EDUARDO FERREIRA JOSÉ

DESENVOLVIMENTO DE TÉCNICAS DE APRENDIZAGEM PERSONALIZADA ADAPTATIVA PARA SISTEMAS INCREMENTAIS DE RECOMENDAÇÃO

Curitiba - PR, Brasil

2020

EDUARDO FERREIRA JOSÉ

DESENVOLVIMENTO DE TÉCNICAS DE APRENDIZAGEM PERSONALIZADA ADAPTATIVA PARA SISTEMAS INCREMENTAIS DE RECOMENDAÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito para obtenção do título de mestre em Informática.

Pontifícia Universidade Católica do Paraná - PUCPR

Programa de Pós-Graduação em Informática - PPGIa

Orientador: Prof. Dr. Fabrício Enembreck

Coorientador: Prof. Dr. Jean Paul Barddal

Curitiba - PR, Brasil

2020

Agradecimentos

Aos meus orientadores Prof. Dr. Fabrício Enembreck e Prof. Dr. Jean Paul Barddal, que tornaram possível a realização dessa pesquisa. Prof. Fabrício sempre com críticas assertivas e comentários relevantes, conseguiu me apresentar o pensamento científico e desenvolver a minha capacidade analítica ao decorrer de nossa caminhada. Prof. Jean sempre que necessário me amparando, com uma enorme disponibilidade, apresentando o caminho em momentos de dificuldade. Um profundo agradecimento a ambos, que conseguiram contribuir para a transformação de devaneios em ideias, e de ideias em resultados.

A Profª. Dr. Deborah Ribeiro Carvalho, que ofereceu incontáveis oportunidades para meu desenvolvimento, sempre solícita para olhar de outra perspectiva e contribuir de inúmeras maneiras para a pesquisa realizada.

A Profª. Dr. Cleybe H. Vieira e toda equipe da Coordenação de Iniciação Científica, por desafiarem alunos e permitirem projetos de tal magnitude.

A Associação Paranaense de Cultura e a Pontifícia Universidade Católica pelo suporte financeiro cedido ao projeto.

Por fim, a Ana Jéssica pelo apoio psicológico oferecido nesse momento de dedicação e a minha família pelo incentivo dado para as minhas conquistas, sempre me apoiando e torcendo para o êxito da pesquisa. Sem a base de tudo, nada disso seria possível.

"Passion is what gets you through the hardest times that might otherwise make strong men weak, or make you give up."
Neil deGrasse Tyson

Resumo

Sistemas adaptativos de recomendação mostram cada vez mais a sua necessidade, visto que o processo de definição de perfil de usuário tende a ser um problema dinâmico. O objetivo é atualizar os modelos de recomendação à medida que novas interações ocorrem, adaptando-se rapidamente a desvios no comportamento e nos desejos do usuário e no público-alvo do item. No entanto, os algoritmos de recomendação existentes geralmente não apresentam um bom desempenho durante as mudanças de conceito e acabam demorando muito para se adaptar às alterações, gerando atualizações sub-ideais devido a ter o mesmo impacto para todos os perfis. Neste trabalho, propomos dois algoritmos, ADADRIFT e BP, para lidar com mudanças de conceito baseados em usuários e itens em sistemas de recomendação adaptativos, usando taxas de aprendizado personalizadas com base em estatísticas de perfil. Enquanto a primeira abordagem utiliza duas médias móveis para controlar o erro e aumentar ou diminuir a taxa de aprendizagem, a última cria momentos de aprendizagem elevada com detectores de mudança de conceito, sendo avaliado o EDDM e o DDM para esse trabalho. As experiências usando sistemas de recomendação baseados em fluxo (ISGD e BRISMF) em quatro conjuntos de dados diferentes mostram que o ADADRIFT ultrapassa o ADADELTA e esses baselines com melhorias significativas nas taxas de recomendação. Já o resultado do BP é mais tímido, mostrando um ganho constante em relação ao baseline, mas não tão significativo. Os melhores resultados aparecem quando os fluxos de dados têm um longo histórico de interações dos usuários ou itens e os desvios se tornam visíveis.

Palavras-chave: Sistemas de Recomendação, Mineração de Fluxos de Dados, Sistemas Incrementais de Recomendação, Aprendizagem Adaptativa.

Abstract

Adaptive recommendation systems increasingly show their need, as the user profile definition process tends to be a dynamic problem. Its goal is to update recommendation models as new interactions occur, quickly adapting to concept drifts in user behavior and desires and the item's target audience. However, existing recommendation algorithms usually do not perform well during drifts, as they take long to adapt to changes, or these updates are suboptimal since they account for all profiles' preferences equally, which is often untrue as each individual and its changes are unique. In this work, we propose two algorithms, ADADRIFT, and BP, to deal with user-based and item-based concept drift in adaptive recommendation systems, using personalized learning rates based on profile statistics. While the former approach uses two moving averages to control the error and increase or decrease the learning rate, the latter creates moments of high learning with a concept drift detector, being evaluated with EDDM and DDM for this work. Experiments using stream-based recommendation systems (ISGD and BRISMF) on four different datasets show that ADADRIFT surpasses ADADELTA and these baselines with significant improvements in recommendation rates. BP's result is timider, showing a steady gain concerning the baseline, but not as significant. The best results appear when data streams have a long history of user or item interactions and drifts become visible. Experimentation in this work highlights the importance of dealing with concept drift in recommendation systems.

Keywords: Recommender Systems, Data Stream Mining, Incremental Recommender Systems, Adaptive Learning.

Lista de ilustrações

Lista de tabelas

Índice de algoritmos

Lista de abreviaturas e siglas

BP	Background Profiling
MF	Matrix Factorization
BRISMF	Biased Regularized Incremental Simultaneos MF
SGD	Stochastic Gradient Descent
ISGD	Incremental SGD
UI	Forget Unpopular Items
UFF	User Factor Fading
SD-UFF	SD-based User Factor Fading
T5	Transformed to positive-only with rating 5
PL	Personality
NF	Netflix
NF5	Netflix com somente *ratings* iguais a 5.
LFM	Last FM 1K Users
CD	Tratamento para Mudança de Conceito
DDM	Drift Detection Method
EDDM	Early Drift Detection Method
ADWIN	Adaptative Sliding Window Algorithm
RMSE	Root Mean Squared Error
FluRS	Flu-* (Flux, Fluid, Fluent) RecommenderSystems
CF	Collaborative Filtering
CBF	Content-based Filtering

Lista de símbolos

$p_t(y\|X)$	Probabilidade no instante t de y dado o conjunto X
G_n	Preferência musical n
U_n	Usuário n
I_n	Item n
A_u	Matriz de fatoração dos usuários A na posição do usuário u
λ	Parâmetro regularizador da fatoração de matriz
η	Taxa de aprendizagem
k	Número de fatores latentes da matriz
θ	Threshold de conversão para *positive-only*
R	*Rating* real
$\hat{R}$	*Rating* predito
T	Conjunto de teste
t	Instante de tempo represetados em número inteiro.
ϵ	Valor insignificante para evitar erros de divisão
ρ	Taxa de decaimento
α	Parâmetro de impacto do método
δ_L	Tamanho da janela de longo prazo
δ_S	Tamanho da janela de curto prazo
μ_L	Média móvel de longo prazo
σ_L	Desvio padrão da média de longo prazo
μ_S	Média móvel de curto prazo
p	Perfil representando tanto usuário quanto item p
η_p	Taxa de aprendizagem para o perfil p

Q	Matriz onde será aplicada a técnica. Pode ser tanto A quanto B
S	Matriz complementar a Q
D	Detectores de mudança
C	Matriz auxiliar do Background Profiling

Sumário

Parte I

Preâmbulo

1 Introdução

Com o consumo diário dos usuários em plataformas on-line, sejam plataformas de e-commerce (e.g. eBay, Amazon) ou de entretenimento (e.g. Netflix, Spotify, YouTube), a experiência do usuário se tornou algo relevante. É vital fornecer a cada usuário o que ele está procurando com rapidez e eficiência, diminuindo o tempo perdido e proporcionando uma melhor experiência com a plataforma.

Guiados pelos avanços na área de Ciência de Dados e estruturados na grande quantidade de dados gerada pela navegação do usuário, pesquisadores e profissionais dedicam seus esforços à identificação do perfil do usuário. Essa tarefa é chamada *profiling* (VELOSO et al., 2018; BOBADILLA et al., 2013), de maneira a aprimorar a experiência na plataforma ao facilitar a busca pelos produtos desejados por meio das recomendações oferecidas pela plataforma. Depois que os perfis de usuário são criados, eles podem ser usados para fazer recomendações.

Grandes competições realizadas na área de recomendações impulsionadas principalmente pelo alto financiamento de empresas como Netflix[1] (BENNETT; LANNING, 2007) e desafios anuais associadas à conferência, como RecSys Challenge[2] ou a WSDM Cup[3] levaram ao desenvolvimento de novos algoritmos e a avanços significativos nos sistemas de recomendação.

A maioria dos esforços nos sistemas de recomendação foi dedicada a cenários *batch*, ou seja, onde os modelos são treinados uma vez e usados para oferecer recomendações. No entanto, os modelos *batch* possuem um problema intrínseco: a falta de adaptação ao longo do tempo (JORGE et al., 2017). Portanto, é necessário treinar constantemente novos modelos para manter a precisão competitiva, especialmente quando o comportamento e as preferências do usuário mudam ao longo do tempo e novos itens são incorporados ao sistema.

Portanto, são necessários modelos de recomendação incrementais que observem as tendências do usuário e se adaptem a elas sem nenhum esforço adicional e períodos de baixo desempenho.

Essas atualizações realizadas de maneira recorrente ao modelo seguem um mesmo impacto no perfil, ou seja, ignoram cenários onde necessitem de uma aprendizagem mais rápida (e.g. usuário novo na plataforma ou mudança de perfil de consumo) ou uma aprendizagem mais lenta (e.g. perfil estável e grande quantidade de acertos para o usuário).

[1] <https://www.netflix.com>
[2] <https://recsys.acm.org/>
[3] <http://www.wsdm-conference.org/>

É possível encontrar uma série de métodos de aprendizado adaptativo em outros campos da literatura, mas nenhum deles aborda diretamente o problema de mudança de conceito, sendo a grande maioria desenvolvidos para cenários *batch*, não sendo adaptados para relações temporais. Suas adaptações visam compreender o conceito de maneira mais rápida e precisa, mas muitas técnicas não estão preparadas para a mudança constante desse conceito, o que é muito característico dos ambientes de streaming.

Além disso, as técnicas existentes para o aprendizado adaptativo em geral não são otimizadas para sistemas de recomendação, onde é necessário lidar com um grande número de usuários e as adaptações precisam agir individualmente. A aprendizagem adaptativa com abordagens individuais (orientado ao perfil do usuário ou item) não é necessária nos cenários clássicos, com alguns estudos similares para sistemas de recomendação.

Neste ponto, é importante mencionar sobre *sequence-aware recommender systems*. Conforme exposto em Quadrana, Cremonesi e Jannach (2018), um dos problemas enfrentado por esse tipo de sistemas de recomendação é analisar tendências individuais com *collaborative-filtering*. Esse será o problema endereçado nessa dissertação.

As mudanças no perfil que ocorrem geralmente em sistemas de recomendação são causadas por novas preferências do usuário, popularização de produtos, rejeição a antigas preferências ou até mesmo compartilhamento de conta entre diferentes pessoas. Visto a ausência de técnicas similares endereçadas a sistemas de recomendação, a proposta desse trabalho visa explorar soluções para aprendizagem personalizada adaptativa, ou seja, definir a taxa de aprendizagem de acordo com a necessidade observada pelo próprio algoritmo, em diferentes cenários de recomendação.

1.1 Motivação

Tendo em vista a grande preocupação na área de mineração de fluxos de dados com as mudanças de conceito (GAMA et al., 2013), a motivação dessa dissertação consiste em observar se todos os perfis se comportam de maneira similiar ao decorrer do tempo, ou se é necessário fazer um tratamento para esse tipo de problema em sistemas incrementais de recomendação.

Sabendo disso, essa dissertação irá propor e validar duas alternativas para mitigar o problema supramencionado. Outro importante aspecto é que as técnicas aqui apresentadas sejam compatíveis com os vários algoritmos de fatoração de matriz da literatura, e por isso todas as técnicas serão analisadas com dois dos principais algoritmos, BRISMF e ISGD.

1.2 Objetivos

A questão de pesquisa dessa dissertação consiste em *"Técnicas de aprendizagem personalizada adaptativa apresentam um maior desempenho em cenários de instabilidade de perfis de usuários/itens de sistemas de recomendação?"*. A partir dessa questão foi definido o objetivo principal, que consiste em explorar cenários onde o ajuste da taxa de aprendizagem possa ser realizado, de maneira a se adequar ao perfil em questão.

1.2.1 Objetivos Específicos

Com base no objetivo principal, os seguintes objetivos específicos foram definidos para essa pesquisa:

(i) Analisar o comportamento das mudanças de perfil identificado em bases de dados de benchmark disponíveis na literatura sobre sistemas de recomendação;

(ii) Encontrar cenários onde o problema de mudança de conceito é encontrado;

(iii) Comparar técnicas de aprendizagem adaptativa, voltadas a mudança de conceito, com métodos estado da arte usados em *batch*;

(iv) Desenvolver soluções para aprendizagem em momento de instabilidade de perfil.

1.3 Hipóteses

Considerando a proposta de aprendizagem adaptativa, as seguintes hipóteses foram levantadas para a sua comprovação.

H1. Existe um ganho significativo na personalização e adaptação da taxa de aprendizagem em sistemas incrementais de recomendação.

H2. É possível observar uma mudança de conceito de um usuário utilizando da estabilidade do perfil.

1.4 Estrutura do documento

A dissertação estará seccionada em três partes. A primeira parte diz respeito aos itens: (i) introdução e apresentação do documento; (ii) revisão da literatura pertinente ao trabalho; (iii) protocolo de validação dos algoritmos futuramente apresentados e (vi) trabalhos relacionados à aprendizagem adaptativa. Os seguintes capítulos compõem a primeira parte do trabalho:

- Capítulo 1, aqui apresentado, visa oferecer ao leitor um panorama geral sobre o contexto no qual se insere este trabalho de pesquisa, apresentando as principais motivações da área de recomendações, definindo os objetivos e o problema de pesquisa.

- Capítulo 2 aprofunda o referencial teórico inicial descrito no Capítulo 1, focando especialmente em técnicas de detecção e adaptação a mudanças de conceito e sistemas incrementais de recomendação, ambos necessários para compreensão do trabalho.

- Capítulo 3 apresenta informações relevantes acerca de todo o processo de validação dos algoritmos propostos. Apresentará o ambiente de experimentação, o protocolo utilizado, as bases definidas e métricas utilizadas para validação.

- Capítulo 4 apresenta os algoritmos pertinentes encontrados na literatura ao se tratar de aprendizagem adaptativa e levanta as lacunas deixadas pelos mesmos.

A segunda parte irá trazer as principais contribuições da dissertação. As seguintes técnicas serão propostas: (i) Background Profiling; (ii) ADADRIFT. Esse conteúdo será veículado nos seguintes capítulos:

- Capítulo 5 apresentará o Background Profiling, sendo essa outra técnica de aprendizagem adaptativa voltada para mudança de conceito. Da mesma forma, será analisado seu custo computacional, resultados e seleção dos parâmetros utilizando o protocolo de validação apresentado no Capítulo 3.

- No Capítulo 6 é onde será proposta a primeira técnica de aprendizagem adaptativa, voltada para mudança de conceito, chamada de ADADRIFT. Será feita uma apresentação do contexto em que a técnica deve ser utilizada, seu custo computacional, resultados preliminares e seleção de parâmetros.

Por fim, na última parte será feita uma (i) comparação dos algoritmos anteriormente propostos e (ii) apresentadas as considerações finais do trabalho, junto com uma discussão sobre trabalhos futuros.

- Capítulo 7 traz uma visão geral dos algoritmos propostos, visando uma comparação entre os dois e seleção dos mais adequados para cada cenário, junto com as considerações finais.

- Capítulo 8 apresenta, por fim, as possibilidades para trabalhos futuros abertos com as constribuições dadas.

1.5 Considerações do Capítulo

O exposto capítulo apresenta as ideias iniciais sobre a área de recomendação e a estruturação da pesquisa em si. A motivação do trabalho, como anteriormente apresentado, é devido aos bons resultados obtidos por técnicas que lidem com mudança de conceito em mineração de fluxos de dados, e a ausência de métodos que preconizem tal problema para modelos adaptativos de recomendação.

Considerando isso, o objetivo dessa pesquisa é desenvolver técnicas que realizem o ajuste da taxa de aprendizagem de maneira personalizada, ou seja, façam com que a aprendizagem se ajuste de maneira individual ao perfil.

O Capítulo 2 irá trazer os principais conceitos para o entendimento do trabalho. Para isso, serão separadas em duas principais frentes: (i) Mineração de Fluxos de Dados e (ii) Sistemas de Recomendação. A primeira parte irá trazer os conceitos já conhecidos da área de Mineração de Fluxos de Dados pertinentes ao cenário de recomendação, sendo já utilizados para o entendimento da segunda parte, de Sistemas de Recomendação.

2 Revisão da Literatura

Para melhor estruturação da Revisão da Literatura presente nesse trabalho, a mesma foi separada em duas principais seções, sendo elas Detecção e Adaptação a Mudanças de Conceito, abrangendo a área de Mineração de Fluxos de dados de uma maneira geral e adentrando na definição e taxonomia de Mudanças de Conceito, e a outra seção sobre Sistemas de Recomendação, abrangendo os conceitos importantes da área e identificando conceitos paralelos necessários para a compreensão do método proposto.

2.1 Detecção e Adaptação a Mudanças de Conceito

Mineração de fluxos de dados é uma subárea da mineração de dados onde sua grande diferença é que a entrada para os modelos de predição se apresenta em formato de um fluxo contínuo e possivelmente infinito de dados. Essa característica implica diretamente que os modelos lidem com a falta de memória e que se ajustem com poucas iterações, havendo geralmente uma única iteração para atualização interna por instância.

O processamento de dados em fluxo provoca outro ponto crucial para a área, a existência da possibilidade de mudança ao decorrer do tempo. Uma mudança de conceito, definida na Equação 2.1, consiste na predição de classe diferente para o mesmo conjunto de atributos X em diferentes instantes de tempo (GAMA et al., 2013). Para isso, $p_{t_0}(y \mid X)$ é a função de probabilidade, no instante de tempo t_0, para a classe y dado o conjunto de atributos X.

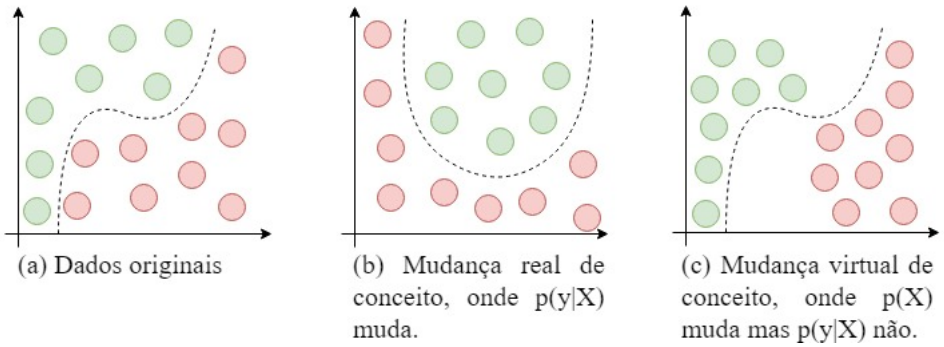
$$\exists X : p_{t_0}(y \mid X) \neq p_{t_1}(y \mid X) \tag{2.1}$$

2.1.1 Mudanças Reais e Virtuais de Conceito

A mudança de conceito ainda pode se apresentar de diferentes maneiras. Uma mudança virtual de conceito consiste de uma alteração na distribuição de classes para diferentes instantes de tempo. Apesar dessa alteração, o conceito ainda se mantém o mesmo, não provocando a definição apresentada na Equação 2.1.

Por outro lado, é possível observar a mudança real de conceito. Essa mudança consiste na alteração do conceito vigente do modelo em diferentes instantes de tempo, ou seja, para o mesmo conjunto de atributos X, é possível obter diferentes respostas (descrito na Equação 2.1) (GAMA et al., 2013).

Figura 1 – Tipos de Mudanças de Conceito



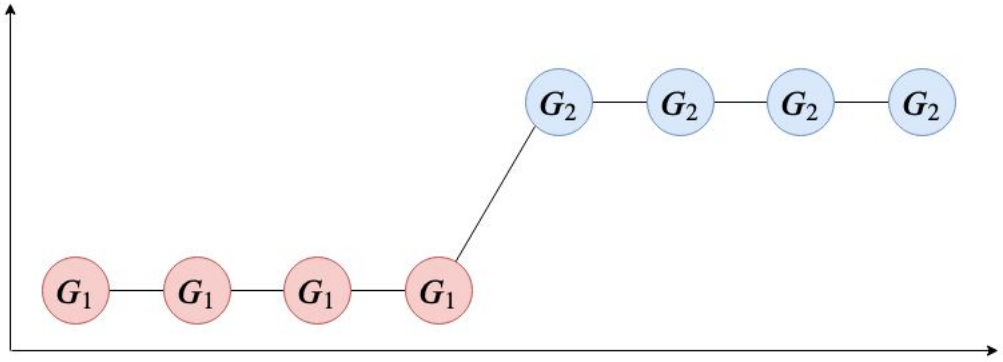
Fonte: Autor, 2019. Adaptado de (GAMA et al., 2013)

2.1.2 Taxa de Mudanças de Conceito

A mudança pode se apresentar em taxas diferentes e essas taxas são importantes para conseguir definir de maneira mais adequada o método a ser utilizado. Para futuras exemplificações, será considerado um cenário de recomendação em uma plataforma musical para um usuário U que procura identificar as preferências dos usuários. O conceito inicial do usuário U será composto, principalmente, por um conjunto de preferências G_1 e a mudança de conceito ocorrerá para um conjunto diferente de preferências G_2. É importante ressaltar que a ausência de consumo de músicas relacionadas à G_1, por exemplo, não implica necessariamente numa mudança real de conceito. O usuário precisa apresentar rejeição para as músicas relacionadas à G_1 para se tornar uma mudança real de conceito, enquanto somente a ausência de consumo é uma mudança virtual de conceito.

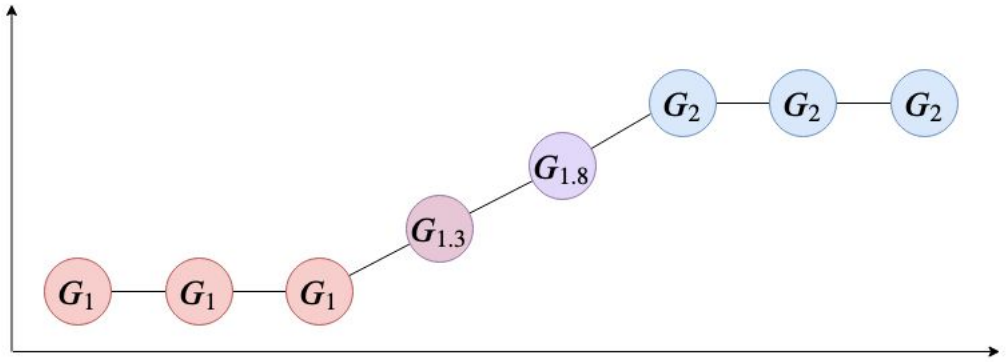
A primeira forma de mudança é a mudança abrupta de conceito, onde não existe

Figura 2 – Mudança Abrupta de Conceito



Fonte: Autor, 2019. Adaptado de (GAMA et al., 2013)

Figura 3 – Mudança Incremental de Conceito



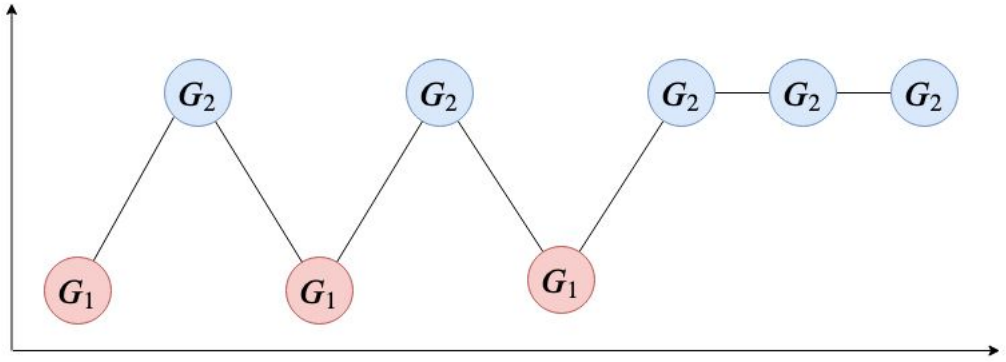
Fonte: Autor, 2019. Adaptado de (GAMA et al., 2013)

uma transição entre os diferentes conceitos (e.g. O usuário U_2 começa a utilizar o mesmo perfil de U_1, considerando as preferências de U_2 como G_2, sendo $G_1 \neq G_2$. A mudança ocorrerá de forma abrupta para o sistema visto que o usuário U_2 possivelmente não terá muitas interações positivas com G_1). Esse cenário foi ilustrado na Figura 2.

Outra possibilidade de mudança de conceito é a incremental. Conforme apresentado na Figura 3, a mudança incremental utiliza de conceitos intermediários até a conclusão da mudança (e.g. O usuário U_1 começa a consumir subgêneros musicais de G_1, inserindo incrementalmente elementos relacionados a G_2, até finalmente apresentar diretamente interações positivas com G_2, deixando de consumir e apresentando rejeição à G_1).

Diferentemente de uma mudança incremental de conceito, a mudança gradual varia entre os dois conceitos G_1 e G_2 diretamente, até finalmente estabilizar o último conceito. O cenário apresentado na Figura 4 apresenta esse tipo de mudança de conceito: o usuário

Figura 4 – Mudança Gradual de Conceito

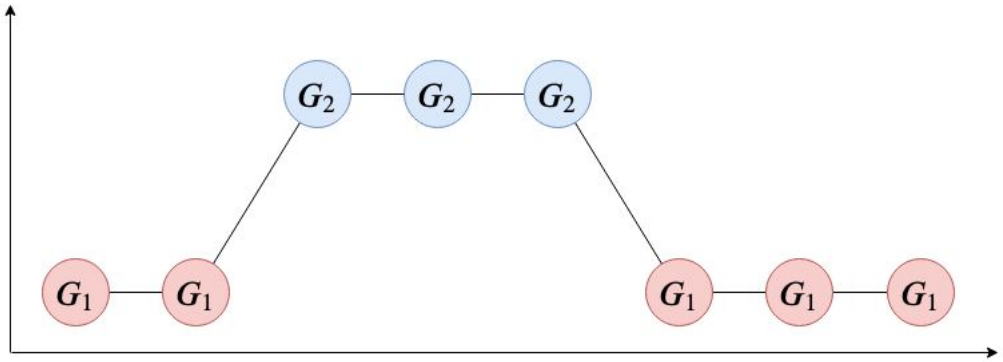


Fonte: Autor, 2019. Adaptado de (GAMA et al., 2013)

U_1 começa a adotar as preferências musicais expressas em G_2. Por um tempo, o usuário escuta músicas de ambos conjuntos de preferências G_1 e G_2, até um determinado momento onde ocorre a perda de interesse de músicas relacionadas à G_1, escutando somente G_2.

Mudanças periódicas de conceito, conforme expressa na Figura 5, estão relacionadas a conceitos recorrentes, ou seja, conceitos que deixa de estar em vigor por um determinado tempo mas acabam voltando a se apresentar para o modelo. Por exemplo, o usuário deixa temporariamente de escutar músicas referentes à G_1, tendo somente as preferências G_2. Depois de um determinado tempo, o usuário U_1 volta a escutar G_1 e deixa dessa vez de consumir músicas relacionadas ao G_2. Esse período de transição pode variar para cada caso.

Figura 5 – Mudança Periódica de Conceito



Fonte: Autor, 2019. Adaptado de (GAMA et al., 2013)

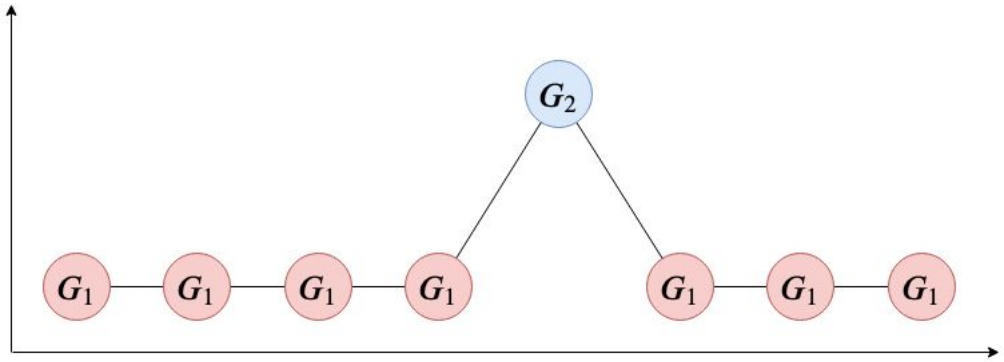
Além de identificar o tipo de mudança que um conceito pode sofrer, é importante lembrar o problema com *outliers* quando há mudanças de conceito. É possível que apareça um conceito C_2 em algum momento, porém ele não seja exatamente com uma mudança no perfil de consumo. Esses casos devem ser tratados como ruído e não provocar a adaptabilidade do modelo acionada para um conceito equivocado (e.g. o usuário realiza um *miss click* em uma música de G_2).

2.1.3 Detectores de Mudança

Os detectores são um artifício amplamente utilizado em mineração de fluxos de dados, sendo responsáveis por detectar as mudanças de contexto ao decorrer do fluxo, impactando algum procedimento interno do algoritmo para adaptação a essa mudança (GONÇALVES et al., 2014).

Tipicamente os detectores são separados em 4 categorias: Gráfico de Controle, Análise Sequencial, Monitoramento de Distribuições e Contextual (GAMA et al., 2013). O presente trabalho focará em detectores de Gráfico de Controle uma vez que eles são

Figura 6 – Outlier de Conceito



Fonte: Autor, 2019. Adaptado de (GAMA et al., 2013)

melhor explorados na literatura e se adaptam bem ao erro retornado pelos algoritmos utilizados nesse trabalho, que futuramente serão apresentados.

A categoria de detectores de gráfico de controle realizam o monitoramento do erro do modelo. Para isso, são definidas 3 regiões do gráfico: estabilidade, alerta e mudança. Como pode ser observado na Figura 7, o erro encontra-se em um estado de estabilidade e à medida que começa a aumentar, aciona o nível de alerta. Acionado o nível de alerta, é possível realizar procedimentos de preparação para uma mudança de conceito no modelo. Caso o erro continue aumentando, será acionado o nível de mudança e então tomadas as providências necessárias.

Outra catergoria muito presente na literatura é a Análise Sequencial. Com um dos grandes representantes sendo o ADWIN (BIFET; GAVALDÀ, 2007), esse tipo de detector trabalha com uma janela de instâncias e procura dividir essa janela em dois conceitos, buscando assim identificar a mudança.

Figura 7 – Níveis dos Detectores

Fonte: Autor, 2019. Adaptado de (CASILLAS; WANG; YAO, 2019)

Todas as implementações utilizadas foram retiradas do Scikit Multiflow (MONTIEL et al., 2018), que consiste atualmente na principal biblioteca de mineração de fluxos de dados para Python. A versão utilizada do Scikit Multiflow foi a 0.4.1. A implementação do ADWIN[1] não tem suporte para as duas zonas de mudança (*warning* e *out of control*). Devido a isso, esse trabalho manteve somente os outros dois detectores (DDM e EDDM) para a busca. Além do mais, uma análise exaustiva dos detectores está fora do escopo desse trabalho. Os métodos escolhidos têm fácil parametrização e bom desempenho (GONÇALVES et al., 2014), enquanto outros métodos como o Page-Hinkley (PAGE, 1954) tem um ajuste mais complexo de ser realizado, sendo também descartado.

2.1.3.1 Drift Detection Method (DDM)

O primeiro método que se apresenta como uma possibilidade é o *Drift Detection Method* (DDM) (GAMA et al., 2004). Apesar de ser o percursor na área de detecção, seu desempenho ainda se mantém competitivo. Conforme apresentado em Gonçalves et al. (2014), o DDM é o melhor detector de mudança para *datasets* que sofrem de mudança gradual de conceito, considerando tanto acurácia quanto o ranque médio. Além disso, tem um desempenho bom para mudanças abruptas de conceito.

Para seu funcionameto, é necessário a medida de erro no instante i, representada por p_i. Além disso, será também calculado o desvio padrão no instante i, represetado por s_i. A cada nova medida de erro p_i, será observado se $p_i + s_i < p_{min} + s_{min}$. Caso essa condicional seja verdadeira, os novos valores p_{min} e s_{min} serão atualizados com p_i e s_i respectivamente. Os hiperparâmetros do método são os níveis δ_w para definição do *warning* e δ_c para a definição da zona de *out of control*. Os parâmetros foram utilizados seguiram uma confiança para o *warning* de 95%, ou seja, $2 * s_{min}$ e *out of control* de 99%, ou seja, $3 * s_{min}$. Essa é uma configuração apresentada em Gama et al. (2004), onde o DDM foi proposto.

Em seguida, as duas Equações 2.2 e 2.3 mostram se o modelo está em uma zona de *warning* ou de *out of control* respectivamente.

$$p_i + s_i \geq p_{min} + 2 * s_{min} \tag{2.2}$$

$$p_i + s_i \geq p_{min} + 3 * s_{min} \tag{2.3}$$

Com um funcionamento bem simples e um custo computacional baixo, o DDM tem ótimos resultados para detecção de mudança de conceito.

[1] <https://scikit-multiflow.github.io/scikit-multiflow/_autosummary/skmultiflow.drift_detection.ADWIN.html>

2.1.3.2 Early Drift Detection Method (EDDM)

Uma outra possibilidade é o *Early Drift Detection Method* (EDDM) (BAENA-GARCÍA et al., 2006), que também utiliza do gráfico presente na Figura 7, mas se diferencia por usar a distância entre os erros para definir uma mudança de conceito. O método também utiliza dos níveis de *warning* e de *out-of-control* para o modelo. Esse método tem um desempenho aprimorado em mudanças lentas e graduais de conceito.

Seu funcionamento consiste no armazenamento de dois valores, a distância média entre dois pontos p_i e o desvio padrão s_i. São também armazenados esses valores ao atingir o valor máximo da função $p_i + 2 \cdot s_i$ representado por p_{max} e s_{max}.

$$\frac{p_i + 2 \cdot s_i}{p_{max} + 2 \cdot s_{max}} \tag{2.4}$$

O erro monitorado pelo modelo é obtido através da Equação 2.4, e são definidas duas constantes α e β, sendo a primeira para definir a zona de *warning* e a segunda *out of control*. Os valores de α e β são utilizados respectivamente como $0,95$ e $0,9$. Portanto, caso a Equação 2.4 seja menor que α, é tratado como zona de *warning* e quando menor que β é considerado *out of control*.

2.1.4 Técnicas de Adaptação

Tendo em vista que nessa área é necessário lidar com fluxos possivelmente infinitos de dados, o gerenciamento de memória se torna um grande problema. É necessário manter modelos treinados ao mesmo tempo que não se pode armazenar uma grande quantidade de instâncias antigas. Para isso, é necessário considerar duas principais questões, a aprendizagem com novas instâncias e as técnicas de esquecimento, que consistem em formas de realizar a eliminação das informações que deixaram de ser relevante para o modelo.

2.1.4.1 Aprendizagem do Modelo

Em termos de aprendizagem, pode ser optado por duas principais estratégias: trabalhar somente com a instância presente ou considerar uma janela de instâncias. A primeira estratégia é característica por aplicar operações na nova instância de maneira a criar uma representação interna do conhecimento, desvinculando do modelo usual de representação (e.g. em forma de árvore). Esse tipo de técnica está sempre em busca de um balanceamento entre sensibilidade e estabilidade (CARPENTER; GROSSBERG; ROSEN, 1991). Um exemplo para a área é a Hoeffding Tree (DOMINGOS; HULTEN, 2000), que transforma as novas instâncias em nós na árvore quando necessário.

Ao trabalhar com a segunda estratégia, uma outra questão é levantada: *"Qual é a quantidade ideal de instâncias a ser considerada?"*. Devido a isso, os algoritmos que utilizam janelamento são categorizados em dois conjuntos: janela deslizante fixa e variável. Surgem com essa questão técnicas de aprendizagem do tamanho de janela e outros desafios. Um exemplo de algoritmo para a representação dessa categoria é a versão incremental do Naïve Bayes, que considera uma janela de instâncias para realizar a predição e do CVFDT (*Concept-adapting Very Fast Decision Tree*) (HULTEN; SPENCER; DOMINGOS, 2001) .

2.1.4.2 Técnicas de Esquecimento

Assim como as técnicas de aprendizagem, as técnicas de esquecimento também podem ser categorizadas em dois conjuntos: graduais e abruptas (GAMA et al., 2013). As técnicas graduais são características por não removerem por completo as instâncias obsoletas do sistema. Ao invés disso, cada vez mais suas implicações são reduzidas até um ponto que deixam de impactar significativamente as decisões do modelo.

Em termos de técnicas de esquecimento abruptas, suas características são justamente a eliminição das instâncias obsoletas por completo. O exemplo clássico é justamente o já apresentado anteriormente: janela deslizante. Pode ser considerada uma técnica de esquecimento também devido a constante remoção de instâncias, sendo o critério comum a ultrapassagem no limite de instâncias na janela, podendo ser o tempo decorrido desde sua chegada.

No Capítulo 4 serão detalhadas algumas técnicas de esquecimento para sistemas de recomendação de ambas as categorias expostas. Na seção seguinte, serão apresentadas a conceituação sobre a área, algoritmos de recomendação, a incrementalidade em sistemas de recomendação e formas de avaliação dos mesmos.

2.2 Sistemas de Recomendação

Sistemas de Recomendação começaram a ter maior importância a medida que plataformas online começaram a aumentar a quantidade de informações que precisavam ser disponibilizadas para o usuário. Uma plataforma que conhece seu usuário e apresenta somente as informações relevantes proporciona boas experiências, consequentemente levando a uma maior quantidade de vendas e de usuários que voltariam a utilizar.

Por trás das opções oferecidas aos usuários de maneira personalizada existem sistemas para realizar essas recomendações. A tarefa principal consiste em ordenar uma grande quantidade de itens e apresentar o top N, sendo N geralmente de 1, 5 ou 10 itens.

Existem diversas abordagens para realizar as recomendações. Usualmente a categorização é feita em dois conjuntos (CHANG et al., 2017), sendo elas: a) *Collaborative*

Filtering e b) *Content-based Filtering*. *Collaborative Filtering* (CF) consiste em agrupamento de usuários ou de itens baseando-se nos *ratings* fornecidos, recomendando assim itens que estejam relacionados aos outros usuários com consumo relacionado.

Já técnicas de *Content-based Filtering* (CBF) baseiam-se em identificar itens similares sem a utilização dos *ratings*, aproveitando-se da descrição, categoria, preço, ano e outras características referentes ao conteúdo do produto. O presente trabalho irá se limitar às técnicas de CF.

CF acaba trabalhando com informações mais simples, sendo geralmente no formato $< U\ I\ R >$ considerando U como o usuário, I como um item e R como o *rating* dado ao item pelo usuário U. Algumas bases de dados ainda utilizam o padrão binário, onde se encontra somente $< U\ I >$, representando uma iteração positiva entre o usuário U para o item I, deixando relações negativas e desconhecidas de lado (VINAGRE; JORGE; GAMA, 2014). As bases ainda precisam de uma informação adicional de *timestamp* caso seja abordado o problema de maneira incremental.

2.2.1 Ratings

Na literatura, é possível encontrar diversas ações por parte do usuário que podem ser consideradas *ratings*. Além de um *rating* explícito, como a votação de 1 a 5 estrelas a um determinado item ou um *upvote* numa rede social, existem *ratings* implícitos, que são extraídos sem a interação direta do usuário com a avaliação do produto, como por exemplo um *click* em um produto, uma compra efetuada ou até mesmo a visualização de uma grande parte do vídeo em uma plataforma como o YouTube. Isso tudo deve ser considerado na escolha de uma técnica de recomendação.

Outro tipo de base de dados utilizada segue o padrão *positive-only*, ou seja, mantém somente as interações positivas dos usuários, tratando as interações negativas e ausentes como iguais. Esse tipo de base de dados representa bem cenários como por exemplo o usuário U ouviu a música I.

Para transformar um conjunto de dados de uma representação explícita em *positive-only*, é necessário definir um limite θ de modo que $0 < \theta \leq R_{\max}$, em que $R_{\max}$ é a classificação máxima possível. Portanto, o conjunto de dados é binarizado, transformando todas as instâncias de $\langle U, I, R \rangle$ com $R \geq \theta$ em $\langle U, I \rangle$, enquanto o restante das interações é ignorado.

2.2.2 Algoritmos baseados em Vizinhança

Ao implementar uma técnica que se baseie em vizinhaça, deve ser optado pela centralidade no usuário ou no item (MIRANDA; JORGE, 2008). Quando centradas no item, a matriz de *ratings* deverá ser armazenada de maneira inversa, ou seja, as linhas

representando itens e as colunas o usuário. Quando for necessário retornar os top N itens a serem recomendados, será obtido um item I sobre o qual foi demonstrado imediato interesse e buscados todos os itens que tenham *ratings* parecidos ao item I. A centralidade no item terá o benefício de, no momento em que um usuário entrar pela primeira vez no sistema e demonstrar interesse em seu primeiro item, já será possível apresentar recomendações relevantes ao mesmo.

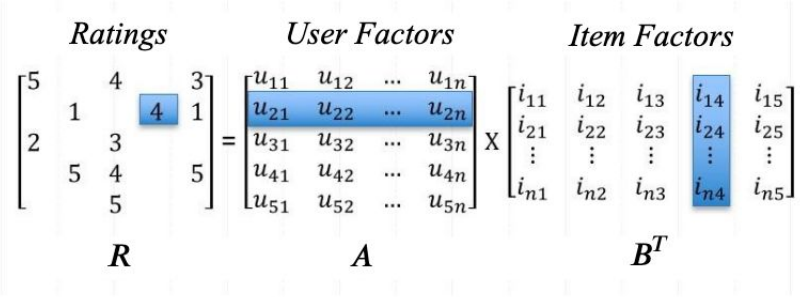
Utilizando a centralidade no usuário, estratégia que apresenta maior taxa de personalização e mais proximidade ao cliente, o sistema de recomendação faz a mesma operação, mas ao invés de aplicar aos itens, é aplicado a um usuário U, de maneira a identificar usuários com *ratings* similares ao perfil do usuário U. Esse método leva a um problema com usuários novos devido a baixa quantidade de *ratings*, e dificuldade de lidar com mudança de perfil, que, dependendo da plataforma a qual o sistema de recomendação está sendo aplicado, pode ser algo comum a alguns usuários. O problema de dificuldade inicial para recomendação é denotado como *cold start*.

2.2.3 Algoritmos de Fatoração de Matriz

Outra técnica muito utilizada para gerar recomendações é baseada em diminuição de dimensionalidade com fatoração de matriz. Proposto em (SARWAR et al., 2001), o algoritmo recebe um valor K como entrada, como pode ser observado nas n dimensões das matrizes na Figura 8, transformando então as matrizes A e B em $(U \times K)$ e $(I \times K)$ respectivamente, um valor η que representa a taxa de impacto das atualizações do modelo e o λ, parâmetro responsável pela penalização de valores elevados para os atributos, trazendo maior generalização do modelo. Esse parâmetro pode ser encontrado como valores diferentes para λ_u e λ_i, mas tipicamente é o mesmo valor para ambos.

$$\hat{R}_{ui} = A_u \cdot B_i^T \tag{2.5}$$

Figura 8 – Fatoração de Matriz



Fonte: Autor, 2019. Adaptado de (JORGE et al., 2017)

Para identificar a compatibilidade entre um item e um usuário, a operação necessária é um produto escalar entre vetores conforme pode ser observado na Equação 2.5. Os elementos da multiplicação podem ser observados em azul na Figura 8.

$$\min_{A,B} \sum_{(u,i)\in D} (R_{ui} - \hat{R}_{ui})^2 + \lambda(\|A_u\|^2 + \|B_i\|^2) \tag{2.6}$$

Os pesos iniciais são definidos aleatoriamente conforme uma distribuição normal, e então aplicado o *Stochastic Gradient Descendent* (SGD) (KIEFER; WOLFOWITZ, 1952) para a minimização de acordo com a Equação 2.6. O primeiro elemento da equação retorna o erro quadrático da predição, enquanto o segundo, que consiste na multiplicação de λ pela soma do quadrado das normas dos vetores, é responsável por manter a generalização do modelo e diminuir os pesos dos atributos. A somatória é aplicada para todos os *ratings* (u, i) no conjunto D, que contém todas as instâncias de treinamento.

$$A_u \leftarrow A_u + \eta(err_{ui} B_i - \lambda A_u) \tag{2.7}$$
$$B_i \leftarrow B_i + \eta(err_{ui} A_u - \lambda B_i) \tag{2.8}$$

Para cada iteração, também são aplicadas as Equações 2.7 e 2.8 para atualização dos vetores latentes de ambos os conjuntos A e B. É importante ressaltar a importância do η, responsável por aumentar ou diminuir o passo de aprendizagem.

2.2.4 Algoritmos Incrementais de Fatoração de Matriz

Devido a existência de uma mudança de perfil no decorrer do tempo, é de extrema importância o uso de classificadores incrementais para manter a performance ao longo do tempo. A mudança de conceito pode ser observada em cenários de recomendação como a criação de novos produtos, queda de popularidade e modificação do perfil do consumidor.

A primeira adaptação de um algoritmo de fatoração de matriz incremental foi o algoritmo BRISMF, proposto em (TAKÁCS et al., 2009). BRISMF é um acrônimo de *Biased Regularized Incremental Simultaneous Matrix Factorization*, e devido a ser *biased*, mantém um valor por usuário e um valor por item que independe de qualquer multiplicação, sendo somado diretamente em $\hat{\mathrm{R}}$. Essa característica permite lidar com *ratings* explícitos numéricos, ou seja, que possam ir além do padrão binário. A Equação 2.9 demonstra essa característica. A diferença para a Equação 2.5 é a adição do bias do usuário b_u e do item b_i.

Considerando temporariamente A_u como o vetor de características X e B_i como Y, é importante levantar um detalhe de implementação do mesmo. Pode ser optado em configurar o bias do usuário na posição X_0 e o bias do item na posição Y_1, ao mesmo

tempo em que nas posições inversas X_1 e Y_0 são atribuídas o valor 1. Com isso, a Equação 2.9 será equivalente a Equação 2.5, considerando que o produto escalar consiste em $X_0 \cdot Y_0 + X_1 \cdot Y_1 + \cdots + X_{k-1} \cdot Y_{k-1} + X_k \cdot Y_k$.

$$\hat{R}_{ui} = A_u \cdot B_i^T + b_i + b_u \tag{2.9}$$

Uma variação chamada de ISGD, específica para lidar com *ratings positive-only*, foi proposta em (VINAGRE; JORGE; GAMA, 2014). Sua principal diferença frente ao BRISMF é a ausência do bias, que acaba se tornando desnecessário nesse cenário. O funcionamento do ISGD é bem similar a fatoração de matriz em *batch* anteriormente apresentada.

Para fazer a adaptação para o ambiente de *streaming*, é inicialmente gerado um modelo *offline* conforme a minimização da Equação 2.6 e atualização dos vetores com as Equações 2.7 e 2.8. Para a segunda etapa, dado um conjunto com a sua ordem temporal preservada, é realizada a avaliação, e logo após a aplicação das Equações 2.7 e 2.8 para efetuar o treinamento. Esse protocolo é aplicado para ambos ISGD e BRISMF, considerando sempre suas nuances. Essa última etapa é conhecida como Protocolo *Prequential* (VINAGRE; JORGE; GAMA, 2015b) e será melhor detalhada na próxima subseção, especificamente em 2.2.5.3, com os detalhes específicos de aplicação para esse trabalho no Capítulo 3.

2.2.5 Avaliação e Validação de Modelos de Recomendação

O treinamento incremental se difere do treinamento em *batch* pelo motivo de ser *one-pass*, ou seja, atualizar os pesos dos vetores somente uma vez. Apesar de ser possível encontrar variações na literatura (VELOSO et al., 2018), o comum é utilizar a versão incremental do recall para a etapa em fluxo enquanto em *batch* utiliza da minimização presente na Equação 2.6, com o erro quadrático.

2.2.5.1 Métrica de Avaliação - RMSE

Uma das primeiras métricas de avaliação utilizadas para medir o desempenho de modelos de recomendação ao decorrer do tempo foi o RMSE. Seu constante uso esteve presente até a competição realizada pelo Netflix (BENNETT; LANNING, 2007). A competição visava aumentar o desempenho dos sistemas de recomendação da empresa utilizando-se do RMSE como métrica, tentando minimizá-lo.

Seu uso é comum em outras áreas da aprendizagem de máquina, e apesar de não ser a melhor possibilidade para medição do desempenho dos modelos de recomendação, pode ser uma alternativa para outras métricas internas do algoritmo (e.g. seu uso para o treinamento em *batch*). Essa métrica não é interessante de ser utilizada por incorporar

informações de, por exemplo, se o modelo acerta um item que não é importante pro usuário. Isso é corrigido ao utilizar o *Recall@N*, métrica que será descrita futuramente.

O RMSE pode ser medido conforme a Equação 2.10, que consiste na raiz quadrada da média do quadrado dos erros pelo tamanho do conjunto T de teste, sendo $\hat{R}_{ui}$ o *rating* predito do usuário u para o item i e R_{ui} o *rating* real dos mesmos.

$$\sqrt{\frac{\sum_{(u,i)\in T}^{N}(\hat{R}_{ui} - R_{ui})^2}{\mid T \mid}} \tag{2.10}$$

2.2.5.2 Métrica de Avaliação - Recall@N

Devido ao problema de distribuição encontrado no cenário de recomendação (a grande maioria dos itens é irrelevante para o usuário), métricas como RMSE e MAE acabam incorporando informações desnecessárias ao considerar previsões de itens com poucas estrelas, ou seja, irrelevantes para o usuário.

A vantagem do recall incremental (reduzido para Recall@N) é usar apenas os itens relevantes, melhorando assim a medição do sistema de recomendação em questão (MATUSZYK; SPILIOPOULOU, 2017).

A métrica Recall (CREMONESI et al., 2010) passou a ser muito utilizada depois que o RMSE foi demonstrado não ser uma boa métrica, devido ao motivo apresentado, na competição realizada pelo Netflix. Apesar de ser importante identificar com acurácia um possível *rating* dado por um usuário a um item, medido pelo RMSE, a grande tarefa sempre foi apresentar N itens mais relevantes aos usuários, medida pelo Recall.

Para realizar a medição do Recall, é necessária a realização dos passos descritos a seguir. Quando encontrada uma interação positiva na base, por exemplo $\{< U\ I\ R\ T > \mid R = 5\}$ ou $< U\ I\ T >$:

1. São selecionados 1000 itens não avaliados pelo usuário U aleatoriamente, e partido-se do pressuposto que serão irrelevantes para o usuário em questão;
2. Todos os 1001 itens, sendo os 1000 itens selecionados aleatoriamente e o item I, são ranqueados utilizando-se algum modelo de recomendação;
3. Caso o item I esteja no top N itens, ele é contado como um *hit*.

Posteriormente, é utilizada a Equação 2.11 para extrair o Recall em tempo real do modelo. Essa equação é a divisão entre a quantidade de *hits* até o atual momento na stream pela quantidade de avaliações já realizadas no conjunto T de teste.

$$Recall(N) = \frac{\mid hits(N) \mid}{\mid T \mid} \tag{2.11}$$

2.2.5.3 Protocolo de Validação

O método de validação clássica utilizada em algoritmos de aprendizagem de máquina é *hold out*[2]. O procedimento padrão consiste na separação da base de dados em dois conjuntos: teste e treino. Essa separação é necessária para ter uma relevância estatística no teste, visto que não se deve violar a independência de observações realizando teste em instâncias que já foram treinadas.

Apesar de ser amplamente utilizada em cenários comuns, segundo os autores (VINAGRE; JORGE; GAMA, 2015b), *hold out* não pode ser utilizado quando tratado de sistemas adaptativos de recomendação por alguns motivos, sendo os principais deles:

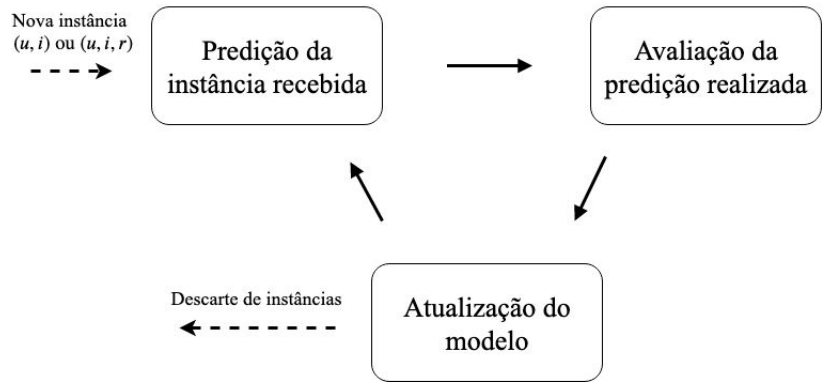
- Ordenamento: a técnica de *hold out* realiza um embaralhamento, o que compromete diretamente o ordenamento natural e temporal dos dados. O ordenamento deve ser preservado em cenários que utilizem de fluxo de dados;
- Temporalidade: análise temporal só é possível se existir essa relação temporal entre as instâncias. Da mesma forma que o item anterior, é possível obter essas informações caso seja preservado o ordenamento natural das instâncias;
- Incrementalidade: visto que o modelo precisa de atualizações incrementais para seu funcionamento, e o *hold out* não permite isso, é necessário um protocolo que consiga suprir essa demanda.

Por esses motivos, o Protocolo *Prequential* (GAMA; SEBASTIÃO; RODRIGUES, 2013) teve a sua adaptação para a área de recomendação em Vinagre, Jorge e Gama (2015b). Seu funcionamento consiste em: para cada instância na base de dados com a ordem temporal dos dados preservada, é realizada a predição e em seguida treinamento, conforme pode ser observado na Figura 9. A independência de observações não é violada devido ao teste nunca ser realizado em uma instância já treinada.

Outro problema que precisa ser considerado para definição do protocolo de validação é o *cold start*. Esse problema consiste na ausência de recomendações para novos usuários, visto que os mesmos não tiveram uma grande quantidade de interações na plataforma. Para evitar esse problema, é necessário gerar um modelo *offline* e depois colocar em um ambiente de *streaming*. Para realizar esse movimento, usualmente a base é dividida em duas principais partes. A primeira parte é responsável por gerar um modelo *offline* aplicando o treinamento em *batch*, expresso na minimização presente na Equação 2.6, e atualização para cada iteração conforme as Equações 2.7 e 2.8. A segunda parte tem uma característica importante a ser mencionada: é necessário a preservação da ordem temporal

[2] Outro método de validação que pode ser encontrado é a validação cruzada. Essa técnica não estará no âmbito dessa pesquisa.

Figura 9 – Protocolo *Prequential*



Fonte: Autor, 2019.

dos dados. Nesse momento, é aplicado o Protocolo *Prequential* conforme anteriormente explicado.

2.3 Considerações do Capítulo

O presente capítulo apresentou conceitos importantes sobre mineração de fluxo de dados. De início, foram discutidos os tipos reais e virtuais de mudanças de conceito, as várias taxas de mudança, os algoritmos de detecção e as técnicas de adaptação a mudanças de conceito.

Posteriormente, foram apresentados conceitos relacionados a sistemas de recomendação. O primeiro conceito importante discutido foi *ratings*, que podem ser classificados em duas categorias, *ratings* explícitos e *ratings* implícitos. É comum que *ratings* implícitos sejam binários, visto a ausência direta do usuário na avaliação. Para isso existem bases *positive-only*, ou seja, onde somente interações positivas são permitidas.

Para ambos os casos existem algoritmos de fatoração de matriz. No caso de bases numéricas, é utilizado o BRISMF devido a sua capacidade de armazenar um *bias* para cada usuário e cada item e somando a regressão, permitindo melhores resultados. A outra técnica deixa de armazenar o *bias*, assim permitindo lidar com bases *positive-only*. Ambas as técnicas utilizam treinamento em batch antes de serem inseridas em ambiente de *streaming*, conforme foi apresentado anteriormente.

O Capítulo 3 irá apresentar o protocolo de experimentação utilizado para validação dos métodos propostos. Com isso, as bases de dados serão expostas, junto com a sua breve origem e características. Além disso, serão apresentados as características para realização dos experimentos, junto aos detalhes do Protocolo *Prequential*.

3 Protocolo de Validação

Para futuras experimentações, o protocolo utilizado será conforme apresentado nesse capítulo. De início, será apresentado brevemente o FluRS, *framework* utilizado para implementação das abordagens. Em seguida os conjuntos de dados selecionados junto com as suas características e origem, finalizando com o método de avaliação para as abordagens. Tendo em vista que é necessário preservar a qualidade de acomplamento dos métodos, serão utilizados ambos ISGD e BRISMF como base. Como ambos demandam diferentes tipos de dados devido a suas respectivas naturezas de *feedback* explicitadas na subseção 2.2.4, as bases apresentadas serão adaptadas para ambos os algoritmos, resultando em quatro bases para cada.

3.1 Framework de Experimentação: FluRS

O *framework* próprio para sistemas de recomendações incrementais FluRS[1], apesar de estar em estágios iniciais, apresenta grande variedade de algoritmos presentes na literatura da área para experimentações. FluRS significa *Flu-* (Flux, Fluid, Fluent) Recommender Systems*, devido a sua natureza incremental e adaptativa de definição de perfil para recomendação.

Devido a isso, sua estrutura básica foi utilizada para a implementação/aprimoramento dos algoritmos mencionados nesse trabalho, sendo eles: BRISMF e ISGD para recomendação e as técnicas propostas ADADRIFT e *Background Profiling*. A implementação foi realizada conforme especificado futuramente nos Capítulos 6 e 5 para os métodos de aprendizagem personalizada adaptativa.

3.2 Base de Dados

Devido ao problema enfrentado nesse trabalho, as bases de dados precisam ser tratadas e selecionadas com cautela. Pressupõe-se que uma base de dados com uma maior quantidade de *ratings* médios por perfil tende a ter uma maior frequência de mudança de conceito nos perfis. Considerando essa premissa, foi definido o critério de seleção das bases de dados: maximizar esse valor.

Devido isso, essa seção é responsável por guiar o leitor no processo para o entendimento dessa seleção, sendo dividida da seguinte maneira:

a) Apresentação das bases de dados preliminarmente selecionadas;

[1] https://takuti.github.io/flurs/

b) Transformação de aumento da quantidade de *ratings* médios por perfil e detalhamento final das bases de dados.

3.2.1 Seleção Preliminar

Considerando que serão utilizados os algoritmos BRISMF e ISGD para este experimento, foram selecionados conjuntos de dados numéricos e *positive-only*, totalizando dois conjuntos de dados por categoria. Além disso, nosso objetivo era observar uma mudança de conceito. Para isso foi necessário realizar alguns tratamentos nos conjuntos de dados selecionados para observar esse problema. A mudança de conceito, como já explicada, afeta usuários e itens. Por esse motivo, a seleção de conjuntos de dados também deve representar esse problema. Assim, foram selecionados dois conjuntos de dados com um longo histórico de interações por usuário e dois com um longo histórico de interações por item. É importante ter em mente que, ao se referir a um perfil, a ideia pode ser aplicada tanto a perfis de usuários quanto a perfis de itens.

Os conjuntos de dados foram selecionados propositadamente com um número considerável de instâncias para, após aplicar transformações, manter um número significativo de *rating* por perfil e ainda um tamanho razoável, com cerca de 1 milhão de *ratings*. Os datasets selecionados serão apresentados em seguida.

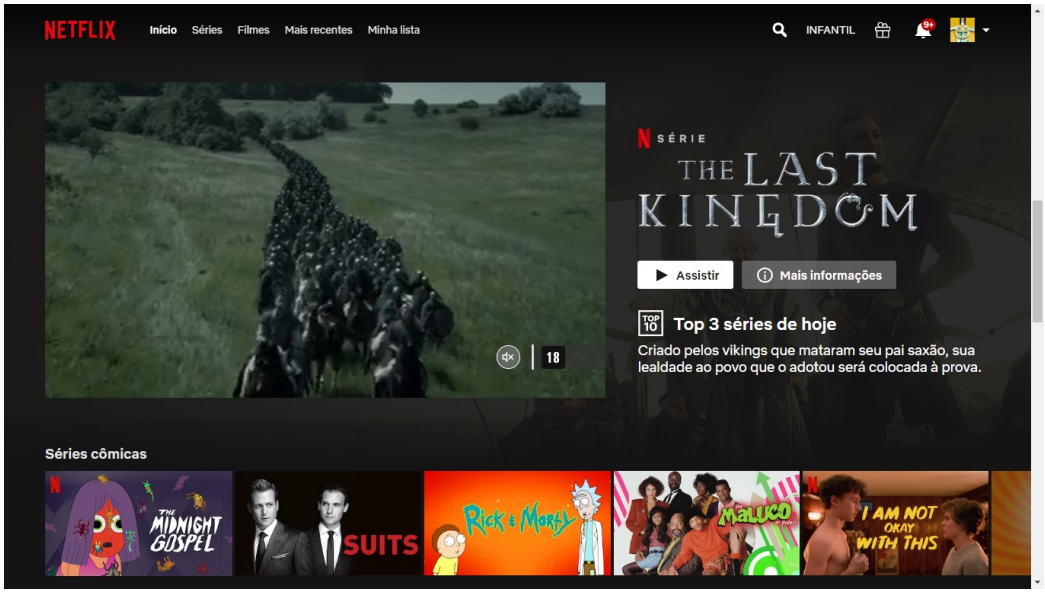


Figura 10 – Plataforma Netflix em 2020. Imagem capturada em 06/05/2020 às 05h55 GMT.

3.2.1.1 Netflix

Disponibilizado atualmente pelo Kaggle[2], o *dataset* do Netflix (BENNETT; LANNING, 2007) consiste em 480.189 usuários e 17.770 filmes da plataforma na época. Sua disponibilização veio junto ao desafio, oferecendo $1.000.000,00 USD para o time que conseguisse realizar um aumento significativo no RMSE. O time vencedor, chamado de "BellKor's Pragmatic Chaos"(BELL; KOREN; VOLINSKY, 2007), propôs uma solução com um ganho de 10,06% no RMSE. Uma curiosidade é que a solução nunca veio a ser utilizada, devido a métrica escolhida não ser muito interessante para medição (MATUSZYK; SPILIOPOULOU, 2017).

Apesar da base ter sido apresentada em 2007, ainda hoje é uma das maiores encontradas na literatura, com 100 milhões de instâncias. Essa grande quantidade de instâncias permite o aporte de diversos filtros, mantendo uma quantidade considerável de *ratings*. A plataforma continua amplamente conhecida, pondendo ser observado a versão atual na Figura 10.

Para esse trabalho, serão selecionadas duas versões desse *dataset*: a versão original e uma versão utilizando a conversão para *positive-only*, apresentada na Subseção 2.2.1, com $\theta = 5$. Mais informações sobre ambos os *datasets* originais podem ser observada na Tabela 1.

3.2.1.2 Personality

Esse *dataset*, disponibilizado pelo GroupLens[4], surgiu com uma pesquisa conduzida com 1.800 usuários em Nguyen et al. (2017). O objetivo dessa pesquisa era definir o quanto é possível inferir configurações de diversidade, popularidade e acaso, métricas importantes para a configuração das recomendações, dados os *ratings* dos usuários. Os pesquisadores concluíram que não é possível inferir essas propriedades somente com os *ratings*, e sugerem incorporar mais informações de traços de personalidade para montar as recomendações.

[2] <https://www.kaggle.com/netflix-inc/netflix-prize-data>

[3] Já como diferentes tipos de mudança de conceito estão sendo observadas, a média de *ratings* por itens foram apresentadas para os *datasets* NF e NF5, e a média de *ratings* por usuário foram apresentadas para os *datasets* LFM e NF5.

[4] <https://grouplens.org/datasets/personality-2018/>

Tabela 1 – Mais informações sobre os *datasets* originais.

Dataset	Tamanho	Usuários	Itens	Média de Interações[3]	Esparsidade
Netflix (NF)	100,480,507	480,189	17,770	5,654	98.82%
Personality (PL)	1,028,751	1,820	35,196	565	98.41%
LastFM 1K (LFM)	19,150,868	992	1,500,661	19,305	99.69%
Netflix GTE 5 (NF5)	23,168,232	463,616	17,775	1304	99,72%

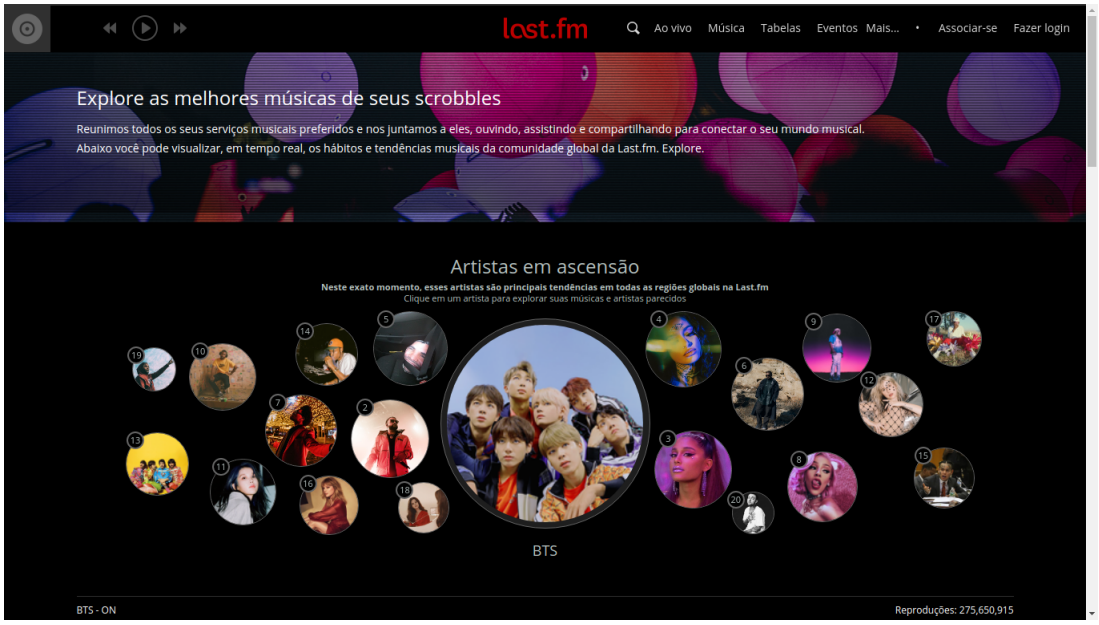


Figura 11 – Plataforma LastFM em 2020. Imagem capturada em 09/05/2020 às 06h45 GMT.

Apesar disso, o assunto desse artigo não será tratado nesse trabalho. Os pesquisadores disponibilizaram o *dataset* oriundo desse projeto, e devido a manter um controle constante ao decorrer de 6 meses, seus dados apresentam uma grande quantidade de *ratings* por usuário. Essas informações podem ser encontradas na Tabela 1.

É importante antes detalhar um processo adicional realizado nessa base. Seus *ratings* estavam no padrão 5 estrelas, com os 10 valores possíveis, sendo eles de 0,5 a 5,0 com incrementos de 0,5. Visto que o *framework* utilizado não suporta *ratings* com ponto flutuante, seus valores foram normalizados aplicando $R_t = R_{t-1} * 2$, se mantendo no padrão de 1 a 10 estrelas.

3.2.1.3 LastFM

Finalizando as bases selecionadas, o LastFM - 1K users[5] (CELMA, 2010) (LFM) foi escolhido devido a sua grande quantidade de *ratings*, beirando 20 milhões de interações com somente 1000 usuários. Esses dados foram obtidos através de *crawlers* na plataforma LastFM[6]. A versão atual da plataforma pode ser observada na Figura 11.

Devido a sua natureza, é possível somente observar interações positivas (usuário u escutou a música i) e uma grande quantidade de *ratings* repetidos, devido a possibilidade do usuário escutar duas vezes a mesma música.

[5] <http://ocelma.net/MusicRecommendationDataset/>
[6] <https://www.last.fm/>

Tabela 2 – Transformações aplicadas para o destaque das mudanças de conceito.

Dataset	Corte de 2M[7]	Valor de Poda	Tipo da Poda	Assinatura
Netflix	X	1K	Item	CD-NF
Personality		150	Usuário	CD-PL
LastFM 1K	X	5K	Usuário	CD-LFM
Netflix GTE 5	X	1K	Item	CD-NF5

3.2.2 Transformações

Os conjuntos de dados foram selecionados propositalmente com um número considerável de instâncias para poderem ser removidas, mantendo um número significativo de *ratings* por perfil e ainda um tamanho razoável, com cerca de 1 milhão de *ratings*. Ambos os *datasets* derivados do Netflix (NF e NF5) têm uma grande histórico de interações para filmes. Portanto, esses *datasets* têm uma maior probabilidade de apresentar uma mudança de conceito do perfil do item. Ao considerar os outros dois *datasets* (PL e LFM), é possível obervar um longo histórico de interações por usuário. Da mesma maneira que as mudanças de conceito baseadas em itens, esses *datasets* serão tratados para realçar as mudanças de conceito baseadas em usuários.

Para os três maiores *datasets* (LastFM, Netflix, and Netflix GTE 5), um corte foi realizado nas últimas 2 milhões de instâncias. Esse corte foi realizado porque também não é desejado que tenha pouquíssimos perfis para validação. O trabalho ainda se trata de *Collaborative-filtering*, ou seja, uma técnica que envolve encontrar usuários similares para fazer recomendações. Limitar drasticamente um *dataset* pode causar efeitos adversos indesejáveis. Esse valor foi definido devido a equilibrar bem uma quantidade razoável de perfis restantes e uma média alta de interações por perfil.

A partir disso, uma poda for realizada de acordo com os valores presentes na Tabela 2, removendo os usuários com uma menor quantidade de *ratings* para os *datasets* LFM e PL, e da mesma forma itens com menos *ratings* do que o *threshold* definido para os *datasets* NF e NF5. Os *datasets* com esse procedimento aplicado receberam o prefixo CD, e.g., CD-PL, CD-NF, CD-NF5, e CD-LFM.

O resultado desse procedimento pode ser obserado na Tabela 3. Como consequência, foi possivel obter *datasets* com uma maior quantidade média de interações por usuário/item ao comprar com os outros *datasets* da literatura. Isso implica numa maior probabilidade de encontrar mudanças de conceito devido ao longo histórico de interações de usuários e itens.

[7] O corte realizado foram das últimas 2 milhões de instâncias.

Tabela 3 – *Datasets* com tratamento de Mudança de Conceito.

Dataset	Tamanho	Usuários	Itens	Média de Interações[2]	Esparsidade
CD-PL	991,456	1,289	35,144	769	97,83%
CD-NF	810,046	108,269	452	1,792	98,34%
CD-NF5	1,012,145	141,132	468	2,162	98,47%
CD-LFM	1,028,379	119	260,098	8,641	98,72%

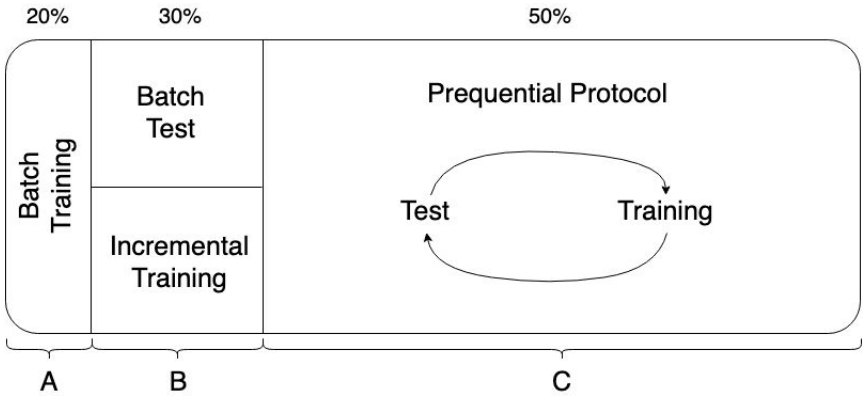


Figura 12 – O conjunto de dados foi separado em três partes, as duas primeiras se referem ao processo de geração da matriz em *batch*, enquanto a última é a aplicação do Protocolo *Prequential* para validação (MATUSZYK; SPILIOPOULOU, 2014).

3.3 Método de Validação

Para observar uma das características dos métodos propostos, a independência do algoritmo de fatoração da matriz, serão realizados experimentos utilizando ISGD e BRISMF. Outros algoritmos que podem ser encontrados na literatura também podem ser adaptados para o uso dos algoritmos de aprendizagem personalizada adaptativa.

Para evitar problemas de *cold start*[8], que não é o escopo desse trabalho, é necessário seguir um procedimento especial. De início, será gerado um modelo usando uma abordagem *offline* e, em seguida, implantá-lo em um ambiente de *streaming*. Esse processo é apresentado na Figura 12, onde as partes A e B são responsáveis pelo treinamento e teste de um modelo usando o método de otimização *Stochastic Gradient Descendent* (SGD) (KIEFER; WOLFOWITZ, 1952) no modo *batch*, sendo feita a minimização da Equação 2.6 na etapa B e a atualização conforme o gradiente dos vetores da Equação 2.7 na etapa A.

Em seguida, ainda na etapa B da Figura 12, foi feito o treinamento incremental *one-pass*. A etapa C ficou responsável pela aplicação do Protocolo *Prequential* (Figura 9)

[8] O problema de *cold start*, como explicado anteriormente, consiste na dificuldade inicial dos sistemas de recomendação para oferecer produtos aos usuários sem histórico na plataforma. Da mesma forma, esse problema pode ocorrer para itens novos sem um grande consumo dos usuários.

de maneira a extrair o *Recall@N* ao decorrer do tempo.

3.4 Considerações do Capítulo

O presente capítulo apresentou o procolo que será utilizado para validação dos experimentos futuramente propostos. Para isso, foi exposto o *framework* gênese de toda a experimentação, junto com as bases de dados que serão operadas no protocolo de experimentação. Para melhor entender os resultados dos algoritmos, foi detalhado a origem e método de extração dos dados. Logo após, o método de avaliação foi definido para doravante utilização.

O Capítulo 4 irá trazer os métodos similares encontrados na literatura. Para isso, serão separados em quatro conjuntos principais: (i) técnicas de aprendizagem adaptativa em outras áreas do conhecimento, (ii) técnicas de busquem trazer adaptação através do esquecimento de informações, (iii) métodos para diminuição da importância dos itens ao decorrer do tempo e (iv) abordagens de otimização encontradas na literatura para a área em questão. Todos esses métodos serão comparados com a proposta dessa dissertação para assim poder ser elencadas as diferenças.

4 Trabalhos Relacionados

A aprendizagem personalizada adaptativa não foi diretamente estudada na literatura para ambientes em *streaming* e sistemas de recomendação. É possível elencar quatro áreas similares à abordagem apresentada nessa dissertação sendo primeiramente as (i) técnicas de aprendizagem adaptativa em outras áreas do conhecimento (RUDER, 2016), (ii) técnicas de esquecimento (MATUSZYK et al., 2018), que trazem essa adaptação mas não implicam na mudança da aprendizagem ao decorrer do tempo, (iii) abordagens para diminuir a importância de itens ao decorrer do tempo (VINAGRE; JORGE; GAMA, 2015a) e (iv) outras que buscam otimizar os hiperparâmetros para o modelo, de maneira tanto personalizada (VELOSO; MALHEIRO; FOSS, 2019) quanto global (VELOSO et al., 2019), mas que não trazem adaptação ao decorrer do tempo. Essas técnicas serão sumarizadas no presente capítulo.

4.1 Aprendizagem Adaptativa

Técnicas de otimização para o *Gradient Descent* são vastamente estudadas e são usuais em tarefas de *deep learning*, estando presentes em diversos *frameworks* conhecidos como Lasagne[1] (DIELEMAN et al., 2015), Tensorflow[2] (ABADI et al., 2015) and Keras[3] (CHOLLET et al., 2015). Essa grande adesão é devido ao ganho obtido ao utilizá-los, sendo significativo para esse segmento.

Apesar disso, essas técnicas foram idealizadas para o ambiente *batch*, e.g., otimização da taxa de aprendizagem para o treinamento em um conjunto finito e bem definido de dados. A recente adaptação para o ambiente de *streaming* das técnicas de fatoração de matriz, e.g., BRISMF e ISGD, têm um problema diferente em mãos, que é a aprendizagem contínua e adaptação as mudanças de conceitos encontradas.

Já é conhecido que o funcionamento do ADAGRAD (DUCHI; HAZAN; SINGER, 2011) leva a taxas de aprendizagem extramemente pequenas (ZEILER, 2012) por constantemente adicionar o erro no divisor de sua equação. Da mesma forma, outras técnicas conhecidas de aprendizagem adaptativa tendem a convergir para um ponto ótimo (RUDER, 2016) como RMSProp (DAUPHIN et al., 2015), Momentum (QIAN, 1999) e Adam (KINGMA; BA, 2014). Ao convergir para um ponto ótimo, essas técnicas vão eventualmente levar a taxa de aprendizagem para níveis muito baixos, falhando em incorporar novas informações aos perfis e indo contra os princípios básicos da mineração de fluxos de dados,

[1] <https://lasagne.readthedocs.io/en/latest/>
[2] <https://www.tensorflow.org/>
[3] <https://keras.io/>

que necessitam de uma constante aprendizagem e adaptação a novas informações. Devido a esse problema diferente, essas técnicas não serão utilizadas nesse trabalho.

ADADELTA (ZEILER, 2012) é uma das bem-conhecidas técnicas de otimização e, apesar de também ser desenvolvida para ambiente *batch*, ela utiliza de um coeficiente de decaímento ρ para uma adesão de novas informações e para evitar o problema anterior do ADAGRAD, de convergir para um ponto infinitesimal. Devido a essa característica, é possível realizar uma adaptação do seu uso para o ambiente de *streaming*, considerando que não existe a convergência para um ponto em específico, mas algo similar a uma janela deslizante de adaptação. Ele será utilizado para comparação e melhor detalhado devido a essa característica.

4.1.1 ADADELTA

O desenvolvimento do ADADELTA (ZEILER, 2012) parte da afirmativa que o limite de aprendizagem e essa constante adição do erro deveria ser janelado. O hiper-parâmetro de decaímento ρ é utilizado para realizar essa janela deslizante. Ele deve se manter no intervalo $0 < \rho < 1$. A Equação 4.1 faz a acumulação do erro, sendo x o erro do gradiente, sendo ele $(R_{ui} - \hat{R}_{ui})^2$.

$$\mu^{t+1} = \mu^t * \rho + (\rho - 1) * x \tag{4.1}$$

Para realizar a atualização do gradiente, é necessário utilizar as Equações 4.2 e 4.3. Os hiper-parâmetros η e λ são os mesmos detalhados na subseção 2.2.3, que em suma, η representa a taxa de aprendizagem base do modelo e λ é um hiper-parâmetro para limitar os vetores e manter a generalização. A taxa de aprendizagem é dividida pela soma do erro acumulado com a variável ϵ, que tem a função de evitar um valor inválido para a operação. O valor sugerido para esse hiper-parâmetro foi de $\epsilon = 1^{-6}$.

$$A_u \leftarrow A_u + \frac{\eta}{\sqrt{\mu^t + \epsilon}}(err_{ui} B_i - \lambda A_u) \tag{4.2}$$

$$B_i \leftarrow B_i + \frac{\eta}{\sqrt{\mu^t + \epsilon}}(err_{ui} A_u - \lambda B_i) \tag{4.3}$$

4.2 Técnicas de Esquecimento

Como detalhado anteriormente, na subseção 2.1.4, as técnicas de esquecimento podem ser divididas em duas categorias: graduais e abruptas. É possível obter mais informações sobre técnicas de esquecimento voltadas para sistemas de recomendação em (MATUSZYK et al., 2018; MATUSZYK et al., 2015; MATUSZYK; SPILIOPOULOU, 2014).

Dado o cenário de recomendação, ainda pode ser feito uma outra categorização mais comum na área nos seguintes conjuntos: *Rating-based Forgetting* e *Latent Factor Forgetting*. A grande diferença entre as duas categorias é a maneira com a qual é feito o esquecimento. Enquanto *Rating-based Forgetting*, como o nome sugere, trabalha diretamente nas avaliações do usuário, a categoria *Latent Factor Forgetting* reune técnicas que possui como principal características trabalhar diretamente nos fatores latentes, ou seja, na matriz de usuários e itens. Pode-se destacar as seguintes técnicas de esquecimento:

4.2.1 Last N retention

Muito similar à amplamente utilizada em mineração de fluxos de dados, essa técnica consiste do uso de uma janela deslizante. Sua grande diferença é que a janela é definida para cada usuário, não impactando assim usuários com poucos *ratings*. Devido a utilizar diretamente as avaliações para realizar seu esquecimento, essa técnica é categorizada como *Rating-based Forgetting*.

Uma variação pode ser encontrada na literatura chamada *Recent N retention* (MATUSZYK; SPILIOPOULOU, 2014). Sua grande diferença é utilizar um intervalo de tempo para definição da janela, diferente do *Last N retention* que utiliza a quantidade de instâncias.

4.2.2 User Factor Fading

De maneira bem simples, essa técnica multiplica os fatores latentes por uma constante α, que precisa se manter no intervalo $0 > \alpha > 1$ (como pode ser observado na Equação 4.4), aplicando um processo constante de *fading* no vetor de preferências do usuário. Por trabalhar diretamente no vetor de características, esse técnica é categorizada como *Latent Factor Forgetting*.

$$A_u^{t+1} = \alpha \cdot A_u^t \tag{4.4}$$

4.2.3 SD-based User Factor Fading

Similar ao *User Factor Fading*, a singularidade dessa técnica é de observar a estabilidade de perfil e esquecer com maior rapidez quando observada uma mudança. A diferença, medida entre o vetor antes e depois de aplicar a atualização, tende a ser maior à medida que a estabilidade do perfil diminuir, ou seja, começar a consumir itens diferentes do seu perfil comum.

Seu funcionamento é dado pela Equação 4.5, sendo α um valor para ajustar a penalização da função, Δ_u a diferença entre os vetores A_u antes e depois da realização do treino, $SD(\Delta_u)$ o desvio padrão dessa diferença, A_u^t como o vetor após ser efetuado o

treino e A_u^{t+1} o novo vetor com o esquecimento aplicado. A função $\alpha^{SD(\Delta_u)}$ sempre estará no intervalo $[0, 1)$ para $\alpha > 1$.

$$A_u^{t+1} = \alpha^{SD(\Delta_u)} \cdot A_u^t \tag{4.5}$$

4.2.4 Forgetting Unpopular Items

Esse método consiste na penalização dos itens não-populares, de maneira a aumentar a recomendação de itens populares em detrimento a produtos para usuários de nicho.

$$A_u^{t+1} = (-\alpha^{-|R(i)|} + 1) \cdot A_u^t \tag{4.6}$$

Conforme pode ser observado na Equação 4.6, $\mid R(i) \mid$ consiste na quantidade de *ratings* dado ao item i (sendo indiferente de usuários em questão) e α uma constante que controla a quantidade de penalização de itens não-populares. A função expressa no primeiro elemento da equação tem um comportamento exponencial em relação à quantidade de *ratings*, e para $\alpha > 1$ se limita entre (0, 1).

4.3 Otimização de Hiperparâmetros

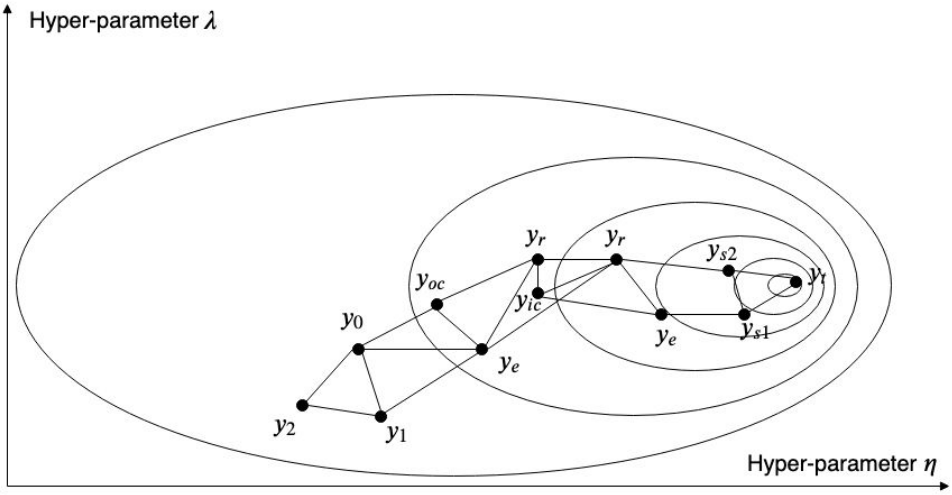
Trabalhos na literatura relativos a otimização de hiperparâmetros para sistemas incrementais de recomendação tiveram um grande destaque recentemente, com duas soluções propostas em 2019. Ambas as soluções buscam pela parametrização ideal do modelo no período em *batch* e a utilizam no cenário de *stream*. O primeiro estudo (VELOSO et al., 2019) utiliza um algoritmo de otimização para encontrar o melhor conjunto de hiperparâmetros, enquanto o segundo realiza essa busca de maneira individualizada (VELOSO; MALHEIRO; FOSS, 2019). Ambas abordagens serão apresentadas nas próximas subseções.

4.3.1 Otimização Global utilizando Nelder-Mead

Essa abordagem utiliza um conhecido algoritmo de otimização chamado Nelder-Mead (NELDER; MEAD, 1965), que utiliza três pontos iniciais em um espaço e aplica quatro operações nesses pontos para encontrar a melhor solução. Essas operações podem ser reflexão, expansão, contração interior e contração exterior. Mais detalhes podem ser encontrados no artigo (VELOSO et al., 2019), mas o processo de otimização foi ilustrado na Figura 13, sendo y_0, y_1 e y_2 pontos iniciais no formato (η, λ), gerados aleatoriamente e y_t o ponto com o valor (η_t, λ_t), encontrado pela solução com o melhor desempenho.

Apesar de buscar otimização dos parâmetros como as soluções propostas nessa dissertação, essa técnica não é personalizada. Os parâmetros encontrados são globais, ou seja, impactam todos os perfis da mesma forma.

Figura 13 – Otimização de Nelder-Mead.



Fonte: Autor, 2020. Adaptado de (OZAKI; YANO; ONISHI, 2017).

4.3.2 Individual Hyper-parameters

Outra abordagem encontrada na literatura para seleção de parâmetros é a *Individual Hyper-parameters* (IHP) (VELOSO; MALHEIRO; FOSS, 2019). Essa técnica aplica um *grid search* nos parâmetros η e λ na base de treinamento em *batch* e assume esses parâmetros, de maneira individualizada, para a fase em fluxo.

O Algoritmo 1 detalha esse funcionamento. São feitos três laços de repetição, sendo dois para explorar os parâmetros λ e η, e outro para iterar pela base de treinamento. Em seguida, é feita a predição do *rating* na linha 4, calculados os erros da linha 5 até 9, atualização do perfil do usuário na linha 10 e, caso o erro seja menor para aquele usuário, os parâmetros são substituídos nas linhas 13 e 14.

Com η_u e λ_u é possível fazer o treinamento de forma personalizada. Apesar disso, o algoritmo ainda sofre com possíveis mudanças de conceito devido a falta de adaptação.

4.4 Recency-based Negative Feedback

A tarefa de recomendação com somente interações positivas dos usuários traz um problema chamado *One-class Collaborative Filtering* (OCCF), similar ao problema de

Algoritmo 1: Otimização dos Hiperparâmetros Individuais

```
   /* Input: Conjunto de treinamento T                                  */
   /* Output: A, B e uma lista com η e λ por usuário.                   */
 1 for λ = 0 → 1 com incrementos de 0.01 do
 2     for η = 0 → 1 com incrementos de 0.01 do
 3         for r_{u,i} ← T do
 4             R̂_{u,i} = A_u · B_i^T
 5             e_{u,i} = (r_{u,i} − r̂_{u,i})^2
 6             e = e + e_{u,i}
 7             rmse = sqrt(e/n)
 8             e_u = e_u + e_{u,i}
 9             rmse_u = sqrt(e_u/n)
10             A_u ← A_u + η(e_{u,i} B_i − λ A_u)
11             if rmse ≤ rmse_old then
12                 if rmse ≤ rmse_{u,old} then
13                     η_u ← η
14                     λ_u ← λ
15             n = n + 1
```

One-class Classification conhecido na área de aprendizagem de máquina. Esse problema é decorrente do fato de estar lidando com bases compostas somente de interações positivas. A ausência de exemplos negativos faz com que o modelo perca sua generalização e faça a predição de todas as interações como positivas, indiferente do perfil do usuário.

Para mitigar o problema de OCCF, o procedimento encontrado na literatura é a inserção de exemplos negativos no sistema (VINAGRE; JORGE; GAMA, 2015a). O método proposto mantém uma FIFO (*First-In-First-Out*) com todos os itens conhecidos de cada usuário. Itens que tenham sido interagidos recentemente com um usuário são movidos para o topo da estrutura de dados. Os itens na cauda da FIFO, ou seja, os itens que não tenham tido interações recentes com o usuário, são transformados em interações negativas. Para evitar múltiplas interações negativas com os mesmos itens, aqueles selecionados para interações negativas são movidos para o topo da FIFO.

O Algoritmo 2 detalha esse procedimento. A FIFO Q é inicializada na linha 1, e logo em seguida começa a iteração pelo conjunto de *ratings* em fluxo D. Da linha 3 até a linha 7 são realizadas as interações negativas do usuário. Para isso, são selecionados l itens menos interagidos com o usuário, ou seja, no início de Q e gerado uma interação com $R_{u,i} = 0$. Em seguida, os itens que tiveram essa interação são colocados no final de Q. Por fim, a *rating* Ru, i é atualizado no modelo, removido da FIFO Q e também colocado no final.

Apesar de gerar uma adaptação, o problema solucionado por essa técnica não

Algoritmo 2: RA-ISGD: Recency-Adjusted ISGD

```
   /* Input: Conjunto de dados em fluxo D, λ, η, l                    */
   /* Output: A, B                                                    */
 1 Q ← initqueue()
 2 for (u, i) ∈ D do
 3     for k ← 1tomin(l, #Q) do
 4         j ← dequeue(Q)
 5         err_{u,j} ← 0 − A_u · B_j^T
 6         A_u ← A_u + η(err_{u,j} B_j − λA_u)
 7         enqueue(Q, j)
 8     err_{u,i} ← 1 − A_u · B_i^T
 9     A_u ← A_u + η(err_{u,i} B_i − λA_u)
10     B_i ← B_i + η(err_{u,i} A_u − λB_i)
11     if i ∈ Q then
12         remove(Q, i)
13     enqueue(Q, i)
```

faz parte do escopo desse trabalho. Além disso, a adaptação gerada não é através da aprendizagem diretamente, mas sim de geração de interações negativas para evitar que o modelo acabe por somente definir $\hat{R} = 1$ no momento de predição.

4.4.1 Considerações do Capítulo

O Capítulo 4 apresentou quatro áreas similares ao objeto de estudo do trabalho. A primeira foi (i) técnicas de apredizagem adaptativa para *batch*, que não foram feitas idealmente para se lidar com mudança de conceito mas serão utilizadas como base de comparação por ser as técnicas mais similares encontradas. A segunda são as (ii) técnicas de esquecimento, que trazem adaptabilidade aos modelos, algumas tem um tratamento personalizado mas não são técnicas de aprendizagem. A adaptação gerada por elas é através do *fading* dos perfis. Outra área é a (iii) otimização de hiperparâmetros, onde foram aplicados diferentes algoritmos para identificar o melhor conjunto (η, λ) para o modelo. Apesar disso, a primeira abordagem apresentada (Otimização com Nelder-Mead) não é personalizada, ou seja, atualiza todos os perfis da mesma forma, enquanto a segunda não trata de mudança de conceito, visto que não é refeito o treinamento igual a primeira abordagem. Por fim, a última técnica similar foi (iv) *Recency-based Negative Feedback*, que traz algum tipo de adaptação, mas para o problema de OCCF, problema que foge do escopo desse trabalho.

Com isso, foi possível demonstrar a lacuna encontrada na literatura. Com exceção das técnicas de esquecimento, não foram encontrados métodos onde seja feita a mudança de parâmetros de maneira a gerar adaptação e a solucionar problemas importantes da área como mudança de conceito para o cenário de recomendação incremental. Considerando

esse aspecto, serão utilizadas técnicas de esquecimento como *baseline* de comparação para os métodos propostos, apesar de divergirem em alguns aspectos teóricos.

Os próximos capítulos, contemplando a Parte II dessa dissertação, serão expostas as contribuições científicas dadas pela atual pesquisa, estabelecendo assim a área de aprendizagem adaptativa personalizada. O Capítulo 5 apresenta um método para lidar com mudanças de conceito, *Background Profiling*, utilizando dessa vez os detectores de mudança vistos na subseção 2.1.3. Por fim, no Capítulo 6 também será apresentado o método ADADRIFT, desenvolvido para lidar com o mesmo problema de mudança de conceito.

Parte II

Contribuições

5 Background Profiling

Um dos maiores problemas na mineração de fluxos de dados é a adaptabilidade dos modelos. Conseguir prontamente identificar uma mudança de conceito e se ajustar a ela vem se demonstrando um processo árduo e complexo computacionalmente em muitos casos. Uma solução encontrada para aprimorar essa adaptabilidade é a utilização de uma técnica chamada *background learner*.

Visto que esse problema também pode ser encontrado na área de sistemas de recomendação, a presente proposta de dissertação tem como objetivo utilizar os conceitos principais dessa técnica e adaptá-los para a tarefa de *profiling*, que consiste na definição do perfil de consumo do usuário.

5.1 Background Learner e Background Profiling

Background learner é uma técnica que consiste em treinar um *learner* em segundo plano para assumir no momento em que for identificada uma mudança de conceito. Utilizando esse mecanismo, é possível ter uma maior adaptabilidade com a preparação especialmente para a mudança, ao mesmo tempo que o conceito defasado é descartado. Essa abordagem pode ser observada obtendo bons resultados em Gomes et al. (2017) e Ikonomovska et al. (2009), por exemplo.

Essa abordagem é possível devido aos detectores, que trabalham nas zonas de *in control*, *warning* e *out of control*. A zona de *in control* remete a um conceito estável, ou seja, não são necessários procedimentos adicionais. Ao atigir na zona de *warning*, é iniciado o treinamento de um *learner* sem afetar os resultados do algoritmo, ou seja, as novas instâncias serão direcionadas para ambos os *learners* enquanto as predições serão feitas apenas pelo *learner* principal. Por fim, quando detectada de fato uma mudança de conceito, o *background learner* assume a posição principal do modelo, enquanto o principal é descartado, partindo do pressuposto que o classificador treinado em segundo plano irá apresentar uma maior adaptabilidade justamente por ter sido apresentado com maior ênfase para o novo conceito vigente.

A utilização direta dessa técnica adaptativa para sistemas de recomendação é inviável devido ao motivo que, ao tratar de fatoração de matriz, o *learner* comporta o perfil de todos os usuários e items do sistema. Dessa forma, assim como a aplicação da técnica de esquecimento *Last N retention* (detalhada na subseção 4.2.1) necessita de adaptação para janelas deslizantes por usuário, um *background learner* teria a mesma necessidade. Para isso, portanto, é necessário realizar a tarefa de *profiling* em segundo plano, aplicando

o mesmo processo de ajuste de perfil, descrito nas Equações 2.7 e 2.8, em ambos os perfis ao mesmo tempo. A adaptação deverá ser de acordo com o tipo de perfil treinado, sendo a Equação 2.7 para usuário e a Equação 2.8 para items.

5.2 O Método

A proposta desse capítulo apresenta o mecanismo de treinamento BP, acrônimo para *Background Profiling*. BP se baseia na definição de perfil em segundo plano, e no aumento da taxa de aprendizagem visando elevar a adaptabilidade do modelo. Esse processo de aumento significativo da taxa de aprendizagem será referido como superaprendizagem daqui em diante.

Um grande problema da superaprendizagem é com a aprendizagem de conceitos equivocados. Isso torna esse processo inviável para métodos que não tenham ferramentas para lidar com *outliers*. Levando em consideração que o método em questão é amparado por detectores, esse processo acaba se tornando viável caso o detector em questão não tenha uma taxa elevada de falsos positivos. A vantagem da superaprendizem é a adaptabilidade dos modelos. É possível obter perfis mais ajustados em uma menor quantidade de tempo.

O seu funcionamento pode utilizar como base diversos algoritmos de fatoração de matriz (e.g. BRISMF e ISGD), e realiza uma complementação de etapas somente no processo de treinamento dos mesmos. O custo computacional adicionado com o BP é $O(1)$ e custo espacial de $O(k \times n)$, com o custo computacional de $O(k)$ para atualização/predição e custo espacial base da técnica de fatoração de matriz $O(k \times n)$, sendo k o número de dimensões e n o número de perfis (usuários e itens). O custo espacial varia de acordo com a incidência de mudanças de conceito, e só ocorrendo $k \times n$ se todos os perfis estiverem em uma mudança de conceito ao mesmo tempo, o que é no mínimo um cenário incomum. O caso médio tende a ser inferior a esse valor. De qualquer forma, no pior caso, o método não tem um impacto da ordem do custo espacial base da técnica de fatoração.

Visto que é feita somente uma complementação no processo de treinamento, esse mecanismo se torna facilmente acomplável a outros algoritmos de fatoração de matriz presentes na literatura. Esse diferencial pode ser observado de maneira mais clara na Figura 14, onde o modelo incremental de recomendação consiste em somente um dos componentes do BP. Após o treinamento do modelo principal é realizada uma série de procedimentos para, caso necessário, efetuar o treinamento do perfil em segundo plano ou ainda concluir a substituição do perfil principal. Antes de entrar em detalhes de implementação, é necessária uma definição dos parâmetros utilizados pelo método. O BP necessita de uma taxa de superaprendizagem, utilizada no momento em que é necessário um aumento de aprendizagem para acompanhar a mudança de conceito. Esse parâmetro será representado por η_{BP}. Além disso, será necessário um detector. O detector por si só

Algoritmo 3: BP-ISGD

Input : q_{id}, s_{id}, R
Parâmetros do BP : D, η_{BP}
Parâmetros do MF : η, λ
Variáveis Internas : Matrizes Q, S, C

/* A matriz Q será aplicada o BP. Já a matriz S será o domínio auxiliar, onde não será aplicado a técnica. A matriz D deve ser iniciada com uma instância de detector para cada perfil. A matriz C será a auxiliar, onde o perfil terá a superaprendizagem e não será utilizada para recomendação. */

$oldFactors \leftarrow Q[q_{id}]$
/* Predição do rating */
$\hat{R} \leftarrow Q[q_{id}] \cdot S[s_{id}]$
$err_{ui} \leftarrow R - \hat{R}$
/* Atualização dos vetores */
$Q[q_{id}] \leftarrow Q[q_{id}] + \eta(err_{ui} S[s_{id}] - \lambda Q[q_{id}])$
$S[s_{id}] \leftarrow S[s_{id}] + \eta(err_{ui} Q[q_{id}] - \lambda S[s_{id}])$
$difference \leftarrow oldFactors - Q[q_{id}]$
$lastStatus \leftarrow D[q_{id}].status()$
/* Update no detector do perfil */
$D[q_{id}].update(\sigma_{difference})$
/* Quando for detectado uma mudança no status */
if $lastStatus \neq D[q_{id}].status()$ **then**
  if $D[q_{id}].status() = In\ Control$ **then**
    $C[q_{id}] \leftarrow \emptyset$
  else if $D[q_{id}].status() = Warning$ **then**
    $C[q_{id}] \leftarrow clone(Q[q_{id}])$
  else if $D[q_{id}].status() = Out\ of\ Control$ **then**
    $Q[q_{id}] \leftarrow C[q_{id}]$
    $C[q_{id}] \leftarrow \emptyset$
if $D[q_{id}].status() = Warning$ **then**
  $C[q_{id}] \leftarrow C[q_{id}] + \eta_{BP}(err_{ui} S[s_{id}] - \lambda C[q_{id}])$
$lastStatus \leftarrow D[q_{id}].status()$

necessitará de ajustes na parametrização.

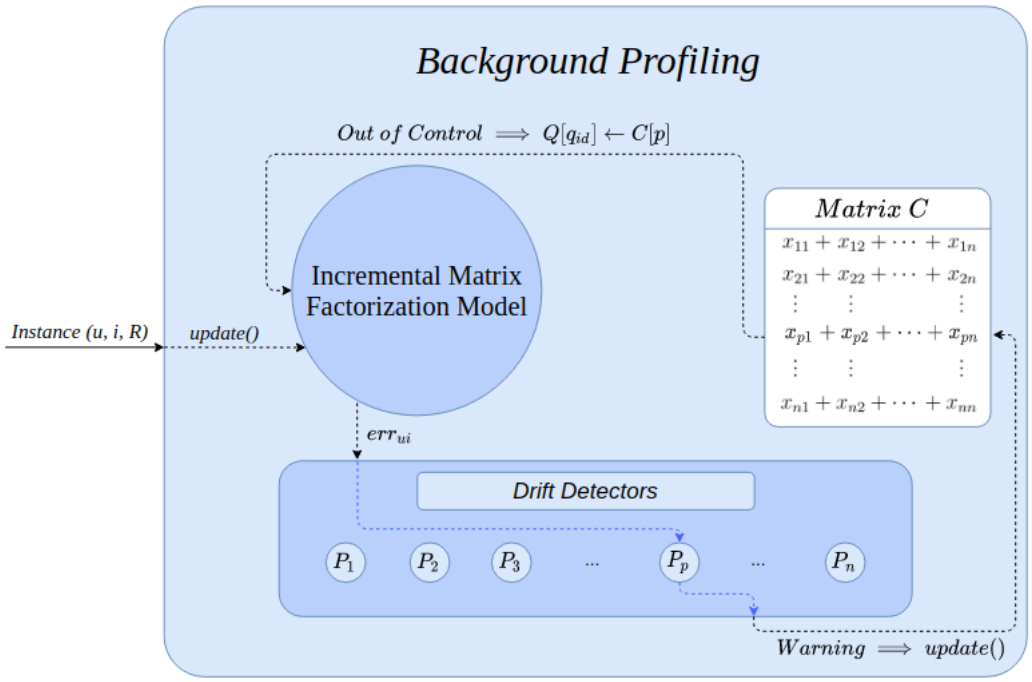
Conforme o protocolo de validação exposto no Capítulo 3, as bases utilizadas terão o enfoque de mudança de conceito em somente um dos domínios por vez, seja usuário ou item. Como aplicar para ambos usuários e itens ao mesmo tempo pode não ser vantajoso devido a probabilidade de falsos positivos em um local com baixa incidência de mudanças de conceito, o algoritmo será aplicado somente em um domínio por vez. A matriz Q representará o grupo onde será aplicada a técnica (usuário ou item) e S o outro domínio onde não será aplicada. Por exemplo: CD-NF e CD-NF5, que são bases com mudanças de conceito orientadas a itens, terão B como a matriz Q, de itens, e A como a matriz S, de usuários. Já para CD-PL e CD-LFM, que possuem mudanças de conceito orientadas

a usuários, o cenário será o contrário. A matriz A será atribuída a Q e B será S. É importante ter essa notação em mente porque ela aparecerá no Algoritmo 3 e nas outras Figuras 14 e 15, detalhadas na sequência. Outro detalhe importante a ser mencionado sobre o Algoritmo 3 é o vetor D, que contém os detectores para cada usuário. Esse detector necessita de dois métodos, $status()$ e $update(error)$. O primeiro deverá retornar o status atual do modelo entre *In Control*, *Warning* e *Out of Control*, enquanto o $update(error)$ irá tomar os devidos procedimentos de atualização para cada detector, dado o erro do modelo. O erro utilizado será o $\sigma_{difference}$, que é o desvio padrão da diferença entre os perfis antes e depois da atualização.

A adição de etapas realizadas pelo BP pode ser observada no treinamento retratado no Algoritmo 3 e na máquina de estados ilustrada na Figura 15. As linhas de 1 a 5 são relativas ao treinamento padrão da matriz de fatoração. O exemplo utilizado é o ISGD, que não necessita de um *bias* adicional.

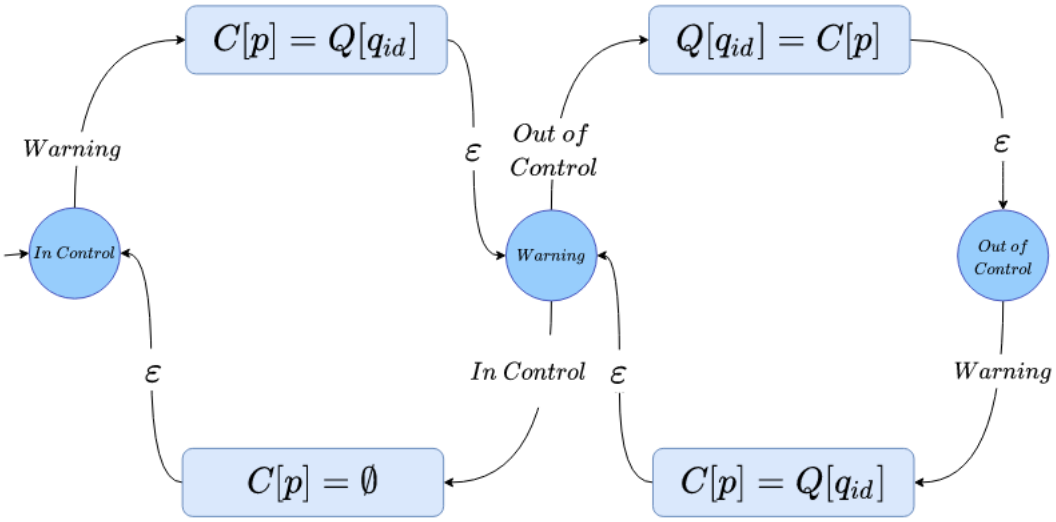
A partir da linha 6, em conjunto com a linha 1, temos a adição das instruções para o BP. Primeiro será calculada a diferença entre os perfis antes e depois da atualização. Logo em seguida, será extraído o desvio padrão dessa diferença e colocado como *input* para o detector. Enquanto o estado do detector estiver sendo categorizado com *in control*, da mesma forma como no processo de *background learner*, não é necessário nenhum

Figura 14 – Diagrama Esquemático do BP.



Fonte: Autor, 2019.

Figura 15 – Máquina de Estado do BP.



Fonte: Autor, 2019.

procedimento adicional.

Ao entrar na zona de *warning*, o perfil do usuário será clonado (conforme pode ser observado na linha 9) e começará a realizar o treino junto ao principal, com uma taxa de aprendizagem elevada. Esse treinamento está representado na linha 18, e o controle da taxa de aprendizagem será permitido pelo hiperparâmetro η_{BP}, que deve permanecer no intervalo $\eta_{BP} > \eta$. Enquanto estiver na zona de *warning*, o perfil em segundo plano irá aprender a mudança de conceito com maior velocidade ao mesmo tempo em que, como dito anteriormente, caso volte ao estado de estabilidade, terá esse perfil de superapredizagem descartado.

Ao ser detectada a mudança de conceito, o perfil de superaprendizagem será definido como principal e o anterior descartado, conforme pode ser observado nas linhas 15 e 16 do algoritmo. Todas essas transições também podem ser acompanhadas pela Figura 15. As transições para o segundo estado são realizadas na linha 13 do Algoritmo 3, as transições para o terceiro estado na linha 15, e as transições para o primeiro estado na linha 11.

5.2.1 Detectores de Mudança de Conceito

Como apresentado anteriormente, detectores de mudança de conceito são técnicas que tem como o objetivo identificar o momento de troca no conceito vigente do fluxo de dados considerando as suas três regiões de estabilidade no conceito: *in control*, *warning* e *out of control*.

Para conseguir realizar essa detecção, é necessário que seja utilizado algum tipo de métrica como entrada para o detector. Usualmente é utilizado o erro do modelo como entrada e partido do pressuposto que quando houver um aumento significativo, isso está diretamente ligado com uma mudança de conceito[1]. Para essa experimentação, será utilizada o coeficiente de estabilidade devido a estar diretamente ligada com uma métrica de erro do modelo.

Para obter o coeficiente de estabilidade, é primeiramente realizado o treino do perfil e observada a diferença $(Q^t_{q_{id}} - Q^{t+1}_{q_{id}})$. O desvio padrão dessa diferença representa o impacto do treinamento no perfil, podendo também ser interpretado como a estabilidade do perfil. Perfis que estejam passando por mudança têm uma tendência a maiores valores do que perfis que estejam reafirmado suas preferências.

5.3 Resultados

O BP precisa de ajuste para o parâmetro η_{BP} e a definição do detector. Os detectores separados para uso foram os dois apresentados anteriormente de gráfico de controle: EDDM e DDM. O EDDM necessita da definição de dois *thresholds* α e β, geralmente utilizados como $0,95$ e $0,90$ para definir (GONÇALVES et al., 2014), respectivamente, as duas zonas (*warning* e *out of control*). Já o DDM, que também necessita de dois valores níveis (*warning* e *out of control*), foi também definido como o padrão de 2 para o *warning* e 3 para o *out of control*. Um *grid search* foi realizado com o objetivo de determinar a configuração mais adequada para o algoritmo. O universo explorado foi $\eta_{BP} \in \{0,04; 0,08; 0,16\}$, e ambos os detectores DDM e EDDM. O número de fatores latentes k foi definido como 60 e o fator de regularização $\lambda = 0,001$ para ambos BRISMF e o ISGD.

Para realizar essa análise, o BP seguiu o protocolo apresentado no Capítulo 3, aplicando a diferentes esquemas de aprendizagem nos diferentes conjuntos de dados. A melhor configuração global [2] do BP foi $\eta_{BP} = 0.16$ e o EDDM como detector.

Para o ADADELTA, o hiper-parâmetro principal é a taxa de decaimento, que foi explorada no conjunto $\rho \in \{0,99; 0,95\}$. O melhor resultado mostrado foi $\rho = 0,99$. Usamos $\epsilon = 1^{-6}$, que foi recomendado no artigo em que o ADADELTA foi proposto (ZEILER, 2012).

[1] Outros dois problemas podem causar aumento de erro: *cold start* e *shilling attack*. Apesar da diminuição do *cold start*, ainda é possível que ele ocorra, mas ele também seria solucionado pela técnica. *Shilling attack* consiste na inserção mal intencionada de *ratings* no sistema para aumentar a recomendação, e ele não fez parte do escopo do trabalho e o algoritmo ainda apresentaria vulnerabilidades para esse problema.

[2] É importante destacar que o processo de ajuste foi realizado globalmente; isto é, a melhor configuração foi obtida considerando todos os conjuntos de dados. Esse procedimento foi escolhido para evitar um otimismo dos resultados, mas também implica que o resultado poderá ser aprimorado se a pesquisa de hiperparâmetros for realizada considerando as nuances de cada conjunto de dados.

Tabela 4 – Resultado do BP em ganho percentual em relação ao modelo base em Recall@N com $N \in \{1, 5, 10, 20\}$.

Algoritmo	BRISMF		ISGD		Média
	CD-PL	CD-NF	CD-NF5	CD-LFM	
Recall@1					
Base Model	1.31×10^{-1}	5.56×10^{-3}	1.20×10^{-3}	1.07×10^{-2}	-
ADADELTA	−92.68%	+105.14%	+514.15%	+27.80%	+138.60%
BP	+0.78%	+6.91%	+37.83%	+14.52%	+15.01%
Recall@5					
Base Model	2.57×10^{-1}	1.69×10^{-2}	3.88×10^{-3}	2.49×10^{-2}	-
ADADELTA	−87.90%	+75.46%	+419.52%	+36.81%	+110.97%
BP	+1.25%	+1.59%	+61.37%	+5.18%	+17.35%
Recall@10					
Base Model	3.55×10^{-1}	2.96×10^{-2}	8.05×10^{-3}	4.85×10^{-2}	-
ADADELTA	−83.18%	+67.58%	+331.82%	+10.25%	+81.62%
BP	+0.88%	+0.09%	+64.27%	+3.05%	+17.07%
Recall@20					
Base Model	4.82×10^{-1}	5.25×10^{-2}	1.77×10^{-2}	1.20×10^{-1}	-
ADADELTA	−75.65%	+63.66%	+255.42%	−34.01%	+52.36%
BP	+1.04%	−1.62%	+69.88%	+0.19%	+17.37%

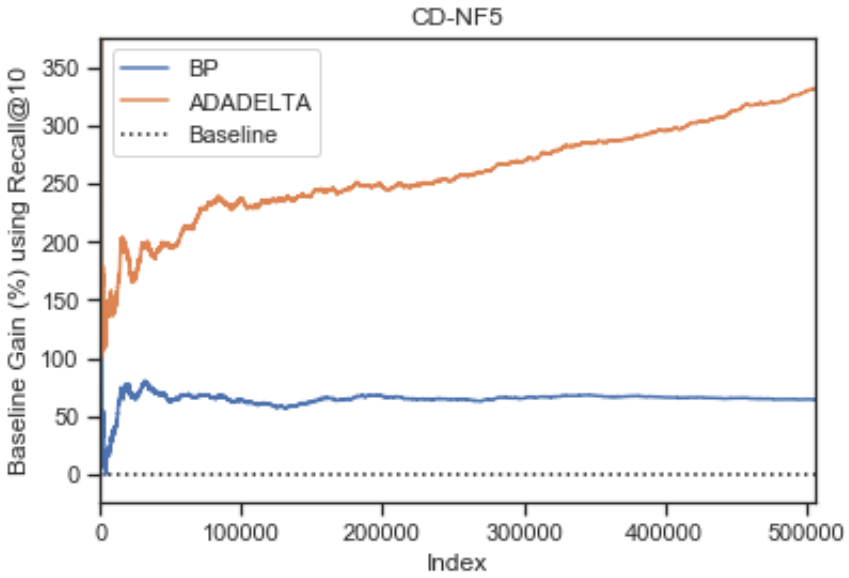


Figura 16 – Os ganhos das variações comparadas ao modelo base (ISGD), medidos em porcentagem de ganho do Recall@10. O conjunto de dados usado foi o CD-NF5, que consiste no tratamento de desvio de conceito realizado no conjunto de dados Netflix *positive-only*.

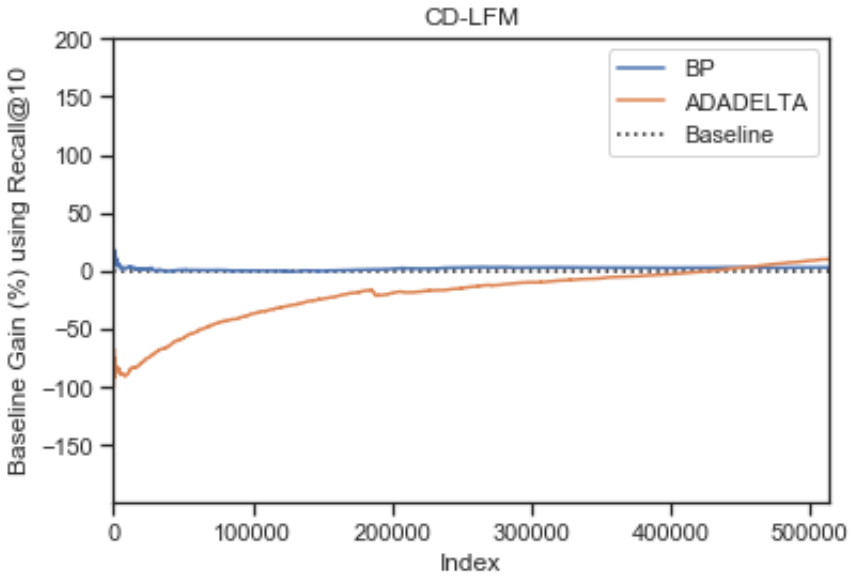


Figura 17 – Os ganhos das variações comparadas ao modelo base (ISGD), medidos em ganho percentual do Recall@10. O conjunto de dados utilizado foi o CD-LFM, que consiste no tratamento de desvio de conceito realizado em usuários do LastFM 1K.

Os resultados dos experimentos foram apresentados na Tabela 4, considerando o percentual de ganho em relação ao modelo base, isto é, BRISMF e ISGD, sem nenhuma outra técnica. Essa abordagem foi usada devido à escala de Recall, que geralmente permanece pequena e dificulta a mensuração/comparação de ganhos. O cálculo para se obter a o ganho relativo pode ser observada na Equação 5.1, sendo $P_{absoluta}$ a performance absoluta do modelo, $P_{baseline}$ a performance do *baseline* e $G_{relativo}$ o ganho de utilização do algoritmo.

$$G_{relativo} = \frac{P_{absoluta} - P_{baseline}}{P_{baseline}} \tag{5.1}$$

O resultado obtido pelo BP foi tímido. Sua principal característica foi a estabilidade diante do baseline. Isso demonstra que o algoritmo está falhando em identificar uma mudança de conceito ou não está conseguindo ter tempo para essa adaptação. Seu melhor desempenho foi no *dataset* CD-NF5, que pode ser observado na Figura 16. O BP performa de maneira estável, sem grandes crescimentos ao decorrer do tempo. O fato de limitar a taxa de aprendizagem a dois parâmetros, η e η_{BP}, não parece ter um benefício muito visível. O ADADELTA, ao realizar essa aprendizagem de maneira dinâmica, tem um resultado bem mais significativo. Apesar disso, essa liberdade acaba sendo prejudicial em alguns momentos, como no caso da CD-PL. Essa liberdade pode trazer uma grande alteração aprendizagem em momentos onde é necessário ponderar, e uma aprendizagem reduzida em momentos onde deveria ter mais adaptabilidade.

O cenário encontrado no CD-LFM demonstra bem isso. Conforme pode ser acompanhado na Figura 17, no começo do fluxo existe uma queda significativa para o ADADELTA, que volta a se recuperar depois de um tempo. Enquanto isso, o BP mantém um resultado bem similar ao baseline, sendo superior ao ADADELTA por um bom tempo. O único desempenho negativo do BP é para o Recall@20 do CD-NF, mesmo assim sendo por apenas -1.62%. BP se mostra uma versão bem conservadora de adaptação para sistemas incrementais de recomendação.

5.4 Considerações do Capítulo

Técnicas de *background learner* podem ser encontradas em mineração de fluxos de dados, e consistem na definição de modelo em segundo plano para efetuar a troca em um momento de instabilidade de conceito. O presente capítulo apresentou uma adaptação de *background learner* para sistemas adaptativos de recomendação, com o método BP. Em relação ao BP, foi definida sua máquina de estado, na Figura 15, e o pseudo código no Algoritmo 3. Ele foi validado de acordo com o protocolo exposto no Capítulo 3, e seu resultado representa um comportamento conservador, onde se mantém bem até em cenário com redução da média de interações.

O Capítulo 6 irá trazer outra abordagem de aprendizagem personalizada adaptativa, dessa vez referida como ADADRIFT. Seu funcionamento visa deixar a taxa de aprendizagem mais dinâmica do que o BP, podendo tanto aumentar em momentos de mudança de conceito quanto diminuir nos momentos de estabilidade do perfil. Da mesma forma que esse capítulo, o método será apresentado com uma breve experimentação de validação.

6 ADADRIFT

Sistemas de recomendação devem ser capazes de se adaptar de acordo com as mudanças nas preferências dos usuários e interações com itens ao decorrer do tempo. A hipótese para o proposta do ADADRIFT é que nem todos os perfis experienciam as mudanças nos mesmos passos. Existem perfis passando por grande estabilidade no conceito enquanto outros buscam novas experências e novos estilos musicais, por exemplo.

Com isso em mente, este capítulo propõe o ADADRIFT, que consiste numa técnica de aprendizagem dinâmica. Esse método visa acompanhar os diferentes ritmos de aprendizagem, de uma maneira personalizada e independente da técnica de fatoração utilizada, e.g., BRISMF e ISGD. Como pode ser observado no diagrama apresentado na Figura 18, ADADRIFT adiciona somente uma etapa instantânea no processamento do modelo, gerando um η_p personalizado para cada usuário ou item p. O ADADRIFT demanda um custo computacional de ordem $O(1)$ e um custo em memória $O(n)$, não impactando no custo computacional de $O(k)$ do algoritmo base para atualização/predição e custo espacial de $O(k \times n)$ por padrão das técnicas de fatoração de matriz, sendo k a quantidade de fatores latentes e n a quantidade de perfis (usuários e itens). A próxima seção serão minusciosamente detalhados os requisitos do ADADRIFT.

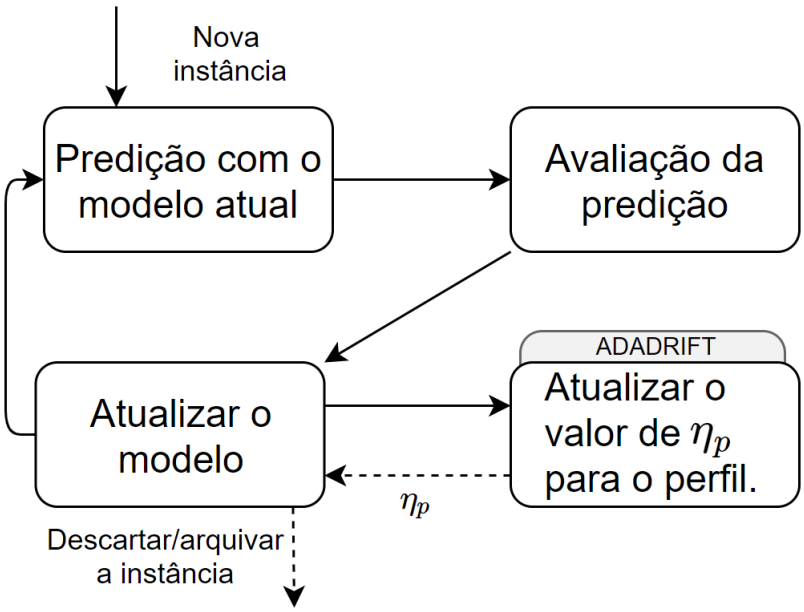


Figura 18 – Protocolo Prequential ao utilizar o ADADRIFT. Adaptado de Jorge et al. (2017).

6.1 O Método

ADADRIFT possui três hiper-parâmetros: δ_L, δ_S e α. O hiper-parâmetro α controla o tamanho do impacto do método na aprendizagem, sendo futuramente necessário para obter a taxa de aprendizagem dinâmica η_p.

Os parâmetros δ_L e δ_S definem o tamanho de duas janelas móveis usadas para aumentar ou diminuir a taxa de aprendizado de cada perfil. Eles trabalham em pares com δ_L em relação à média de longo prazo, enquanto δ_S representa a média de curto prazo. O relacionamento deles mostra se um desvio de conceito está acontecendo ou não. Para que o δ_S consiga captar mudanças de curto prazo, e δ_L mudanças de longo prazo, é necessário sempre manter $\delta_S < \delta_L$.

Depois de definir os valores δ_L e δ_S, o ADADRIFT calcula as médias móveis μ_L e μ_S respectivamente, de acordo com a Equação 6.1. O μ^t representa a média móvel μ no início da interação, ou seja, no momento t, enquanto μ^{t+1} é a média móvel no final da interação, ou seja, no momento $t+1$. Essa mesma notação também é usada nas Equações 6.2 e 6.3, com as variáveis σ e η nesses casos.

O valor de n_L representa o número de elementos x já na janela e permanece no intervalo $1 < n_L \leq \delta_L$, fixando em δ_L após a conclusão da janela. O n_S segue o mesmo processo, com δ_S como o valor máximo possível. As variáveis n_S e n_L integrarão as médias móveis mu_S e mu_L.

O valor x representa a estabilidade do usuário ao longo do tempo e é usado para ambas as médias móveis μ_L e μ_S. A estabilidade do usuário (variável x) pode ser obtida pelo desvio padrão do gradiente já calculado nas Equações 2.7 e 2.8, representado respectivamente por $(err_{ui}B_i - \lambda A_u)$ e $(err_{ui}A_u - \lambda B_i)$, e usado no treinamento do perfil do usuário/item.

$$\mu^{t+1} = \frac{\mu^t * (n-1) + x}{n} \tag{6.1}$$

A média móvel μ_L refere-se à média de longo prazo, ou seja, tem menos sensibilidade e representa a estabilidade de longo prazo do usuário. Em conjunto com μ_L, σ_L é o desvio padrão incremental e é usado para definir a intensidade das alterações. Sua atualização é responsável pela média móvel e pode ser observada na Equação 6.2.

Como $\delta_S < \delta_L$, o comportamento de μ_S é mais instável, buscando um equilíbrio entre detectar rapidamente um desvio de conceito (pela média móvel de curto prazo μ_S) e ter resiliência a *outliers* (pelo média móvel de longo prazo μ_L). Essa relação entre as duas médias móveis permite a identificação de uma mudança de conceito.

$$\sigma_L^{t+1} = \frac{\sigma_L^t * (n_L - 1) + (x - \mu_L)^2}{n_L} \tag{6.2}$$

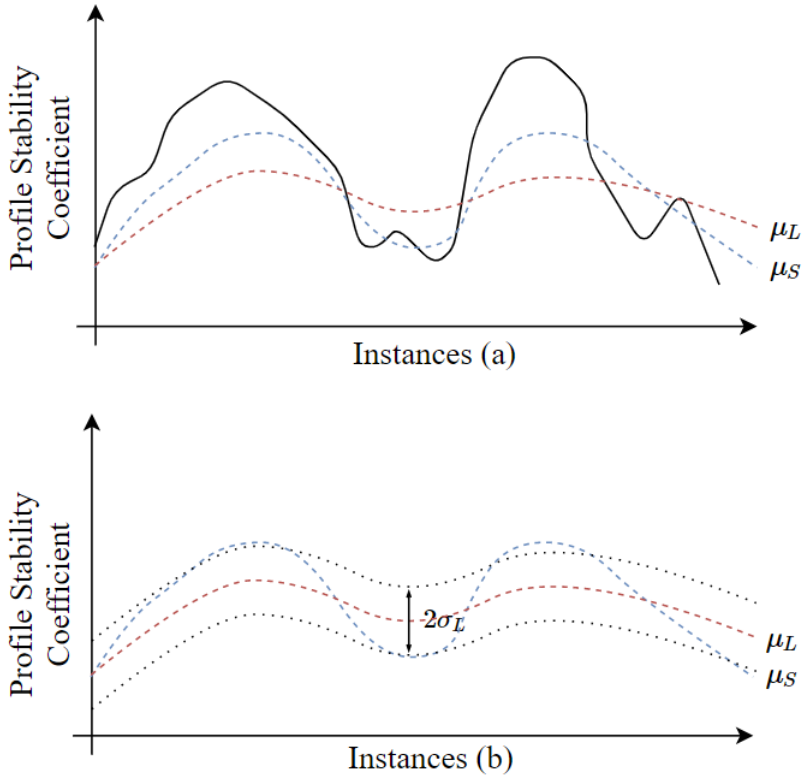


Figura 19 – Mudanças de perfil ao longo do tempo e o relacionamento com médias de curto e longo prazo.

O comportamento desse erro ao longo do tempo pode ser visto na Figura 19a, onde μ_L e μ_S estão representados. Nos momentos em que μ_S é maior que μ_L, a taxa de aprendizado dinâmica aumenta proporcionalmente à sua diferença, ou seja, $(\mu_S - \mu_L)$, enquanto a taxa de aprendizado dinâmica diminui no cenário oposto.

A Equação 6.3 usa σ_L para observar quão fora do padrão são os novos valores do coeficiente de estabilidade, impactando diretamente a taxa de aprendizado dinâmico conforme a diferença $(\mu_S - \mu_L)$ cresce. O relacionamento com σ_L pode ser visto na Figura 19b. Essa alteração na taxa de aprendizado do usuário é calculada com a Equação 6.3.

É importante ressaltar que cada usuário/item deve ter médias móveis privadas, desvio padrão móvel e taxa de aprendizado personalizada, porque a mudança de conceito ocorre individualmente, não para o modelo inteiro.

$$\eta^{t+1} = \eta^t * \alpha^{[\frac{\mu_S - \mu_L}{\sigma_L}]} \tag{6.3}$$

O pseudocódigo pode ser observado no Algoritmo 4. A notação com η_p representa a variável personalizada η para o perfil p. Da mesma forma, outras variáveis como μ_{L_p}, μ_{S_p}, σ_{L_p} representam respectivamente os valores personalizados μ_L, μ_S, σ_L, para o perfil p. Cada variável têm $\mid A \mid + \mid B \mid$ elementos, sendo um para cada perfil.

As linhas 2 a 4 atualizam os valores de μ_{L_p}, μ_{S_p} e σ_{L_p}. Essas linhas usam ambas as Equações 6.1 e 6.2. É importante notar que x é um vetor da diferença entre os fatores latentes depois da atualização, portanto o valor utilizado para o processamento é o desvio padrão desse vetor, representado por x_σ. Com esses valores definidos para o perfil, a Equação é 6.3 é aplicada para as linhas 5 e 6, resultado em η_p. Logo após, os valores de n_L e n_S são atualizados respeitando respectivamente os limites δ_L e δ_S. Finalmente, na linha 11, o valor de η_p é retornado.

Algoritmo 4: ADADRIFT

/* **Input**: $(err_{ui}B_i - \lambda A_u)$ and p */
/* **Output**: η_p */
/* **Variáveis**: δ_L, δ_S, α and arrays μ_L, μ_S, η, σ_L n_L, n_S */
/* **Inicialização**: Matrizes com tamanho $\mid A \mid$. η é uma matriz e cada posição é inicializada com o valor padrão da técnica de fatoração da matriz. As variáveis μ_L, σ e μ_S não requerem valores de inicialização, e n é inicializado em 1 para cada usuário. */

1 $x \leftarrow (err_{ui}B_i - \lambda A_u)$
2 $\mu_{L_p} \leftarrow (\mu_{L_p} * (n_{L_p} - 1) + x_\sigma)/n_{L_p}$
3 $\mu_{S_p} \leftarrow (\mu_{S_p} * (n_{S_p} - 1) + x_\sigma)/n_{S_p}$
4 $\sigma_{L_p} \leftarrow \left(\sigma_{L_p} * (n_{L_p} - 1) + (x_\sigma - \mu_{L_p})^2\right)/n_{L_p}$
5 $s \leftarrow (\mu_{S_p} - \mu_{L_p})/\sigma_{L_p}$
6 $\eta_p \leftarrow \eta_p * \alpha^s$
7 **if** $n_{L_p} < \delta_L$ **then**
8 $\quad n_{L_p} \leftarrow n_{L_p} + 1$
9 **if** $n_{S_p} < \delta_S$ **then**
10 $\quad n_{S_p} \leftarrow n_{S_p} + 1$
11 **return** η_p

Após calcular o novo η_p para o perfil p, os fatores latentes de p são atualizados de acordo com as Equações 6.4 e 6.5, substituindo ambas as Equações 2.7 e 2.8. Esse processo de substituição de η pode ser aplicado a todos os algoritmos que usam esse parâmetro, mostrando assim versatilidade e adaptabilidade do método proposto.

$$A_u \leftarrow A_u + \eta_p(err_{ui}B_i - \lambda A_u) \tag{6.4}$$

$$B_i \leftarrow B_i + \eta_p(err_{ui}A_u - \lambda B_i) \tag{6.5}$$

6.2 Resultados

O ADADRIFT possui 3 parâmetros que requerem ajuste: α, δ_L, δ_S. Portanto, um *grid search* foi realizado com o objetivo de determinar a configuração mais adequada. O

Tabela 5 – Resultado do ADADRIFT em ganho percentual em relação ao modelo base em Recall@N com $N \in \{1, 5, 10, 20\}$.

Algoritmo	BRISMF		ISGD		Média
	CD-PL	CD-NF	CD-NF5	CD-LFM	
Recall@1					
Base Model	1.31×10^{-1}	5.56×10^{-3}	1.20×10^{-3}	1.07×10^{-2}	-
ADADELTA	−92.68%	+105.14%	+514.15%	+27.80%	+138.60%
ADADRIFT	−4.28%	+156.91%	+401.65%	+908.22%	+365.63%
Recall@5					
Base Model	2.57×10^{-1}	1.69×10^{-2}	3.88×10^{-3}	2.49×10^{-2}	-
ADADELTA	−87.90%	+75.46%	+419.52%	+36.81%	+110.97%
ADADRIFT	−4.19%	+99.68%	+295.51%	+622.41%	+253.35%
Recall@10					
Base Model	3.55×10^{-1}	2.96×10^{-2}	8.05×10^{-3}	4.85×10^{-2}	-
ADADELTA	−83.18%	+67.58%	+331.82%	+10.25%	+81.62%
ADADRIFT	−4.30%	+71.02%	+211.25%	+346.06%	+156.01%
Recall@20					
Base Model	4.82×10^{-1}	5.25×10^{-2}	1.77×10^{-2}	1.20×10^{-1}	-
ADADELTA	−75.65%	+63.66%	+255.42%	−34.01%	+52.36%
ADADRIFT	−4.99%	+47.85%	+143.51%	+113.75%	+75.03%

universo explorado foi $\alpha \in \{1,001; 1,1\}$; $(\delta_L, \delta_S) \in \{(20, 10); (100, 50); (200, 100)\}$. O número de fatores latentes k foi definido como 60 e o fator de regularização $\lambda = 0,001$ para ambos BRISMF e o ISGD. Para realizar essa análise, o ADADRIFT seguiu o protocolo apresentado no Capítulo 3, aplicando diferentes esquemas de aprendizagem nos diferentes conjuntos de dados. A melhor configuração global [1] do ADADRIFT foi $\alpha = 1.1$ e $(\delta_L, \delta_S) = (200, 100)$. Para o ADADELTA, o hiper-parâmetro principal é a taxa de decaimento, que foi explorada no conjunto $\rho \in \{0,99; 0,95\}$. O melhor resultado mostrado foi $\rho = 0,99$. Usamos $\epsilon = 1^{-6}$, que foi recomendado no artigo em que o ADADELTA foi proposto (ZEILER, 2012).

Os resultados dos experimentos estão apresentados na Tabela 5, considerando o percentual de ganho em relação ao modelo base, isto é, BRISMF e ISGD, sem nenhuma outra técnica. Essa abordagem foi usada devido à escala de Recall, que geralmente permanece pequena e dificulta a mensuração/comparação de ganhos. Devido também a essa escala, alguns ganhos ficaram bem elevados. Um ganho pequeno comparado a um resultado menor ainda acaba trazendo porcentagens grandes. Esse ganho relativo foi calculado conforme a Equação 5.1, sendo $P_{absoluta}$ a performance absoluta do algoritmo e $P_{baseline}$ a performance do algoritmo nativo.

[1] É importante destacar que o processo de ajuste foi realizado globalmente; isto é, a melhor configuração foi obtida considerando todos os conjuntos de dados. Esse procedimento foi escolhido para evitar um otimismo dos resultados, mas também implica que o resultado poderá ser aprimorado se a pesquisa de hiperparâmetros for realizada considerando as nuances de cada conjunto de dados.

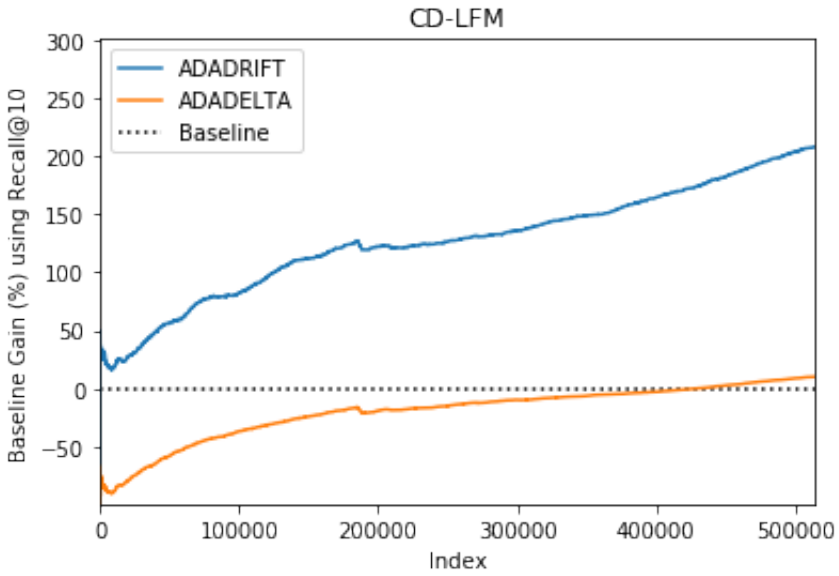


Figura 20 – Os ganhos das variações comparadas ao modelo base (ISGD), medidos em ganho percentual do Recall@10. O conjunto de dados utilizado foi o CD-LFM, que consiste no tratamento de desvio de conceito realizado em usuários do LastFM 1K.

Para observar o ganho ao longo do tempo, os resultados obtidos para o Recall@10 para os conjuntos de dados CD-LFM e CD-NF5 podem ser vistos nas Figuras 20 e 21, respectivamente. O conjunto de dados CD-LFM possui o maior número médio de classificações por perfil, com cerca de 4 vezes mais que o segundo maior, CD-NF5. Por esse motivo, a incidência de mudanças de conceito tende a ser maior. Essa característica mostrou um grande impacto nos resultados, como pode ser visto na Figura 20. O ADADRIFT possui um ganho significativo e constante ao longo do tempo em comparação com o baseline e o ADADELTA. Essa melhoria disparou nas últimas 125.000 instâncias, atingindo a marca de 346,06% na última instância [2]. Esse crescimento é ainda maior, resultando em um desempenho de 908,22% para o Recall@1 em comparação com o ganho de apenas 27,80% ao usar o ADADELTA, representando 32,67 vezes mais eficaz. Esse alto desempenho deve-se à alteração das preferências dos usuários e às adaptações do ADADRIFT, que estão acumulando nos resultados finais.

Por outro lado, quando o número médio de classificações por perfil diminui para os conjuntos de dados, há uma degradação no ganho do algoritmo. Em relação ao conjunto de dados CD-NF5, é possível observar uma vantagem do ADADELTA, embora o ADADRIFT ainda apresente ganhos em relação ao modelo base e seja um algoritmo competitivo. A versão numérica do conjunto de dados da Netflix, CD-NF, tem um resultado muito semelhante entre os dois algoritmos, com pequenas vantagens alternando entre as métricas

[2] Devido ao funcionamento da Recall detalhado anteriormente, o último ponto representa todos os acertos divididos pelo número de elementos testados. O último ponto deve ser usado para medição.

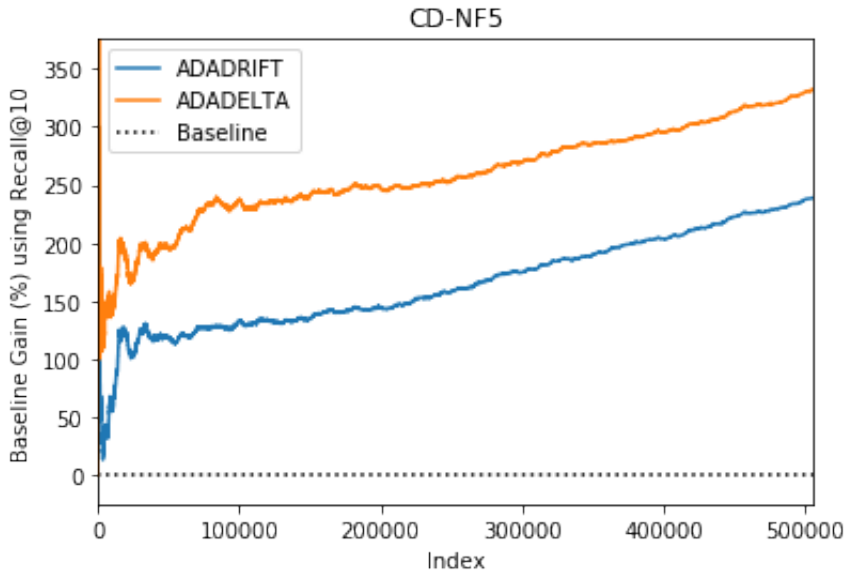


Figura 21 – Os ganhos das variações comparadas ao modelo base (ISGD), medidos em porcentagem de ganho do Recall@10. O conjunto de dados usado foi o CD-NF5, que consiste no tratamento de desvio de conceito realizado no conjunto de dados Netflix *positive-only*.

usadas. Como o ADADRIFT permanece competitivo, é possível obter um resultado melhor no ganho médio do Recall@$N|N \in \{1, 5, 10, 20\}$. Essa degradação é esperada devido à diminuição da incidência de desvios de conceito, que podem gerar adaptação onde não é necessário e ocasionando erros. O gráfico na Figura 21 mostra um crescimento semelhante para ambos os algoritmos, com o ADADELTA tendo uma vantagem constante de cerca de 100% do desempenho do ISGD. Essa diferença permanece ao longo do fluxo, com ADADELTA sendo 1,39 vezes mais eficaz que ADADRIFT na última medição.

O ADADELTA acaba se degradando ainda mais ao diminuir o número médio de *ratings* por perfil, com o ADADRIFT mantendo um resultado semelhante ao BRISMF. A importância do número médio de *ratings* é clara ao observar que a ordenação por essa característica (CD-PL, CD-NF, CD-NF5 e CD-LFM) é igual à ordenação por ganho médio do ADADRIFT em comparação com a linha de base para Recall@$N|N \in \{1, 5, 10\}$. Esse efeito é esperado devido ao problema em que o ADADRIFT foi pensado, que aparecerá com maior incidência quando tiver um longo histórico de interação na plataforma.

O desempenho altamente correlacionado com o tamanho do histórico de iterações é desejado para identificar quais cenários seu uso é rentável ou não. Considerando que a maioria das aplicações reais de sistemas de recomendação possuem um número grande de *ratings* por perfil, o ADADRIFT prosperará com a adaptação aos desvios de conceito.

6.3 Considerações Finais

Este capítulo propôs a técnica de aprendizado adaptativo ADADRIFT. O ADADRIFT aumenta a taxa de aprendizado para se adaptar de maneira mais eficaz à mudança de conceito e diminui durante momentos de estabilidade do conceito, evitando alterações não necessárias no processo de aprendizado e diminuição das taxas de erro. O ADADRIFT foi aplicado a diferentes esquemas de aprendizado em diferentes conjuntos de dados.

Foram realizadas experiências envolvendo 4 conjuntos de dados, 2 algoritmos de base diferentes (ISGD e BRISMF), 2 cenários diferentes (desvio de conceito baseado no usuário e no item) e uma técnica de aprendizagem adaptativa diferente. O ADADRIFT produziu, em média, melhores resultados devido à estabilidade do algoritmo dada pela relação dualística entre médias móveis.

O desempenho do ADADRIFT tem uma alta correlação com o tamanho médio do histórico de iterações, um comportamento esperado, pois os desvios de conceito precisam de tempo para aparecer. O ADADRIFT teve o melhor desempenho no conjunto de dados modificado de usuários do LastFM 1K (CD-LFM), que possivelmente teve uma maior incidência de desvios de conceito. Ele ainda teve um bom desempenho para os outros conjuntos de dados utilizados, apesar de perder um pouco para o ADADELTA na versão com apenas interações positivas do conjunto de dados da Netflix, CD-NF5. Ao usar o conjunto de dados com o menor histórico de interação, não foi possível observar nenhum ganho em uso, resultando no efeito oposto: uma degradação ao usar o algoritmo. Apesar disso, houve uma leve degradação do ADADRIFT em comparação com um resultado pior para o ADADELTA.

É importante notar que o problema de mudança de conceito não ocorre exclusivamente para sistemas de recomendação baseados em fluxo, mas com toda a área de mineração de fluxo de dados. O ADADRIFT pode ser avaliado em outros tipos de aplicativos que extrapolam sistemas de recomendação, ou seja, sempre que técnicas de descida de gradiente são aplicadas em ambientes de streaming, como problemas de classificação e regressão. Essa hipótese deverá ser validada em pesquisas futuras.

Os sistemas de recomendação devem poder se adaptar às preferências e ao comportamento dos usuários ao longo do tempo. O principal problema tratado por esse método é que nem todos os usuários experimentam mudanças no mesmo ritmo. Alguns usuários estão em um momento de maior estabilização em um conceito, enquanto outros estão passando por períodos de maior mudança.

A Parte III irá concluir o trabalho, apresentando os resultados obtidos na Parte II sumarizados e as possibilidades de trabalhos futuros dessa dissertação.

Parte III

Conclusão e Trabalhos Futuros

7 Conclusão e Trabalhos Futuros

Esse trabalho apresenta os métodos de aprendizagem personalizada adaptativa *Background Profiling* (BP) e ADADRIFT, que consistem em técnicas acopláveis a algoritmos de fatoração incremental de matriz, de maneira a não impactar no seu funcionamento interno, mas funcionando como uma camada adicional aumentando a adaptabilidade do modelo. Essa característica viabiliza o uso em conjunto com uma série de algoritmos da literatura, sendo ISGD e BRISMF somente dois exemplos de algoritmos viáveis para utilização em conjunto com elas.

Com esse trabalho, foi possível comprovar o ganho significativo para casos onde existam um histórico de interações do perfil. Outros cenários onde há baixo nível de interação dos usuários não fizeram parte do escopo dessa pesquisa. Isso confirma a hipótese H1 anteriormente posta a prova. A outra hipótese era a observação da estabilidade do perfil para detecção de uma mudança de conceito. Essa hipótese também foi comprovada com o ADADRIFT, que conseguiu se adaptar dinamicamente às mudanças de conceito. Ambas as hipóteses se encontram na seção 1.3.

No Capítulo 5 foi apresentado o BP. O BP teve resultados tímidos, mas mesmo assim apresentou um melhora em relação aos baselines. Apesar disso, vale a pena ressaltar que teve ganho na maioria dos cenários, demonstrando um perfil conservador de risco do seu uso. Já no Capítulo 6 foi apresentado o ADADRIFT. O ADADRIFT obteve um desempenho superior em geral ao ADADELTA, se mantendo com um melhor resultado na maioria dos cenários, com competitividade nos cenários onde perdeu. Diferente do BP, seu comportamento é altamente dinâmico e permite comportar desde períodos de grande mudança quanto momentos de elevada estabilidade.

Apesar disso, todas as técnicas observadas (ADADRIFT, ADADELTA e BP) contribuíram para a adaptação do modelo ao decorrer do tempo em geral. Características como essa são desejadas justamente por aprimorar a experiência do usuário em plataformas de consumo. Usuários que comecem a consumir produtos com um outro conceito necessitam de um sistema que se adapte a isso, principalmente se as mudanças estiverem atreladas a uma rejeição ao conceito anterior, ou seja, uma mudança real de conceito. Usuários que desenvolverem rejeição a um antigo conceito terão uma péssima experiência com uma plataforma que não tenha adaptabilidade de modelos.

Visto a grande quantidade de contribuições dada por esse trabalho, é possível elencar três frentes de trabalhos futuros: (i) lacunas gerais, (ii) lacunas do ADADRIFT e (iii) lacunas do *Background Profiling*.

De maneira geral, não foram explorados impactos causados por hiper-parâmetros das

técnicas de fatoração além da taxa de aprendizagem (λ e k). Além disso, as técnicas podem ser observadas em cenários sem uma grande incidência de mudanças de conceito, situações não exploradas nesse trabalho. Cabe ressaltar também que as técnicas de aprendizagem personalizada adaptativas não são necessariamente voltadas para adaptação à mudança de conceito. Outro problema que tem potencial de ser explorado é o uso para *cold start*.

Uma continuidade natural das pesquisas com o ADADRIFT consiste em adaptá-lo para a mineração de fluxos de dados. Neste trabalho sua aplicação se deu de maneira individualizada devido ao cenário para qual foi desenvolvido, mas é possível explorar seu uso conjunto a outros algortimos, regulando outros parâmetros além da taxa de aprendizagem. Além disso, soluções para definição de tamanho ideal das janelas deslizantes podem acarretar em maior adaptação ao usuário, considerando que cada usuário terá a sua taxa de mudança de conceito. Já em relação ao *Background Profiling*, investigações complementares podem ser feitas com outros tipos de detectores e modelos.

Referências

ABADI, Martín; AGARWAL, Ashish; BARHAM, Paul; BREVDO, Eugene; CHEN, Zhifeng; CITRO, Craig; CORRADO, Greg S.; DAVIS, Andy; DEAN, Jeffrey; DEVIN, Matthieu; GHEMAWAT, Sanjay; GOODFELLOW, Ian; HARP, Andrew; IRVING, Geoffrey; ISARD, Michael; JIA, Yangqing; JOZEFOWICZ, Rafal; KAISER, Lukasz; KUDLUR, Manjunath; LEVENBERG, Josh; MANé, Dan; MONGA, Rajat; MOORE, Sherry; MURRAY, Derek; OLAH, Chris; SCHUSTER, Mike; SHLENS, Jonathon; STEINER, Benoit; SUTSKEVER, Ilya; TALWAR, Kunal; TUCKER, Paul; VANHOUCKE, Vincent; VASUDEVAN, Vijay; VIÉGAS, Fernanda; VINYALS, Oriol; WARDEN, Pete; WATTENBERG, Martin; WICKE, Martin; YU, Yuan; ZHENG, Xiaoqiang. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <http://tensorflow.org/>. Citado na página 55.

BAENA-GARCÍA, Manuel; CAMPO-ÁVILA, José del; FIDALGO, Raúl; BIFET, Albert; GAVALDA, R; MORALES-BUENO, R. Early drift detection method. *Fourth international workshop on knowledge discovery from data streams*, v. 6, n. January, p. 77–86, 2006. Citado na página 37.

BELL, Robert M.; KOREN, Yehuda; VOLINSKY, Chris. *The BellKor solution to the Netflix Prize*. [S.l.], 2007. Citado na página 49.

BENNETT, James; LANNING, Stan. The Netflix Prize. 2007. Citado 3 vezes nas páginas 25, 42 e 49.

BIFET, Albert; GAVALDÀ, Ricard. Learning from Time-Changing Data with Adaptive Windowing a. 2007. Citado na página 35.

BOBADILLA, J.; ORTEGA, F.; HERNANDO, A.; GUTIÉRREZ, A. Recommender systems survey. *Knowledge-Based Systems*, Elsevier BV, v. 46, p. 109–132, jul 2013. Disponível em: <https://doi.org/10.1016%2Fj.knosys.2013.03.012>. Citado na página 25.

CARPENTER, Gail A.; GROSSBERG, Stephen; ROSEN, Dabid B. Fuzzy ART : Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System I +. v. 4, p. 759–771, 1991. Citado na página 37.

CASILLAS, Jorge; WANG, Shuo; YAO, Xin. Concept drift detection in histogram-based straightforward data stream prediction. *IEEE International Conference on Data Mining Workshops, ICDMW*, v. 2018-November, p. 878–885, 2019. ISSN 23759259. Citado na página 35.

CELMA, O. *Music Recommendation and Discovery in the Long Tail*. [S.l.]: Springer, 2010. Citado na página 50.

CHANG, Shiyu; ZHANG, Yang; TANG, Jiliang; YIN, Dawei; CHANG, Yi; HASEGAWA-JOHNSON, Mark A.; HUANG, Thomas S. Streaming Recommender Systems. *Proceedings of the 26th International Conference on World Wide Web - WWW '17*, p. 381–389, 2017.

Disponível em: <http://dl.acm.org/citation.cfm?doid=3038912.3052627>. Citado na página 38.

CHOLLET, François et al. *Keras*. [S.l.]: GitHub, 2015. <https://github.com/fchollet/keras>. Citado na página 55.

CREMONESI, Paolo; MILANO, Politecnico; KOREN, Yehuda; TURRIN, Roberto. Performance of Recommender Algorithms on Top-N Recommendation Tasks Categories and Subject Descriptors. n. September, 2010. Citado na página 43.

DAUPHIN, Yann N.; VRIES, Harm de; CHUNG, Junyoung; BENGIO, Yoshua. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *CoRR*, abs/1502.04390, 2015. Disponível em: <http://dblp.uni-trier.de/db/journals/corr/corr1502.html#DauphinVCB15>. Citado na página 55.

DIELEMAN, Sander; SCHLüTER, Jan; RAFFEL, Colin; OLSON, Eben; SøNDERBY, Søren Kaae; NOURI, Daniel; MATURANA, Daniel; THOMA, Martin; BATTENBERG, Eric; KELLY, Jack; FAUW, Jeffrey De; HEILMAN, Michael; ALMEIDA, Diogo Moitinho de; MCFEE, Brian; WEIDEMAN, Hendrik; TAKáCS, Gábor; RIVAZ, Peter de; CRALL, Jon; SANDERS, Gregory; RASUL, Kashif; LIU, Cong; FRENCH, Geoffrey; DEGRAVE, Jonas. *Lasagne: First release*. 2015. Disponível em: <http://dx.doi.org/10.5281/zenodo.27878>. Citado na página 55.

DOMINGOS, Pedro; HULTEN, Geoff. Mining High-Speed Data Streams. 2000. Citado na página 37.

DUCHI, John; HAZAN, Elad; SINGER, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, v. 12, n. Jul, p. 2121–2159, 2011. Citado na página 55.

GAMA, Joao; BIFET, Albert; PECHENIZKIY, Mykola; BOUCHACHIA, Abdelhamid. 1 A Survey on Concept Drift Adaptation ~. v. 1, n. 1, 2013. Citado 7 vezes nas páginas 26, 31, 32, 33, 34, 35 e 38.

GAMA, João; MEDAS, Pedro; CASTILLO, Gladys; RODRIGUES, Pedro. Learning with drift detection. In: *Advances in Artificial Intelligence – SBIA 2004*. Springer Berlin Heidelberg, 2004. p. 286–295. Disponível em: <https://doi.org/10.1007/978-3-540-28645-5_29>. Citado na página 36.

GAMA, João; SEBASTIÃO, Raquel; RODRIGUES, Pedro Pereira. On evaluating stream learning algorithms. p. 317–346, 2013. Citado na página 44.

GOMES, Heitor Murilo; BIFET, Albert; READ, Jesse; BARDDAL, Jean Paul. Adaptive Random Forests for Evolving Data Stream Classification. n. June, 2017. Citado na página 65.

GONÇALVES, Paulo M.; De Carvalho Santos, Silas G.T.; BARROS, Roberto S.M.; VIEIRA, Davi C.L. A comparative study on concept drift detectors. *Expert Systems with Applications*, v. 41, n. 18, p. 8144–8156, 2014. ISSN 09574174. Citado 3 vezes nas páginas 34, 36 e 70.

HULTEN, Geoff; SPENCER, Laurie; DOMINGOS, Pedro. Mining time-changing data streams. ACM Press, 2001. Disponível em: <https://doi.org/10.1145/502512.502529>. Citado na página 38.

IKONOMOVSKA, Elena; GAMA, João; SEBASTIÃO, Raquel; GJORGJEVIK, Dejan. Regression trees from data streams with drift detection. In: *Discovery Science*. Springer Berlin Heidelberg, 2009. p. 121–135. Disponível em: <https://doi.org/10.1007/978-3-642-04747-3_12>. Citado na página 65.

JORGE, Alípio M.; VINAGRE JOÃO, Vinagre1; MARCOS, Domingues; JOÃO, Gama; CARLOS, Soares; MATUSZYK, Pawel; SPILIOPOULOU, Myra. Scalable Online Top-N Recommender Systems. v. 278, p. 3–20, 2017. Disponível em: <http://link.springer.com/10.1007/978-3-319-53676-7>. Citado 4 vezes nas páginas 11, 25, 40 e 75.

KIEFER, J.; WOLFOWITZ, J. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, Institute of Mathematical Statistics, v. 23, n. 3, p. 462–466, set. 1952. Disponível em: <https://doi.org/10.1214/aoms/1177729392>. Citado 2 vezes nas páginas 41 e 52.

KINGMA, Diederik P.; BA, Jimmy. *Adam: A Method for Stochastic Optimization*. 2014. Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. Disponível em: <http://arxiv.org/abs/1412.6980>. Citado na página 55.

MATUSZYK, Pawel; SPILIOPOULOU, Myra. Hoeffding-CF: Neighbourhood-Based Recommendations on Reliably Similar Users. v. 7, p. 146–157, 2014. Citado 4 vezes nas páginas 11, 52, 56 e 57.

MATUSZYK, Pawel; SPILIOPOULOU, Myra. Stream-based semi-supervised learning for recommender systems. *Machine Learning*, Springer US, v. 106, n. 6, p. 771–798, 2017. ISSN 15730565. Citado 2 vezes nas páginas 43 e 49.

MATUSZYK, Pawel; VINAGRE, João; SPILIOPOULOU, Myra; JORGE, Alípio Mário; GAMA, João. Forgetting methods for incremental matrix factorization in recommender systems. p. 947–953, 2015. Citado na página 56.

MATUSZYK, Pawel; VINAGRE, João; SPILIOPOULOU, Myra; JORGE, Alípio Mário; GAMA, João. Forgetting techniques for stream-based matrix factorization in recommender systems. *Knowledge and Information Systems*, Springer London, v. 55, n. 2, p. 275–304, 2018. ISSN 02193116. Citado 2 vezes nas páginas 55 e 56.

MIRANDA, Catarina; JORGE, Alípio M. Incremental collaborative filtering for binary ratings. *Proceedings - 2008 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2008*, p. 389–392, 2008. Citado na página 39.

MONTIEL, Jacob; READ, Jesse; BIFET, Albert; ABDESSALEM, Talel. Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, v. 19, n. 72, p. 1–5, 2018. Disponível em: <http://jmlr.org/papers/v19/18-251.html>. Citado na página 36.

NELDER, John A.; MEAD, Roger. A simplex method for function minimization. *Computer Journal*, v. 7, p. 308–313, 1965. Citado na página 58.

NGUYEN, Tien T.; HARPER, F. Maxwell; TERVEEN, Loren; KONSTAN, Joseph A. User personality and user satisfaction with recommender systems. *Information Systems*

Frontiers, Springer Science and Business Media LLC, v. 20, n. 6, p. 1173–1189, set. 2017. Disponível em: <https://doi.org/10.1007/s10796-017-9782-y>. Citado na página 49.

OZAKI, Yoshihiko; YANO, Masaki; ONISHI, Masaki. Effective hyperparameter optimization using nelder-mead method in deep learning. *IPSJ Transactions on Computer Vision and Applications*, Springer Science and Business Media LLC, v. 9, n. 1, nov. 2017. Disponível em: <https://doi.org/10.1186/s41074-017-0030-7>. Citado na página 59.

PAGE, E. S. CONTINUOUS INSPECTION SCHEMES. *Biometrika*, Oxford University Press (OUP), v. 41, n. 1-2, p. 100–115, 1954. Disponível em: <https://doi.org/10.1093/biomet/41.1-2.100>. Citado na página 36.

QIAN, Ning. On the momentum term in gradient descent learning algorithms. *Neural Networks*, v. 12, n. 1, p. 145–151, 1999. Disponível em: <http://dblp.uni-trier.de/db/journals/nn/nn12.html#Qian99>. Citado na página 55.

QUADRANA, Massimo; CREMONESI, Paolo; JANNACH, Dietmar. Sequence-aware recommender systems. *ACM Computing Surveys*, Association for Computing Machinery (ACM), v. 51, n. 4, p. 1–36, sep 2018. Disponível em: <https://doi.org/10.1145%2F3190616>. Citado na página 26.

RUDER, Sebastian. *An overview of gradient descent optimization algorithms.* 2016. Disponível em: <http://arxiv.org/abs/1609.04747>. Citado na página 55.

SARWAR, Badrul M.; KARYPIS, George; KONSTAN, Joseph A.; RIEDL, John T. Application of Dimensionality Reduction in Recommender System – A Case Study. *KDD*, v. 35, n. 4, p. 429–443, 2001. ISSN 00335177. Citado na página 40.

TAKÁCS, Gábor; PILÁSZY, István; NÉMETH, Bottyán; TIKK, Domonkos. Scalable Collaborative Filtering Approaches for Large Recommender Systems. *Journal of Machine Learning Research*, v. 10, p. 623–656, 2009. Citado na página 41.

VELOSO, Bruno; GAMA, João; MALHEIRO, Benedita; VINAGRE, João. Self hyper-parameter tuning for stream recommendation algorithms. In: MONREALE, Anna; ALZATE, Carlos; KAMP, Michael; KRISHNAMURTHY, Yamuna; PAURAT, Daniel; SAYED-MOUCHAWEH, Moamar; BIFET, Albert; GAMA, João; RIBEIRO, Rita P. (Ed.). *ECML PKDD 2018 Workshops*. Cham: Springer International Publishing, 2019. p. 91–102. ISBN 978-3-030-14880-5. Citado 2 vezes nas páginas 55 e 58.

VELOSO, Bruno; LEAL, Fátima; VÉLEZ, Horacio González; MALHEIRO, Benedita; BURGUILLO, Juan Carlos. Highlights 1 . Parallel data stream algorithm 2 . Scalable Recommendation Engine 3 . Profiling and Recommendation algorithms using scalable data analytics recommendations. *J. Parallel Distrib. Comput.*, Elsevier Inc., 2018. ISSN 0743-7315. Disponível em: <https://doi.org/10.1016/j.jpdc.2018.06.013>. Citado 2 vezes nas páginas 25 e 42.

VELOSO, Bruno; MALHEIRO, Benedita; FOSS, Jeremy. Stream recommendation using individual hyper-parameters. In: . [S.l.: s.n.], 2019. Citado 3 vezes nas páginas 55, 58 e 59.

VINAGRE, João; JORGE, Alípio Mário; GAMA, João. Fast Incremental Matrix Factorization for Recommendation with Positive-Only Feedback. p. 459–470, 2014. Citado 2 vezes nas páginas 39 e 42.

VINAGRE, João; JORGE, Alípio Mário; GAMA, João. Collaborative filtering with recency-based negative feedback. p. 963–965, 2015. Citado 2 vezes nas páginas 55 e 60.

VINAGRE, João; JORGE, Alípio Mário; GAMA, João. Evaluation of recommender systems in streaming environments. 2015. Citado 2 vezes nas páginas 42 e 44.

ZEILER, Matthew D. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. Disponível em: <http://dblp.uni-trier.de/db/journals/corr/corr1212.html#abs-1212-5701>. Citado 4 vezes nas páginas 55, 56, 70 e 79.