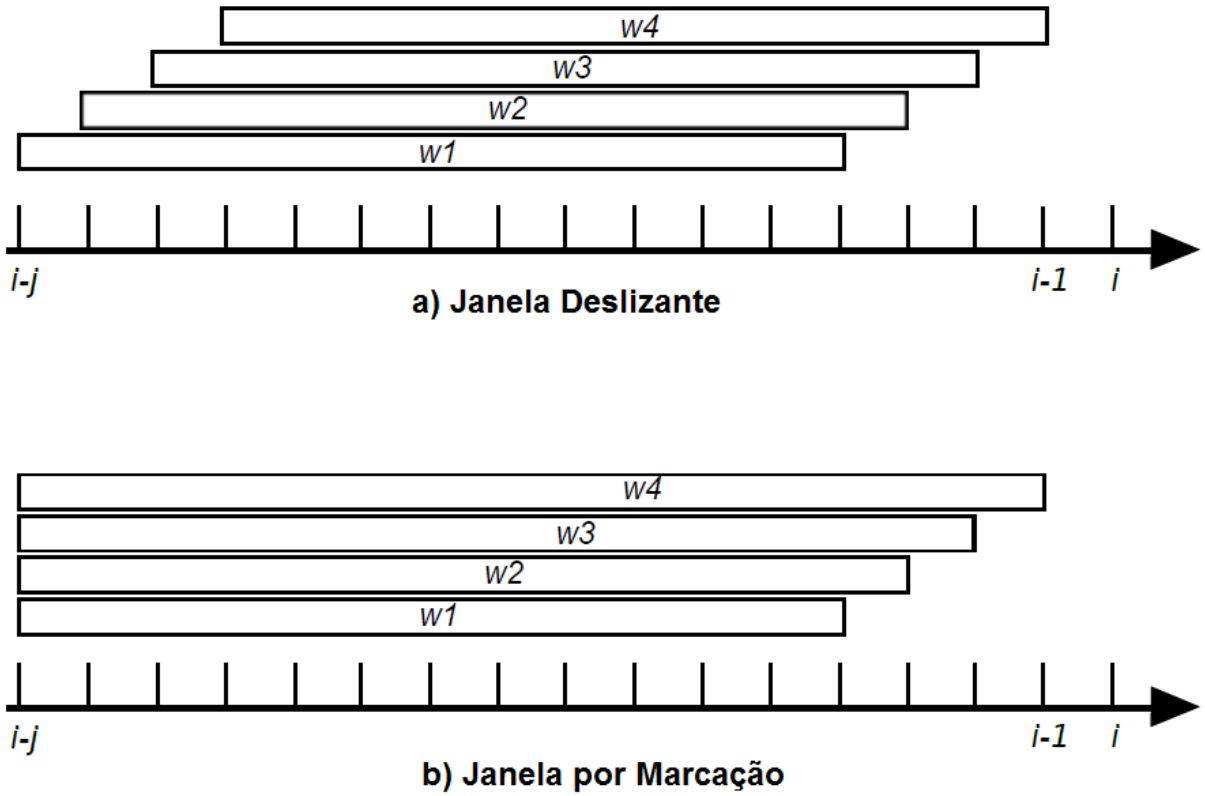


Figura 2 – Exemplo de janela deslizante *(a)* e janela por marcação *(b)* em quatro períodos diferentes de tempo (GAMA, 2010))

Partindo da mesma ideia central das janelas, que é dar relevância aos exemplos mais recentemente observados no fluxo de dados, existem métodos que atribuem pesos aos exemplos, de forma que o peso de um exemplo diminua com o passar do tempo. Um ponto de contraste com a abordagem utilizando janelas é que com o uso de pesos um exemplo não é descartado

abruptamente, e sim perde sua importância gradualmente com o passar do tempo, a medida em que se torna antigo e potencialmente desatualizado.

2.3.2 Adaptação com detecção de mudanças

Já nos métodos que buscam detectar a ocorrência de uma mudança de conceito antes de atualizar o modelo de classificação, uma abordagem comum é o monitoramento de determinados indicadores ao longo do tempo, de modo que o classificador seja atualizado quando estes indicadores atinjam um certo limiar. Podem ser utilizados como indicadores medidas de desempenho (GAMA *et al.*, 2004; BAENA-GARCÍA *et al.*, 2006; NISHIDA; YAMAUCHI, 2007), como a acurácia, análises sobre os próprios dados brutos (BIFET; GAVALDA, 2007; PATIST, 2007), ou mesmo os parâmetros do próprio algoritmo utilizado (SU; SHEN; XU, 2008). Dries e Rückert (2009) desenvolveu métodos de detecção de mudanças de conceito baseados nas três estratégias. Uma vantagem interessante deste tipo de abordagem é que obtém-se naturalmente informações significativas sobre o problema em questão, como os pontos em que ocorreram as mudanças de conceito.

De modo geral, métodos baseados na detecção da mudança de conceito baseados no monitoramento do desempenho do classificador assumem que em processos estacionários o erro deste classificador mantém-se aproximadamente constante ao longo do tempo. Por outro lado, é provável que um classificador não atualizado apresente uma queda de desempenho acentuada em problemas não estacionários quando ocorre uma mudança de conceito, o que indica a necessidade de atualização de seu modelo de classificação. Contudo, existem alguns problemas associados a este tipo de método, sendo um dos principais a diferenciação entre mudanças reais de conceito e algum tipo de ruído presente nos dados. Outra dificuldade, que possui impacto em algumas aplicações, é a de obter os rótulos corretos necessários para o monitoramento dos indicadores de erro.

Segundo Gama (2010) uma forma de detectar mudanças de conceito através do monitoramento do desempenho do algoritmo é a criação de três estados de alerta de acordo com os dados observados no fluxo. Supondo uma sequência de exemplos x_i, y_i, onde x_i representa os valores dos atributos e y_i representa a classe. Para cada exemplo o algoritmo infere $\hat{y}_i$ que pode ser uma predição correta ($\hat{y}_i = y_i$) ou errada ($\hat{y}_i \neq y_i$). Para um conjunto de exemplos o erro do algoritmo é uma variável aleatória de uma distribuição de Bernoulli. A distribuição binomial fornece a forma geral da probabilidade para a variável aleatória que representa o número de erros em um conjunto de n exemplos. Para cada ponto i na sequência, a taxa de erro é a probabilidade de se observar uma predição errada, p_i, com desvio padrão dado por $s_i = \sqrt{(p_i(1-p_i))/i}$. O método de detecção de mudanças de conceito trabalha com dois registros durante a fase de treinamento do classificador, p_{min} e s_{min}. Assim, para cada novo exemplo processado i se $p_i + s_i < p_{min} + s_{min}$ esses registros são atualizados.

Para um número suficientemente grande de exemplos, a distribuição binomial pode

ser aproximada por uma distribuição normal com a mesma média e variância. Considerando que a distribuição de probabilidade não se altera em um contexto estático, então o intervalo de confiança $1-\alpha/2$ para p com $n > 30$ exemplos é aproximadamente $p \pm z \times s_i$, sendo o parâmetro z dependente do nível de confiança desejado.

Supondo-se que em um fluxo de dados existe um exemplo j com os respectivos valores de p_j e s_j, e levando-se em consideração os parâmetros α e β, são definidos três possíveis estados para o classificador, sendo eles:

1. Sob controle: Enquanto $p_j + s_j < p_{min} + \beta \times s_{min}$. O erro se encontra estável. Sendo assim o exemplo j é gerado a partir da mesma distribuição dos exemplos passados.

2. Fora de controle: Sempre que $p_j + s_j > p_{min} + \alpha \times s_{min}$. O erro está aumentando e atingiu um patamar significativamente maior do que os exemplos recentes. O exemplo atual pode ter sido gerado por outra distribuição com uma probabilidade de $1-\alpha/2$. Ao atingir este estado conclui-se que houve uma mudança de conceito e o classificador deve ser atualizado utilizando os exemplos que estavam sendo armazenados em um *buffer* durante o estado *Em alerta*.

3. Em alerta: Em quaisquer situações intermediárias aos dois estados anteriores. O erro está aumentando, mas ainda está em um patamar baixo. O aumento no erro pode ter sido causado por ruídos nos dados e não uma mudança de conceito. São necessários mais exemplos para que se tenha uma conclusão e possivelmente uma ação. Quando o algoritmo está nesta fase, começa-se a armazenar os exemplos observados em uma espécie de *buffer*.

A Figura 3 mostra as possíveis transições entre esses estados.

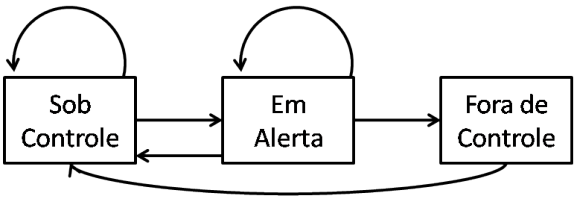


Figura 3 – Possíves transições entre os estados.

2.4 Avaliação em fluxos de dados

Para se avaliar um classificador em um cenário de fluxo de dados existem duas alternativas viáveis apresentadas na literatura (GAMA, 2010). A primeira opção é preparar um conjunto independete de teste onde o classificador será aplicado em determinados intervalos de tempo ou conjunto de exemplos. Esta abordagem possibilita uma estimativa não viesada da performance do algoritmo.

Já a segunda alternativa consiste em computar a performance do classificador a partir de uma sequência de exemplos[4]. Neste cenário uma medida de desempenho é calculada para cada exemplo processado pelo classificador. O *prequential-error S* é computado através da soma da função de perda L entre o que foi predito pelo classificador e o valor real observado, como mostra a Equação 2.3.

$$S = \sum_{i=1}^{n} L(y_i, \hat{y}_i). \tag{2.3}$$

A segunda abordagem resulta em uma curva que monitora a evolução do aprendizado como um processo. Também é possível obter uma curva semelhante usando um conjunto de treinamento independente, aplicando-se o modelo atual ao conjunto de teste em intervalos regulares. Nas duas metodologias o resultado pode ser afetado pela ordem na qual os exemplos são apresentados ao algoritmo. Além disso, o *Prequential error* tende a ser uma abordagem pessimista, isto é, sob as mesmas condições ele apresenta um erro um pouco maior do que outras formas de medição de desempenho (GAMA, 2010). Um ponto fraco desta técnica é que a estimativa final pode ser muito influenciada pelas análises iniciais do fluxo, quando poucos exemplos foram avaliados. Sendo assim, esta estimativa pode ser feita utilizando mecanismos de esquecimento, como pesos ou janelas, para aumentar a importância de exemplos mais recentes no cálculo da estimativa.

Usando pesos, o *Prequential Error* pode ser calculado como $E_i = \frac{S_i}{N}$, onde $S_1 = L_1$ e $S_i = L_i + \alpha \times S_{i-1}$. Contudo, neste cenário E_i acaba convergindo para 0 quando $N \to +\infty$, o que obriga a utilização de um fator de correção como mostra a Equação 2.4. Nesta equação n_i é o número de exemplos utilizados para calcular L_i. Isto é, quando L_i é calculado para cada exemplo apresentado ao classificador $n_i = 1$.

$$E_i = \frac{S_i}{B_i} = \frac{L_i + \alpha \times S_{i-1}}{n_i + \alpha \times B_{i-1}}. \tag{2.4}$$

Já o uso de janelas de tamanho k faz com que o cálculo seja limitado em um conjunto menor de exemplos (geralmente os mais recentes) como mostra a Equação 2.5.

$$S = \sum_{i=t}^{t+k} L(y_i, \hat{y}_i). \tag{2.5}$$

O cálculo do *Prequential error* utilizando-se tanto janelas quanto pesos acaba convergindo para o cálculo desta estimativa no cenário onde um conjunto de testes independente é utilizado (GAMA, 2010).

[4] *Prequential* (DAWID, 1984; GAMA, 2010)

2.5 Considerações finais

O cenário de fluxo de dados está presente atualmente em diversas aplicações práticas. Este cenário representa um ramo relativamente novo na computação, visto que o crescimento das aplicações cujos dados são derados neste formato só foi possível a partir de meados da década de 90. Além de características únicas que o difere do aprendizado tradicional em lote, como a incerteza dos intervalos de tempo entre a apresentação de dois exemplos subsequentes, o cenário de fluxos contínuos de dados foco deste projeto apresenta uma característica que acrescenta dificuldade em seu tratamento: as mudanças de conceito.

Para trabalhar com fluxos contínuos de dados com mudanças de conceitos, também conhecidos como não estacionários, este projeto utiliza o aprendizado incremental, que será melhor descrito no Capítulo 3.

CAPÍTULO

3

APRENDIZADO INCREMENTAL

3.1 Considerações iniciais

Neste capítulo são apresentadas as principais características presentes no aprendizado incremental, com o intuito de diferencia-lo de abordagens convencionais tais como o aprendizado em lote.

3.2 O aprendizado incremental

A tendência atual para fontes de dados dinâmicas é clara, tanto na área acadêmica quanto para aplicações reais (READ *et al.*, 2012). Tais fontes de dados não são apenas dinâmicas como também geram dados em alta velocidade em tempo real. Exemplo disso são aplicações que envolvem agentes inteligentes, monitoramento de redes, exploração de redes sociais, entre outras, que frequentemente atuam em ambientes dinâmicos, isto é, ambientes que se modificam com o passar do tempo. Por mais que tal fato possa parecer natural, muitas vezes é preciso levar em consideração que o processo de aprendizagem ocorre durante o tempo. Um importante problema presente no processo de mineração de dados é a presença de novos dados em contínua evolução, sendo essencial que as técnicas de classificação e agrupamnto tratem esses dados de alguma forma (JOSHI; KULKARNI, 2012).

Na literatura são encontradas duas abordagens predominantes para se lidar com este tipo de cenário: *batch-incremental*, doravante chamado de *mini-lote*, e *instance-incremental*, doravante chamado de *incremental* (READ *et al.*, 2012).

Existe uma controvérsia a respeito da definição de aprendizado incremental. Sharma (1998) e Lange e Grieser (2002) discutem se em um cenário de aprendizado incremental os exemplos mais antigos podem ser revistos pelo classificador. Já Zhou e Chen (2002) discorre sobre a necessidade de algoritmos incrementais lidarem com o surgimento inexperado de novas

classes. Além disso, He e Chen (2008) e Muhlbaier, Topalis e Polikar (2009) dizem que as abordagens incrementais devem ser capazes de aprender novas informações mantendo seus conhecimentos anteriores, sem ter acesso aos exemplos já vistos anteriormente.

No paradigma de mini-lote um classificador tradicional é treinado com mini-lotes de dados, isto é, cada w novos exemplos formam um mini-lote, e quando este mini-lote está cheio ele é apresentado ao classificador para que este gere seu modelo. Como principais desvantegens desse método pode-se citar:

1. Necessidade de especificação do tamanho do mini-lote;
2. Descarte dos modelos antigos para a criação de um novo modelo baseado no mini-lote mais recente;
3. O classificador não é atualizado com novos exemplos até que um mini-lote esteja cheio.

Descartar modelos antigos pode afetar a capacidade do modelo de aprender conceitos completos, enquanto a incapacidade de aprender com novos exemplos imediatamente após a sua chegada pode afetar a capacidade do algoritmo de se adequar a novos conceitos (READ *et al.*, 2012).

Já os métodos incrementais buscam incorporar os exemplos ao modelo assim que eles são apresentados ao algoritmo. Esta abordagem basea-se no comportamento humano que, segundo Pinto (2005), pode ser visto como um processo gradual, no qual o Homem tem a habilidade aprender com fatos e incorporar incrementalmente novos conhecimentos nestes fatos já estruturados a medida em que novas observações são feitas. Sendo assim, o aprendizado incremental pode ser definido como um paradigma de aprendizado de máquina no qual o processo de aprendizagem se dá a qualquer momento em que novos dados surgem, causando um ajuste do conhecimento já existente de acordo com esses novos dados (ADE *et al.*, 2013).

Esta categoria inclui algoritmos como o k vizinhos mais próximos (BERINGER; HüLLERMEIER, 2007; ZHANG *et al.*, 2011), o *Naive Bayes* (JOHN; LANGLEY, 1995) e o *Hoeffding Trees* (DOMINGOS; HULTEN, 2000). Devido a sua natureza naturalmente incremental, estes métodos são muitas vezes preferidos em relação aos métodos de mini-lote, mas também possuem desvantegens (READ *et al.*, 2012). Uma delas é a tendência de precisar de uma quantidade grande de exemplos para aprenderem um novo conceito. Um exemplo disso são as *Hoeffding Trees*, que crescem de forma bastante lenta em relação ao número de exemplos observados. Métodos "gulosos" como o *kNN* tendem a aprender novos conceitos mais rapidamente, mas são muito limitados em relação a quantidade de exemplos utilizados para gerar o modelo, o que acaba gerando a necessidade de discartar exemplos com o passar do tempo, assim como os métodos de mini-lote.

Lidar com o problema de manter um modelo atualizado com novos exemplos sem perder as informações do modelo anterior é o principal objetivo dos algotitmos incrementais (JOSHI;

KULKARNI, 2012). Devido a suas características básicas de incorporação de novo conhecimento aos já existentes, os algoritmos incrementais tendem a consumir menos recursos computacionais, tanto em termos de memória quanto em tempo de processamento (PINTO, 2005). Este é um fator importante para o presente projeto, visto que com uma abordagem incremental é possível levar em consideração as mudanças de conceito, isto é, quando o ambiente no qual o agente está inserido se modifica, o algoritmo é capaz de integrá-las a seu modelo de mundo.

Para ser considerado incremental um algoritmo deve possuir algumas características. Uma definição bem aceita é a dada por Langley (1995), que diz: "Um algoritmo *L* é incremental se *L* for capaz de introduzir na sua base de dados de conhecimento um exemplo de cada vez sem rever qualquer exemplo já processado e retiver em memória apenas uma estrutura de conhecimento".

Como fluxos de dados podem estar sujeitos a mudanças de conceito, lidar com essas mudanças também é um ponto importante para os algoritmos incrementais e de mini-lote. Enquanto algoritmos de mini-lote e o *kNN* possuem naturalmente uma ferramenta para se adaptar às mudanças, devido ao descarte de informações antigas, os algoritmos incrementais como o *Hoeffding Tree* tendem a funcionar melhor em ambientes com um detector de mudanças explicito (BIFET; GAVALDA, 2007). O *ADWIN* (BIFET; GAVALDA, 2007) pode ser usado para se manter uma janela de tamanho variável sobre os exemplos mais recentes com a intenção de detectar mudanças de conceito, e pode ser utilizada de forma combinada com diversos métodos, como mostrado em Bifet e Gavaldà (2009).

Neste cenário também é possível a utilização de *ensembles* de classificadores, como o *Accuracy Weighted Ensemble* (*Ensemble* Ponderado pela Acurácia) (WANG *et al.*, 2003). Neste método tipicamente o modelo mais velho é descartado quando o número máximo de modelos é atingido, sendo re-construído a partir dos exemplos mais recentes. Também neste tipo de *ensemble*, os membros tem seus votos ponderados pela sua performance de classificação.

3.3 Considerações finais

O aprendizado incremental se difere intrinsecamente do aprendizado tradicional. Isso se deve ao fato do aprendizado em lote trabalhar analisando uma grande quantidade de dados de uma só vez, assumindo-se que dados suficientemente representativos estão disponíveis durante a fase de treinamento (ADE *et al.*, 2013), formando um único modelo de mundo. Enquanto isso, modelos incrementais "evoluem" e atualizam seu modelo quando novas instâncias são processadas.

CAPÍTULO

4

MODELO DE MISTURAS GAUSSIANAS

4.1 Considerações iniciais

Neste capítulo são apresentadas as técnicas relevantes para este projeto. Na Seção 4.2 é descrito o Modelo de Misturas Gaussianas convencional, com suas equações e definições fundamentais. Já na Seção 4.3 é apresentada a versão incremental do GMM proposta por Engel e Heinen (2010). Por fim, na Seção 4.4, as modificações propostas neste projeto para o algoritmo incremental também são descritas, e o algoritmo modificado é apresentado.

4.2 Modelo de misturas gaussianas tradicional

O modelo de misturas gaussianas é uma ferramenta de modelagem estatística que já foi utilizada com sucesso em diversas aplicações, como classificação supervisionada e não supervisionada. Na classificação supervisionada um exemplo, visto como um vetor D-dimensional de características, é atribuído a um conjunto pré determinado de classes, possuindo uma certa probabilidade de pertencer a cada uma delas (ENGEL; HEINEN, 2010).

Segundo Malheiro (2003), quando se fala em reconhecimento estatístico de padrões, classificar um exemplo desconhecido consiste na estimativa da função densidade de probabilidade (*fdp*) para os vetores de características de cada classe. O modelo de misturas gaussianas (GMM) visa estimar essas funções, modelando cada classe como uma mistura de várias funções de densidade de probabilidade.

Os algoritmos que utilizam este modelo buscam maximizar uma função verossimilhança, que fornece uma medida da forma com que a *fdp* se ajusta ao conjunto de dados. Cada *fdp* é composta, basicamente, por três parâmetros: w_i (probabilidade a priori da componente i), μ_i (centróide da distribuição gaussiana i) e Σ_i (variância ou covariâncias da distribuição i) (REYNOLDS, 2009). Tendo em vista tais parâmetros, o algoritmo buscará estimá-los de forma

que o ajuste da *fdp* aos dados de treinamento seja o melhor possível, isto é, que o valor da função de verossimilhança seja o maior possível. Para realizar tal estimativa costuma-se utilizar o algoritmo Expectation-Maximization (EM) (REDNER; WALKER, 1984; KUMAR; SATOOR; BUCK, 2009), que garante que a função de verossimilhança é não decrescente e que converge pelo menos para um máximo local.

O algoritmo EM trabalha em duas etapas que se repetem até que se atinja uma convergência, sendo que em cada iteração os parâmetros w_i, μ_i e Σ_i são ajustados para maximizar a função de verossimilhança. Contudo, os valores iniciais devem ser passados como parâmetro para o algoritmo EM e, para tal, pode ser utilizado o algoritmo de agrupamento k-means (DUDA; HART; STORK, 2000) para se obter um agrupamento inicial dos dados e, a partir de tal, calcular-se as médias, covariâncias e probabilidades a priori dos grupos. É importante ressaltar que com diferentes valores iniciais, o algoritmo EM pode produzir resultados finais distintos. Desta forma, uma estratégia que pode ser utilizada para aumentar as chances de bons resultados é a execução repetitiva deste algoritmo para diferentes valores iniciais, selecionando no fim os valores para os atributos que gerem a maior saida para a função de verossimilhança.

Sendo θ o conjunto de parâmetros das misturas de gaussianas (μ_i, Σ_i e w_i), x um vetor D-dimensional de dados contínuos (como atributos, por exemplo), um modelo de misturas gaussianas é uma soma ponderada de M componentes gaussianas, dada pela Equação 4.1.

$$p(x|\theta) = \sum_{i=1}^{M} w_i g(x|\mu_i, \Sigma_i) \tag{4.1}$$

Nesta equação $g(x|\mu_i, \Sigma_i)$ representa cada componente gaussiana D-dimensional, definida pela Equação 4.2.

$$g(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} exp\left\{\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right\} \tag{4.2}$$

Após a obtenção de todos os valores, é possível calcular probabilidade *a posteriori* da componente i dada a ocorrência do exemplo x_j, representada pela Equação 4.3.

$$Pr(i|x_j, \theta) = \frac{w_i g(x_j|\mu_i, \Sigma_i)}{\sum_{k=1}^{N} w_k g(x_j|\mu_k, \Sigma_k)} \tag{4.3}$$

Assim como já mencionado, o algoritmo EM é utilizado para estimar os parâmetros θ que maximizem a verossimilhança entre os dados e o modelo de misturas gaussianas que os representará. Dado um conjunto de treinamento $X = (x_1, x_2, x_3, ..., x_n)$ a função de verossimilhança assume independência entre os exemplos (REYNOLDS, 2009) e pode ser escrita como

na Equação 4.4.

$$\mathscr{L}(\theta, X) = \prod_{i=1}^{n} p(x_i|\theta) \tag{4.4}$$

Contudo, esta função não é linear em θ, o que torna sua maximização direta impossível (REYNOLDS, 2009), o que justifica o uso do EM para realizar a estimativa de tais parâmetros.

O algoritmo EM busca, a partir de um dado θ inicial, estimar um novo $\overline{\theta}$ tal que $\mathscr{L}(X|\overline{\theta}) \geq \mathscr{L}(X|\theta)$. Na próxima iteração, o novo modelo se torna o modelo inicial, e o processo se repete até que haja uma determinada convergência.

A estimativa dos parâmetros θ de cada iteração do algoritmo EM segue as Equações 4.5, 4.6 e 4.7 a seguir.

$$\overline{w}_i = \frac{1}{n} \sum_{j=1}^{n} Pr(i|x_j, \theta) \tag{4.5}$$

$$\overline{\mu}_i = \frac{\sum_{j=1}^{n} Pr(i|x_j, \theta) x_j}{\sum_{j=1}^{n} Pr(i|x_j, \theta)} \tag{4.6}$$

$$\overline{\Sigma}_i = \frac{\sum_{j=1}^{n} Pr(i|x_j, \theta) x_j^2}{\sum_{j=1}^{n} Pr(i|x_j, \theta)} - \overline{\mu}_i^2 \tag{4.7}$$

As probabilidades a priori w_i e as componentes gaussianas são ajustadas de forma a obedecer as Equações 4.8 e 4.9, respectivamente:

$$\sum_{i=1}^{M} w_i = 1, 0 \leq w_i \leq 1 \tag{4.8}$$

$$\int g(x|\mu_i, \Sigma_i) d(x) = 1 \tag{4.9}$$

É importante notar que o algoritmo EM original não é incremental. Sendo assim, as equações de atualização dos parâmatros (4.5, 4.6 e 4.7) requerem o conhecimento das probabilidades a posteriori de todos os exemplos vistos até o momento. Tal requisito é incompatível com as características de um fluxo de dados.

No caso do algoritmo incremental, a ordem em que os exemplos são apresentados é relevante. Portanto, a notação utilizada até o momento será modificada para acrescentar um sobrescrito (t) indicando o instante no tempo em que a variável assume determinado valor.

4.3 Modelo de misturas gaussianas incremental

Assim como o algoritmo EM, o Modelo de Misturas Gaussianas Incremental, do inglês IGMM (*Incremental Gaussian Mixture Model*), também realiza a modelagem das misturas de distribuições de probabilidade gaussianas. Contudo, a abordagem incremental permite que os parâmetros de cada componente gaussiana sejam ajustados assim que cada novo exemplo é computado, seguindo-se equações aproximadamente incrementais, ao contrário das utilizadas pelo EM (ENGEL; HEINEN, 2010). Desta forma o modelo pode ser atualizado assim que novas informações relevantes chegam pelo fluxo de dados.

Porém, assim como em outros modelos, existe a dificuldade de se aliar a adaptabilidade rápida às mudanças (flexibilidade) a robustez em períodos sem mudanças de conceito (estabilidade), conhecida como dilema da estabilidade *versus* sensibilidade (ENGEL; HEINEN, 2010). Tal dificuldade se justifica por, assim como já citado, muitas vezes algoritmos que se adequam rapidamente às mudanças se tornam muito sensíveis a ruídos e, na outra extremidade, algoritmos robustos a ruídos e estáveis em fases sem mudanças de conceito acabam sendo lentos para adaptar-se às mudanças (TSYMBAL, 2004).

Assim como no GMM convencional, o IGMM assume que a fdp de um conjunto de dados pode ser definida como:

$$p(x|\theta) = \sum_{i=1}^{M} w_i g(x|\mu_i, \Sigma_i) \tag{4.10}$$

Sendo θ o conjunto de parâmetros das misturas de gaussianas (μ_i, Σ_i e w_i), x um vetor D-dimensional de dados contínuos (como atributos, por exemplo), um modelo de misturas gaussianas é uma soma ponderada de M componentes gaussianas. As probabilidades a priori w_i e as componentes gaussianas tambpem são ajustados como no GMM tradicional para obedecer as Equações 4.11 e 4.12.

$$\sum_{i=1}^{M} w_i = 1, 0 \leq w_i \leq 1 \tag{4.11}$$

$$\int g(x|\mu_i, \Sigma_i) d(x) = 1 \tag{4.12}$$

A probabilidade de se observar o vetor x D-dimensional na componente gaussiana i também segue a Equação 4.2, utilizada pelo GMM.

O IGMM utiliza algumas medidas para controlar o número de componentes necessárias para se representar os dados já vistos. O modelo começa com uma única componente possuindo probabilidade *a priori* $w_1^{(1)} = 1$, com média μ_1 igual ao primeiro exemplo visto, no instante $t = 1$. Além disso é utilizada uma matriz de covariância diagonal padrão, como $\Sigma_1^1 = \sigma_{ini}^2 I$, com σ_{ini} sendo definido previamente pelo usuário e I é a matriz identidade (ENGEL; HEINEN, 2010).

As novas componentes são adicionadas ao modelo assim que se fazem necessários, utilizando-se um critério de verossimilhança mínima de um dado exemplo $x^{(t)}$ em relação a alguma componente já existente. Sendo assim, para cada novo dado $x^{(t)}$, é verificado se este se adapta a cada componente gaussiana i com uma probabilidade $p(x^{(t)}|i) = g(x^{(t)}|\mu_i^{(t-1)}, \Sigma_i^{(t-1)})$ acima de um valor mínimo. Caso $p(x^{(t)}|i)$ não atinja o valor mínimo para nenhuma componente, o dado é considerado uma nova informação e uma componente gaussiana é adicionada ao modelo. É importante ressaltar a dificuldade na escolha do valor do critério de mínima verossimilhança, devido ao dilema entre estabilidade e sensibilidade já citado.

O valor do critério de mínima verossimilhança C_{ver} pode ser definido como uma fração do valor máximo da função de verossimilhança, de forma que a adição de uma nova componente após a análise de um dado $x^{(t)}$ ocorre quando:

$$p(x^{(t)}|i) < \frac{C_{ver}}{(2\pi)^{D/2}\sqrt{|\Sigma_i^{(t-1)}|}}, \forall i \tag{4.13}$$

Já os pontos que obedecem a esse critério e, portanto, se adequam a alguma das componentes já existentes, devem ser incorporados ao modelo, levando a necessidade de atualização dos parâmetros do mesmo. O IGMM se baseia em uma versão incremental do processo EM já mencionado.

Quando um exemplo $x^{(t)}$ é atribuído a uma componente i, é necessário saber as probabilidades a posteriori de todos os exemplos já analisados para se calcular o peso do novo dado nesta componente. Isso se deve ao fato de que a componente i será atualizada proporcionalmente a probabilidade a posteriori $p(i|x^{(t)})$ e a soma das probabilidades $p(i|x)$ de todos os exemplos já vistos. Contudo, manter todos esses valores em memória ao se lidar com um fluxo de dados aumentaria muito o custo computacional e de memória do algoritmo. Para evitar este problema, o IGMM armazena esta soma em uma variável denominada sp_i, que é reiniciada periodicamente para se evitar uma possível saturação. Tendo em vista esta necessidade, as Equações 4.14, 4.15, 4.16 e 4.17 apresentam a forma com que os parâmetros do modelo são atualizados.

$$sp_i^{(t)} = sp_i^{(t-1)} + Pr(i|x^{(t)}, \theta^{(t-1)}) \tag{4.14}$$

$$\mu_i^{(t)} = \mu_i^{(t-1)} + \frac{Pr(i|x^{(t)}, \theta^{(t-1)})}{sp_i^{(t)}}(x^{(t)} - \mu_i^{(t-1)}) \tag{4.15}$$

$$\Sigma_i^{(t)} = \Sigma_i^{(t-1)} - (\mu_i^{(t)} - \mu_i^{(t-1)})(\mu_i^{(t)} - \mu_i^{(t-1)})^T + \frac{Pr(i|x^{(t)}, \theta^{(t-1)})}{sp_i^{(t)}}[(x^{(t)} - \mu_i^{(t)})(x^{(t)} - \mu_i^{(t)})^T - \Sigma_i^{(t-1)}] \tag{4.16}$$

$$w_i^{(t)} = \frac{sp_i^{(t)}}{\sum_{j=1}^{M} sp_j^{(t)}} \tag{4.17}$$

Desta forma, após a chegada do primeiro exemplo $x^{(1)}$, a componente 1 terá valores de $sp_1^{(1)} = 1$, $\mu_1^{(1)} = x^{(1)}$ e $\Sigma_1^{(1)} = \sigma_{ini}^2 I$. Esta inicialização é importante e reforça a diferença entre o GMM tradicional e sua versão incremental. Enquanto o GMM tradicional gera seu modelo revisitando todos os exemplos já vistos e calculando seus parâmetros a partir deles, o IGMM mantém apenas os parâmetros em memória e os atualiza sem a necessidade de rever exemplos antigos.

O IGMM possui apenas dois parâmetros que necessitam ser definidos antes de sua execução. Segundo Engel e Heinen (2010), o parâmetro σ_{ini} não é crítico, necessitando ser grande o suficiente apenas para evitar que componentes englobem apenas um exemplo. Nos experimentos realizados no trabalho de Engel e Heinen (2010) σ_{ini} foi inicializado com o valor $(x_{max} - x_{min})/10$, onde x_{max} representa os maiores valor observados para os atributos e x_{min} os menores. Contudo, como já mencionado, a definição de um valor para o parâmetro C_{ver} é mais crítica, visto que este influencia a sensibilidade do algoritmo em relação às mudanças de conceito. Em outras palavras, este parâmetro indica o quão distante um exemplo $x^{(t)}$ deve estar do centro $\mu_i^{(t)}$ para não ser considerado membro da componente i. Sendo assim, um valor de $C_{ver} = 0,01$, por exemplo, indica que somente quando $p(x^{(t)}|i) < 0,01$ o exemplo $x^{(t)}$ será considerado não pertencente à componente i.

4.4 IGMM-CD - Modelo de misturas gaussianas incremental com mudanças de conceito

Neste trabalho, o modelo de misturas gaussianas incremental de Engel e Heinen (2010), originalmente proposto para ser aplicado em problemas de agrupamento, foi utilizado como base para criação de um novo método de classificação para fluxo de dados, denominado IGMM-CD[1].

Como descrito na Seção 4.3, o IGMM é uma ferramenta para se estimar a *fdp* de um conjunto de dados. Quando utilizada no cenário de agrupamento, cada grupo pode ser definido como uma componente gaussiana e cada dado pode pertencer a um grupo com uma certa probabilidade.

Contudo, neste projeto esta-se interessado em problemas de classificação. Desta forma, é necessário expandir o IGMM para que este seja capaz de lidar com a tarefa de classificar novos exemplos. Esta adaptação pode ser feita criando-se um GMM para cada classe, ou mantendo-se um GMM para todas as classes e atribuindo cada componente a uma classe. Neste projeto a segunda abordagem foi escolhida, mas é provável que ambas apresentam resultados semelhantes.

[1] Do Inglês: Incremental Gaussian Mixture Model with Concept Drifts

Para classificar um novo exemplo em um fluxo de dados, primeiramente recebe-se o exemplo sem rótulo $x^{(t)}$ e prediz-se uma classe $\bar{y}^{(t)}$ para este exemplo, baseada no atual GMM com parâmetros $\theta^{(t-1)}$. Após a predição, assume-se que o algoritmo receberá o rótulo correto $y^{(t)}$ para o exemplo x^t, e assim poderá atualizar seu modelo para $\theta^{(t)}$.

Os passos do IGMM-CD são mostrados a seguir no Algoritmo 1.

Algoritmo 1: IGMM-CD

Entrada:
$X = (x^{(1)}, \ldots, x^{(n)})$ {Sequência ordenada de exemplos para classificação}
$Y = (y^{(1)}, \ldots, y^{(n)})$ {Sequência ordenada dos rótulos corretos das classes dos exemplos de X}
σ_{ini} {Matriz de covariância inicial}
C_{ver} {Critério de mínima verossimilhança}
Saída: $\bar{Y} = (\bar{y}^{(1)}, \ldots, \bar{y}^{(n)})$ {Sequência ordenada de rótulos preditos}
$\theta^{(1)} = \emptyset$
para cada $x^{(t)} \in X$ **faça**
    $\bar{y}^{(t)} \leftarrow classifica(x^{(t)}, \theta^{(t-1)})$
    $y^{(t)} \leftarrow rótulo\_correto(x^{(t)})$
    $nova\_componente \leftarrow TRUE$
    para $i \leftarrow 1$ *to* M **faça** /* M = número de componentes */
        se $p(x^{(t)}|i) \geq \frac{C_{ver}}{(2\pi)^{D/2}|\Sigma_i|^{1/2}}$ ***e*** $class(i) = y^{(t)}$ **então**
            $nova\_componente = FALSE$
    se $nova\_componente = TRUE$ **então**
        $cria\_nova\_componente(x^{(t)}, y^{(t)}, \sigma_{ini})$
    senão
        $\theta^{(t)} = atualiza\_componentes(x^{(t)}, \theta^{(t-1)})$
    $remove\_componentes()$

O IGMM-CD se utiliza de quatro métodos principais. O método *Classifica* é responsável por predizer uma classe $\bar{y}^{(t)}$ para um novo exemplo $x^{(t)}$ que chega pelo fluxo de dados. São propostas duas formas de realizar a classificação de cada exemplo. A primeira delas, denominada *global*, usa um critério de máxima probabilidade a posteriori para classificar o exemplo como pertencente à classe cuja soma da probabilidade a posteiori seja a maior. Esta abordagem é mostrada no Algoritmo 2, onde $[class(i) = c]$ retorna 1 caso a classe da componente i seja c e 0 caso contrário. .

Neste projeto, levanta-se a hipótese de que uma abordagem local para atualização das componentes e classificação dos exemplos no cenário de fluxos de dados com mudanças de conceito, pode produzir melhores resultados em termos de acurácia. Tal hipótese se baseia na ideia de que exemplos que representam novos conceitos devem ter pequena ou nenhuma influência em componentes desatualizadas, agilizando a criação de novas componentes representando estes novos conceitos. Apesar da abordagem local poder ser implementada em diversos níveis, neste

Algoritmo 2: Método *classica* global

Entrada:
1 $x^{(t)}$ {Exemplo a ser classificado}
2 $\theta^{(t-1)}$ {Conjunto de parâmetros do GMM}
Saída: c (classe com máxima probabilidade a posteriori)
3 **retorna** $\underset{c}{arg\,max}\ p(c|x^{(t)},\theta^{(t-1)}) = \sum_{i=1}^{M} g(x^{(t)}|\mu_i^{(t-1)},\Sigma_i^{(t-1)})w_i^{(t-1)}[class(i)=c]$;

projeto analisa-se o caso extremo onde somente uma componente é atualizada e utilizada para classificação. Desta forma, um exemplo desconhecido é classificado como pertencente a mesma classe da componente com maior probabilidade a posteriori individual, e não mais do maior somatório dentre as classes. O Algoritmo 3 mostra o procedimento de classificação local.

Algoritmo 3: Método *classifica* local

Entrada:
1 $x^{(t)}$ {Exemplo a ser classificado}
2 $\theta^{(t-1)}$ {Conjunto de parâmetros do GMM}
Saída: c (classe da componente com maior probabilidade a posteriori)
3 **retorna** $classe(\underset{j}{arg\,max}\ g(x^{(t)}|\mu_j^{(t-1)},\Sigma_j^{(t-1)})w_j^{(t-1)})$

De modo similar, existem duas formas de atualizar as componentes do modelo quando um exemplo passa pelo critério de mínima verossimilhança. O método global atualiza todas as componentes que pertencem a mesma classe do exemplo recém classificado. As regras de atualização são as mesmas apresentadas nas Equações 4.14, 4.15, 4.16 e 4.17, e são apresentadas no Algoritmo 4.

O Algoritmo 5 apresenta a versão local do método de atualização de componentes. A ideia é atualizar apenas a componente que forneceu o rótulo para o exemplo atual. Em outras palavras, somente a componente gaussiana com maior probabilidade a posterior é atualizada com cada novo exemplo apresentado ao modelo, sendo as demais componentes modificadas apenas em relação a sua probabilidade a priori. O intuito desta modificação é que haja uma convergência e adaptação mais ágil do modelo às mudanças de conceito. Contudo, tanto na abordagem global quanto na abordagem local, quando $\bar{y}^{(t)} \neq y^{(t)}$, o IGMM-CD ao invés de atualizar as componentes, cria uma nova baseada em $x^{(t)}$ e $y^{(t)}$ como descreve o Algoritmo 1.

Quando nenhuma das componentes já existentes passam pelo critério de mínima verossimilhança dado um novo exemplo, uma nova componente deve ser adicionada ao modelo. Isto é feito pelo método *cria_componente* onde a nova componente é iniciada com média igual ao próprio exemplo e com matriz de covariância igual a σ_{ini}.

Apesar de Engel e Heinen (2010) destacar apenas dois parâmetros, existe uma terceira variável no algoritmo que se mostra importante: a sp. Como já mencionado, esta variável é utilizada para manter as informações da soma das probabilidades das componentes ao longo do

Algoritmo 4: Método *atualiza_componentes* global

Entrada:

$x^{(t)}$ {Exemplo atual}

$y^{(t)}$ {Rótulo correto de $x^{(t)}$ }

$\theta^{(t-1)}$ {Conjunto de parâmetros do GMM}

Saída:

$\theta^{(t)}$ {Conjunto de parâmetros do GMM atualizados}

$sp^{(t)}$ {Variável sp atualizada}

para $i \leftarrow 1$ *to* M **faça**

  se $classe(i) = y^{(t)}$ **então**

    $sp_i^{(t)} = sp_i^{(t-1)} + Pr(i|x^{(t)}, \theta^{(t-1)})$

    $\mu_i^{(t)} = \mu_i^{(t-1)} + \frac{Pr(i|x^{(t)}, \theta^{(t-1)})}{sp_i^{(t)}}(x^{(t)} - \mu_i^{(t-1)})$

    $\Sigma_i^{(t)} = \Sigma_i^{(t-1)} - (\mu_i^{(t)} - \mu_i^{(t-1)})(\mu_i^{(t)} - \mu_i^{(t-1)})^\intercal + \frac{Pr(i|x^{(t)}, \theta^{(t-1)}}{sp_i^{(t)}}[(x^{(t)} - \mu_i^{(t)})(x^{(t)} - \mu_i^{(t)})^\intercal - \Sigma_i^{(t-1)}]$

    $w_i^{(t)} = \frac{sp_i^{(t)}}{\sum_{j=1}^{M} sp_j^{(t)}}$

Algoritmo 5: Método *atualiza_componentes* local

Entrada:

$x^{(t)}$ {Exemplo atual}

$y^{(t)}$ {Rótulo correto de $x^{(t)}$ }

$\theta^{(t-1)}$ {Conjunto de parâmetros do GMM}

Saída:

$\theta^{(t)}$ {Conjunto de parâmetros do GMM atualizados}

$sp^{(t)}$ {Variável sp atualizada}

$i \leftarrow \underset{j}{arg\,max}\ g(x^{(t)}|\mu_j^{(t-1)}, \Sigma_j^{(t-1)})w_j^{(t-1)}$

$sp_i^{(t)} = sp_i^{(t-1)} + Pr(i|x^{(t)}, \theta^{(t-1)})$

$\mu_i^{(t)} = \mu_i^{(t-1)} + \frac{Pr(i|x^{(t)}, \theta^{(t-1)})}{sp_i^{(t)}}(x^{(t)} - \mu_i^{(t-1)})$

$\Sigma_i^{(t)} =$
$\Sigma_i^{(t-1)} - (\mu_i^{(t)} - \mu_i^{(t-1)})(\mu_i^{(t)} - \mu_i^{(t-1)})^\intercal + \frac{Pr(i|x^{(t)}, \theta^{(t-1)}}{sp_i^{(t)}}[(x^{(t)} - \mu_i^{(t)})(x^{(t)} - \mu_i^{(t)})^\intercal - \Sigma_i^{(t-1)}]$

$w_i^{(t)} = \frac{sp_i^{(t)}}{\sum_{j=1}^{M} sp_j^{(t)}}$

tempo, para que seja possível realizar-se uma normalização ponderada dos valores atuais ao se atualizar os valores de média, matriz de covariância e probabilidade a priori das componentes. Além disso, essa variável deve ser reiniciada periodicamente para evitar que seu valor cresça descontroladamente e acabe dando pouca relevância para os novos exemplos. No trabalho de Engel e Heinen (2010) não fica claro qual é o critério utilizado para a realização do *reset* da variável (tempo, valor máximo pré-definido, etc), e nem a forma com que a reinicialização acontece. Nos estudos realizados neste projeto é utilziada uma forma de reinicialização baseada no número de componentes existentes no modelo, mas não são realizados testes específicos com o intuito de otimizar tal abordagem.

A abordagem proposta por Engel e Heinen (2010) apresenta bons resultados seus experimentos. Contudo, ela foi aplicada em um ambiente estacionário, isto é, onde não existe evolução significativa dos dados durante o tempo. Desta forma, os resultados obtidos aplicando-se o método de Engel e Heinen (2010) apenas adaptado à classificação no cenário de interesse deste projeto foram ruins, visto que todas as bases de dados utilizadas nos experimentos representavam cenários de fluxos de dados com mudanças de conceito. Mesmo nos conjuntos de dados considerados fáceis o algoritmo não conseguiu apresentar boa acurácia, como mostra a Figura 4. Neste conjunto de dados existem apenas duas classes definidas por dois atributos, que se movem horizontalmente de forma perpendicular ao eixo X.

Além disso, o algoritmo se mostrou pouco escalável neste cenário, apresentando um grande aumento no custo computacional para se processar novos exemplos a medida em que o tempo passava. O aumento no custo para se processar novos exemplos é atribuido a falta de um critério de eliminação de componentes gaussianas que perdem sua relevância para o modelo no decorrer do tempo e é mostrado na Figura 5.

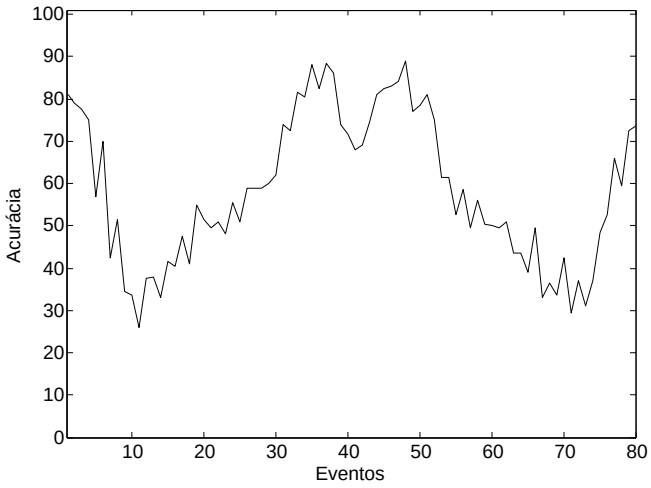


Figura 4 – Variação ao longo do tempo da acurácia do algoritmo no conjunto de dados *2CHT*.

Tendo em vista os resultados obtidos pelo IGMM original (mostrados no Capítulo 5), foram proposta algumas modificações com intuito solucionar o problema da quantidade de com-

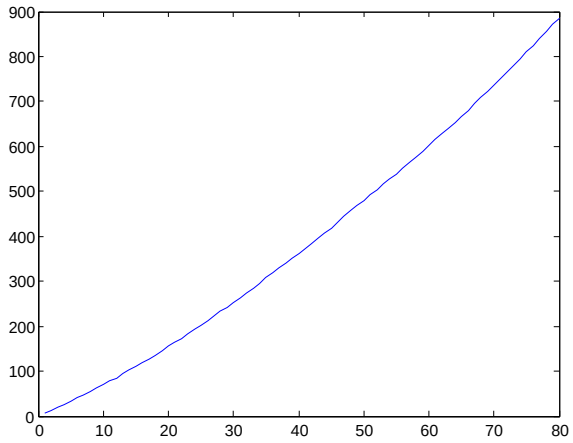


Figura 5 – Variação ao longo do tempo do custo computacional para se processar a base *2CDT* com o algoritmo sem eliminação de gaussianas.

ponentes gaussianas existentes no modelo, que tendia a crescer indefinidamente. Estas abordagens foram implementadas através do método *remove_componentes*. Existem diversas abordagens na literatura possíveis para se lidar com este problema, tais como a junção de componentes semelhantes (SONG; WANG, 2005; ARANDJELOVIC; CIPOLLA, 2006; BOUCHACHIA; VANARET, 2011), o uso de exemplos negativos para eliminar conceitos considerados errados e/ou desatualizados (KRISTAN; SKOČAJ; LEONARDIS, 2010) e a eliminação de informação utilizando critérios como tempo ou relevância, como é o caso do uso de janelas ou mesmo em algoritmos que também realizam a junção e/ou divisão de componentes (BOUCHACHIA; VANARET, 2011). No presente trabalho foram avaliadas duas estratégias: 1)A simples eliminação de componentes cuja probabilidade a priori é inferior a $0,01$, ou 1% e 2) A adição de um terceiro parâmetro T utilizado como limite médio de componentes permitidas por classe.

Tais alteração, apesar de pequena, possibilita um controle sobre o crescimento do modelo e consequente custo computacional do algoritmo, como mostrado no Capítulo 5.

O parâmetro T é um limiar de componentes permitidas no modelo, e tem como intuito realizar uma estratégia de eliminação de componentes mais inteligente. Sendo assim, no lugar de eliminar todas as gaussianas que possuem probabilidades a priori inferiores a 1%, apenas quando existe uma quantidade muito grande de componentes as que possuem menor probabilidade a priori são eliminadas. A quantidade máxima permitida é baseada no parâmetro T e no número de classes existentes no modelo, isto é, o modelo permite, em média, que haja T componentes representando cada classe. É importante ressaltar que esta abordagem não obriga a existência de T componentes por classe, ela apenas utiliza o número de classes existentes para ponderar o limiar máximo permitido. Em outras palavras, quando o modelo é representado por mais que $T \times numero\_de\_classes$ componentes, o algoritmo elimina a componente com menor valor de probabilidade *a priori* (w), visto que esta componente representa a menor quantidade de exemplos. Esta abordagem é interessante por se adaptar à cada base de dados, visto que é

provável que em casos onde existem poucas classes sejam necessárias menos componentes para formar um bom modelo do que em casos onde existam muitas.

4.5 Considerações finais

Neste projeto foram avaliados alguns métodos baseados no GMM. O primeiro consiste em uma versão adaptada para classificação do método proposto por Engel e Heinen (2010), no qual não existem política de eliminação de componentes desatualizadas. Neste método, chamado apenas de IGMM, foi aplicada a política global de classificação e atualização de componentes, visto que esta se adequa melhor aos conceitos originais do GMM. Além deste, foram testados os métodos IGMM-CD com política global e local de atualização de componentes e classificação de exemplos. Foram testadas duas técnicas de esquecimento com o IGMM-CD: a eliminação de componentes com probabilidade a priori inferior a 1% e a remoção baseada no parâmetro T, descrita neste capítulo.

CAPÍTULO

5

RESULTADOS E DISCUSSÕES

5.1 Considerações iniciais

Neste capítulo são apresentados os resultados mais relevantes obtidos com o desenvolvimento deste projeto de mestrado. A Seção 5.2 evidencia como as técnicas propostas foram avaliadas, apresentando os critérios e ferramentas utilizadas, e com quais métodos elas são comparadas, apresentando diversos algoritmos presentes na literatura. Já a Seção 5.3 apresenta uma breve descrição dos conjuntos de dados utilizados para validação das hipóteses do projeto. Na Seção 5.4 são apresentados os resultados obtidos com a versão incremental do GMM proposta por Engel e Heinen (2010) enquanto as Seções 5.4.2, 5.4.3, 5.4.4 e 5.4.5 apresentam os resultados obtidos com os algoritmos com as modificações propostas neste projeto.

5.2 Métodos de avaliação

Assim como já descrito, o principal objetivo deste projeto é avaliar o comportamento de técnicas de aprendizado incremental quando utilizadas em cenários de fluxos de dados não estacionários. Para ser considerada aplicável neste cenário, uma técnica deve ser capaz de classificar os exemplos com boa acurácia, mesmo na presença de mudanças de conceito, tendo como base um conjunto de treinamento inicial de exemplos já rotulados, e manter-se atualizada utilizando informações dos rótulos dos exemplos do fluxo.

Um fator a ser considerado ao se discutir a viabilidade de um método de classificação utilizado neste cenário é sua eficiência em termos de tempo de processamento. Sendo assim, mesmo que algum método possua uma desvantagem pequena em termos de acurácia na classificação, este pode se tornar interessante desde que possua uma alta eficiência em termos de custo computacional. A relevância do fator custo computacional se deve à necessidade de algoritmos aplicados em fluxos de dados estarem sempre preparados para receber um novo exemplo, visto

que o intervalo entre a chegada dos dados do fluxo pode não ser constante.

Dessa maneira os algoritmos de classificação incrementais estudados neste projeto de pesquisa são avaliados tanto em termos de eficiência de processamento quanto em termos de eficácia de classificação.

Os algoritmos incrementais são analisados em termos de eficiência de processamento, caracterizada pelo tempo de execução necessário para se processar um fluxo. Dessa maneira, deseja-se comparar entre os diversos modelos incrementais qual é o custo computacional em se atualizar uma hipótese incrementalmente com a adição de um novo exemplo.

Os algoritmos também são avaliados em termos de desempenho de classificação, sendo comparados utilizando medidas de desempenho derivadas da matriz de confusão, como a acurácia. Além disso, os classificadores também são avaliados no contexto de classificação de fluxo de dados, analisando-se seus comportamentos ao longo do tempo. Para tal, alguns gráficos foram gerados e seus pontos representam o desempenho médio dos algoritmos em determinado período de tempo. Além disso, em casos onde a base de dados possui muitos exemplos, o intervalo de tempo representado por cada ponto foi aumentado, com o intuito de manter fácil a compreensão dos gráficos. Como exemplo pode-se citar as bases *1CDT* e *MG_2C_2D*, onde a primeira possui poucos exemplos sendo seu gráfico traçado com 80 pontos, com cada ponto representando 200 exemplos, enquanto a segunda possui seu gráfico traçado com 100 pontos, com cada ponto representando 2000 exemplos.

Para a avaliação dos métodos propostos são utilizados conjuntos de dados reais e de *benchmark*, descritos na Seção 5.3. No caso de avaliação de mudanças de conceito, as avaliações são geralmente realizadas na literatura utilizando conjuntos de dados sintéticos que simulam as mudanças de conceito, uma vez que somente com dados artificiais é possível controlar as mudanças de conceito e avaliar o desempenho do algoritmo no momento em que ocorrem.

Para a comparação com métodos da literatura é utilizado o ambiente computacional MOA[1] (*Massive Online Analysis*), derivado do conhecido ambiente Weka[2]. O ambiente inclui uma coleção de algoritmos de aprendizado de máquina e ferramentas para avaliação de problemas de classificação, regressão e aglomeramento de fluxo de dados. Os algoritmos da plataforma utilizados foram o *Naive Bayes*[3], *Hoeffling Adaptive Tree*[4] (HULTEN; SPENCER; DOMINGOS, 2001; BIFET; GAVALDÀ, 2009; BIFET *et al.*, 2011), Active Classifier[5] (ŽLIOBAITĖ *et al.*,

1 <http://moa.cs.waikato.ac.nz/>
2 <http://www.cs.waikato.ac.nz/ml/weka/>
3 <http://www.cs.waikato.ac.nz/~abifet/MOA/API/classmoa_1_1classifiers_1_1bayes_1_1_naive_bayes.html#details>
4 <http://www.cs.waikato.ac.nz/~abifet/MOA/API/classmoa_1_1classifiers_1_1trees_1_1_hoeffding_tree.html#details>
5 <http://www.cs.waikato.ac.nz/~abifet/MOA/API/classmoa_1_1classifiers_1_1active_1_1_active_classifier.html#details>

2011; CESA-BIANCHI; GENTILE; ZANIBONI, 2006), *Drift Detection Method*[6], *Perceptron*[7] e o *Accuracy Updated Ensemble*[8] (BRZEZIŃSKI; STEFANOWSKI, 2011). A escolha por estes algoritmos foi baseada tanto na disponibilidade destes no ambiente MOA quanto em uma busca por variabilidade de abordagens, visto que a lista inclue algoritmos naturalmente incrementais, como o *Naive Bayes*, métodos baseados em *ensemble*, como o *Accuracy Updated Ensemble*, baseados na detecção de mudanças de conceito, como o *Drift Detection Method*, árvores de decisão, como o *Hoeffling Adaptive Tree*, entre outros.

5.3 Conjuntos de dados utilizados

Nesta seção são apresentados e descritos as bases de dados utilizadas para o teste dos algoritmos a serem comparados. A Tabela 1 mostra algumas das características dos conjuntos de dados artificiais utilizados. Vídeos mostrando a variação ao longo do tempo destas bases de dados são encontrados em Souza (2015). A coluna *Mudança a cada X exemplos* indica o número de exemplos entre mudanças de conceito consecutivas.

Tabela 1 – Conjuntos de dados sintéticos.

Base de dados	Número de classes	Número de atributos	Número de exemplos	Mudança a cada X exemplos
1CDT	2	2	16.000	400
2CDT	2	2	16.000	400
1CHT	2	2	16.000	400
2CHT	2	2	16.000	400
4CR	4	2	144.400	400
4CRE-V1	4	2	125.000	1.000
4CRE-V2	4	2	183.000	1.000
5CVT	5	2	40.000	1.000
1CSurr	2	2	55.283	600
UG_2C_2D[9]	2	2	100.000	1.000
MG_2C_2D[10]	2	2	200.000	2.000
FG_2C_2D[10]	2	2	200.000	2.000
UG_2C_3D[10]	2	3	200.000	2.000
UG_2C_5D[10]	2	5	200.000	2.000
GEARS_2C_2D[10]	2	2	200.000	2.000

A seguir cada conjunto de dados é descrito brevemente.

1. *2CDT* - Duas classes se movimentam diagonalmente ao longo do tempo no mesmo sentido, mantendo-se próximas. Essa movimentação é ilustrada na Figura 6.

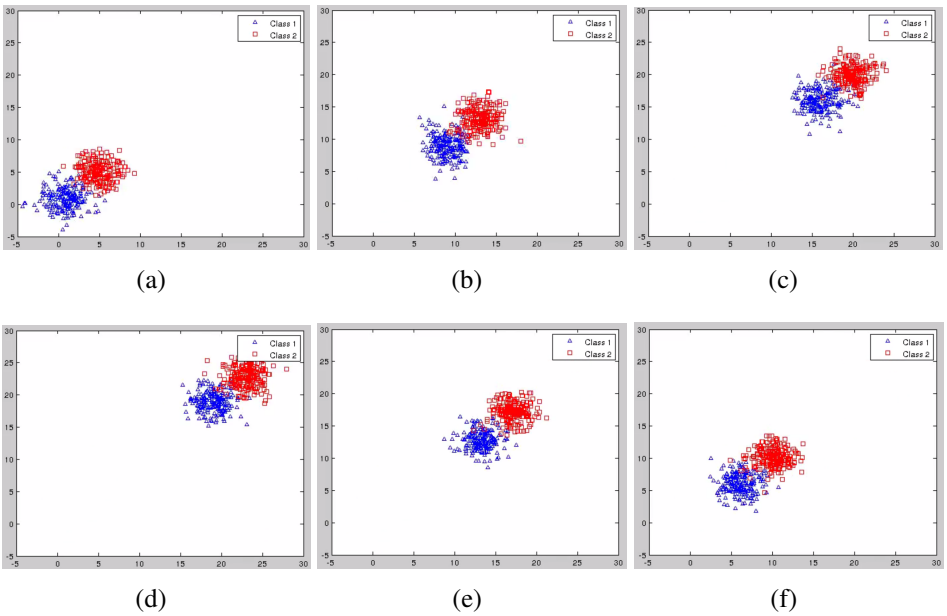
[6] <http://moa.googlecode.com/hg-history/a7212c57797928a0508dc20e22859e4388d0011c/moa/src/main/java/moa/classifiers/drift/DriftDetectionMethodClassifier.java>

[7] <http://www.cs.waikato.ac.nz/~abifet/MOA/API/classmoa_1_1classifiers_1_1functions_1_1_perceptron.html#details>

[8] <http://www.cs.waikato.ac.nz/~abifet/MOA/API/classmoa_1_1classifiers_1_1meta_1_1_accuracy_updated_ensemble.html#details>

[9] Dyer, Capo e Polikar (2014)

[10] Ditzler e Polikar (2013)



(a) (b) (c)

(d) (e) (f)

Figura 6 – Variação ao longo do tempo das classes no conjunto de dados *2CDT*.

2. *1CDT* - Semelhante ao *2CDT*, contudo somente uma das classes se movimenta ao longo do tempo, enquanto a outro permanece centrada no ponto *(0,0)*.

3. *2CHT* - Semelhante ao *2CDT*, contudo as classes se movimentam horizontalmente ao invés de diagonalmente.

4. *1CHT* - Semelhante ao *2CHT*, contudo somente uma das classes se movimenta ao longo do tempo, enquanto a outra permanece centrada no ponto *(2,2)*.

5. *4CRE-V2* - 4 classes se movimentam em uma trajetória circular. Em alguns momentos estas se aproximam e se sobrepõe. A Figura 7 mostra esse comportamento.

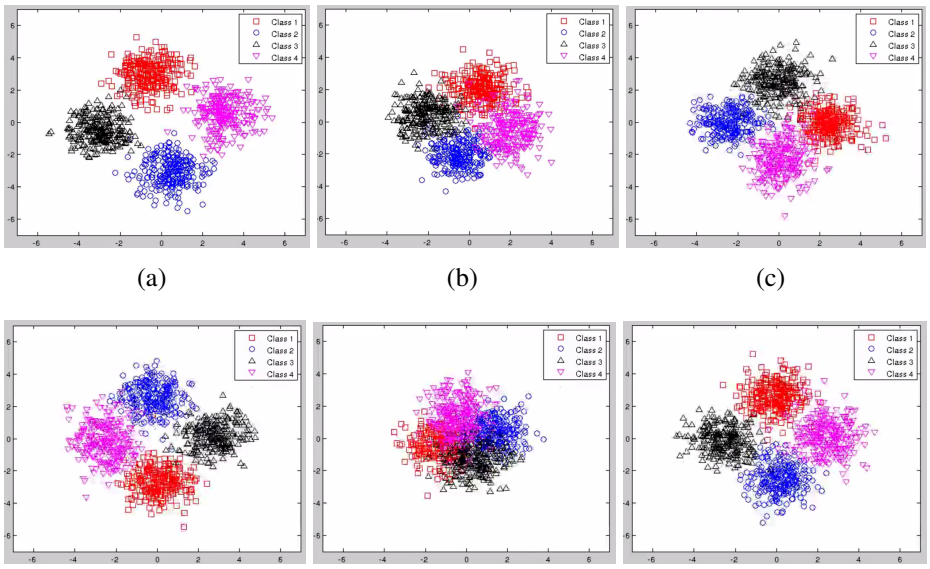
6. *4CRE-V1* - semelhante a *4CRE-V2*, contudo a sobreposição das classes é menor.

7. *4CR* - 4 classes se movimentam em uma trajetória circular, sem se sobreporem.

8. *5CVT* - 5 classes se movimentam verticalmente, mantendo-se próximas com pequenas sobreposições.

9. *1CSurr* - Uma classe menos esparça se movimenta circundando uma classe mais esparça. Em um dado momento a classe menos esparça se movimenta diagonalmente no interior da classe mais esparça. A Figura 8 mostra o deslocamento das classes ao longo do tempo.

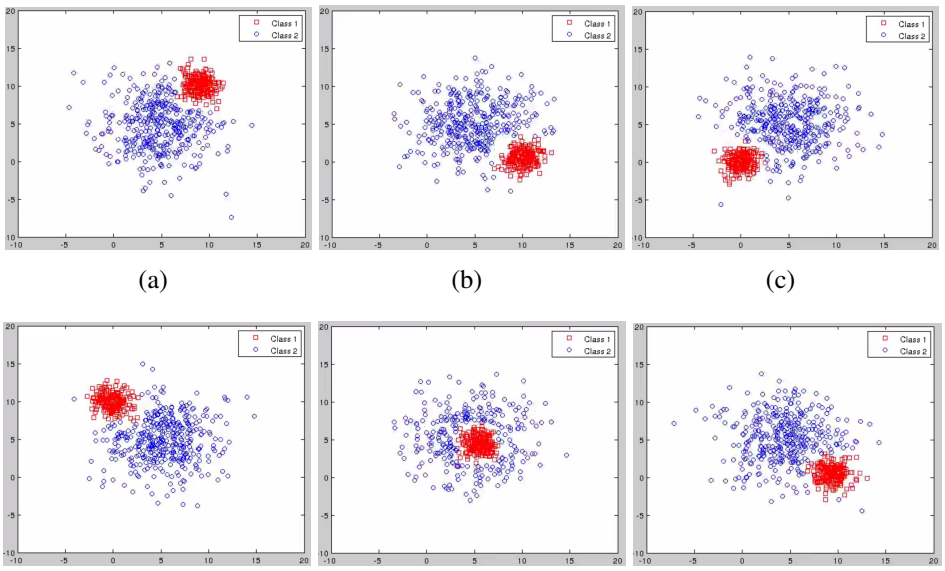
10. *MG_2C_2D* - No início as classes 1 e dois estão distantes. A classe 1 é descrita por duas gaussianas, enquanto a classe 2 é descrita por apenas uma. A medida em que se movimentam e se aproximam, a classe 1 passa a ser descrita por apenas uma gaussiana e a classe dois se divide em duas gaussianas. Neste momento as classes possuem uma grande



(a) (b) (c)

(d) (e) (f)

Figura 7 – Variação ao longo do tempo das classes no conjunto de dados *4CRE-V2*.



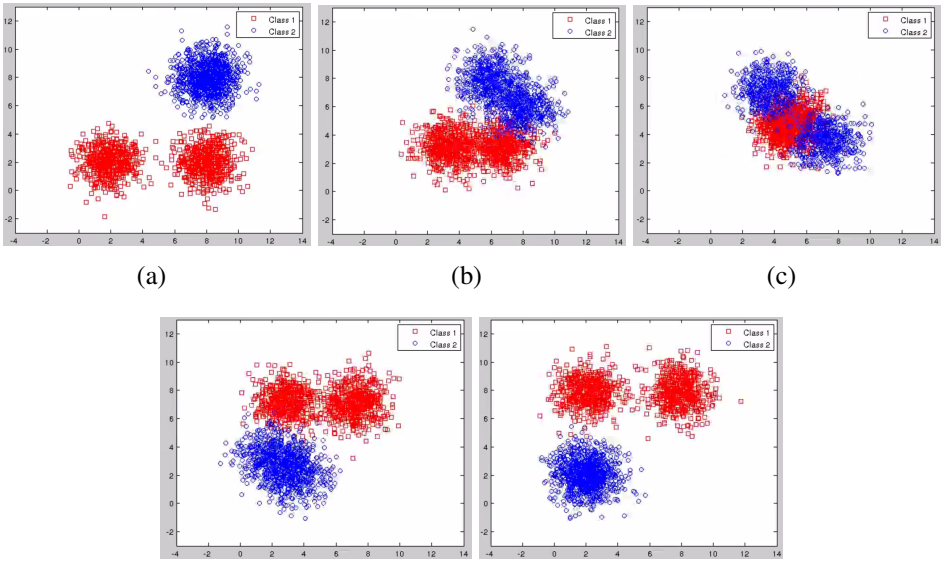
(a) (b) (c)

(d) (e) (f)

Figura 8 – Variação ao longo do tempo das classes no conjunto de dados *1CSurr*.

sobreposição e, após ele, elas continuam se movimentando e voltando a ser descritas como eram inicialmente. O deslocamento das classes ao longo do tempo é mostrado na Figura 9.

11. *FG_2C_2D* - No início a classe 1 é descrita por 3 gaussianas e a classe 2 por apenas uma. Elas formam uma espécia de losango no espaço, sendo a classe 1 os vértices superior, inferior e a esquerda, e a classe 2 o vértice a direita. Ao longo do tempo os vértices superior e inferior (parte da classe 1) se movimentam para a direita enquanto o vértice da direta (classe 2) se aproxima do vértice da esquerda (classe 1). Quando os vértices da esquerda e direita começam a se sobrepor, o vértice da direta direciona-se diagonalmente para cima,

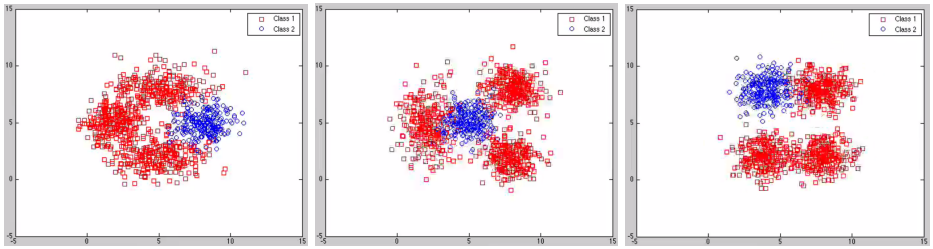


(a) (b) (c)

(d) (e)

Figura 9 – Variação ao longo do tempo das classes no conjunto de dados *MG_2C_2D*.

enquanto o vértice inicialmente a esquerda começa a se mover verticalmente para baixo. No final das movimentações as classes se assemelham a um quadrado, sendo a classe 2 o vértice superior a esquerda e os três vértices restantes pertencentes a classe 1. Existe sobreposição das classes durante todo o processo de movimentação no espaço. A Figura 10 mostra a movimentação das classes.



(a) (b) (c)

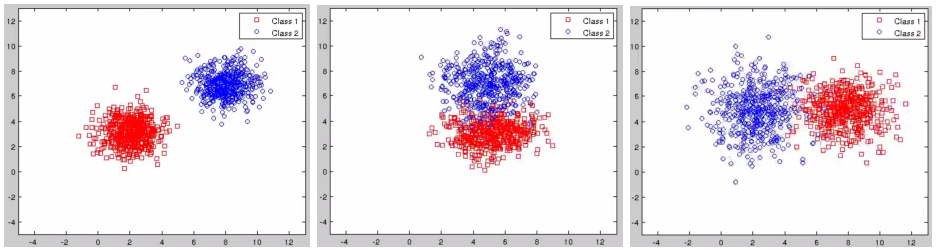
Figura 10 – Variação ao longo do tempo das classes no conjunto de dados *FG_2C_2D*.

12. *UG_2C_2D* - 2 classes se aproximam e fazem alguns movimentos circulatórios próximos uma a outra, com sobreposição variando de pequena a média, como mostra a Figura 11.

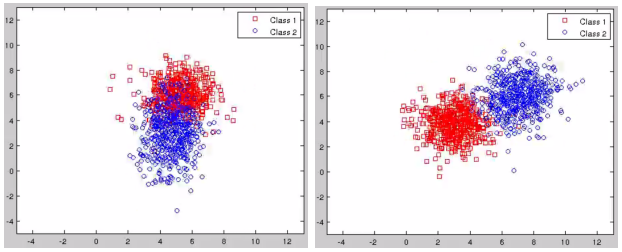
13. *UG_2C_3D* - Semelhante ao *UG_2C_2D*, porém os exemplos possuem 3 atributos.

14. *UG_2C_5D* - Semelhante ao *UG_2C_2D*, porém os exemplos possuem 5 atributos.

15. *GEARS_2C_2D* - As classes possuem formato semelhante a uma estrela e se movimentam de forma circular sobre seus próprios eixos. A classe 1 gira no sentido anti-horário, enquanto a classe 2 gira no sentido horário. A Figura 12 mostra a posição das classes em um dado instante de tempo.



(a) (b) (c)



(d) (e)

Figura 11 – Variação ao longo do tempo das classes no conjunto de dados *UG_2C_2D*.

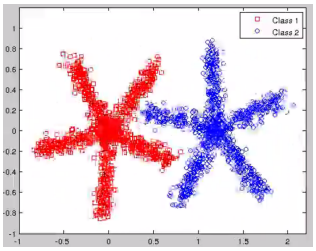


Figura 12 – Conjunto de dados *GEARS_2C_2D*.

Vídeos mostrando a variação ao longo do tempo destas bases de dados são encontrados em Souza (2015).

Os conjuntos de dados reais utilizados são descritos a seguir:

1. *Poker-lsn* - Possui 1.000.000 de instâncias e 10 atributos. Cada instância representa um exemplo de "mão" de *poker* contendo cinco cartas retiradas de um baralho comum de 52 cartas, sendo cada carta descrita por dois atributos (*suit* e *rank*).

2. *ElecNormNew* - Conjunto de dados amplamente utilizado descrito por Harries e Wales (1999). Estes dados foram coletados a partir do mercado de eletricidade de *New South Wales*, um estado da Austrália. Neste mercado, os preços não são fixos e são afetados pela demanda e oferta do mercado. Eles são ajustados a cada cinco minutos. O conjunto de dados contém 45.312 instâncias. Cada classe identifica a alteração do preço relativamente a uma média variável das últimas 24 horas. Foi utilizada a versão normalizada deste conjunto de dados, possuindo 8 atributos e 2 classes.

3. *Keystroke* - Conjunto de dados advindo de características de digitação de usuários. É

baseado no conjunto de dados *CMU* apresentado por Killourhy e Maxion (2010). Possui 4 classes, 10 atributos e 1600 instâncias.

.

A Tabela 2 apresenta um resumo das bases de dados reais utilizadas.

Tabela 2 – Conjuntos de dados reais.

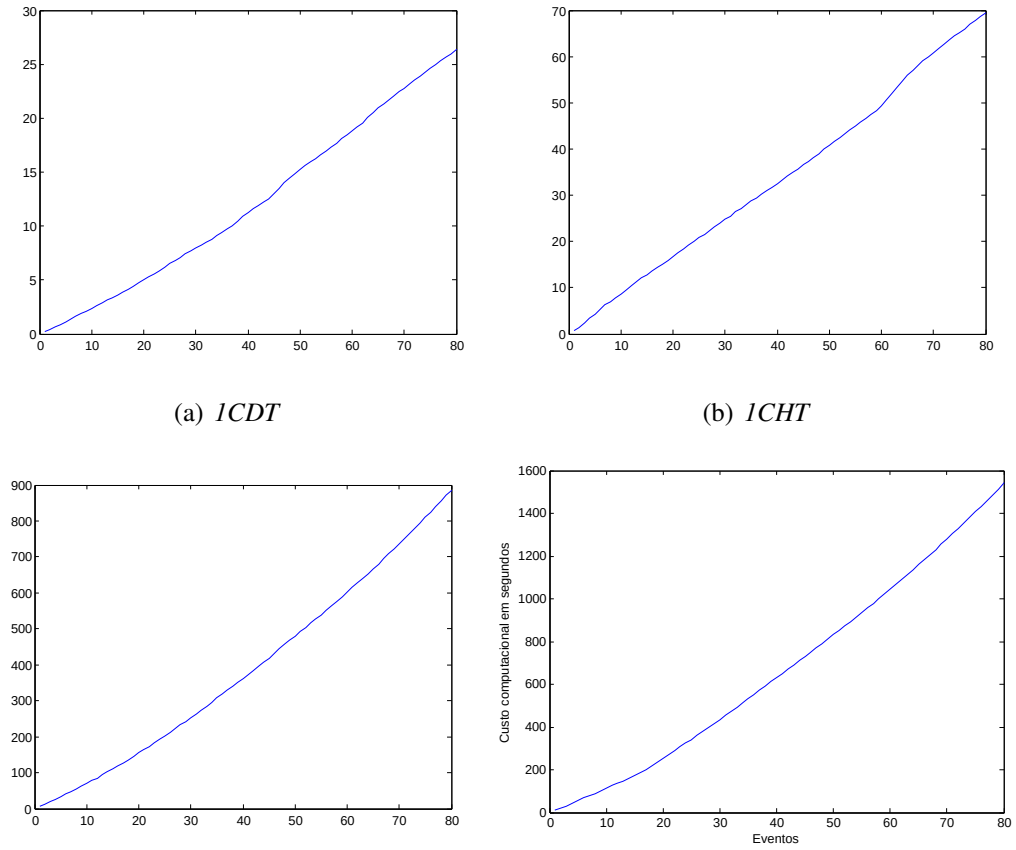
Base de dados	Número de classes	Número de atributos	Número de exemplos
Poker-lsn	10	10	1.000.000
ElecNormNew	2	8	45.312
keystroke	4	10	1.600

5.4 A estratégia incremental

5.4.1 Modelo sem eliminação de gaussianas

Inicialmente o Modelo de Misturas Gaussianas Incremental de Engel e Heinen (2010) foi implementado e executado exatamente como o proposto pelos autores, exceto pela adaptação para o problema de classificação, assim como o mostrado na Seção 4.3. Em outras palavras, manteve-se no modelo todas as componentes gaussianas criadas. Entretanto, logo no início dos experimentos percebeu-se que tal abordagem não possuia um custo de tempo de processamento aproximadamente constante ao longo do tempo, sendo que este aumentava a medida em que novos dados eram apresentados ao algoritmo e novas componentes gaussianas eram criadas. Tal comportamento esta associado ao fato de não haver uma estratégia de eliminação de componentes ao longo do tempo. Na presença de mudanças de conceito, gaussianas associadas a dados gerados a partir de conceitos obsoletos ficam representadas no modelo, aumentando o custo computacional e possivelmente reduzindo a acurácia do modelo. A Figura 13 mostra o aumento do custo computacional de se processar novos exemplos ao longo do tempo para os conjuntos de dados avaliados. Cada ponto no gráfico mostra o tempo em segundos acumulado necessário para processar (classificar e re-treinar o modelo) durante instantes de tempo no fluxo de dados, utilizando os parâmetros covariância inicial = 2 e critério de mínima verossimilhança = 0,01. A escolha dos parâmetros foi baseada nas recomendações presentes em Engel e Heinen (2010).

Devido ao alto custo computacional para a aplicação deste algoritmo foram realizados experimentos apenas nas bases de dados com menor quantidade de exemplos. Além do alto custo computacional devido ao aumento do número de componentes gaussianas, o algoritmo foi incapaz de acompanhar as mudanças de conceito ao longo do tempo, apresentando resultados ruins mesmo pra algumas das bases de dados mais simples. Esta incapacidade está associada a presença das componentes gaussianas que, apesar de representarem conceitos obsoletos, ainda são mantidas no modelo. A Figura 14 mostra o comportamento da acurácia ao longo do tempo para os conjuntos de dados avaliados.



(a) *1CDT* (b) *1CHT*

(c) *2CDT* (d) *2CHT*

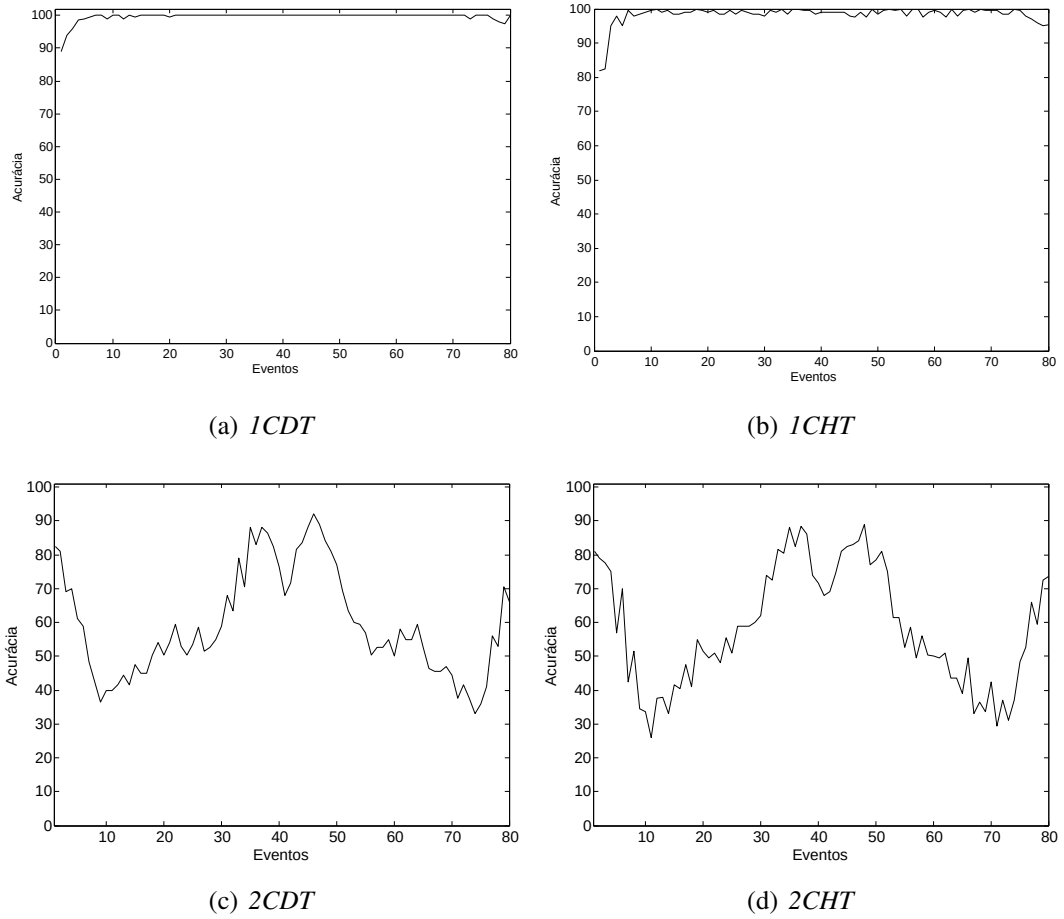
Figura 13 – Variação ao longo do tempo do custo computacional de se processar os conjunto de dados *1CDT* (a), *1CHT* (b), *2CDT* (c) e *2CHT* (d).

5.4.2 Modelo com eliminação de gaussianas

Tendo em vista os resultados encontrados nos primeiros experimentos, foi realizada uma adaptação no algoritmo de Engel e Heinen (2010) para que este eliminasse algumas componentes gaussianas ao longo do tempo, com o intuito de controlar o custo computacional e permitir uma avaliação mais ampla do modelo, possibilitando a inclusão de bases de dados maiores. A estratégia inicial foi o corte de gaussianas que possuíam um valor de probabilidade a priori inferior a 1% (um por cento). Sendo assim, quando um novo exemplo é apresentado ao algoritmo é verificado se, após a adição deste exemplo, alguma das componentes passa a ter seu valor de probabilidade a priori reduzido a menos de 0,01 e, em caso afirmativo, essa componente é eliminada do modelo.

Em termos de acurácia os resultados não são suficientes para concluir se a abordagem de eliminação de gaussianas é mais eficiente. Isso se deve ao fato do comportamento das técnicas variarem nas bases de dados avaliadas, estando algumas vezes próximos e em outras distantes, como mostra a Figura 15.

Já em relação ao custo de processamento a vantagem obtida ao se eliminar gaussianas



(a) *1CDT* (b) *1CHT*

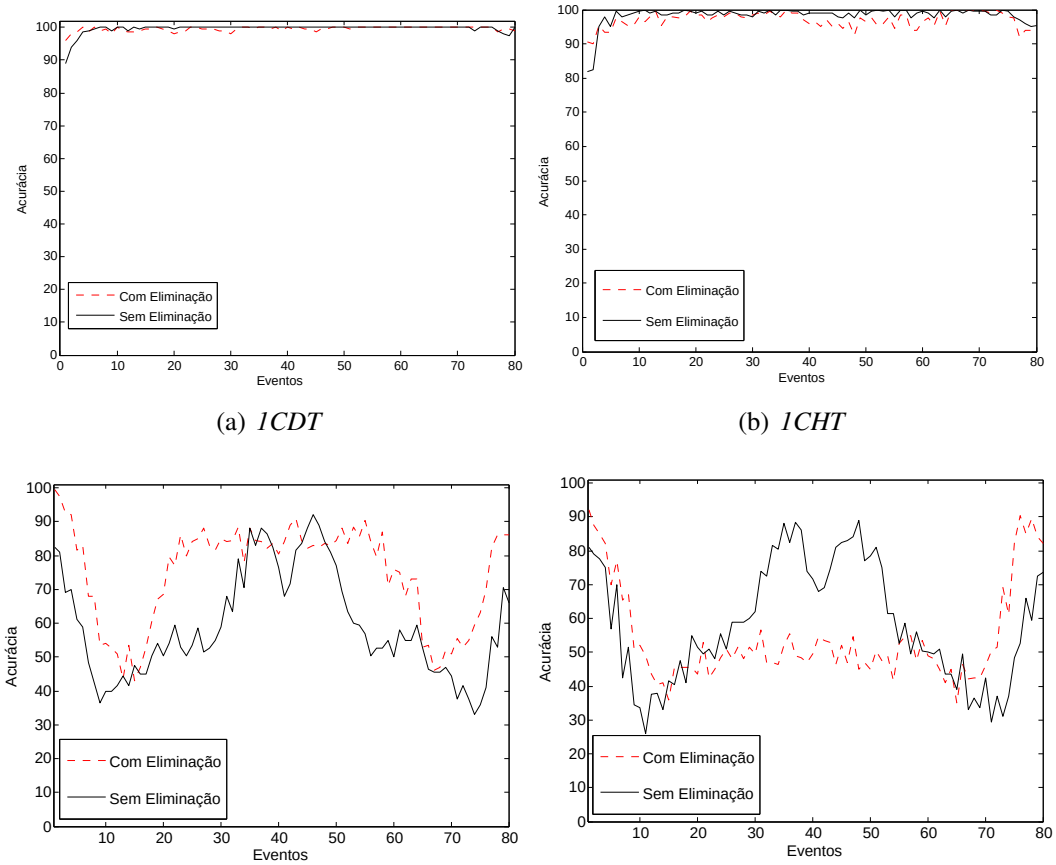
(c) *2CDT* (d) *2CHT*

Figura 14 – Variação ao longo do tempo da acurácia do algoritmo IGMM sem eliminação de componentes nos conjuntos de dados *1CDT* (a), *1CHT* (b), *2CDT* (c) e *2CHT* (d).

foi relevante em todos os conjuntos de dados avaliados, chegando a reduzir o tempo necessário para se processar a base de dados *2CHT* em aproximadamente 32 (trinta e duas) vezes. A Figura 16 mostra a comparação do custo computacional nas bases de dados avaliadas.

A adição de um critério de eliminação de gaussianas proporcionou ao algoritmo uma grande economia em termos de custo computacional, tornando possível analisar as maiores bases de dados utilizadas no estudo em poucos minutos. Enquanto o algoritmo que utiliza a estratégia sem eliminação de gaussianas gasta aproximandamente 25 (vinte e cinco) minutos para processar a base de dados *2CHT*, o algoritmo com eliminação de gaussianas realiza a mesma tarefa em cerca de 14 segundos.

A grande economia em termos de custo de processamento proporcionada pela eliminação de gaussianas torna possível a avaliação dos demais conjuntos de dados utilizados no presente projeto. Sendo assim, foram realizados estudos em relação a variação dos parâmetros do algoritmo, isto é, a variação do critério de mínima verossimilhança e a covariância inicial. A variação da covariância inicial foi feita em um intervalo de 0,2 a 2,0, mantendo-se o critério de mínima verossimilhança fixado em 0,01. Já a variação do critério de mínima verossimilhança foi feita em um intervalo de 0,01 a 0,1, mantendo-se a covariância inicial fixada em 2,0.



(a) *1CDT* (b) *1CHT*

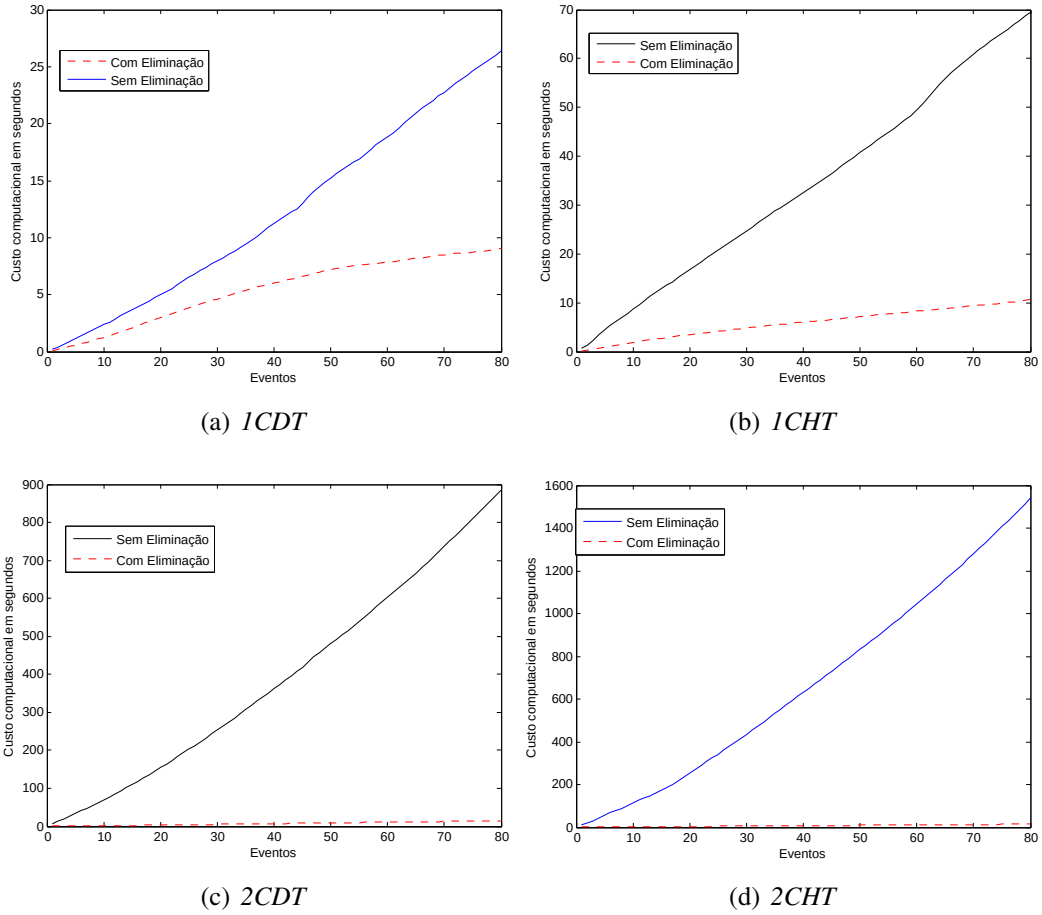
(c) *2CDT* (d) *2CHT*

Figura 15 – Variação ao longo do tempo da acurácia dos algoritmos nos conjuntos de dados *1CDT* (a), *1CHT* (b), *2CDT* (c) e *2CHT* (d).

Apesar de mostrar alguns resultados interessantes para bases de dados considerados complicadas (como a *MG_2C_2D*), o algoritmo teve problemas de desempenho relativo a acurácia em algumas bases de dados consideradas simples (como a *2CDT*). A Tabela 3 mostra os melhores e piores resultados ao se variar o parâmetro covariância inicial e o critério de mínima verossimilhança.

Para a variação do critério de mínima verossimilhança a covariância inicial foi fixada em 2,0, e para a variação da covariância inicial o critério de mínima verossimilhança foi fixado em 0,01, novamente seguindo as recomendações presentes em Engel e Heinen (2010).

Apesar da Tabela 3 mostrar os melhores e piores resultados levando-se em consideração a variação dos dois parâmetro do algoritmo, notou-se que esta variação não causa grande impacto na acurácia do algoritmo na maioria das bases de dados. Tal característica é ainda mais marcante ao se variar o parâmetro de mínima verossimilhança, onde a variação do parâmetro não causa mudança de mais do que 1% na acurácia na maioria das bases de dados. As Figuras 17 e 18 evidenciam essa observação.



(a) *1CDT* (b) *1CHT*

(c) *2CDT* (d) *2CHT*

Figura 16 – Variação ao longo do tempo do custo computacional dos algoritmos para processar os conjuntos de dados *1CDT* (a), *1CHT* (b), *2CDT* (c) e *2CHT* (d).

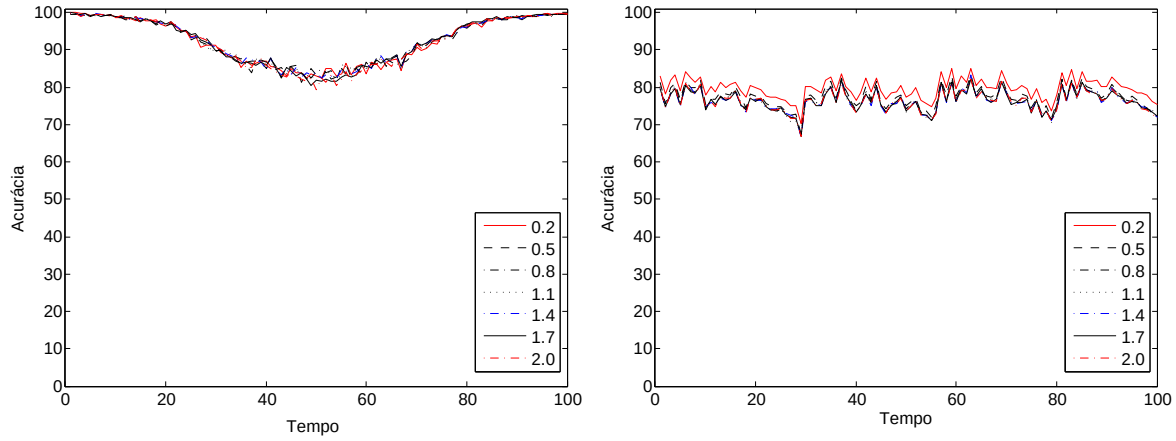
5.4.3 Modelo com atualizações de componentes locais

Tendo em vista a baixa acurácia média obtida pelo IGMM-CD mesmo após a adição de um critério de eliminação de gaussianas, foi proposta uma alteração na forma com que o algoritmo atualiza seu modelo. Tal alteração consiste na atualização apenas da gaussiana cuja probabilidade a posteriori em relação a um dado exemplo seja a maior dentre todas as componentes do modelo. A ideia é que que apenas a componente que melhor represente o exemplo seja atualizada, já que no algoritmo original mesmo gaussianas cuja probabilidade a posteriori fossem pequenas seriam afetadas pelo exemplo. Em outras palavras, pode-se dizer que a atualização do modelo passou a ser local, e não global como no algoritmo original. Mais detalhes sobre as modificações feitas no algoritmo são encontrados na Seção 4.4. Os resultados onde houve maior variação da acurácia comparando-se o IGMM global e local são mostrados na Tabela 4.

Apesar de ter tido um desempenho inferior ao algoritmo com atualizações globais nas bases de dados *ElecNormNew* e *Keystroke*, a proposta de atualização local apresentou resultados próximos ou superiores em todos os demais conjuntos de dados, com destaque para os exibidos na Tabela 4.

Tabela 3 – Melhores e piores resultados de acurária - média (desvio padrão) ao longo de todo o fluxo - para o algoritmo IGMM-CD com eliminação de componentes gaussianas com probabilidade a priori inferior a 0,01.

	Parâmetro 1		Parâmetro 2	
Base de dados	**Melhor Acurácia**	**Pior Acurácia**	**Melhor Acurácia**	**Pior Acurácia**
ElecNormNew	78,69 (6,54)	48,91 (16,6)	78,69 (6,54)	78,69 (6,54)
FG_2C_2D	95,30 (2,49)	95,09 (2,75)	95,29 (2,51)	95,02 (2,84)
GEARS_2C_2D	93,47 (1,96)	91,06 (5,57)	91,93 (4,59)	91,22 (5,83)
1CSurr	78,07 (14,17)	72,52 (14,10)	76,03 (15,38)	74,55 (16,3)
Keystroke	90,19 (4,12)	29,31 (7,19)	88,00 (5,01)	88,00 (5,01)
MG_2C_2D	92,57 (6,26)	92,25 (6,71)	92,37 (6,60)	92,16 (6,90)
poker_lsn	80,46 (6,19)	77,10 (6,60)	77,15 (6,62)	77,04 (6,60)
4CRE-V1	45,82 (36,57)	27,58 (35,41)	30,61 (39,47)	26,28 (38,36)
4CRE-V2	35,95 (37,13)	27,46 (34,49)	35,29 (37,87)	28,08 (34,28)
4CR	35,08 (36,52)	26,28 (38,51)	26,88 (38,57)	25,74 (38,71)
1CDT	99,49 (0,71)	98,97 (1,97)	99,51 (0,7)	99,21 (1,51)
2CDT	74,55 (14,84)	55,06 (14,94)	73,28 (13,41)	54,72 (15,03)
1CHT	98,13 (2,08)	97,06 (2,28)	97,91 (2,18)	96,13 (3,20)
2CHT	66,86 (12,03)	53,38 (13,98)	66,72 (12,24)	54,74 (14,65)
5CVT	52,19 (22,65)	43,69 (25,94)	53,17 (23,07)	45,98 (28,04)
UG_2C_2D	94,78 (4,01)	90,91 (11,50)	94,42 (4,29)	88,15 (15,15)
UG_2C_3D	93,86 (6,39)	93,11 (7,74)	93,60 (6,09)	81,08 (13,15)
UG_2C_5D	78,86 (11,69)	74,39 (15,33)	78,55 (11,37)	78,09 (11,41)



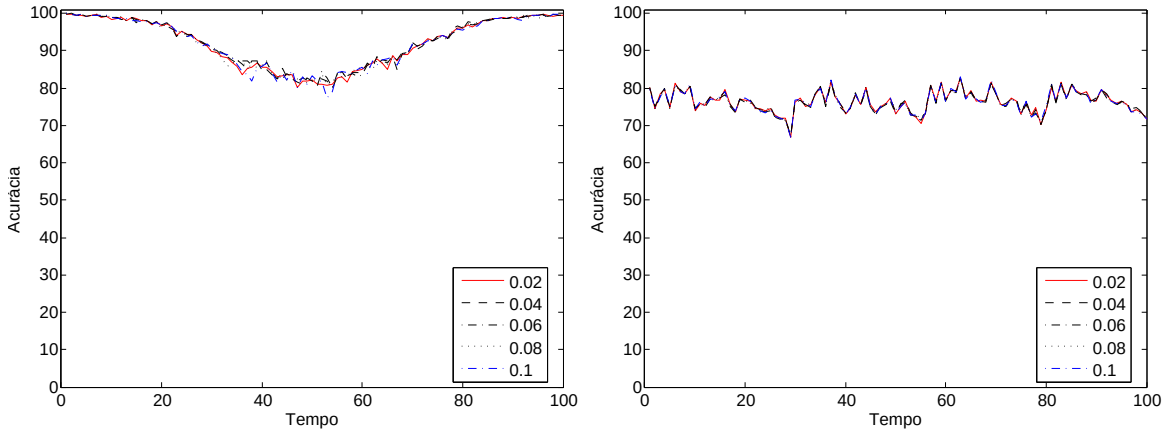
(a) *MG_2C_2D.* (b) *poker-lsn.*

Figura 17 – Variação ao longo do tempo da acurácia do algoritmo para as bases *MG_2C_2D (a)* e *poker-lsn (b)* variando-se o parâmetro da covariância inicial.

5.4.4 Comparações com métodos da literatura

Após a comparação entre o IGMM-CD com atualizações globais e o IGMM-CD com atualizações locais fez-se necessário uma comparação dos dois métodos com outros algoritmos utilizados no mesmo cenário. Os algoritmos utilizados para a comparação foram os citados na Seção 5.2. A Tabela 5 mostra os resultados obtidos por todos os algoritmos, sendo os melhores algoritmos em cada base de dados destacados em negrito.

O algoritmo IGMM-CD com atualização global, assim como já mencionado, obteve um desempenho baixo em diversas bases de dados sintéticas. Um ponto em comum sobre estes



(a) *MG_2C_2D*. (b) *poker-lsn*.

Figura 18 – Variação ao longo do tempo da acurácia do algoritmo para as bases *MG_2C_2D (a)* e *poker-lsn (b)* variando-se o parâmetro da mínima verossimilhança.

Tabela 4 – Resultados de acurária - média (desvio padrão) ao longo de todo o fluxo - para os algoritmo IGMM-CD com atualização global (G) e local (L).

Base de dados	**IGMM-CD (G)**	**IGMM-CD (L)**
ElecNormNew	78,69 (6,54)	51,74 (11,38)
1CSurr	78,07 (14,17)	96,57 (1,55)
Keystroke	88,00 (5,01)	44,56 (6,30)
4CRE-V1	27,58 (35,41)	94,91 (6,02)
4CRE-V2	27,46 (34,49)	87,25 (8,82)
4CR	26,70 (38,59)	99,80 (0,32)
1CDT	99,49 (0,71)	99,65 (1,24)
2CDT	74,55 (14,84)	93,43 (2,00)
1CHT	97,06 (2,28)	98,79 (2,73)
2CHT	54,12 (14,01)	85,42 (3,11)
5CVT	46,02 (28,01)	86,94 (2,87)

conjuntos de dados é que as classes se mantêm próximas enquanto se movimentam pelo espaço[11].

Apesar de não ser o melhor algoritmo em nenhuma das bases de dados, é possível notar que o IGMM-CD com atualizações locais se mantém na maioria dos casos com uma acurácia próxima a dos melhores resultados. As exceções são os conjuntos de dados *ElecNormNew* e *keystroke*, onde o algoritmo apresenta uma média de acurácia muito abaixo da obtida por seus concorrentes.

5.4.5 Resultados com o parâmetro T

Tendo em vista o fato de que ainda haviam bases de dados em que os algoritmos IGMM-CD com atualização global e local possuiam desempenhos considerados ruins, foi proposta uma forma diferente de eliminação de gaussianas. Assim como já mencionado, inicialmente as gaussianas com probabilidade a priori inferiores a 1% eram eliminadas, com o intuito de minimizar o custo computacional e retirar do modelo informações desnecessárias. Contudo, este

[11] Ver https://sites.google.com/site/nonstationaryarchive/home

Tabela 5 – Resultados de acurária - média (desvio padrão) ao longo de todo o fluxo - para os algoritmo IGMM-CD com atualização global (G) e local (L), *active learning*, *drift detection*, *tree*, *naive bayes*, *perceptron* e *ensenmble*.

Base de dados	IGMM-CD (G)	IGMM-CD (L)	Active	Drift Det.	Tree	N. Bayes	Perceptron	Ensemble
ElecNormNew	78,69 (6,54)	51,74 (11,38)	75,49 (11,67)	81,19 (9,7)	**83,79 (8,29)**	73,20 (12,09)	79,11 (5,90)	77,30 (11,42)
FG_2C_2D	95,09 (2,75)	92,00 (3,39)	92,64 (4,23)	91,28 (5,06)	95,07 (2,66)	84,97 (10,84)	75,00 (2,89)	**95,39 (2,32)**
GEARS_2C_2D	91,06 (5,57)	83.76 (6,25)	95,83 (1,36)	95,82 (1,37)	97,81 (1,42)	95,82 (1,37)	96,00 (1,38)	**98,99 (1,12)**
1CSurr	78,07 (14,17)	96,57 (1,55)	95,82 (3,73)	**97,65 (2,01)**	96,85 (2,61)	65,49 (16,02)	66,44 (9,08)	96,72 (3,56)
Keystroke	**88,00 (5,01)**	44,56 (6,30)	81,69 (5,59)	73,88 (7,93)	83,25 (7,36)	63,56 (11,03)	85,94 (3,83)	75,94 (13,84)
MG_2C_2D	92,25 (6,71)	89,46 (7,61)	88,69 (8,05)	88,36 (8,59)	92,69 (6,05)	55,45 (33,37)	47,73 (11,89)	**93,19 (5,85)**
poker_lsn	**77,10 (6,60)**	72,46 (11,16)	60,93 (19,24)	62,01 (19,01)	66,90 (16,30)	59,48 (19,60)	0,39 (1,32)	66,82 (18,12)
4CRE-V1	27,58 (35,41)	94,91 (6,02)	94,73 (8,73)	97,10 (4,73)	77,39 (32,81)	22,20 (37,40)	**98,18 (4,68)**	95,16 (9,05)
4CRE-V2	27,46 (34,49)	87,25 (8,82)	83,81 (11,13)	88,08 (8,45)	88,96 (8,90)	24,17 (31,82)	**92,19 (7,83)**	91,70 (8,24)
4CR	26,70 (38,59)	99,80 (0,32)	99,41 (2,79)	99,87 (0,51)	99,57 (0,83)	24,84 (38,31)	98,93 (5,58)	**99,94 (1,36)**
1CDT	99,49 (0,71)	99,65 (1,24)	99,60 (0,97)	99,65 (0,89)	99,65 (0,86)	99,65 (0,89)	**99,85 (0,64)**	99,41 (2,89)
2CDT	74,55 (14,84)	93,43 (2,00)	90,28 (6,29)	**94,11 (3,06)**	85,85 (10,85)	59,56 (16,14)	55,25 (13,23)	86,17 (6,34)
1CHT	97,06 (2,28)	98,79 (2,73)	98,47 (2,07)	98,58 (2,02)	98,49 (2,33)	98,57 (2,05)	**99,20 (2,02)**	98,81 (3,74)
2CHT	54,12 (14,01)	85,42 (3,11)	77,42 (9,95)	**86,36 (4,68)**	84,21 (5,72)	59,46 (13,88)	57,56 (12,55)	78,72 (5,98)
5CVT	46,02 (28,01)	86,94 (2,87)	83,93 (8,64)	**89,63 (4,98)**	87,36 (5,37)	66,05 (15,27)	64,42 (12,10)	86,62 (5,55)
UG_2C_2D	90,91 (11,50)	92,88 (4,49)	94,22 (5,41)	94,82 (4,21)	95,61 (3,80)	58,09 (30,81)	75,43 (20,20)	**95,89 (3,84)**
UG_2C_3D	93,28 (7,74)	91,47 (7,44)	93,92 (5,85)	94,39 (5,60)	94,58 (5,53)	61,38 (28,70)	91,89 (7,54)	**94,94 (5,47)**
UG_2C_5D	78,00 (13,06)	89,65 (7,08)	91,67 (6,37)	92,86 (5,42)	92,78 (5,57)	79,17 (11,45)	89,21 (8,30)	**93,42 (5,34)**

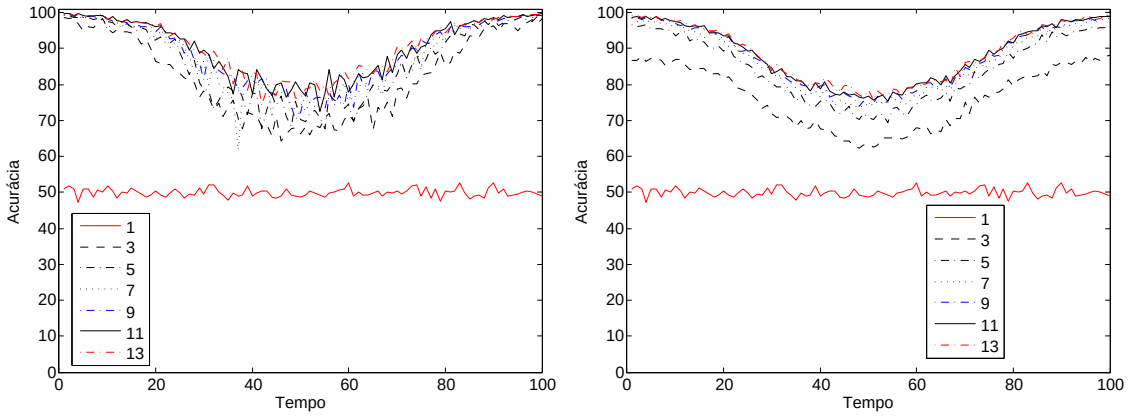
limiar de 1% era fixo para todas as bases de dados, independentemente da quantidade de classes encontradas nelas. Sendo assim, a nova estratégia de eliminação deveria levar em consideração o número de classes existentes no conjunto de dados. Desta forma, como o descrito na Seção 4.4, foi inserido nos algoritmos um novo parâmetro T que se refere a um limite da quantidade média de componentes gaussianas permitidas para representar cada classe. Os melhores resultados encontrados com a utilização deste parâmetro podem ser vistos na Tabela 6, onde os resultados destacados em negrito são aqueles que representaram uma melhoria na acurácia se comparados ao algoritmo que utiliza o limite baseado em probabilidades a priori. Além disso, são exibidos os melhores resultados obtidos para cada base de dados, levando-se em consideração todos os algoritmos testados.

Tabela 6 – Resultados de acurária - média (desvio padrão) ao longo de todo o fluxo - para o algoritmo IGMM-CD global (G) e local (L) com eliminação de componentes gaussianas através do parâmetro T.

Base de dados	IGMM-CD (G)	Par.	IGMM-CD (L)	Par.	Melhor Resultado
ElecNormNew	**85,32 (4,25)**	1	**85,32 (4,25)**	1	83,79 (8,29) - Tree
FG_2C_2D	93,54 (3,78)	13	91,33 (3,69)	13	95,39 (2,32) - Ensemble
GEARS_2C_2D	91,58 (4,48)	13	**86,33 (4,66)**	13	98,99 (1,12) - Ensemble
1CSurr	96,04 (4,08)	11	95,79 (1,80)	13	97,65 (2,01) - Drift
Keystroke	77,56 (10,41)	13	39,06 (6,66)	7	88,00 (5,01) IGMM (G)
MG_2C_2D	90,58 (8,31)	13	88,7 (8,26)	13	93,19 (5,85) - Ensemble
poker_lsn	74,19 (8,04)	1	**73,91 (8,27)**	9	77,10 (6,60) - IGMM (G)
4CRE-V1	**95,78 (7,95)**	3	**96,65 (5,64)**	13	98,18 (4,68) - Perceptron
4CRE-V2	**89,54 (9,73)**	5	**88,38 (9,57)**	13	92,19 (7,83) - Perceptron
4CR	**99,89 (0,26)**	3	**99,89 (0,24)**	13	99,94 (1,38) - Ensemble
1CDT	**99,74 (0,90)**	7	99,65 (1,17)	13	99,85 (0,64) - Perceptron
2CDT	**92,42 (3,29)**	5	93,37 (2,09)	13	94,11 (3,06) - Drift
1CHT	**98,88 (2,57)**	7	98,76 (2,78)	13	99,20 (2,02) - Perceptron
2CHT	**82,77 (5,92)**	5	84,64 (3,07)	13	86,36 (4,68) - Drift
5CVT	**67,00 (7,98)**	1	**88,44 (3,37)**	13	89,63 (4,98) - Drift
UG_2C_2D	94,40 (4,28)	11	92,61 (4,83)	13	95,89 (3,84) - Ensemble
UG_2C_3D	92,65 (7,63)	13	90,99 (8,05)	13	94,94 (5,47) - Ensemble
UG_2C_5D	**81,20 (12,41)**	3	87,41 (8,07)	13	93,42 (5,34) - Ensemble

Tais resultados são interessantes devido ao fato de que, através da variação do parâmetro proposto, foi possível chegar em bons resultados para todos os conjuntos de dados, tanto para o IGMM-CD com atualização global quanto para o com atualização local. Fica claro que este parâmetro é mais relevante para o funcionamento do algoritmo do que os parâmetros propostos por Engel e Heinen (2010), tendo grande influência sobre o seu desempenho. Além disso, através da variação do parâmetro T, pode-se controlar o custo computacional em tempo do processamento de novos exemplos visto que, quanto menor é a quantidade média de componentes permitida por classe, menor será o tempo gasto tanto para atualizar o modelo quanto para classificar um novo exemplo. A Figura 19 mostra a variação da acurácia ao longo do tempo para o conjunto de dados *MG_2C_2D* para os algoritmos com atualização global e local ao se modificar o valor do parâmetro proposto.

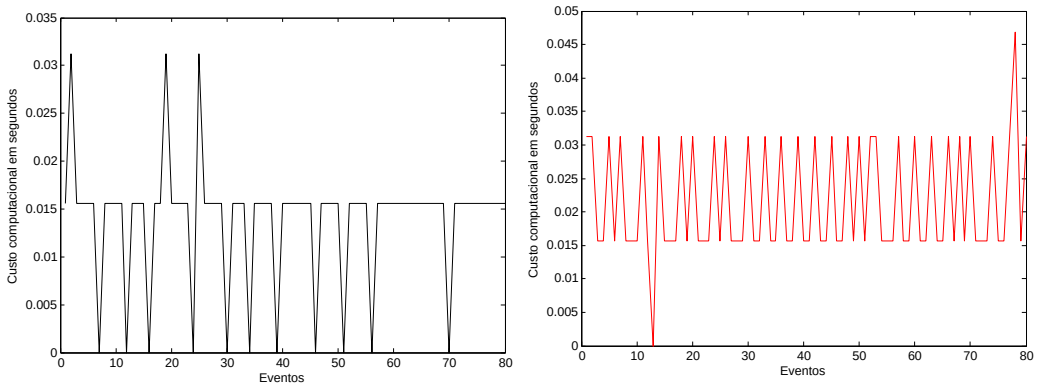
Assim como já mencionado, o novo parâmetro também possibilita o controle do custo computacional do algoritmo. O algoritmo de atualização global com a estratégia de eliminação



(a) Atualização global. (b) Atualização local.

Figura 19 – Variação ao longo do tempo da acurácia do algoritmo IGMM-CD com atualização global (A) e local (B) para a base *MG_2C_2D* variando-se o parâmetro proposto.

de componentes baseada na probabilidade a priori aplicado ao conjunto de dados *2CDT* obteve um custo médio de 0,17 segundos, com pico em 0,23, para processar 200 exemplos, enquanto o melhor resultado utilizando o novo parâmetro obteve um custo de tempo médio de 0,01 segundos, com pico em 0,03, para processar a mesma quantidade de exemplos. Já o algoritmo com atualização local possuia um custo médio de 0,037 segundos, com pico em 0,078, e seu melhor resultado utilizando o novo parâmetro teve como custo médio 0,021 com pico em 0,046, também para processar 200 exemplos. A Figura 20 mostra a variação ao longo do tempo do custo computacional dos algoritmos com atualização global e local para o conjunto de dados *2CDT* para o melhor resultado em relação a acurácia obtido ao se variar o parâmetro proposto.



(a) Atualização global. (b) Atualização local.

Figura 20 – Variação ao longo do tempo do custo computacional para se processar 200 exemplos do algoritmo IGMM-CD com atualização global (a) e local (b) para a base *2CDT*.

5.5 Considerações finais

Os resultados mostram que o IGMM-CD possui performance parecida com o estado da arte. É importante ressaltar que o IGMM-CD é um classificador simples, que não possui mecanismos de detecção de mudanças de conceito. Mesmo com estas características, este método apresenta resultados superiores em relação ao *Naive Bayes* que possui características similares. Contudo, tanto a abordagem local quanto a global apresentam desempenhos ruins em algumas bases de dados, o que indica que uma abordagem intermediária pode possuir melhor desempenho nestes cenários.

Os algoritmos de *Ensemble* e *Drift Detection* apresentaram boa acurácia em todos os conjuntos de dados. *Ensembles* são algoritmos complexos que se utilizam de diversos modelos gerados por diversos algoritmos para a realização da predição, apresentando uma boa acurácia média, porém com um alto custo computacional.

Já o algoritmo de *Drift detection* apresenta um comportamento interessante, visto que utiliza como base o algoritmo *Naive Bayes*. Enquanto o *Naive Bayes* puro apresenta baixa acurácia em diversas bases de dados, a simples adição de um método de detecção explicita de mudanças de conceito fez com que o algoritmo se tornasse uma das melhores técnicas avaliadas.

É importante ressaltar que no atual estado da pesquisa é complicado fazer comparações em relação ao custo computacional entre o IGMM-CD e os demais algoritmos avaliados. O motivo é a otimização do algoritmo IGMM-CD, que ainda não foi feita completamente, possibilitando uma melhoria de desempenho assim que algumas características e métodos forem implementados. Exemplo disso é uma política de inserção de gaussianas que mantenha o vetor de componentes ordenado por probabilidade a priori. Tal medida otimizaria a remoção das componentes, tanto no método de eliminação baseada na probabilidade a priori quanto no método utilizando o limite de gaussianas.

Atualmente as novas componentes são inseridas no final de um vetor, com custo computacional *O(1)*. Enquanto isso para se localizar a(s) componente(s) gaussiana(s) a serem eliminadas é necessário percorrer todo o vetor, isto é, um custo computacional de *O(n)*. Contudo, é possível utilizar uma política de inserção de componentes de forma que o vetor seja sempre mantido ordenado através de uma busca binária $O(\log_2 n)$. Dessa forma a eliminação de componentes seria constante, já que bastaria eliminar a(s) componente(s) na(s) última(s) posições do vetor. Além disso, é necessário testar a substituição da matriz de covariância cheia, isto é, com todos os valores preenchidos, por uma matriz de covariância diagonal. Tal mudança significaria uma grande economia de tempo de processamento, visto que o cálculo de Σ^{-1} é bastante alto. Essa estratégia já foi utilizada em outros trabalhos e se mostrou eficiente, mantendo-se competitiva e economizando processamento (BIMBOT *et al.*, 2004).

CAPÍTULO

6

CONCLUSÃO E TRABALHOS FUTUROS

Os algoritmos incrementais estudados obtiveram bons resultados ao serem comparados com diversos algoritmos presentes na plataforma MOA. Contudo, ainda existem casos representados por alguns conjuntos de dados onde as técnicas apresentam um baixo desempenho. Sendo assim, é possível que o uso de outras técnicas já utilizadas em alguns trabalhos na literatura melhorem o desempenho do IGMM-CD, tais como estratégias de junção de componentes semelhantes e divisão de componentes muito abrangentes.

Por outro lado, em relação ao cenário de latência extrema mostrado no Apêndice A, a abordagem de fixação de parâmetros se mostrou promissora. Contudo, a ideia de fixar parâmetros de um algoritmo que visa acompanhar a evolução dos dados e se adaptar a mudanças de conceito parece contra-intuitiva, de forma que outras abordagens podem ser testadas. Uma alternativa é utilizar uma forma de controlar o quanto um dado parâmetro pode ser alterado a cada iteração. Isso faz sentido no cenário de interesse deste projeto, visto que com mudanças de conceito incrementais o modelo não deve se alterar drasticamente em poucas iterações. Além disso, o problema encontrado quando classes são descritas por mais de uma distribuição gaussiana pode ser resolvido aumentando o número máximo de componentes permitido no modelo, de forma a torná-lo mais flexível.

Sendo assim, uma mescla das duas abordagens também pode ser estudada. Um algorítmo incremental cujos parâmetros das componentes gaussianas tem sua variação limitada pode ser aplicado em cenários de latência de verificação extrema, enquanto a estratégia de mini-lote pode ter seu desempenho melhorado se for adicionada a ela a possibilidade de criação de novas componentes ao longo do tempo. Tendo estas possibilidades em mente o presente trabalho poderá ser continuado de diversas formas.

APÊNDICE

A

CENÁRIO COM LATÊNCIA DE VERIFICAÇÃO EXTREMA

A.1 Considerações iniciais

Grande parte das aplicações envolvendo classificação de fluxos de dados com mudanças de conceito assumem que os rótulos corretos dos exemplos classificados estarão disponíveis imediatamente após a classificação ou com um pequeno atraso (GAMA*et al.*, 2004; NISHIDA; YAMAUCHI, 2007; BIFET; GAVALDA, 2007). Os algoritmos utilizados nestas aplicações buscam verificar se seus modelos estão atualizados a partir da avaliação da performance de classificação obtida nos exemplos mais recentes, utilizando-se de métricas como a acurácia.

Assumir a disponibilidade dos rótulos corretos após curtos intervalos de tempo é possível em diversas aplicações, como a previsão do gasto de energia elétrica de uma população. Um algoritmo que prevê o consumo de uma população com três dias de antecedência, por exemplo, terá como verificar se sua predição foi correta após esse intervalo de tempo.

Contudo, existe uma classe de aplicações, principalmente relacionadas a sensores e a robótica, na qual não é possível garantir que os algoritmos receberão retorno a respeito da corretude de suas predições. Tais aplicações geralmente necessitam de um alto custo para se obter os rótulos corretos das classes, muitas vezes envolvendo a classficação manual e individual dos eventos por um especialista (SOUZA V.M.A.; SILVA, 2015). O tempo necessário entre a classificação do exemplo e a obtenção de seus rótulo correto é denominado latência de verificação[1] (MARRS; HICKEY; BLACK, 2010).

Dependendo da aplicação, a latência pode ser fixa ou aleatória. No caso da predição do consumo de energia elétrica com três dias de antecedência, o algoritmo terá um retorno a respeito de sua indução, isto é, uma latência de verificação, após três dias se passarem. Já no

[1] do inglês *verificcation latency, também conhecida como label delay (KUNCHEVA, 2008).*

caso de uma aplicação de verificação de fraude em cartões de crédito este retorno pode ocorrer com espaços de tempos diferentes, o que pode até mesmo gerar um fornecimento dos exemplos fora da ordem cronológica original (MARRS; HICKEY; BLACK, 2010).

Quando esta latência tende ao infinito, tem-se a chamada latência de verificação extrema (do inglês *extreme verification latency*), onde nenhum rótulo é fornecido ao algoritmo após o início da classificação. Contudo, existe uma quantidade inicial de exemplos rotulados que podem ser utilizados para o treinamento do modelo, o que torna possível classificar problemas com estas características como problemas de classificação, e não de agrupamento.

Tal característica difere-se do que é comumente utilizado em diversos algoritmos de classificação de fluxos de dados, tais como *On Line Information Network* - OLIN (LAST, 2002) e *Ultra Fast Forest of Trees* - UFFT (GAMA *et al.*, 2004). Este cenário também se difere de problemas de aprendizado semi-supervisionado, nos quais alguns exemplos possuem latência de verificação extrema enquanto outros podem possuir uma latência igual a zero. Também existe diferença com o cenário de aprendizado ativo, no qual o algoritmo seleciona uma pequena parcela dos exemplos vistos para ser rotulada por um agente externo (SOUZA V.M.A.; SILVA, 2015).

A.2 A estratégia em mini-lote

Após a análise dos algoritmos em um cenário onde os modelos são atualizados com rótulos corretos, voltou-se a atenção para outro cenário interessante: a latência de verificação extrema. Como já descrito, neste cenário não é possível fornecer ao algoritmo rótulos corretos das instâncias classificadas, o que dificulta bastante a classificação. Foram avaliados os métodos k-vizinhos mais próximos (kNN) e o modelo de misturas gaussianas (*Gaussian Mixture Model* - GMM), utilizando-se bases de *benchmark* disponibilizadas por Souza, Gama e Batista (2014)[2] já descritas neste trabalho. Inicialmente pretendia-se fazer uma comparação entre as acurácias dos dois métodos em todas as bases. Contudo, o classificador kNN apresentou resultados ruins mesmo para as bases de dados consideradas fáceis quando atualizado com os rótulos preditos pelo próprio algoritmo, o que fez com que o foco fosse voltado para o GMM nos demais testes.

Para que fosse possível a adaptação dos classificadores GMM e kNN para o conceito de fluxo de dados, foram utilizadas janelas deslizantes sobre o fluxo, como mostra a Figura 21.

Tanto os pontos de treinamento quanto o tamanho da janela são definidos previamente e são fixos ao longo do fluxo. A primeira variável trata de quantos em quantos exemplos o modelo será retreinado, enquanto a segunda se refere ao tamanho da janela utilizada para a realização deste retreinamento. Sendo assim, se considerarmos *pontos de retreinamento = 200* e *janela = 500* o algoritmo terá um treinamento inicial com *500* exemplos e, após esse treinamento, classificará os próximos 200 eventos. Após a classificação dos 200 eventos o algoritmo será

[2] Descritas em https://sites.google.com/site/nonstationaryarchive/conjuntos de dados

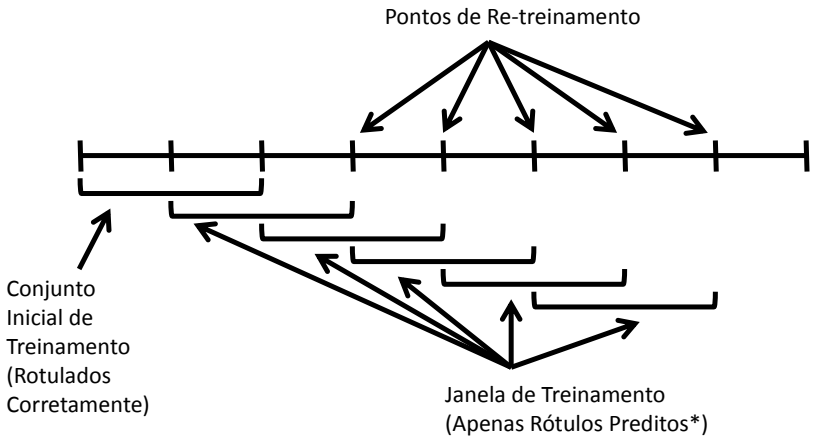


Figura 21 – Projeto experimental utilizado para atualização dos modelos de classificação ao longo do tempo.

retreinado com os últimos 500 eventos aos quais foi exposto, incluindo aqueles que o próprio algoritmo classificou. Desta forma, o algoritmo segue os seguintes passos, definindo-se as variáveis como J e R:

1. Algoritmo treinado com J exemplos rotulados corretamente (conjunto de treinamento).
2. Algoritmo classifica R eventos.
3. Modelo é atualizado utilizando-se J eventos mais recentes, incluindo os últimos R eventos rotulados pelo próprio algoritmo.
4. Repete-se os passos 2 e 3.

Um ponto importante sobre o projeto experimental é a não suposição da disponibilidade de rótulos verdadeiros durante a classificação do fluxo de dados. Tal suposição é comumente encontrada em diversos algoritmos de classificação de fluxos de dados, tais como *On Line Information Network* - OLIN (LAST, 2002) e *Ultra Fast Forest of Trees* - UFFT (GAMA *et al.*, 2004). Entretanto, para as aplicações consideradas neste apêndice, não se pode assumir a existência de rótulos verdadeiros durante o fluxo, condição conhedida como latência extrema (KREMPL, 2011; DYER; CAPO; POLIKAR, 2014; SOUZA; GAMA; BATISTA, 2014).

Foram utilizadas janelas de tamanhos variados, de acordo com o tamanho da própria base de dados em que os algoritimos foram aplicados. Além disso, também foram variados os pontos de re-treinamento, o que modificava a frequência com que o modelo de classificação era adaptado. Para o algotimo kNN foram utilizados valores para k que variavam entre 1 e 20. Já para o GMM, utilizou-se o número de gaussianas igual à quantidade de classes existente em cada base.

A Tabela 7 apresenta os resultados obtidos para quatro bases de *benchmark*. A coluna "Corretos"indica que o modelo foi atualizados pelos rótulos corretos e equivalem a latência de

verificação zero. Já a coluna "Preditos"indica que o modelo foi atualizado utilizando os rótulos preditos pelo próprio algoritmo e equivale a latência de verificação extrema.

Tabela 7 – Resultados de acurária - média (desvio padrão) ao longo de todo o fluxo - para os algoritmos GMM e kNN, com seus modelos atualizados utilizando-se os rótulos corretos ou apenas rótulos preditos.

	GMM		kNN	
Conjunto de dados	**Corretos**	**Preditos**	**Corretos**	**Preditos**
1CDT	99,93 (0,25)	99,92 (0,26)	99,88 (0,44)	99,49 (0,71)
2CDT	94,16 (3,09)	83,16 (8,52)	90,64 (3,79)	52,62 (9,59)
1CHT	99,54 (1,14)	99,54 (1,16)	99,4 (1,71)	93,46 (4,35)
2CHT	88,23 (3,86)	87,79 (3,92)	83,5 (4,13)	51,73 (7,76)

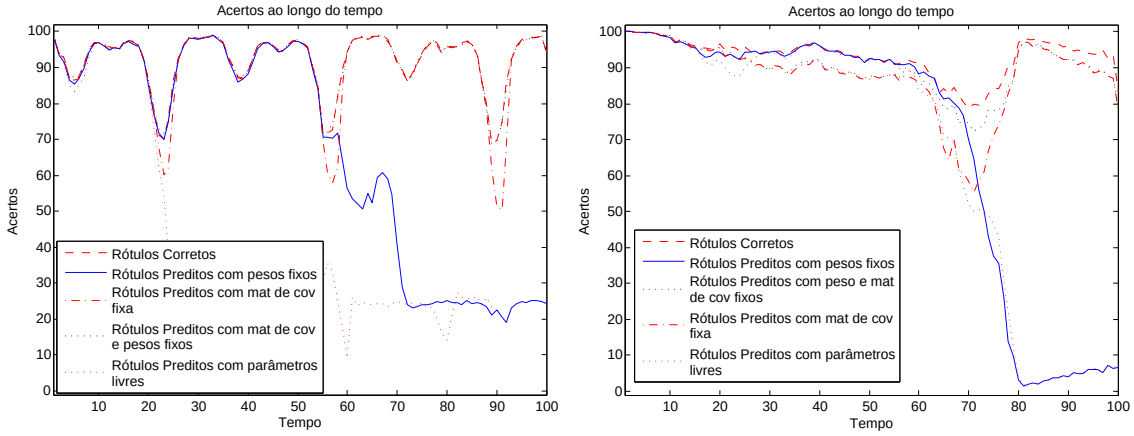
Nota-se que os resultados obtidos pelo classificador kNN são inferiores aos obtidos com o GMM, principalmente quando os rótulos corretos não estão disponíveis. Este é um ponto muito importante para o projeto, visto que busca-se uma técnica deve ser capaz de classificar os eventos com boa acurácia, mesmo na presença de mudanças de conceito, tendo como base um conjunto de treinamento inicial de exemplos já rotulados. Contudo, na fase de testes em diante, não assume-se que seja possível fornecer os rótulos corretos para o classificador, isto é, o algoritmo não será informado se suas predições estão corretas ou não.

A.3 A estratégia de fixação de parâmetros

Tendo esses resultados em mente, voltou-se a atenção dos estudos para o algoritmo GMM, de forma a aplicá-lo nas demais bases. Contudo, observou-se que em alguns casos onde a sobreposição das classes era maior o algoritmo acabava acumulando uma grande quantidade de erros ao longo do tempo, até que em um dado momento o modelo não fosse mais capaz de manter uma boa acurácia de classificação. Observando-se a evolução dos parâmetros do GMM ao longo do tempo, notou-se que aos poucos tanto as probabilidades a priori quanto a matriz de covariância das classes do modelo atualizado com rótulos preditos, se afastavam cada vez mais daqueles obtidos através dos rótulos corretos. Sendo assim, foram realizados experimentos mantendo um ou dois parâmetros do modelo fixados. A Figura 22 mostra os resultados de acurácia ao longo do tempo do algoritmo GMM fixando-se os parâmetros da matriz de covâriancia e/ou probabilidades a priori para as bases de dados *4CRE-V2* e *UG_2C_5D*.

Os melhores resultados obtidos foram observados quando as probabilidades a priori e a matriz de covariância eram mantidas fixas a partir do início do fluxo de dados. Uma comparação dos resultados obtidos com o GMM com parâmetros livres e com parâmetros fixados no início do fluxo de dados é mostrada na Tabela 8.

Contudo, tais resultados ainda não são suficientes para se afirmar que fixação dos parâmetros do algoritmo GMM sempre resultará em uma boa acurácia de classificação. O principal fator que impossibilita esta afirmação é o modo com que as bases de dados utilizadas para os testes iniciais foram geradas, já que estas são criadas a partir de distribuições normais



(a) 4CRE-V2 (b) UG_2C_5D

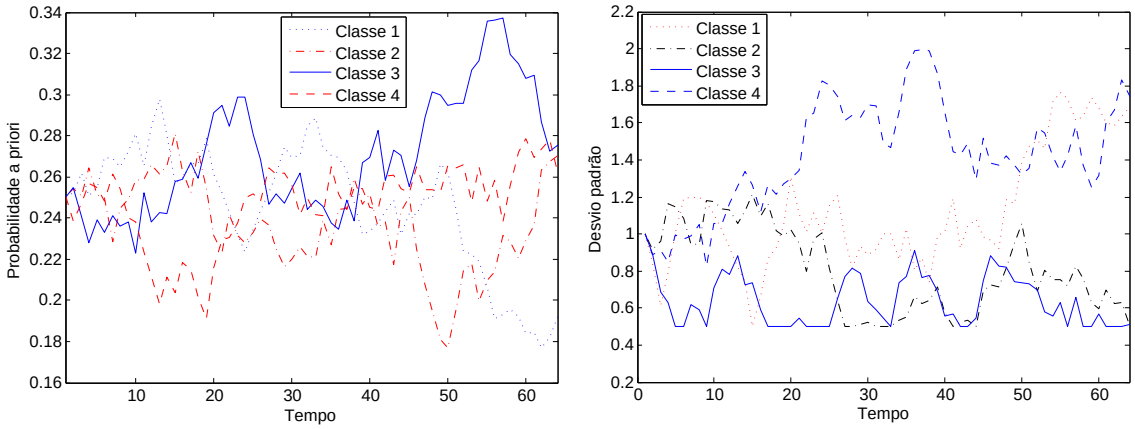
Figura 22 – Variação ao longo do tempo da acurácia do algoritmo GMM atualizado com rótulos corretos e atualizado com rótulos preditos mantendo-se os parâmetros fixos ou livres nos conjuntos de dados *4CRE-V2* (a) e *UG_2C_5D* (b).

Tabela 8 – Resultados de acurária - média (desvio padrão) ao longo de todo o fluxo - para o algoritmo GMM com parâmetros livres e fixados no início do fluxo, com seus modelos atualizados utilizando-se os rótulos corretos ou apenas rótulos preditos.

	GMM		**GMM com Parâmetros Fixos**	
Conjunto de dados	**Corretos**	**Preditos**	**Corretos**	**Preditos**
FG_2C_2D	81,88 (13,31)	62,65 (17,60)	81,88 (13,31)	73,81 (18,56)
GEARS_2C_2D	95,61 (1,19)	95,23 (1,29)	95,61 (1,19)	95,65 (1,18)
1CSurr	97,41 (2,58)	95,61 (5,36)	97,41 (2,58)	97,05 (2,88)
4CRE-V1	96,87 (6,23)	56,5 (29,35)	96,87 (6,23)	96,32 (7,28)
4CRE-V2	91,98 (7,77)	36,97 (30,00)	91,98 (7,77)	92,07 (7,72)
1CDT	99,93 (0,25)	99,92 (0,26)	99,93 (0,25)	99,92 (0,29)
2CDT	94,16 (3,09)	83,16 (8,52)	94,16 (3,09)	92,80 (3,67)
1CHT	99,54 (1,14)	99,54 (1,16)	99,54 (1,14)	99,59 (1,10)
2CHT	88,23 (3,86)	87,79 (3,92)	88,23 (3,86)	88,34 (3,63)
5CVT	87,61 (5,44)	50,80 (13,00)	87,61 (5,44)	83,17 (6,08)
UG_2C_2D	95,31 (3,90)	88,69 (13,63)	95,31 (3,90)	95,68 (3,84)
UG_2C_5D	93,17 (5,20)	67,8 (37,03)	93,17 (5,20)	89,83 (6,71)

que variam apenas suas médias ao longo do tempo. Neste cenário, fica claro que a fixação dos parâmetros de probabilidades a priori e matriz de covariância ao longo do tempo produziriam bons resultados, já que eles realmente não se alteram durante o fluxo. Tendo isto em mente, foram criados alguns conjuntos de dados também gerados a partir de distribuições normais, porém com a característica de variação em todos os seus parâmetros ao longo do tempo (médias, probabilidades a priori e matriz de covariância). Foram criadas variações das bases *4CRE-V2* e *1CSurr*, e uma base completamente nova. Elas foram chamadas de *4CRE-V2-var*, *1CSurr-var* e *4CRE-V3*, respectivamente, e em breve estarão disponíveis *online*. A variação ao longo do tempo das probabilidades a priori e desvios padrões da base *4CRE-V3* são mostradas na Figura 23

Após a geração destes conjuntos de dados, alguns testes foram feitos com o intuito de analisar o impacto da fixação de parâmetros do classificador GMM em um cenário onde estes parâmetros variam ao longo do tempo. Os resultados obtidos são exibidos na Tabela 9.



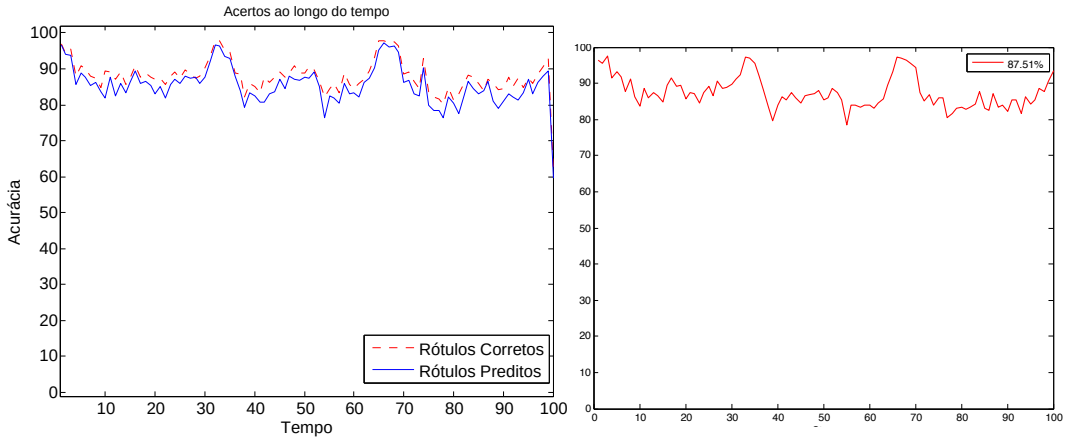
(a) Probabilidades a priori (b) Desvio Padrão

Figura 23 – Variação ao longo do tempo das probabilidades a priori (a) e desvios padrões (b) das quatro classes do conjunto de dados *4CRE-V3*.

Tabela 9 – Resultados de acurária - média (desvio padrão) ao longo de todo o fluxo - para o algoritmo GMM com parâmetros livres e fixados, com seus modelos atualizados utilizando-se os rótulos corretos ou apenas rótulos preditos, para as bases onde médias, probabilidades a priori e matriz de covariância variam ao longo do tempo.

	GMM		**GMM com Parâmetros Fixos**	
Conjunto de dados	**Corretos**	**Preditos**	**Corretos**	**Preditos**
1CSurr-var	97,96 (1,34)	97,48 (1,64)	97,96 (1,34)	96,84 (2,65)
4CRE-V2-var	79,89 (6,93)	23,96 (17,35)	79,89 (6,93)	77,25 (7,39)
4CRE-V3	88,02 (4,83)	63,55 (24,32)	88,02 (4,83)	85,55 (5,36)

Ao se comparar o desempenho do GMM utilizando a estratégia de mini-lote com fixação de parâmetros com o algoritmo SCARGC[3] (SOUZA V.M.A.; SILVA, 2015) percebe-se que o GMM apresenta resultados satisfatórios. A Figura 24 mostra essa comparação.



(a) GMM (b) SCARGC

Figura 24 – Variação ao longo do tempo da acurácia dos algoritmos GMM com mini-lote e parâmetros fixados (a) e SCARGC (b) na base de dados *4CRE-V3*.

[3] Stream Classification Algorithm Guided by Clustering

Tais resultados fortalecem ainda mais a proposta de fixação de parâmetros visto que, mesmo neste cenário onde estes não estão realmente fixos durante o fluxo de dados, o classificador consegue manter um bom desempenho. Entretanto, tal abordagem necessita ser melhor explorada, visto que aparenta ser contrária à ideia de evolução dos dados através das mudanças de conceito. Além disso, como mostra a Figura 25, esta técnica ainda apresenta resultados ruins em alguns

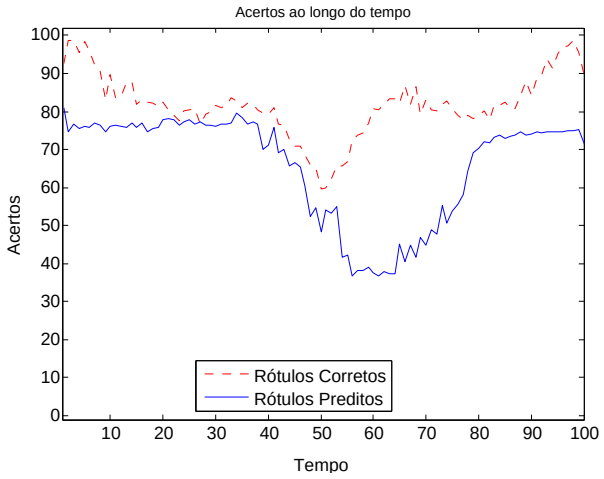


Figura 25 – Variação ao longo do tempo da acurácia dos algoritmos com parâmetros fixos e livres no conjunto de dados *MG_2C_2D*.

casos, como quando uma classe deve ser descrita por mais de uma componente gaussiana, tal como na base de dados *MG_2C_2D*.

Uma das possíveis formas de se resolver este problema é o uso de um número de componentes gaussianas variável para representar cada classe, tal como a estratégia do IGMM.

A.4 Considerações finais

A abordagem de fixação de parâmetros se mostrou promissora. No entando, a abordagem de fixar parâmetros de um algoritmo que visa acompanhar a evolução dos dados e se adaptar a mudanças de conceito parece contra-intuitiva, o que acaba levantando questionamentos sobre a eficiência de tal técnica.

Sendo assim outras abordagens podem ser testadas. Uma alternativa é utilizar uma forma de controlar a variação de um dado parâmetro a cada iteração, o que torna o algoritmo menos rígido. Isso faz sentido no cenário de mudanças de conceito incrementais, já que nestes casos o modelo não deve se alterar drasticamente em poucas iterações. Além disso, o problema encontrado quando classes são descritas por mais de uma distribuição gaussiana pode ser resolvido aumentando o número máximo de componentes permitido no modelo, de forma a torná-lo mais flexível.

REFERÊNCIAS

ADE, M.; GHRIET, P.; DESHMUKH, P.; SCOE&T, A. Methods for incremental learning: a survey. **International Journal of Data Mining & Knowledge Management Process**, v. 3, n. 4, p. 119–125, 2013. Citado 2 vezes nas páginas 36 e 37.

ARANDJELOVIC, O.; CIPOLLA, R. Incremental learning of temporally-coherent gaussian mixture models. **Society of Manufacturing Engineers (SME) Technical Papers**, IEEE, p. 1–1, 2006. Citado 2 vezes nas páginas 23 e 49.

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 31.

BERINGER, J.; HüLLERMEIER, E. An efficient algorithm for instance-based learning on data streams. In: **Proceedings of the 7th industrial conference on Advances in data mining: theoretical aspects and applications**. Berlin, Heidelberg: Springer-Verlag, 2007. (ICDM'07), p. 34–48. ISBN 978-3-540-73434-5. Disponível em: <http://dl.acm.org/citation.cfm?id=1770770.1770776>. Citado na página 36.

BIFET, A.; GAVALDA, R. Learning from time-changing data with adaptive windowing. In: **Proceedings of the 7th SIAM International Conference on Data Mining**. Minneapolis, Minnesota, USA: [s.n.], 2007. p. 443–448. Citado 3 vezes nas páginas 31, 37 e 71.

BIFET, A.; GAVALDÀ, R. Adaptive learning from evolving data streams. In: **Advances in Intelligent Data Analysis VIII**. [S.l.]: Springer, 2009. p. 249–260. Citado 3 vezes nas páginas 22, 37 e 52.

BIFET, A.; HOLMES, G.; KIRKBY, R.; PFAHRINGER, B. **Data Stream Mining: A Practical Approach**. [S.l.], 2011. Citado 2 vezes nas páginas 22 e 52.

BIFET, A.; PFAHRINGER, B.; READ, J.; HOLMES, G. Efficient data stream classification via probabilistic adaptive windows. In: **Proceedings of the 28th Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2013. (SAC '13), p. 801–806. ISBN 978-1-4503-1656-9. Disponível em: <http://doi.acm.org/10.1145/2480362.2480516>. Citado 2 vezes nas páginas 25 e 26.

BIMBOT, F.; BONASTRE, J.-F.; FREDOUILLE, C.; GRAVIER, G.; MAGRIN-CHAGNOLLEAU, I.; MEIGNIER, S.; MERLIN, T.; ORTEGA-GARCÍA, J.; PETROVSKA-DELACRÉTAZ, D.; REYNOLDS, D. A. A tutorial on text-independent speaker verification. **EURASIP journal on applied signal processing**, Hindawi Publishing Corp., v. 2004, p. 430–451, 2004. Citado na página 68.

BOUCHACHIA, A.; VANARET, C. Incremental learning based on growing gaussian mixture models. In: IEEE. **Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on**. [S.l.], 2011. v. 2, p. 47–52. Citado 2 vezes nas páginas 21 e 49.

BRZEZIŃSKI, D.; STEFANOWSKI, J. Accuracy updated ensemble for data streams with concept drift. In: **Hybrid Artificial Intelligent Systems**. [S.l.]: Springer, 2011. p. 155–163. Citado 2 vezes nas páginas 22 e 53.

CESA-BIANCHI, N.; GENTILE, C.; ZANIBONI, L. Worst-case analysis of selective sampling for linear classification. **The Journal of Machine Learning Research**, JMLR. org, v. 7, p. 1205–1230, 2006. Citado 2 vezes nas páginas 22 e 53.

DAWID, A. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. **Journal of the Royal Statistical Society. Series A (General)**, JSTOR, n. 147, p. 278–292, 1984. Citado na página 33.

DITZLER, G.; POLIKAR, R. Incremental learning of concept drift from streaming imbalanced data. **Knowledge and Data Engineering, IEEE Transactions on**, IEEE, v. 25, n. 10, p. 2283–2301, 2013. Citado na página 53.

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 36.

DRIES, A.; RÜCKERT, U. Adaptive concept drift detection. **Statistical Analysis and Data Mining: The ASA Data Science Journal**, Wiley Online Library, v. 2, n. 5-6, p. 311–327, 2009. Citado na página 31.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern Classification**. [S.l.]: Wiley and Sons, 2000. Citado 2 vezes nas páginas 27 e 40.

DYER, K. B.; CAPO, R.; POLIKAR, R. Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. **IEEE Trans. Neural Netw. Learning Syst.**, v. 25, n. 1, p. 12–26, 2014. Citado 4 vezes nas páginas 22, 23, 53 e 73.

ELWELL, R.; POLIKAR, R. Incremental learning of concept drift in nonstationary environments. **IEEE Transactions on Neural Networks**, v. 22, n. 10, p. 1517–1531, oct. 2011. ISSN 1045-9227. Citado na página 21.

ENGEL, P. M.; HEINEN, M. R. Incremental learning of multivariate gaussian mixture models. In: **Proceedings of the 20th Brazilian Conference on Advances in Artificial Intelligence**. Berlin, Heidelberg: Springer-Verlag, 2010. (SBIA'10), p. 82–91. ISBN 3-642-16137-5, 978-3-642-16137-7. Disponível em: <http://dl.acm.org/citation.cfm?id=1929622.1929633>. Citado 14 vezes nas páginas 21, 22, 23, 39, 42, 44, 46, 48, 50, 51, 58, 59, 61 e 66.

GABER, M.; ZASLAVSKY, A.; KRISHNASWAMY, S. Mining data streams: a review. **ACM Sigmod Record**, ACM, v. 34, n. 2, p. 18–26, 2005. Citado na página 21.

GAMA, J. **Knowledge Discovery from Data Streams**. 1st. ed. [S.l.]: Chapman & Hall/CRC, 2010. 233 p. ISBN 978-1-4398-2611-9. Citado 5 vezes nas páginas 21, 30, 31, 32 e 33.

GAMA, J. a.; MEDAS, P.; CASTILLO, G.; RODRIGUES, P. Learning with drift detection. In: BAZZAN, A.; LABIDI, S. (Ed.). **Advances in Artificial Intelligence - SBIA 2004**. Springer Verlag, 2004. (Lecture Notes in Computer Science, 17), p. 286–295. Disponível em: <http://dx.doi.org/10.1007/978-3-540-28645-5_29>. Citado 4 vezes nas páginas 31, 71, 72 e 73.

GEHRKE, J.; KORN, F.; SRIVASTAVA, D. On computing correlated aggregates over continual data streams. **ACM SIGMOD**, ACM, v. 30, n. 2, p. 13–24, 2001. Citado na página 30.

GHOLIPOUR, A.; HOSSEINI, M. J.; BEIGY, H. An adaptive regression tree for non-stationary data streams. In: **Proceedings of the 28th Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2013. (SAC '13), p. 815–817. ISBN 978-1-4503-1656-9. Disponível em: <http://doi.acm.org/10.1145/2480362.2480519>. Citado na página 26.

GOLAB, L.; OZSU, M. T. Issues in data stream management. **SIGMOD Record**, v. 32, p. 5–14, 2003. Citado na página 25.

GOMES, H. M. **Teoria De Redes Sociais Aplicada Ao Problema de Classificação Online Com Mudança de Conceito**. Tese (Dissertação de Mestrado) — Pontifícia Universidade Católica do Paraná, Curitiba, Brasil, 2012. Citado 4 vezes nas páginas 26, 27, 28 e 29.

HARRIES, M.; WALES, N. **Splice-2 comparative evaluation: Electricity pricing**. [S.l.], 1999. Citado na página 57.

HE, H.; CHEN, S. Imorl: Incremental multiple-object recognition and localization. **Neural Networks, IEEE Transactions on**, IEEE, v. 19, n. 10, p. 1727–1738, 2008. Citado na página 36.

HULTEN, G.; SPENCER, L.; DOMINGOS, P. Mining time-changing data streams. In: ACM. **Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2001. p. 97–106. Citado 2 vezes nas páginas 22 e 52.

JOHN, G. H.; LANGLEY, P. Estimating continuous distributions in bayesian classifiers. In: MORGAN KAUFMANN PUBLISHERS INC. **Proceedings of the Eleventh conference on Uncertainty in artificial intelligence**. [S.l.], 1995. p. 338–345. Citado na página 36.

JOSHI, P.; KULKARNI, P. Incremental learning: areas and methods—a survey. **International Journal of Data Mining & Knowledge Management Process**, Academy & Industry Research Collaboration Center(AIRCC), v. 2, n. 5, p. 43–51, 2012. Citado 2 vezes nas páginas 35 e 37.

JÚNIOR, J. R. B. **Classificação de Dados Estacionários e não Estacionários Baseada em Grafos**. Tese (Tese de Doutorado) — Instituto de Ciências Matemáticas e de Computacão, Universidade de São Paulo, 2010. Citado 3 vezes nas páginas 21, 28 e 29.

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 27.

KILLOURHY, K.; MAXION, R. Why did my detector do that?! In: SPRINGER. **Recent Advances in Intrusion Detection**. [S.l.], 2010. p. 256–276. Citado na página 58.

KREMPL, G. The algorithm apt to classify in concurrence of latency and drift. In: GAMA, J.; BRADLEY, E.; HOLLMéN, J. (Ed.). **IDA**. Springer, 2011. (Lecture Notes in Computer Science, v. 7014), p. 222–233. ISBN 978-3-642-24799-6. Disponível em: <http://dblp.uni-trier.de/db/conf/ida/ida2011.html#Krempl11>. Citado 3 vezes nas páginas 22, 23 e 73.

KRISTAN, M.; SKOČAJ, D.; LEONARDIS, A. Online kernel density estimation for interactive learning. **Image and Vision Computing**, Elsevier, v. 28, n. 7, p. 1106–1116, 2010. Citado 3 vezes nas páginas 22, 23 e 49.

KUMAR, N.; SATOOR, S.; BUCK, I. Fast parallel expectation maximization for gaussian mixture models on gpus using cuda. In: IEEE. **High Performance Computing and Communications, 2009. HPCC'09. 11th IEEE International Conference on**. [S.l.], 2009. p. 103–109. Citado na página 40.

KUNCHEVA, L. I. Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. In: **Proceedings of the 2nd Workshop SUEMA**. [S.l.: s.n.], 2008. v. 2008, p. 5–10. Citado na página 71.

LANGE, S.; GRIESER, G. On the power of incremental learning. **Theoretical Computer Science**, Elsevier, v. 288, n. 2, p. 277–307, 2002. Citado na página 35.

LANGLEY, P. **Learning in humans and machines: Towards an interdisciplinary learning science**. [S.l.]: Oxford, U.K. ; New York : Pergamon, 1996, 1995. Capítulo 9 - Order Effects in Incremental Learning p. Citado na página 37.

LAST, M. Online classification of nonstationary data streams. **Intelligent Data Analysis**, IOS Press, v. 6, n. 2, p. 129–147, 2002. Citado 2 vezes nas páginas 72 e 73.

MALHEIRO, R. M. d. S. **Sistemas de Classificação Automática em Gêneros Musicais**. Tese (Tese de Doutorado) — Departamento de Engenharia Informática, Faculdade de Ciências e Tecnologia, Universidade de Coimbra, 2003. Citado na página 39.

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 2 vezes nas páginas 71 e 72.

MUHLBAIER, M. D.; TOPALIS, A.; POLIKAR, R. Learn. nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes. **Neural Networks, IEEE Transactions on**, IEEE, v. 20, n. 1, p. 152–168, 2009. Citado na página 36.

NARASIMHAMURTHY, A.; KUNCHEVA, L. A framework for generating data to simulate changing environments. In: ACTA PRESS. **Proceedings of the 25th conference on Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications**. [S.l.], 2007. v. 549, p. 389. Citado na página 27.

NISHIDA, K.; YAMAUCHI, K. Detecting concept drift using statistical testing. In: SPRINGER. **Discovery Science**. [S.l.], 2007. p. 264–269. Citado 2 vezes nas páginas 31 e 71.

PATIST, J. P. Optimal window change detection. In: IEEE. **Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on**. [S.l.], 2007. p. 557–562. Citado na página 31.

PINTO, C. M. S. **Algoritmos Incrementais para Aprendizagem Bayesiana**. Tese (Tese de Doutorado) — Departamento de Ciência de Computadores, Faculdade de Economia da Universidade do Porto, 2005. Citado 2 vezes nas páginas 36 e 37.

READ, J.; BIFET, A.; PFAHRINGER, B.; HOLMES, G. Batch-incremental versus instance-incremental learning in dynamic and evolving data. In: **Advances in Intelligent Data Analysis XI**. [S.l.]: Springer, 2012. p. 313–323. Citado 3 vezes nas páginas 25, 35 e 36.

REDNER, R. A.; WALKER, H. F. Mixture densities, maximum likelihood and the em algorithm. **SIAM review**, SIAM, v. 26, n. 2, p. 195–239, 1984. Citado na página 40.

REYNOLDS, D. Gaussian mixture model. **Encyclopedia of Biometrics**, p. 659–663, July 2009. Citado 3 vezes nas páginas 39, 40 e 41.

SEBASTIÃO, R.; GAMA, J. A study on change detection methods. In: **Proceedings of the Fourteenth Portuguese Conference on Artifcial Intelligence**. Aviero, Portugal: [s.n.], 2009. p. 353–364. Citado na página 29.

SHARMA, A. A note on batch and incremental learnability. **Journal of Computer and System Sciences**, Elsevier, v. 56, n. 3, p. 272–276, 1998. Citado na página 35.

SONG, M.; WANG, H. Highly efficient incremental estimation of gaussian mixture models for online data stream clustering. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. **Defense and Security**. [S.l.], 2005. p. 174–183. Citado 2 vezes nas páginas 22 e 49.

SOUZA, V. M. A. **Nonstationary Environments - Archive**. 2015. <https://sites.google.com/site/nonstationaryarchive/home>. [Online; acessado 24-Abril-2015]. Citado 2 vezes nas páginas 53 e 57.

SOUZA, V. M. A. de; GAMA, J.; BATISTA, G. E. A. P. A. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: **Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)**. [S.l.: s.n.], 2014. Citado 4 vezes nas páginas 22, 23, 72 e 73.

SOUZA V.M.A.; SILVA, D. G. J. B. G. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: . [S.l.: s.n.], 2015. p. 1–9. Citado 3 vezes nas páginas 71, 72 e 76.

SU, B.; SHEN, Y.-D.; XU, W. Modeling concept drift from the perspective of classifiers. In: IEEE. **Cybernetics and Intelligent Systems, 2008 IEEE Conference on**. [S.l.], 2008. p. 1055–1060. Citado na página 31.

TSYMBAL, A. **The problem of concept drift: Definitions and related work**. [S.l.], 2004. Citado 4 vezes nas páginas 26, 27, 28 e 42.

WANG, H.; FAN, W.; YU, P.; HAN, J. Mining concept-drifting data streams using ensemble classifiers. In: ACM. **Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2003. p. 226–235. Citado na página 37.

WIDMER, G.; KUBAT, M. Effective learning in dynamic environments by explicit context tracking. In: SPRINGER. **Proceedings of European Conference on Machine Learning**. [S.l.], 1993. p. 227–243. Citado na página 27.

____. Learning in the presence of concept drift and hidden contexts. **Machine learning**, Springer, v. 23, n. 1, p. 69–101, 1996. Citado na página 28.

ZHANG, P.; GAO, B. J.; ZHU, X.; GUO, L. Enabling fast lazy learning for data streams. In: IEEE. **Data Mining (ICDM), 2011 IEEE 11th International Conference on**. [S.l.], 2011. p. 932–941. Citado na página 36.

ZHOU, Z.-H.; CHEN, Z.-Q. Hybrid decision tree. **Knowledge-based systems**, Elsevier, v. 15, n. 8, p. 515–528, 2002. Citado na página 35.

ZLIOBAITE, I. **Learning under concept drift: an overview**. Lithuania, 2009. Citado 5 vezes nas páginas 21, 26, 27, 28 e 29.

ŽLIOBAITĖ, I.; BIFET, A.; PFAHRINGER, B.; HOLMES, G. Active learning with evolving streaming data. In: **Machine Learning and Knowledge Discovery in Databases**. [S.l.]: Springer, 2011. p. 597–612. Citado 2 vezes nas páginas 22 e 53.