RECOMENDAÇÃO *ONLINE* DE MÚSICAS USANDO *FEEDBACK* IMPLÍCITO

BRUNO LAPORAIS PEREIRA

RECOMENDAÇÃO *ONLINE* DE MÚSICAS USANDO *FEEDBACK* IMPLÍCITO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais - Departamento de Ciência da Computação como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Rodrygo Luis Teodoro Santos
Coorientador: Nivio Ziviani

Belo Horizonte
Março de 2017

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Pereira, Bruno Laporais

P436r Recomendação *Online* de Músicas usando *Feedback* Implícito / Bruno Laporais Pereira. — Belo Horizonte, 2017
xxii, 81 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais - Departamento de Ciência da Computação

Orientador: Rodrygo Luis Teodoro Santos
Coorientador: Nivio Ziviani

1. Sistemas de Recomendação. 2. Recomendação *Online* de Músicas. 3. Aprendizado *Online*. 4. *Feedback Implícito*. *I. Orientador. II. Coorientador. III. Título.*

CDU 519.6*73(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS.
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Recomendação online de músicas usando feedback implícito

BRUNO LAPORAIS PEREIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. RODRYGO LUIS TEODORO SANTOS - Orientador
Departamento de Ciência da Computação - UFMG

PROF. NIVIO ZIVIANI - Coorientador
Departamento de Ciência da Computação - UFMG

PROF. ADRIANO ALONSO VELOSO
Departamento de Ciência da Computação - UFMG

PROF. LEANDRO BALBY MARINHO
Departamento de Sistemas e Computação - UFCG

Belo Horizonte, 31 de março de 2017.

À minha namorada Letícia por me oferecer a cada dia o amor mais puro. Aos meus pais José Donizete e Regina Célia pelo carinho e apoio incondicionais. À minha irmã Daniela por ser meu exemplo de perseverança. Sem vocês, mesmo que fosse possível, nenhuma conquista valeria a pena.

Agradecimentos

Nesta singela homenagem, gostaria de agradecer a todos que não mediram esforços em me apoiar durante esses dois anos de trabalho intenso. O apoio de cada um tem uma marca definitiva sobre o meu futuro.

Primeiramente agradeço a Deus pelo dom da vida. Agradeço à minha namorada Letícia pelo amor incondicional e fonte de motivação para buscar cada sonho, que não são mais só meus. Aos meus Pais José Donizete e Regina Célia por estarem ao meu lado todos os dias, e por muitas vezes se superarem em prol do meu aprendizado e crescimento. À minha irmã Daniela pelo incentivo, perseverança e disposição a todo momento. Aos meus familiares agradeço pelo carinho e cuidado dedicado.

Ao professor Wladmir obrigado pelo empenho sobre minha formação ao me indicar e ajudar no processo de seleção para o mestrado. Agradeço também à todos os membros da Universidade Federal de Minas Gerais, em especial aos professores Nivio e Rodrygo pela oportunidade de trabalharmos juntos, pelos ensinamentos, lições e contribuições. Sem dúvidas essa parceria foi essencial e desejo que se estenda para trabalhos futuros.

Por fim, agradeço aos meus amigos que sempre torceram por mim. Aos parceiros de curso Adriano, Armando, Giuseppe, Raphael, Sérgio e Vaux. Em particular aos amigos do laboratório LATIN, Alberto, Felipe, Jordan, Rafael, Raul e Sabir. A todos obrigado pelas discussões construtivas, conhecimentos compartilhados e pelo companheirismo.

"A persistência é o caminho do êxito."
(Charles Chaplin)

Resumo

O sucesso dos serviços de *streaming* tem gerado grandes desafios para a recomendação de músicas. Em um cenário de *streaming*, as músicas são consumidas sequencialmente dentro de uma sessão, a qual deve atender não somente preferências históricas, mas também preferências eventuais ocasionadas por mudanças repentinas de contexto do usuário. Nesta dissertação, propomos uma nova abordagem de aprendizado *online* para recomendação de músicas, com vistas a aprender continuamente a partir de interações com o usuário. Em contraste a abordagens de aprendizado *online* existentes para recomendação de música, utilizamos o *feedback* implícito como único sinal de preferência do usuário em cada ponto no tempo. Em um universo de milhões de músicas, representamos cada música em um espaço de baixa dimensionalidade utilizando um conjunto de atributos contínuos. Uma avaliação rigorosa utilizando sessões coletadas do Last.fm demonstra a eficácia da nossa abordagem em aprender mais rápido e melhor comparada a abordagens do estado-da-arte para aprendizado *online*.

Palavras-chave: Sistemas de Recomendação, Recomendação *Online* de Músicas, Aprendizado *Online*, *Feedback* Implícito.

Abstract

The prominent success of music streaming services has brought increasingly complex challenges for music recommendation. In particular, in a streaming setting, songs are consumed sequentially within a listening session, which should cater not only for the user's historical preferences, but also for eventual preference drifts, triggered by a sudden change in the user's context. In this dissertation, we propose a novel online learning-to-rank approach for music recommendation, aimed to continuously learn from the user's listening feedback. In contrast to existing online learning approaches for music recommendation, we leverage implicit feedback as the only signal of the user's preference at each point in time. In a space of millions of songs, we represent each song in a lower dimensional space of continuous features. Our thorough evaluation using listening sessions from Last.fm demonstrates the effectiveness of our approach at learning faster and better compared to state-of-the-art online learning approaches.

Keywords: Recommender Sytems, Online Music Recommendation, Online Learning, Implicit Feedback.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Com o advento da internet, as pessoas podem acessar uma enorme quantidade de informação, produtos e serviços. Esse novo horizonte de possibilidades vem transformando o comportamento dos usuários, gerando a necessidade de recuperar itens que atendam ou superem suas expectativas. Ao longo dos anos, várias abordagens foram introduzidas com a proposta de auxiliar os usuários a encontrarem itens relevantes, normalmente a partir de consultas formuladas em formato de texto. O uso de consultas é muito comum por exemplo em sistemas de busca, como Google,[1] Bing,[2] e etc. No entanto, em outros cenários os sistemas precisam predizer as preferências e as necessidades dos usuários de acordo com informações adicionais como localização, horário e o próprio perfil do usuário. Nesses casos, são utilizadas sugestões, seja para surpreender o usuário ou para facilitar a utilização de um determinado serviço. A esse tipo de necessidade se aplicam os sistemas de recomendação [Ricci et al., 2015].

Os sistemas de recomendação têm apresentado grande evolução nos últimos anos, produzindo resultados sólidos, tanto para o mercado quanto em pesquisas científicas. Segundo Ricci et al. [2015], esses sistemas têm como objetivo a recuperação e sugestão de itens aos usuários, possivelmente desconhecidos, mas que atendam ou superem as expectativas de consumo do usuário. Para os sistemas de recomendação, os itens representam o que é sugerido ao usuário, ou seja, o que será recomendado. Os sistemas atuais trabalham sobre milhares de itens que podem, ou não, fazer parte de um mesmo grupo, como as recomendações de filmes e séries, livros, músicas, etc. Por outro lado, em outros contextos, como no caso de sites de compras pela internet, possuem itens variados em seu catálogo. Além das particularidades de cada cenário, a recomendação de cada um cada um desses itens pode ser modelada de maneira distintas, considerando

[1] http://www.google.com
[2] http://www.bing.com

específicas de cada um.

No caso da recomendação de músicas os desafios são ainda maiores, devido à mudanças drásticas no comportamento dos usuários que consomem música [Celma, 2010]. Além disso, nos sistemas de *streaming* de música como Spotify[3] ou Google Play Music,[4] o consumo de mídia está relacionado ao contexto em que o usuário está inserido. Dessa forma, um mesmo usuário pode escutar diferentes tipos de música enquanto está trabalhando ou dirigindo seu automóvel. Além disso, cada usuário possui preferências únicas, e mesmo em contextos semelhantes, a recomendação deve ser personalizada.

É importante destacar que as preferências dos usuários podem mudar, ou seja, itens relevantes no passado podem se tornar irrelevantes, assim como itens potencialmente similares podem não condizer com a necessidade real do usuário. Um dos desafios está na adaptação da recomendação às preferências dos usuários que podem estar ligadas ao histórico, ao contexto, ao sentimento, entre outros fatores que geram influência sobre os gostos dos usuários.

Um dos fatores mais importantes em sistemas de recomendação se refere a utilização do *feedback* como forma de inferência das preferências dos usuários. O *feedback* ocorre na forma de interações entre o usuário e o sistema. Em recomendação, podemos considerar dois tipos de *feedback*, o explícito e o implícito. No contexto de recomendação de músicas, o *feedback* explícito pode ser observado quando um usuário expressa diretamente sua preferência sobre algumas músicas. Por outro lado, o *feedback* implícito ocorre de forma mais abundante a partir das interações do usuário com o sistema ao utilizar comandos do tipo *play* ou *skip*. Ainda que ruidosos, esses sinais carregam informações importantes sobre as preferências do usuário [Jawaheer et al., 2010].

Existem diversas abordagens para recomendação *online* de músicas utilizando aprendizado de máquina e *feedback* explícito [Wang et al., 2014; Hariri et al., 2015; Liebman et al., 2015]. Porém, é difícil adquirir *feedback* explícito suficiente a partir de uma porção de usuários. Essa dificuldade pode ser explicada devido o esforço necessário pelo usuário para prover o *feedback* explícito, entre outros fatores como o desânimo [Jawaheer et al., 2010].

King & Imbrasaitė [2015] apresenta uma abordagem para geração automática de *playlist* utilizando aprendizado por reforço a partir do *feedback* implícito. Entretanto, o aprendizado por reforço é aplicado para escolha de grupos de músicas e não da próxima música a ser recomendada, conforme tratado nesta dissertação. Para escolha da próxima música, King & Imbrasaitė [2015] propõe uma heurística sobre as músicas do grupo.

[3]https://www.spotify.com
[4]https://play.google.com/music/

O melhor algoritmo que encontramos na literatura com o objetivo de servir como *baseline* para comparação com o nosso algoritmo é o LINUCB [Li et al., 2010]. O LINUCB é um algoritmo para recomendação de notícias representadas por um conjunto de características que formulam o contexto. O algoritmo é capaz de selecionar uma notícia para recomendação a cada interação com o usuário, dentre uma lista de opções. A partir da recomendação, o LINUCB absorve o *feedback* implícito (no caso de notícias, clique ou não clique) para aprimorar o conhecimento e melhorar a recomendação nas próximas interações.

Nesta dissertação apresentamos um novo algoritmo de aprendizado *online* utilizando *feedback* implícito para recomendação de músicas, chamado Multi-DBGD (*Multi Dueling Bandits Gradient Descent*, ou em português, Descida de Gradiente de Multi Duelos entre Bandidos). Nossa proposta foi inspirada em dois trabalhos: (*i*) Yue & Joachims [2009] propõem DBGD (*Dueling Bandits Gradient Descent*), um algoritmo de aprendizado *online* para ordenar respostas em um sistema de recuperação de informação, o qual escolhe o melhor modelo a partir de um duelo entre dois modelos; (*ii*) Schuth et al. [2016] propõem o algoritmo MGD (em Multileave Gradient Descent), o qual explora os mesmos conceitos do DBGD mas, ao invés de realizar comparações entre dois modelos, são utilizados múltiplos modelos para obter a ordenação a cada interação.

Voltando ao cenário de recomendação *online* de músicas, é possível observar o *feedback* apenas sobre um item a cada interação com o usuário, fato que torna inviável a adaptação do DBGD e do MGD nesse cenário. Sendo assim, propomos os seguintes passos para reutilizar o *feedback* sobre um mesmo item na comparação de múltiplos modelos: (*i*) o *feedback* sobre um mesmo item (selecionado pelo melhor modelo corrente) é reutilizado para simular o *feedback* sobre novos modelos. Essa simulação permite realizar múltiplas comparações entre modelos, que definimos como número de duelos entre o melhor modelo corrente e um modelo candidato; (*ii*) O *feedback* negativo é utilizado com o objetivo de penalizar o melhor modelo corrente, permitindo então explorar novas evidências. Além disso, nas interações onde somente o *feedback* negativo foi observado, os modelos que promovam o item irrelevante também são penalizados na atualização do melhor modelo corrente.

O algoritmo Multi-DBGD visa promover recomendações personalizadas sem a interferência explícita do usuário. O intuito é aumentar a satisfação do usuário ao interagir com sistemas de *streaming* de música, aproximando os itens recomendados da necessidade real de consumo. Para isso, propomos a utilização de sinais observados a partir do comportamento do usuários conforme os comandos de *play* e *skip*. Diferente das abordagens que aprendem a selecionar os itens, devido ao grande número de músicas

no cenário de *streaming*, o algoritmo proposto aprende a importância das características que representam cada música da coleção, sendo esse um espaço de opções menor se comparado ao número de músicas distintas na coleção. Dessa forma, além de utilizar somente o *feedback* implícito a cada interação, é importante destacar que o Multi-DBGD pode ser utilizado sobre características extraídas de diversas fontes, assim como em informações colaborativas, sociais, de conteúdo das músicas e das atributos do áudio.

1.1 Proposta da Dissertação

O presente trabalho tem como finalidade melhorar a recomendação *online* por meio da utilização do *feedback* implícito. Os principais objetivos do presente trabalho são: (*i*) propor Multi-DBGD, um novo algoritmo para recomendação *online* de músicas, capaz de comparar múltiplos modelos lineares para balancear as características que representam cada música, de forma personalizada, interativa e utilizando somente o feedback implícito; (*ii*) proporcionar a reutilização do *feedback* implícito recebido sobre um mesmo item a cada interação; (*iii*) comparar os resultados do Multi-DBGD com o melhor algoritmo encontrado na literatura a partir de uma avaliação utilizando dados publicamente disponíveis.

Na recomendação *online* de músicas, a cada interação com o usuário somente um item recebe *feedback*. Além da pouca informação, o *feedback* observado normalmente é realizado de forma implícita a partir dos comandos de *play* e *skip*, sendo pouco precisos quanto a relevância dos itens [Jawaheer et al., 2010]. Sendo assim, diferentemente do *baseline* LINUCB, propomos com o Multi-DBGD a exploração repetida do *feedback* implícito sobre um mesmo item a cada interação, a fim de promover múltiplas comparações entre modelos para recomendação. Além disso, diferente do cenário de aplicação dos algoritmos DBGD [Yue & Joachims, 2009] e o MGD [Schuth et al., 2016], propomos a exploração do *feedback* implícito negativo, com o objetivo de penalizar o melhor modelo corrente e permitir a exploração de novos modelos candidatos.

Propomos também algumas premissas para o funcionamento do Multi-DBGD, a saber: (*i*) utilização de um modelo inicial que é interativamente aprimorado a partir da reutilização do *feedback* implícito nas comparações de modelos candidatos a fim de atualizar o modelo corrente; (*ii*) propomos calibrar os fatores de exploração e aproveitamento (do inglês *exploration and exploitation*) para otimizar a busca de novos modelos sem desprezar o conhecimento corrente, assim como realizado por Yue & Joachims [2009].

As perguntas de pesquisa a serem respondidas nesta dissertação são:

- Q1: O algoritmo proposto produz melhores resultados que o melhor algoritmo encontrado na literatura?
- Q2: O algoritmo proposto é capaz de aprender mais rápido que o melhor algoritmo encontrado na literatura?
- Q3: Como o modelo de inicialização do algoritmo proposto afeta sua eficácia?
- Q4: Qual impacto do número de duelos na eficácia do algoritmo proposto?

1.2 Contribuições

A contribuições desta dissertação são:

- Proposta de um novo algoritmo chamado Multi-DBGD capaz de aprender sobre um espaço de características consideradas na recomendação *online* de músicas, usando o *feedback* implícito como único sinal sobre as preferências dos usuários a cada ponto no tempo.
- Resultados experimentais mostram que o algoritmo Multi-DBGD é capaz de superar o melhor algoritmo encontrado na literatura, utilizando dados reais e publicamente disponíveis.

1.3 Organização do Texto

Os demais capítulos desta dissertação estão organizadas seguindo a formatação abaixo:

- O Capítulo 2 introduz os conceitos básicos de sistemas de recomendação e aprendizado de máquina e descreve os trabalhos relacionados a esta dissertação.
- O Capítulo 3 apresenta em detalhes o algoritmo Multi-DBGD, assim como o embasamento teórico que motivou sua proposta.
- O Capítulo 4 descreve a configuração dos experimentos realizados, apresenta as características propostas para representar cada música da base de dados utilizada nos experimentos, apresenta os *baselines* e formalização das configurações sobre o modelo de treino, validação e teste. O capítulo conclui com a formulação das métricas de avaliação utilizadas para comparar e avaliar os algoritmos considerados na avaliação experimental.

- O Capítulo 5 apresenta os resultados experimentais, comparando a eficácia do algoritmo Multi-DBGD em relação aos *baselines* a partir de vários testes, a fim de responder as perguntas de pesquisa propostas nesta dissertação.
- O Capítulo 6 apresenta as conclusões, resume as principais contribuições e aponta direções futuras relacionadas a este trabalho.

Capítulo 2

Trabalhos Relacionados

Neste capítulo são formalizados os principais conceitos dos sistemas de recomendação, além de descrever as abordagens para aprendizado de máquina com enfoque principal em trabalhos relacionados ao cenário de recomendação *online* de músicas, conforme tratado nesta dissertação.

2.1 Sistemas de Recomendação

Os sistemas de recomendação vêm mudando a forma como os usuários interagem e consomem itens de um sistema. A área de pesquisa surgiu em meados da década de 90 e é considerada uma sub área da recuperação da informação. Os sistemas de recomendação estudam técnicas para promover sugestões de itens para consumo dos usuários. Dessa forma, o objetivo está na recuperação e sugestão de itens aos usuários, talvez desconhecidos, mas que atendam ou superem as expectativas de consumo do usuário [Ricci et al., 2015].

Para os sistemas de recomendação, os itens representam o que deve ser sugerido ao usuário, ou seja, o que será recomendado. Os sistemas atuais trabalham sobre quantidades massivas de itens. Em alguns cenários, esses itens fazem parte de um mesmo grupo, como as recomendações de filmes e séries, livros, músicas, etc. Ou, em muitos sistemas, como no caso de sites de compras pela internet, possuem itens variados em seu catálogo. Além das particularidades de cada cenário, para cada um desses itens é possível perceber perfis e características específicas.

Conforme Ricci et al. [2015], um dos fatores mais importantes em sistemas de recomendação se refere a utilização do *feedback* como forma de inferência das preferências dos usuários sobre os itens. O *feedback* pode ser observado a partir das interações entre

o usuário e o sistema. Em sistemas de recomendação, podemos observar dois tipos de *feedback*, o explícito e o implícito, descritos abaixo:

- O *feedback* explícito é observado a partir da declaração do usuário sobre suas preferências e representa um sinal de maior relevância, apesar de ser pouco frequente. Por exemplo: a nota (número de estrelas) atribuído a um filme no Netflix;[1] o botão de "curtir" do Facebook.[2]

- O *feedback* implícito é um sinal muito mais abundante e que pode ser observado a partir das interações do usuário com os itens de um determinado sistema, sem expressar diretamente sua preferência. Por exemplo: clicar em um produto de um site de compras pela internet; escutar totalmente ou executar um comando de *skip* para a próxima música no Spotify.

Ao longo dos anos, diversos métodos foram propostos com a motivação de tentar modelar preferências dos usuários e características dos itens a partir do *feedback* [Ricci et al., 2015]. Uma das principais famílias de métodos é a filtragem colaborativa, onde informações de colaboração entre os usuários é utilizada para capturar itens e usuários similares. Primeiramente, é criada uma matriz de usuários por itens, sendo cada célula a representação de relevância, seja a avaliação do usuário sobre um item, seja uma contagem de interações no caso do *feedback* implícito). Seguindo esse raciocínio, foram desenvolvidos diversos trabalhos, assim como Herlocker et al. [1999] que exploram a similaridade entre os usuários com base nos itens consumidos e/ou avaliados, e Sarwar et al. [2001] que propõem a utilização de um modelo baseado na similaridade entre os itens.

Ainda sobre filtragem colaborativa, os melhores resultados foram extraídos a partir da fatoração de matrizes. Nesse contexto, um dos principais trabalhos foi o de Koren et al. [2009], desenvolvido pelos ganhadores da competição do Netflix em 2009. Assim como na teoria de álgebra, a fatoração de matrizes permite decompor uma matriz completa no produto de outras menores do que a original. No caso da recomendação, a ideia é reconstruir a matriz de usuários por item. Alguns desses métodos de fatoração propõem a redução das matrizes intermediárias utilizando fatores latentes, com vistas a representar usuários e itens em um mesmo espaço de fatores. No entanto, em sistemas de recomendação a matriz original não está completa, devido aos usuários interagirem com poucos itens da base, ou seja, o objetivo é predizer as avaliações não observadas

[1] www.netflix.com
[2] www.facebook.com

na matriz. No caso do trabalho de Koren et al. [2009], os autores propõem a utilização de vários métodos para fatoração de matrizes e milhares de fatores latentes.

Entre trabalhos recentes baseados em fatores latentes, destacamos o trabalho de Volkovs & Wei Yu [2015] que, assim como a presente dissertação, propõem a utilização do *feedback* implícito. Em relação as anteriores, da abordagem utiliza da técnica conhecida como Decomposição em Valores Singulares (ou em inglês *Singular Value Decomposition* (SVD)) para decomposição da matrizes original contendo apenas o *feedback* implícito. Dessa forma, as preferências sobre os itens são binárias e representam apenas as interações entre usuários e itens. A abordagem proposta apresentou bons resultados em testes sobre diferentes bases de dados (inclusive de músicas), superando modelos mais complexos e que utilizam *feedback* explícito para recomendação.

No entanto, as abordagens baseadas na filtragem colaborativa são falhas para predição sobre novos usuários ou itens, ou seja, que não existentes na base a priori. Nesse caso, não existem informações colaborativas o que inviabiliza utilizar o método para recomendação. Nesse caso, os modelos baseados em conteúdo são métodos capazes de modelar itens e usuários utilizando as informações de conteúdo dos itens e dos usuários, assim como descrições, categorias, nomes, etc. Portanto, mesmo nos casos de usuários ou itens novos, as informações para recomendação estão disponíveis, ainda que com menos detalhes em relação as informações necessárias para inserção de novos usuários ou itens na base. Porém, a similaridade entre os conteúdos dificulta a diversidade sobre as sugestões, pois essas informações são estáticas [Lops et al., 2011].

Acima foram citados alguns dos métodos clássicos para recomendação. No entanto, como nenhum modelo é capaz de solucionar todos os problemas, uma alternativa é a utilização de hibridização, ou seja, abordagens capazes de combinar as métodos diferentes para recomendação. Nesse cenário, o objetivo é repesar as predições de cada modelo utilizando, por exemplo, aprendizado de máquina [Burke, 2002].

O horizonte de abordagens para recomendação são diversos, tendo em vista desafios cada vez mais complexos e que exploram particularidades de cada cenário. A seguinte seção descreve em detalhes a recomendação de músicas, tratado nesta dissertação.

2.1.1 Recomendação de Músicas

Diversos fatores caracterizam o cenário de recomendação de músicas, como a criação de sessões de reprodução, a forma sequencial como as mídias são consumidas, etc. Além disso, a ação de escutar músicas normalmente compõe uma tarefa de fundo, ou seja, apenas acompanha uma tarefa principal do usuário, onde o objetivo dificilmente será

interagir com apenas um item, mas sim sobre um conjunto de itens em sequência. Ainda como tarefa de fundo, existem frequentes interações, ou *feedback*, que podem ser capturadas de diversas formas, sejam elas explícitas ou implícitas [Celma, 2010].

No contexto de recomendação de músicas, o *feedback* explícito pode ser observado quando um usuário expressa diretamente sua preferência sobre algumas músicas, por exemplo por meio de uma avaliação na forma positiva ou negativa. Por outro lado, de forma mais abundante, o *feedback* implícito está presente nas interações dos usuários com o sistema, por exemplo, ao utilizar os comandos de *play* ou *skip* enquanto escuta musicas em um serviço de *streaming* [Jawaheer et al., 2010]. Nesse sentido, a execução de um *skip* pode significar, por exemplo, uma avaliação negativa e o fato escutar uma música completa (*play*) pode representar a satisfação do usuário em relação à música escutada.

Apesar de ruidoso, o uso do *feedback* implícito é comum em diversas áreas na tentativa de inferir as preferências dos usuários [Kelly & Teevan, 2003]. Por exemplo em recomendação música, Baltrunas & Amatriain [2009] propõem utilizar o *feedback* implícito para modelar as preferências dos usuários sobre as horas do dia.

Mesmo com grandes esforços de pesquisadores sobre a área de recomendação de músicas, ainda existem diversos problemas em aberto, sendo destacado principalmente métodos para melhorar o reconhecimento das necessidades de cada usuário. Ou seja, modelos mais personalizados possíveis, a fim de explorar também o contexto em que o usuário está inserido e promover adequação das músicas recomendadas, às tarefas que estão sendo desenvolvidas [Song et al., 2012].

Conforme as características descritas sobre o cenário de músicas em geral, recomendar apenas um item não é a principal motivação e de fato não representa o problema de recomendação, devido ao consumo interativo dos itens. Sendo assim, existem diversos trabalhos que realizam recomendação explorando a ideia de sequências musicais ou *playlists* do usuário buscando-se a construção de sessões consistentes e automáticas, a saber na seguinte seção.

2.1.2 Recomendação de *Playlists*

A recomendação de *playlists* é um problema muito comum na área de recomendação de músicas. Nesse caso, a tarefa se resume a recomendar não mais um item específico mas um conjunto de músicas sequenciais. Esse tipo de recomendação pode ser encontrado de duas formas: (*i*) a recomendação de *playlists* previamente geradas por especialistas humanos, como o caso da empresa Superplayer.fm[3] e mesmo em algumas casos de uso

[3]https://www.superplayer.fm/

em outros grandes sistemas de *streaming* de música como Spotify, Google Play Music, etc; (*ii*) geração automática de *playlists*.

A recomendação de playlist geradas manualmente exige grande esforço de especialistas humanos a fim de atender o máximo de usuários e gostos possíveis. Com o objetivo de evitar esse esforço, o foco desta dissertação está na geração automática de *playlists*. A tarefa consiste em, dado um conjunto de músicas e características que às representem, produzir uma sequência finita de músicas que atendam as preferências dos usuários da melhor forma possível, com base no conhecimento observado a priori [Bonnin & Jannach, 2015]. Nos últimos anos diversos trabalhos foram apresentados a fim de solucionar ou aprimorar a geração automática de *playlists*. Por exemplo, os trabalhos de Pampalk et al. [2005], Bosteels & Kerre [2009], Flexer et al. [2008] e Hariri et al. [2012]. Cada trabalho possui estratégias específicas a fim de explorar várias definições sobre o cenário de recomendação e geração automática de playlists, porém com o objetivo comum de encontrar de forma automática uma sequência de músicas que atendam às necessidades de cada usuário a compor uma sessão de reprodução.

Pampalk et al. [2005] utilizam o *feedback* implícito a fim de definir interativamente o conjunto de músicas candidatas para a *playlist* do usuário. Os autores propõem que músicas similares às músicas aceitas sejam adicionadas como candidatas e, músicas similares às músicas não aceitas são removidas dos candidatos. O trabalho foi conduzido utilizando a similaridade baseada em características extraídos do áudio e os experimentos realizados sobre usuários fictícios, cujos comportamentos para simulação são propostos pelos autores.

Bosteels & Kerre [2009] apresentam uma modelagem para o cenário de geração dinâmica de *playlist* utilizando várias heurísticas (estáticas e dinâmicas, com base na similaridade entre as músicas aceitas e descartadas pelo usuário). O objetivo é operar sobre um conjunto de músicas candidatas e adicionar na *playlist* corrente do usuário. Bosteels & Kerre [2009] utilizam os mesmos dados de Pampalk et al. [2005] e inclusive a mesma proposta de características baseados em áudio. Nesse caso, a novidade está na apresentação de várias heurísticas sobre uma nova modelagem baseada na teoria de conjunto nebulosos (ou em inglês, *fuzzy sets*). Essa teoria assume uma ligação parcial entre itens e conjuntos, ou seja, um grau representando a pertinência de um item e um conjunto [Zadeh, 1965].

Flexer et al. [2008] introduzem um modelo para geração automática de *playlists* baseado na primeira e na última música de uma lista. A proposta é criar uma transição suave, permitindo ao usuário a descoberta de novas músicas da coleção durante uma sessão. Essa abordagem também utiliza similaridade sobre características extraídas do áudio.

Hariri et al. [2012] apresentam um sistema de recomendação de música sensível ao contexto. A informação contextual é baseada no histórico recente de músicas preferidas pelos usuários. Os autores propõem a mineração de tags para as mídias com o objetivo de determinar um conjunto de tópicos latentes. Por fim, padrões sobre a sequência de tópicos são extraídos utilizando uma base de dados de *playlists* criadas manualmente. Esses padrões são utilizados para predizer o próximo tópico sobre a sessão corrente de um usuário.

Esses últimos cinco trabalhos se assemelham com este pelo fato de utilizarem o *feedback* implícito, além de propor transições suaves entre as músicas, modelando um cenário mais próximo da realidade. Entretanto, apesar de muito explorado, ainda existem várias direções ainda pouco apreciadas em recomendação de *playlists* [Bonnin & Jannach, 2015]. Uma dessas direções são os trabalhos baseados em aprendizado *online* e por reforço, principalmente que utilizam de *feedback* implícito para aprimorar conhecimento na recomendação de cada música da sessão de reprodução de um usuário [Vall, 2015].

2.2 Aprendizado de Máquina

O aprendizado de máquina é um campo da ciência da computação que explora o estudo de técnicas envolvendo conhecimento de diversas áreas tais como probabilidade, estatística, matemática, dentre outras. O objetivo é fornecer capacidade de aprendizado para uma máquina, ou seja, são algoritmos capazes de aprimorar a eficácia de seus resultados conforme os dados são fornecidos para treino. Esse aprendizado pode ser realizado por exemplo, com a utilização da computação de padrões sobre o comportamento dos dados [Murphy, 2012].

Normalmente, esses algoritmos trabalham com a criação de modelos capazes de realizar predições sobre os dados, a partir de uma entrada de exemplo utilizada, o objetivo é computar direções e padrões sobre os dados. A construção desses modelos de aprendizado está diretamente ligada à técnicas de otimização matemática, na tentativa de aprimorar soluções e minimizando uma função de perda ou erro que deve medir a distância entre a solução conhecida e a ideal. O destaque dessas técnicas é a aplicabilidade em diversos problemas como classificação de textos, agrupamento automático, modelos de regressão e inclusive para recomendação de itens [Murphy, 2012].

O processo de aprendizado pode ser visto sob três perspectivas distintas: (i) supervisionado, quando um conjunto de dados é previamente rotulado e utilizado como entrada para treinar o algoritmo de aprendizado. Por exemplo, uma tarefa de classi-

ficação onde todo o conjunto de dados foi previamente classificado manualmente; (*ii*) não supervisionado, quando os dados de entrada para o algoritmo de aprendizado não são previamente rotulados, nesse caso a tarefa consiste em buscar padrões que caracterizem os dados. Por exemplo, o agrupamento de itens semelhantes em relação aos seus conteúdos; (*iii*) semi-supervisionado, quando uma pequena porção dos dados possui uma avaliação prévia no entanto, o algoritmo também utiliza dos dados não rotulados para a tarefa de aprendizado [Chapelle et al., 2010].

Ainda sobre essas três perspectivas é possível observar dois diferentes métodos de aprendizado. Primeiramente, o aprendizado *offline* compõe uma técnica comum na área e refere-se à algoritmos que utilizam de um conjunto de dados estáticos (algumas vezes rotulados a priori) para treinar e aprimorar o modelo. Por outro lado, o aprendizado *online* utiliza dinamicamente de um conhecimento iterativo, ou seja, os dados recebem rótulos conforme são realizadas interações [Ben-David et al., 1997]. A proposta desta dissertação se assemelha à perspectiva de aprendizado *online* supervisionado, conforme destacado na seguinte seção.

2.2.1 Aprendizado *Online* e Aprendizado por Reforço

Aprendizado *online* (do inglês, *online learning*) são métodos em aprendizado de máquina aplicáveis principalmente quando a estrutura no comportamento dos dados é observada sequencialmente. Por exemplo, dados em *streaming* são transmitidos sequencialmente conforme o consumo. Um algoritmo baseado em aprendizado *online* é capaz de interativamente aprimorar o conhecimento corrente (aumentando a eficácia sobre a tarefa) enquanto são observados novos exemplos a cada ponto no tempo [Shalev-Shwartz & Singer, 2007].

Aprendizado por reforço (do inglês, *reinforcement learning*) é um problema em aprendizado de máquina inspirado no comportamento psicológico [Peng et al., 2016]. De modo geral, são modelados um agente de aprendizado e um conjunto de ações possíveis sobre um determinado ambiente, onde cada ação gera uma recompensa desconhecida a priori. O objetivo do agente é interagir com o sistema e maximizar a recompensa acumulada conforme escolhe ações que transformam os estados do ambiente e do agente até um objetivo também desconhecido a priori [Sutton & Barto, 1998].

Ainda sobre aprendizado por reforço, existem abordagens *offline* (utilizando um conjunto de dados históricos) e *online* (interagindo com o ambiente enquanto aprende) [Wiering & Van Otterlo, 2012]. Entretanto, nem todo algoritmo de aprendizado *online* pode ser classificado como aprendizado por reforço. Basicamente, o

aprendizado por reforço tem como objetivo encontrar a melhor sequencia de ações a fim de maximizar a recompensa. O aprendizado *online* tem como objetivo aprimorar um modelo sobre uma determinada tarefa mas não necessariamente sobre uma sequência de ações. Por exemplo, um classificador automático de mensagens maliciosas. A cada nova mensagem, o classificador atribui um rótulo para a mensagem (sim ou não), recebe *feedback* do usuário (acerto ou erro) e melhora o modelo corrente. Dessa forma, o objetivo do classificador é encontrar um modelo para rotular as novas mensagens e não uma sequência das ações.

Voltando aos conceitos de aprendizado *online*, durante os anos vários algoritmos foram propostos para esta família com objetivo de aprimorar os modelos aprendidos conforme novas amostras dos dados são observadas. Em particular, destacamos os métodos de aprendizado *online* usando *Gradient Descent* (GD) (ou em português Descida de Gradiente). GD é um algoritmo clássico na literatura de otimização que iterativamente caminha em busca do gradiente de uma função, onde $J(\theta)$ define uma função objetivo com os parâmetros $\theta \in \mathbb{R}$ e o cálculo do gradiente $\nabla\theta J(\theta)$. A partir de valores de θ_t para cada iteração t, o algoritmo calcula a direção de maior declive que minimiza o resultado $J(\theta_t)$ da função objetivo no tempo t e atualiza os valores dos parâmetros para a próxima iteração θ_{t+1}. Dessa forma, o algoritmo explora novas combinações para os parâmetros até atingir um ponto de convergência, quando a diferença entre os resultados é menor que um critério de parada ϵ, ou seja, $J(\theta_{t+1}) - J(\theta_t) < \epsilon$. Devido a essa condição de parada, o algoritmo não tem garantias de encontrar a solução ótima [Ruder, 2016].

Vários métodos foram propostos com o objetivo de otimizar a busca da melhor configuração e reduzir o número de iterações. Por exemplo, uma simplificação é o *Stochastic Gradient Descent* (SGD) onde o gradiente é estimado a partir de um vetor aleatório a cada iteração. A teoria por trás de GD é extensamente estudada e para mais aprofundamento no assunto indicamos a leitura extra [Bottou, 2010; Ruder, 2016].

No caso de aprendizado *online* usando GD a proposta é que a estimativa do gradiente da função seja realizada sobre um cenário interativo, ou seja, enquanto novas evidencias são observadas sobre o cenário [Flaxman et al., 2005]. Destacamos sobre esse cenário dois trabalhos que motivaram a proposta desta dissertação. Yue & Joachims [2009] introduzem um problema de otimização *online* chamado de *dueling bandits* para o cenário de recuperação da informação. Os autores apresentam um problema de comparação entre dois pontos ($\mathbf{w}'$ e $\mathbf{w}^*$, assim como um duelo) dentro de um espaço W, onde o espaço W pode ser modelado sobre qualquer função. Por exemplo, uma função para um espaço de parâmetros para recuperação de documentos relevantes em uma sistema de busca. A partir dessa definição do problema, Yue & Joachims [2009]

propõem DBGD (*Dueling Bandits Gradient Descent*) um algoritmo de aprendizado *online* capaz de ordenar respostas em um sistema de recuperação de informação, o qual escolhe o melhor modelo a partir de um duelo entre dois modelos a cada consulta recebida pelo sistema, definindo a direção do gradiente.

Formalizando o algoritmo DBGD, seja $\mathbf{w}'$ um vetor de pesos considerado o melhor modelo corrente, o algoritmo explora um vetor unitário $\mathbf{v}$ e conforme um fator de exploração δ produz um novo modelo em torno do melhor modelo corrente, onde $\mathbf{w}^* \leftarrow \mathbf{w}' + \delta\mathbf{v}$. Usando uma técnica conhecida como *Team Draft Interleave* (TDI), o DBGD produz uma lista $\mathbf{l}$ que intercala os resultados entre os modelos $\mathbf{w}'$ e $\mathbf{w}^*$. A lista $\mathbf{l}$ é submetida a uma avaliação *online* realizada pelo usuário a fim de inferir o melhor entre os modelos. Por fim, caso $\mathbf{w}^*$ seja melhor que $\mathbf{w}'$, então o modelo $\mathbf{w}'$ é atualizado utilizando o fator de aproveitamento γ, onde $\mathbf{w}' \leftarrow \mathbf{w}' + \gamma\mathbf{v}$. Nesse caso, γ é utilizado para calibrar a distância do passo entre o melhor modelo corrente $\mathbf{w}'$ e o modelo candidato $\mathbf{w}^*$ a fim de permitir o aproveitamento do conhecimento já adquirido [Yue & Joachims, 2009].

Devido ao processo de TDI, o DBGD é capaz de comparar apenas dois pontos $\mathbf{w}'$ e $\mathbf{w}^*$ para cada consulta. Sendo assim, Schuth et al. [2016] propõem o algoritmo MGD, o qual explora os mesmos conceitos do DBGD mas, ao invés de realizar comparações entre dois modelos, são utilizados múltiplos modelos para obter a ordenação dos resultados a cada consulta. A principal diferença é a introdução do *Team Draft Multileave* (TDM) a fim de permitir a comparação a partir da intercalação de múltiplos modelos sobre os resultados de uma mesma consulta. Além disso, Schuth et al. [2016] propõem duas funções para atualização do melhor modelo corrente. Os resultados apresentam melhoras significativas em relação ao DBGD, encontrando melhores modelos de forma mais rápida.

Ainda sobre aprendizado *online* e aprendizado por reforço, a seguinte seção descreve os principais trabalhos encontrados na literatura que se assemelham a esta dissertação. São destacamos essas abordagens com perspectivas sobre o cenário de aprendizado *online* com foco na recomendação de músicas.

2.2.2 Aprendizado *Online* na Recomendação de Músicas

Assim como os algoritmos de aprendizado de máquina, os métodos baseados em aprendizado *online* e aprendizado por reforço são aplicados em diversos contextos conforme apresentado na seção anterior. Nesta seção são destacados trabalhos relacionados a esta dissertação com uma perspectiva centrada ao cenário de recomendação *online* de músicas e geração automática de *playlists*.

Xing et al. [2014] propõem encontrar o balanceamento entre a exploração e aproveitamento (do inglês *exploration and exploitation*, significa, respectivamente, explorar conhecimento novo e aproveitar o conhecimento já obtido) utilizando uma abordagem baseada em filtragem colaborativa. A hipótese é fundamentada no fato de que a filtragem colaborativa tende a recomendar itens com melhores avaliações, porém sempre associados à um conjunto de itens mais populares. Para tratar esse problema, os autores propõem um novo algoritmo de aprendizado por reforço com base em avaliações explícitas, a fim de encontrar equilíbrio em explorar novos itens ou aproveitar o conhecimento mais popular.

Wang et al. [2014] apresentam uma nova abordagem para recomendação de músicas também baseada em uma formulação para balancear a exploração e o aproveitamento. Os autores propõem uma modelagem para o problema utilizando aprendizado por reforço. A abordagem utiliza um método *bayesiano* para contabilizar as preferências sobre o conteúdos dos áudios e sobre a novidade das recomendações. Por fim, a proposta pode ser utilizada tanto para recomendação de músicas quanto para geração de *playlists*, entretanto também utiliza o *feedback* explícito.

Hariri et al. [2015] propõem uma abordagem interativa para recomendação de músicas, adaptando-se às mudanças nas preferências dos usuários. A hipótese é que as preferências dos usuários podem mudar durante o tempo e, para capturar tais mudanças, é proposto um modelo baseado na estratégia de *multi armed bandit* (ou em português "bandido multi armado"). A cada interação o sistema apresenta uma lista de item e recebe *feedback* indicando a qualidade das recomendações. A intenção então é captar de forma ágil de acordo com as alterações no comportamento do usuário e mudar a direção das recomendações.

O trabalho de Liebman et al. [2015] também descreve uma abordagem baseada em aprendizado por reforço para recomendação de sequências de músicas ou *playlists*. O modelo é baseado na semelhança entre músicas e nas transições, além de ser personalizado para cada usuário. Para ajustar as transições, o modelo utiliza cadeias de Markov onde as músicas representam os estados de transição. Nesse trabalho também é proposta a adequação do modelo de recomendação utilizado o *feedback* explícito a cada interação com o usuário.

Vall [2015] apresenta uma proposta de tese de doutorado definindo caminhos para seus estudos em recomendação de músicas e alguns resultados preliminares. O objetivo principal é a geração de *playlists* automáticas baseada em dados obtidos a partir do usuário. Como resultado parcial, é explorado um problema de recomendação de novos artistas usando os históricos e as atividades dos usuários, obtido por meio de *tags*. É importante destacar os caminhos para trabalhos futuros onde o autor propõe

a utilização do *feedback* implícito para adaptação do modelo de aprendizado, cenário ainda pouco explorado e que é abordado neste trabalho.

O presente trabalho se assemelha aos demais citados pela construção de um modelo com objetivo de recomendar sequências musicais, ou *playlists*, de forma personalizada e automática. Porém, propomos a utilização unicamente do *feedback* implícito a cada interação, dado que melhor representa o comportamento dos usuários em sistemas de *streaming* de música. Todas as abordagens citadas anteriormente exigem o *feedback* explícito para atualização de um algoritmo aprendizado.

Encontramos apenas um trabalho na literatura utilizando *feedback* implícito para recomendação de músicas usando aprendizado por reforço. King & Imbrasaitė [2015] apresentam um abordagem para geração automática de *playlist* utilizando aprendizado por reforço sobre um agrupamento hierárquico baseado em características do áudio. O objetivo é aprender as preferências dos usuários e utilizar do *feedback* implícito na adaptação do modelo por reforço. Foram extraídas características de áudio das mídias com intuito de propor uma aplicação independente de fontes externas e capaz de gerar sequências consistentes a partir da primeira música. Os resultados coletados foram suficientes para perceber redução de *skips*, porém obtidos a partir de experimentos realizados em um pequeno estudo de caso.

Apesar de apresentarem uma ideia também similar ao presente trabalho, a abordagem de King & Imbrasaitė [2015] tem um propósito diferente. Exclusivamente, foram utilizadas apenas características baseados nos áudios. Nesta dissertação, o objetivo independe da definição das características que representam as músicas e os usuários, o que permite a utilização de qualquer evidência e fonte de dados. Além disso, King & Imbrasaitė [2015] propõem o aprendizado por reforço para escolha de grupos de músicas, sendo esse um nível mais genérico se comparado a escolha de cada música da sessão, conforme nesta dissertação.

A presente abordagem envolve também conhecimentos sobre um problema clássico em teoria de probabilidade conhecido como *multi armed bandit*. A seguinte seção define uma visão geral sobre esse problema, assim como a relação esta dissertação.

2.3 *Multi Armed Bandit* (MAB)

Multi armed bandit (MAB) é um problema clássico de decisão em teoria de probabilidade modelado com o seguinte cenário. Um jogador está em frente a um conjunto K de máquinas para jogos de azar. Assim, o problema de decisão é escolher quais máquinas, quantas vezes e em qual ordem deve jogar. Cada máquina oferece uma recompensa

aleatória, ou *payoff*, a partir de uma distribuição de probabilidade desconhecida. Assim, o objetivo do jogador é maximizar o *payoff* acumulado a partir de uma sequência de jogadas [Gittins, 1979].

O objetivo dessa tarefa é basicamente encontrar um balanceamento entre explorar outras máquinas ou aproveitar o conhecimento sobre a probabilidade estimada de cada máquina já jogada. A fim de solucionar esse problema, ao longo dos anos foram apresentadas diversas soluções como o ϵ- *greedy*, a família de abordagens abordagens baseadas no *Upper Confidence Bound* (UCB) (ou em português intervalo de confiança superior) [Lai & Robbins, 1985; Auer et al., 2002; Garivier & Cappé, 2011]. Também encontramos abordagens bem antigas na literatura, assim como o *Thompson Sampling* [Thompson, 1933].

Ainda sobre os trabalhos recentes, encontramos propostas de adaptações do problema clássico de MAB em diversos cenários. Algumas dessas adaptações têm foco principal em problemas sensíveis ao contexto. A definição de *contextual multi armed bandit* (MAB contextual) é similar ao problema clássico de MAB. No entanto, nesse caso o jogador também pode observar informações sobre o contexto para decidir em qual máquina apostar [Langford & Zhang, 2008]. Conforme Li et al. [2010], um algoritmo de MAB contextual, pode ser formalizado da seguinte maneira:

Dado uma sequência de tentativas $t = 1, 2, 3, ...$ Em cada tentativa t:

1. O algoritmo observa o usuário u_t, um conjunto K_t de máquinas e um vetor de características $x_{t,k}$ para todo $k \in K_t$. Nesse caso, $x_{t,k}$ resume as informações do usuário u_t no instante t e do item k representando o contexto.

2. Utilizando o *payoff* observado em tentativas passadas (ou seja, $t = 1, 2, 3, ..., t-1$), o algoritmo escolhe uma máquina $k \in K_t$, e recebe um novo *payoff* r_{t,a_t}, onde a expectativa depende do usuário u_t e seu contexto, além da probabilidade relacionada à máquina selecionada a_t.

3. Para cada t, é observado *feedback* apenas para a máquina selecionada, sendo essa a única observação disponível para que o algoritmo melhore as predições ao estimas as probabilidades de *payoff*, resultando na tripla $(x_{t,a_t}, a_t, r_{t,a_t})$.

Li et al. [2010] apresentam uma adaptação do problema de MAB contextual ao cenário de recomendação de notícias. De modo similar, a recomendação de músicas também permite essa adequação. Nesse caso, o conjunto de K máquinas, são músicas candidatas para recomendação a cada tempo t. O usuário, em ambos os cenários, representa o elemento que utiliza um serviço. Por fim, a recompensa pode ser medida,

por exemplo, a partir do *feedback* implícito, *play* ou *skip*, respectivamente recebendo *payoff* um ou zero.

Conforme observado por Li et al. [2010], um caso especial desse cenário ocorre quando o conjunto K é fixo para todo t, e o contexto dos usuários é também é mantido. Esse problema mais específico é conhecido como livre de contexto, pois não sofre influência do contexto, porém não fornece uma boa adaptação para o cenário de recomendação *online* de músicas. No cenário proposto nesta dissertação, o conjunto de músicas candidatas é alterado conforme as necessidades dos usuários a cada interação, assim como o contexto também pode ser alterado influenciando potencialmente nas preferências dos usuários. Nesta dissertação, propomos a utilização de um conjunto K variando para cada tempo t.

A presente dissertação propõe a utilização de técnicas de aprendizado *online* interagindo sobre esse cenário de MAB contextual. Alguns exemplos de trabalhos que utilizam dessas técnicas são Song et al. [2014b] e Song et al. [2014a], onde os autores apresentam algoritmos de aprendizado *online* para solução do problema de MAB contextual contendo em ambos, avaliações teóricas e empíricas. No entanto, o aprendizado é aplicado sobre o nível de grupos de itens, sendo o trabalho de Song et al. [2014a] uma proposta mais simples que utiliza grupos fixos e Song et al. [2014b] uma abordagem cujo agrupamento pode ser adaptado interativamente.

Outros trabalhos relacionados formalizam a mesma ideia porém em cenários diferentes. Por exemplo, Hou et al. [2016] utilizam técnicas de aprendizado *online* para recomendação em uma grande plataforma de cursos *online*. Outro exemplo recente foi conduzido por Nguyen et al. [2016] com a proposta de um algoritmo de aprendizado *online* baseado em agrupamentos para recomendação de notícias em um cenário livre de contexto. Em ambos os casos, a modelagem adotada é baseada em MAB contextual. Além dos cenários distintos, a proposta desta dissertação é a recomendação de cada música da sessão do usuário sendo este um nível mais específico do que por exemplo a abordagem de Nguyen et al. [2016].

Por fim, sobre essas diversas áreas do conhecimento o melhor algoritmo encontrado a fim de servir como *baseline* para o algoritmo proposto nesta dissertação é o LINUCB [Li et al., 2010]. Apesar de ser utilizado originalmente para recomendação de notícias, o LINUCB possui características adequadas também ao cenário de recomendação *online* de músicas, assim como a sensibilidade ao contexto e a utilização do *feedback* contínuo.

2.4 Metodologias de Avaliação em Sistemas de Recomendação

O objetivo das metodologias de avaliação é apresentar um arcabouço capaz de promover hipóteses, verificar o funcionamento adequado de soluções para uma determinada tarefa. Inicialmente, o poder de predição tinha muito impacto na avaliação dos sistemas de recomendação, ou seja, a eficiência em predizer as escolhas dos usuários. Atualmente, as propriedades relevantes de um recomendador vem sofrendo mutações, onde os usuários tendem a exigir características como diversidade, respostas imediatas, entre outros fatores que as avaliações tradicionais não mediam. Sendo assim, apesar da grande evolução na área a partir de novas propostas de algoritmos, a sofisticação das metodologias de avaliação a fim de comprovar o funcionamento adequado dessas novas técnicas também é essencial para promoção dos trabalhos [Ricci et al., 2015].

As metodologias de avaliação em sistemas de recomendação podem ser vistos em três perspectivas (experimentos *offline*, estudos sobre os usuários, experimentos *online*). Essas perspectivas possuem modelos experimentais que permitem a adaptação sobre cenários distintos conforme a necessidade e a cobertura de cada avaliação [Ricci et al., 2015].

Conforme Ricci et al. [2015], além das configurações experimentais, para garantir coesão nos resultados é importante que os experimentos sigam algumas premissas: (*i*) os experimentos devem condizer com a hipótese que deseja discutir; (*ii*) as variáveis devem ser controladas para manter igualdade dos testes sobre o mesmo cenário; (*iii*) caso o objetivo seja observar conclusões sobre a generalização de um determinado método, é necessária a realização de testes em várias bases de dados diferentes.

2.4.1 Experimentos *Offline*

Em sistemas de recomendação, os experimentos *offline* são realizados sobre uma coleção de dados históricos, normalmente coletados a partir da observação de um conjunto de usuários. Esses dados devem conter interações ou avaliações sobre os itens realizadas em um determinado período. Nesse tipo de experimento, assume-se que o comportamento dos usuários durante o período coletado reflete o comportamento dos usuários sobre o cenário real. Ou seja, assume-se a possibilidade de gerar simulações a partir dos dados históricos, capazes de representar adequadamente o comportamento real dos usuários [Ricci et al., 2015].

O grande benefício desse tipo de experimento é não exigir interações com usuários reais, além de ter um custo baixo de execução aumentando a reprodutibilidade

dos experimentos. Além disso, é possível explorar os problemas mesmo sem acesso ao domínio sobre um ambiente real. O objetivo principal é comparar e filtrar abordagens candidatas para serem submetidas a testes mais caros. Apesar de ser muito importante e comum em várias pesquisas, medir a influência das abordagens sobre o comportamento real dos usuários usando apenas dados históricos é uma tarefa complexa [Ricci et al., 2015].

Um dos desafios da avaliação provém do viés presente nos dados históricos, devido à complexidade em coletar dados suficientes sobre todas as preferências dos usuários [Ricci et al., 2015]. Por exemplo, um usuário de um serviço de *streaming* de música evita interações sobre gêneros que não lhe agrade. Portanto, os dados históricos dificilmente devem carregar informações negativas, já que as interações não são observadas.

Vários estudos importantes são baseados nesse tipo de avaliação, assim como os trabalhos de Herlocker et al. [1999], Sarwar et al. [2001] e Burke [2002]. Nesses casos, foram usadas coleções contendo avaliações explícitas dos usuários, por exemplo a coleção de avaliações em filmes disponibilizada pela Movielens.[4] Essas propostas utilizavam medidas de erro para avaliar o quão distante as predições estão da avaliação real.

Segundo Cremonesi et al. [2010], a predição de notas explicitas não reflete a forma real como os usuários consumem os itens de um sistema real. Dessa forma, os resultados não representavam como os algoritmos se comportariam em ambientes reais. A proposta de Cremonesi et al. [2010] é modelar o consumo de recomendações por meio de uma lista ordenada de itens, onde quanto mais relevante, mais ao topo o item deve ser apresentado. Para isso, cada item do conjunto de teste é misturado em um conjunto de itens aleatoriamente selecionados de um conjunto desconhecido para cada usuário. A proposta então é reordenar a lista e avaliar os resultados a partir de métricas de ranking, por exemplo, a posição média em que os itens relevantes aparecem na lista.

2.4.2 Experimentos com Estudo de Usuários

Apesar de não ser o foco desta dissertação, experimentos baseados em estudos de usuários são muito comuns na literatura de recomendação. Essas abordagens selecionam um conjunto de usuários de teste a fim de questionar observações sobre os resultados de um novo algoritmo. Um exemplo típico desses experimentos consiste em testar a influência de um algoritmo no comportamento navegacional de usuários ao lerem notícias. Nesse caso, o usuário deve ler um conjunto de notícias que lhe interessa e em

[4] https://www.movielens.org

alguns casos são inclusas recomendações do novo algoritmo sendo possível medir como a recomendação foi de fato apreciada pelos usuários [Ricci et al., 2015].

Esse tipo de experimento possui várias vantagens e recebe atenção de muitas áreas, já que permite a aplicação de questionários, entrevistas, entre outras abordagens que medem qualitativamente os resultados [Ricci et al., 2015]. Em sistemas de recomendação, os trabalhos de Hu & Pu [2009] e Hu & Pu [2010] são alguns desses exemplos. Entretanto, vários fatores encarecem essa abordagem como: selecionar usuários de teste, sejam elas voluntárias ou não, a fim de coletar dado suficiente para avaliação; amostrar usuários de diversas características que representem melhor o possível cenário real; entre outros [Ricci et al., 2015]. Sendo assim, com objetivo de reduzir o custo experimental, essa perspectiva não irá compor os experimentos desta dissertação.

2.4.3 Experimentos *Online*

A terceira perspectiva são os experimentos *online*, utilizadas principalmente quando as abordagens levam em consideração do comportamento dos usuários. Nesse caso, a avaliação depende de várias fontes de informação, como contexto, intenção do usuário, personalidade, interface, entre outros [Ricci et al., 2015].

Nesses experimentos é necessário submeter os algoritmos em tarefas com usuários reais, obtendo assim evidências de resultados concretos sobre o comportamento das abordagens no ambiente. Normalmente, uma pequena amostragem dos usuários são expostos a esses experimentos devido ao alto risco de insatisfação e abandono do sistema, já que os resultados são totalmente desconhecidos a priori [Ricci et al., 2015]. Além disso, o custo do acesso a sistemas reais com usuários suficientes para realização desse tipo de teste é uma dificuldade enfrentada em diversos estudos. Em recomendação *online* de músicas por exemplo, seria necessário acesso a um sistema de *streaming* de música produtivo, como o Spotify.

Esses experimentos são comuns em várias áreas, como em recuperação da informação. No entanto, executar experimentos *online* de forma padronizada a fim de comparar resultados entre várias abordagens, resulta em uma tarefa muito custosa e de alto risco, sendo altamente recomendado que os experimentos *offline* sejam executados a priori. Os experimentos *offline* nesse caso tem importante papel de garantir melhores condições para experimentos *online*, aumentando as chances de satisfação dos usuários que utilizam uma nova abordagem [Hofmann, 2013].

2.5 Sumário

Este capítulo descreve os principais trabalhos relacionados aos assuntos tratados nesta dissertação. Os conceitos abordados introduzem um conhecimento geral sobre os sistemas de recomendação e aprendizado de máquina com enfase na recomendação de músicas e aprendizado *online*. São mapeados os principais trabalhos em recomendação *online* a fim contextualizar as direções desta dissertação sobre um cenário pouco explorado em estudos passados. Também é introduzido o LINUCB, o melhor algoritmo encontrado para servir como *baseline* desta proposta. Por fim, são apresentadas as principais configurações experimentais realizadas em sistemas de recomendação, tendo em vista os principais desafios em garantir consistência sobre os experimentos realizados.

O próximo capítulo apresenta o embasamento teórico que motivou a proposta do algoritmo Multi-DBGD. Em detalhes são apresentados os componentes sobre a modelagem para o cenário de recomendação *online* de músicas.

Capítulo 3

Multi-DBGD: *Multi Dueling Bandit Gradient Descent*

O presente capítulo tem como objetivo detalhar os componentes da proposta do Multi-DBGD, um novo algoritmo baseado em aprendizado *online* para recomendação de músicas utilizando *feedback* implícito. A Seção 3.1 introduz uma modelagem para a tarefa de recomendação online de músicas envolvendo o conhecimento explorado e adequação da nossa proposta. A Seção 3.2 apresenta detalhes sobre os componentes componentes do algoritmo Multi-DBGD, assim como os desafios confrontados.

3.1 Definição do Arcabouço

O cenário de recomendação de músicas apresenta vários desafios, tais como a recomendação sequencial de músicas, utilização do *feedback* contínuo, geração automática de *playlists* (sequencia de músicas), entre outros. Nesta dissertação, o arcabouço é baseado na recomendação *online* de cada música de uma sessão de reprodução utilizando a primeira música da sessão e o *feedback* implícito contínuo. Dessa forma, a presente seção discute a modelagem para esse desafio em serviços de *streaming* de música.

A Figura 3.1 modela o comportamento de um usuário ao utilizar um serviço de *streaming* de música tendo em vista o desafio proposto nesta dissertação. O comportamento do usuário u está relacionado à geração de uma sessão $\mathbf{S}$ a partir do consumo sequencial de músicas $\mathbf{S} = \{s_1^{\oplus}, s_2^{\ominus}, s_3^{\oplus}, s_4^{\ominus}, ..., s_t^{\oplus}\}$, onde s representa a música que o usuário interage a cada tempo t. Dessa forma, nosso objetivo é encontrar um modelo, representado por um vetor de pesos $\mathbf{w}$, capaz de compor uma sessão $\mathbf{S}$ personalizada, observando apenas a primeira música da sessão $s_1^{\oplus}$ (semente escolhida pelo usuário para geração de uma *playlist* automática) e o *feedback* sobre as músicas $s_t^{\oplus}$ selecionadas

a cada ponto t no tempo. Na figura cada música selecionada um *feedback* implícito é observado, onde o negativo (*skip*) é representado pelo símbolo $\ominus$ e o positivo (*play*) é representado pelo símbolo $\oplus$.

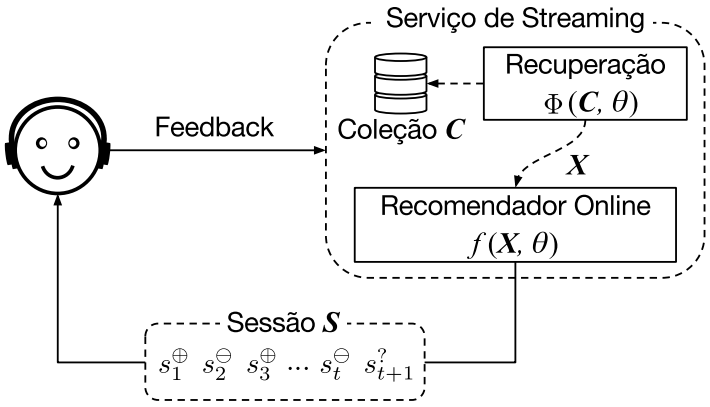

Figura 3.1. Diagrama sobre o comportamento de um usuário ao utilizar um serviço de *streaming* de música.

Ainda sobre a Figura 3.1, definimos o serviço de *streaming* divido em três componentes: (i) coleção $\mathbf{C}$, representando a base de dados composta por milhares de músicas; (ii) recuperação Φ, representado por uma função responsável por recuperar uma lista de músicas candidatas para cada tempo t da interação com o usuário u, onde cada música está representada por um vetor de n características $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ $\forall \mathbf{x} \in \mathbf{X}$; ($iii$) recomendador *online* f: função que utiliza a sessão corrente do usuário e o *feedback* implícito já observado para selecionar uma música $s_{t+1}^{?}$ das músicas candidatas $\mathbf{X}$ recuperadas pela função Φ, a incluir sobre a sessão $\mathbf{S}$ corrente do usuário, cujo *feedback* é desconhecido a priori. Nesta dissertação, propomos a função f utilizando o algoritmo Multi-DBGD. Na modelagem do arcabouço, o usuário consome uma música $s_{t+1}^{?}$ a cada interação e fornece *feedback* a partir dos comandos de *play* ($\oplus$) ao escutar a música completa, ou *skip* ($\ominus$) ao abandonar a música em execução. Esses comandos representam o *feedback* implícito, cuja definição é apresentada na próxima seção.

Definição do *Feedback*

O *feedback* pode ser definido como uma resposta enviada pelo usuário ao sistema. De modo geral, existem dois tipos de *feedback*, o explícito e o implícito. Em um cenário de *streaming* de música, o *feedback* explícito representa uma declaração do usuário sobres suas preferências, assim como uma lista de músicas e artistas preferidos. Esse tipo de resposta é menos frequente devido ao custo atrelado a tarefa de avaliação e análise realizada pelo usuário. O *feedback* implícito representa a observação do comportamento

do usuário sobre o sistema, ou seja, são sinais fornecidos sem a intenção de avaliar um determinado item. Comparado ao *feedback* explícito, o *feedback* implícito ocorre de forma mais abundante e frequente, porém é mais ruidoso. Em um serviço de *streaming* de música por exemplo, o feedback a partir dos comandos de *play* e *skip* pode carregar ruído quando o usuário executa o comando de *skip* involuntariamente sobre músicas que estão entre suas preferidas em um determinado contexto [Jawaheer et al., 2010].

De modo particular, nesta dissertação propomos a utilização do *feedback* implícito como único sinal de preferência a cada ponto no tempo. Esse tipo de *feedback* apesar de ruidoso é sensível ao contexto e pode representar mudanças no comportamento do usuário. Por exemplo, as preferências dos usuários podem ser influenciadas pelo sentimento, clima, atividade, entre outros fatores que potencialmente mudam a forma como os usuários interagem com os comandos de *play* e *skip* [Jawaheer et al., 2010].

Consideramos a utilização de três classificações sobre *feedback* implícito segundo Yang et al. [2012]: (i) o *play* explícito, quando o usuário utiliza o comando de *play* com a intenção de executar uma música específica. No cenário desta dissertação, consideramos como *play* explícito a utilização da primeira música $s_1^{\oplus}$ fornecida pelo usuário a fim de gerar uma *playlist* automaticamente; (ii) o *play* de conclusão, quando o usuário escuta completamente uma música sem executar qualquer *skip*; (iii) o *skip* explícito, quando o usuário utiliza o comando de *skip* para abandonar uma música em execução e solicita ao sistema que avance para uma próxima música.

Por fim, todo o arcabouço definido tem como embasamento o problema clássico de decisão em teoria da probabilidade conhecido como *Multi Armed Bandits* (MAB), particularmente na extensão *Contextual Bandits* (MAB contextual). Em MAB, um jogador u está frente a um conjunto de máquinas de jogos de azar $\mathbf{X}$, cada uma com uma distribuição desconhecida a priori que define as chances de premio. O jogador precisa escolher uma sequência de jogadas $\mathbf{S}$, analisando para cada jogada a probabilidade de prêmios das máquina em $\mathbf{X}$ com o objetivo de maximizar o lucro. Normalmente na literatura de MAB, a cada interação com uma máquina o *feedback* observado é modelado sobre um conceito de *payoff* [Gittins, 1979]. A definição de MAB contextual é similar ao problema clássico de MAB. Entretanto, nesse caso o jogador u também pode observar informações sobre o contexto para decidir em qual máquina apostar [Langford & Zhang, 2008].

Nesta dissertação, o jogador u é representado pelo usuário de um serviço de *streaming* de músicas, as máquinas $\mathbf{X}$ são as músicas candidatas para cada tempo t, onde a sequencia de jogadas $\mathbf{S}$ define a sessão de músicas do usuário u. Além disso, o $payoff = 0$ representa o *feedback* implícito negativo ou *skip* e $payoff = 1$ o *feedback* implícito positivo ou *play*. Dessa forma, sempre que utilizarmos a palavra *feedback*

estamos considerando o sinal observado a partir do usuário e o respectivo valor de *payoff* conforme descrito acima.

Por fim, a modelagem do *feedback* em conjunto ao arcabouço apresentado é similar as condições da tarefa de contextual MAB, estudadas no Capítulo 2. Portanto, algoritmos de contextual MAB podem ser utilizados para compor a função de recomendação *online* f, a fim de escolher um candidato à cada interação com o usuário. Assim, propomos o Multi-DBGD, um novo algoritmo com vistas a este componente, a saber na seguinte seção.

3.2 Descrição do algoritmo

Nesta seção apresentamos um novo algoritmo de aprendizado *online* usando *feedback* implícito chamado Multi-DBGD (*Multi Dueling Bandits Gradient Descent*, ou em português, Descida de Gradiente de Multi Duelos entre Bandidos). O algoritmo Multi-DBGD é baseado nas propostas de dois algoritmos de aprendizado *online* usando *Gradient Descent* (GD) aplicados na área de recuperação da informação. O primeiro algoritmo é o DBGD (*Dueling Bandits Gradient Descent*) [Yue & Joachims, 2009] que utiliza aprendizado *online* capaz de ordenar respostas em um sistema de recuperação da informação. O objetivo é a comparação de dois modelos ($\mathbf{w}^*$ e $\mathbf{w}'$) a cada consulta recebida pelo sistema. Dessa forma, o algoritmo atualiza o modelo para a próxima interação $\mathbf{w}^*_{(t+1)}$ baseado nos resultados do melhor modelo corrente $\mathbf{w}^*_{(t)}$ comparado a um modelo candidato $\mathbf{w}'_{(t)}$. A escolha do melhor modelo a cada consulta estima o gradiente da função, ou seja, a variação sobre o melhor corrente que reduz o erro da função. O segundo algoritmo é o MGD (*Multileave Gradient Descent*) [Schuth et al., 2016], o qual explora os mesmos conceitos do DBGD, porém múltiplos modelos são comparados a cada consulta observada. A comparação de múltiplos modelos proposta para o MGD permite encontrar modelos com melhores resultados de forma mais rápida quando comparado ao DBGD.

Os algoritmos DBGD e MGD dependem da avaliação do usuário sobre todos os modelos (dois no caso do DBGD), para estimar a direção da atualização do melhor modelo corrente. Em ambos os casos, a cada consulta os modelos candidatos e o melhor modelo corrente são igualmente avaliados pelo usuário, dado que as respostas do sistema são ordenadas a partir da intercalação das respostas obtidas por cada modelo. Dessa forma, a principal diferença do MGD é a intercalação das respostas de múltiplos modelos para atualizar o melhor modelo corrente. Entretanto, no cenário de recomendação *online* de músicas, o *feedback* pode ser apenas sobre uma música a cada interação

com o usuário, fato que inviabiliza a utilização do DBGD ou do MGD nesse cenário, devido as técnicas de TDI e TDM que exigem feedback sobre os resultados dos demais modelos. A partir disso, nossa proposta é um algoritmo capaz de realizar múltiplas comparações entre modelos, reutilizando o *feedback* sobre uma mesma música a cada interação a fim de aprimorar as recomendações futuras.

O algoritmo Multi-DBGD é uma função f utilizada para selecionar sequencialmente a música $s_{(t)}$ para cada tempo t a compor a sessão S do usuário u. Sobre a única música recomendada o usuário atribui um *feedback* implícito, sendo *payoff* $= 0$ representando o *feedback* implícito negativo ou *skip* e *payoff* $= 1$ o *feedback* implícito positivo ou *play*. A partir de um modelo inicial $\mathbf{w}^*_{(1)}$ e baseando no *feedback* observado, o algoritmo atualiza o melhor modelo corrente $\mathbf{w}^*_{(t+1)}$ a fim de aprimorar o conhecimento corrente sobre as preferencias dos usuários e reduzir o erro a partir do modelo inicial. Assim como o DBGD e o MGD, assumimos um modelo $\mathbf{w}$ representado por um vetor de n pesos distribuídos sobre a representação das músicas candidatas em $\mathbf{X}$. Cada música $\mathbf{x}$ é representada por um vetor de n características, onde $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ $\forall \mathbf{x} \in \mathbf{X}$. Dessa forma, o modelo atribui pesos sobre a representação das músicas para calcular a nota das músicas candidatas $\mathbf{X}$ e selecionar a música com maior valor.

O Algoritmo 1 descreve a rotina principal do Multi-DBGD, responsável por selecionar a cada tempo t uma música $s_{(t)}$. Os parâmetros necessários para essa rotina são, u representando o usuário, m o número de duelos entre os modelos, γ e δ os fatores de exploração e aproveitamento e $s_1^{\oplus}$ a primeira música da sessão fornecida pelo usuário como semente para geração automática da sessão.

Algoritmo 1 Rotina principal do Multi-DBGD.

Multi-DBGD-Select$(u, m, \delta, \gamma, s_1^{\oplus})$

1 $\mathbf{w}^*_{(1)} \leftarrow$ Multi-DBGD-Initialize$(u, m, \delta, \gamma, s_1^{\oplus})$ ▷ Inicialização do modelo
2 $e^{w^*}_{(1)} \leftarrow 1.0$ ▷ Assume-se inicialmente eficácia máxima
3 **for** $t \leftarrow 1$ **to** ∞ **do**
4 $\mathbf{X} \leftarrow \Phi(u, t)$ ▷ Φ Recupera as músicas candidatas
5 $s_{(t)} \leftarrow \arg\max\langle \mathbf{w}^*_{(t)}, \mathbf{X} \rangle$ ▷ Recupera a música com maior nota
6 **if** *payoff*$(u, t, s_{(t)}) = 0$ **then** ▷ Recebe *feedback* do usuário
7 $\gamma \leftarrow abs(\gamma)$ ▷ Incorporação do *feedback* será positiva
8 $e^{w^*}_{(t+1)} \leftarrow 1.0$ ▷ Assume-se eficácia máxima para a próxima interação
9 **else**
10 $\gamma \leftarrow abs(\gamma) \times (-1)$ ▷ Incorporação do *feedback* será negativa
11 $e^{w^*}_{(t)}, e^{w^*}_{(t+1)} = \text{decaying}(e^{w^*}_{(t)})$ ▷ Política de decaimento
12 **end if**
13 $\mathbf{w}^*_{(t+1)} \leftarrow$ Multi-DBGD-Update$(\mathbf{X}, s_{(t)}, \mathbf{w}^*_{(t)}, e^{w^*}_{(t)}, m, \gamma, \delta)$
14 **end for**

Ainda sobre o Algoritmo 1, a cada tempo t, o usuário fornece *feedback* sobre uma música selecionada pelo melhor modelo corrente $\mathbf{w}^*_{(t)}$ onde o *skip* e o *play* recebem respectivamente *payoff* $= 0$ e *payoff* $= 1$. Esse *feedback* é utilizado para atualizar a eficácia do melhor modelo corrente representada no algoritmo pela variável e^{w^*} e para aprimorar o modelo utilizado na próxima interação.

O Algoritmo 2 é uma sub-rotina executada a partir do Algoritmo 1 e tem como objetivo retornar uma atualização sobre o melhor modelo corrente com base nos seguintes parâmetros: $\mathbf{X}$ representando o conjunto de músicas candidatas e sua características, s a música selecionada pelo melhor modelo corrente, $\mathbf{w}^*$ uma cópia do melhor modelo corrente, e^{w^*} a eficácia do melhor modelo corrente, m o número de duelos entre os modelos e γ e δ os fatores de exploração e aproveitamento.

Algoritmo 2 Sub-rotina para atualizar o modelo a partir do *feedback*.

Multi-DBGD-Update$(\mathbf{X}, s, \mathbf{w}^*, e^{w^*}, m, \delta, \gamma)$

1 **for** $i \leftarrow 1$ **to** m **do** ▷ Múltiplos Duelos
2 $\mathbf{v} \leftarrow$ sample-unit-vector()
3 $\mathbf{w}' \leftarrow \mathbf{w}^* + \delta\mathbf{v}$
4 $e^{w'} \leftarrow$ effectiveness-simulated$(\mathbf{w}^*_{(1)}, \mathbf{X}, s)$ ▷ Simulação da eficácia
5 **if** $e^{w'} > e^{w^*}$ **then** ▷ Compara o vencedor do duelo
6 $\mathbf{w}^* \leftarrow \mathbf{w}^* + \gamma e^{w'}\mathbf{v}$ ▷ Atualiza o melhor modelo corrente
7 **end if**
8 **end for**
9 **return** $\mathbf{w}^*$ ▷ Retorna o novo melhor vetor

As próximas seções descrevem cada componente do algoritmo Multi-DBGD, conforme a seguinte organização:

- A Seção 3.2.1 descreve a simulação da eficácia sobre os modelos candidatos $\mathbf{w}'$ reutilizando o *feedback* sobre uma mesma música observado a cada interação, conforme a linha 4 do Algoritmo 2.

- A Seção 3.2.2 descreve a instanciação do *feedback* a fim de medir a eficácia do melhor modelo corrente $\mathbf{w}^*$ controlada pela variável e^{w^*} e pela política de decaimento aplicada na linha 11 do Algoritmo 1.

- A Seção 3.2.3 descreve a sub-rotina Multi-DBGD-Initialize, responsável por retornar um modelo inicial baseado na primeira música da sessão.

- A Seção 3.2.4 descreve os parâmetros δ e γ respectivamente utilizados para definir a exploração dos modelos candidatos $\mathbf{w}'$ e o aproveitamento na atualização do melhor modelo corrente $\mathbf{w}^*$.

- A Seção 3.2.5 conclui a proposta do Multi-DBGD conforme cada componente e como são utilizados para comparação de múltiplos modelos, realizada a partir de m duelos entre o melhor modelo corrente $\mathbf{w}^*$ e um modelo candidato $\mathbf{w}'$, conforme Algoritmo 2.

3.2.1 Simulação da Eficácia

Conforme arcabouço apresentado nas seções anteriores, no cenário de recomendação *online* de músicas apenas uma música $s_{(t)}$ recebe *feedback* para cada interação com usuário. Nossa proposta é receber o *feedback* sobre a música selecionada $s_{(t)}$ a partir do melhor modelo corrente $\mathbf{w}^*$ para cada tempo t e simular uma avaliação da eficácia de um modelo $\mathbf{w}$ baseada no conjunto de músicas candidatas $\mathbf{X}$ e na música selecionada $s = s_{(t)}$. Dessa forma, definimos a função effectiveness-simulated$(\mathbf{w}, \mathbf{X}, s)$ para simular a eficácia de um modelo sobre a música selecionada s, conforme os seguintes passos:

- Uma lista $\mathbf{l} \leftarrow \langle \mathbf{w}, \mathbf{X} \rangle$ é definida a partir do produto escalar do modelo $\mathbf{w}$ (definido por um vetor de pesos de tamanho n) e o vetor de n características $\mathbf{x} \in \mathbf{X}$ representado cada música candidata.

- A lista $\mathbf{l}$ deve manter os índices de cada música candidata em $\mathbf{X}$, ordenados de forma descendente pela nota calculada a partir do produto escalar descrito anteriormente.

- O *Reciprocal Rank* (RR) é utilizado como medida de eficácia para os modelos e representa a posição de um item em uma lista, a saber:

$$RR = \frac{1}{i}, \tag{3.1}$$

  onde i é a posição da música selecionada s na lista ordenada $\mathbf{l}$. Por exemplo, o RR de um item na segunda posição é $RR = \frac{1}{2} = 0{,}5$.

Essa simulação não é utilizada para definir a eficácia do melhor modelo corrente $\mathbf{w}^*$, sendo essa controlada pela variável e^{w^*}, a saber na seguinte seção.

3.2.2 Eficácia do Melhor Modelo Corrente

No cenário de recomendação *online* de música, o *feedback* é observado exclusivamente sobre uma música $s_{(t)}$ selecionada pelo melhor modelo corrente $w^*_{(t)}$ para cada tempo t. Além disso, o *feedback* implícito permite observar apenas valores binários, ou seja, o

skip recebe *payoff* $= 0$ e o *play* recebe *payoff* $= 1$. Dessa forma, a fim de discretizar o *feedback* em uma avaliação para medir a eficácia do melhor modelo corrente $w^*_{(t)}$ a cada tempo t, propomos uma modelagem a partir da variável $e^{w^*}_{(t)}$, conforme as seguinte premissas:

- Assumimos o valor máximo para a eficácia ($e^{w^*} = 1.0$) na inicialização ($e^{w^*}_{(1)} = 1.0$) e quando um *feedback* implícito positivo (*play*, representado pelo *payoff* $= 1$) é observado.

- Ao observar um *feedback* implícito negativo (*skip*, representado pelo *payoff* $= 0$), simulamos uma avaliação da eficácia a partir de uma política de decaimento sobre a eficácia do melhor modelo corrente $e^{w^*}_{(t)}$. Nesse caso, o valor de $e^{w^*}_{(t)}$ é decrementado em uma posição a partir da inversão da eficácia atribuída pelo *reciprocal rank*, conforme proposto na seção anterior. Dessa forma, o decaimento pode ser calculado conforme a seguinte equação:

$$\text{decaying}(e^{w^*}_{(t)}) = (\frac{1}{e^{w^*}_{(t)}} + 1)^{-1}. \tag{3.2}$$

Por exemplo, seja $e^{w^*}_{(t)} = 1.0$ (simulando um item na primeira posição, ou seja, eficácia máxima), ao observar um *skip* o valor é atualizado para $e^{w^*}_{(t)} = 0.5$ simulando um item na segundo posição e assim por diante. Sendo assim, o *skip* penaliza o $e^{w^*}_{(t)}$ sequencialmente em uma posição da lista, a fim de comparar de forma similar a eficácia sobre o melhor modelo corrente e a eficácia simulada para os modelos candidatos.

3.2.3 Inicialização do Modelo

Conforme apresentado nas seções anteriores, cada música da coleção é representada por um conjunto de características. Dessa forma, o primeiro passo do Algoritmo 1 é a inicialização de um modelo linear representado por um vetor de pesos $\mathbf{w}^*_{(1)}$ utilizado para calibrar os características $\mathbf{x}$ que representam cada música em uma lista de músicas candidatas $\mathbf{X}$. Para essa tarefa, propomos duas diferentes abordagens de inicialização a random-initialize que é aleatória e a static-initialize que é estática.

O Algoritmo 3 define a rotina de inicialização proposta para o Multi-DBGD. Primeiramente, escolhemos uma das abordagens de inicialização. A abordagem aleatória inicializa o modelo $\mathbf{w}^*_{(1)}$ com os pesos a partir de um vetor unitário. Já a abordagem estática, considera o parâmetro γ como peso inicial para todos as características $\mathbf{w}^*_{(1)} = \{\gamma\}$.

Algoritmo 3 Sub-rotina para inicializar o modelo a partir da primeira música da sessão.

Multi-DBGD-Initialize$(u, m, \delta, \gamma, s_1^{\oplus})$

1 $\mathbf{w}^*_{(1)} \leftarrow$ random-initialize() **or** static-initialize(γ) ▷ Inicialização do modelo
2 $\mathbf{X} \leftarrow \Phi(u, 0)$ ▷ Φ Recupera as músicas candidatas
3 $e^{w^*}_{(1)} \leftarrow$ effectiveness-simulated($\mathbf{w^*}_{(1)}, \mathbf{X}, s_1^{\oplus}$) ▷ Simulação da eficácia
4 $\mathbf{w}^*_{(1)} \leftarrow$ Multi-DBGD-Update($\mathbf{X}, s_1^{\oplus}, \mathbf{w}^*_{(1)}, e^{w^*}_{(1)}, m, \gamma, \delta$)
5 **return** $\mathbf{w}^*_{(1)}$

Ainda sobre o Algoritmo 3, a primeira música fornecida pelo usuário $s_1^{\oplus}$ é utilizada para avaliar a eficácia do modelo inicial $\mathbf{w}^*_{(1)}$. Para isso, primeiramente a função Φ para recuperar uma matriz $\mathbf{X}$ de músicas candidatas representadas por um vetor $\mathbf{x}$ de n características, baseado no usuário u para o tempo zero (nesse caso a música $s_1^{\oplus}$ deve estar inclusa na lista $\mathbf{X}$ de candidatas). A partir desse conjunto, o algoritmo simula a eficácia conforme a função definida anteriormente na Seção 3.2.1 e o modelo inicial é atualizado conforme o Algoritmo 2.

3.2.4 Exploração e Aproveitamento

O conceito de exploração e aproveitamento (do inglês *exploration and exploitation*) é visto em diversas áreas. Explorar define uma etapa de busca por conhecimento novo ou novidade, por outro lado aproveitamento define a utilização do conhecimento corrente [March, 1991]. Em sistemas de recomendação, exploração é a ação de sugerir itens desconhecidos pelo usuário mas que são potencialmente relevantes. Por outro lado, aproveitamento é a ação de recomendar utilizando o conhecimento histórico sobre os itens preferidos pelo usuário [Ricci et al., 2015].

No algoritmo proposto, o parâmetro $\delta \leftarrow [0, ..., 1] \in \mathbb{R}^+$ define um fator para calibrar à distância de exploração dos vizinhos ao melhor modelo corrente, motivado pela proposta de Yue & Joachims [2009]. Dessa forma, quanto maior o valor de δ, maior é a distância entre os modelos candidatos $\mathbf{w}'$ e o melhor modelo corrente $\mathbf{w}^*$.

O parâmetro $\gamma \leftarrow [-1, ..., 1] \in \mathbb{R}$ é inicializado como positivo e define o grau de aproveitamento sobre o melhor modelo corrente $\mathbf{w}^*$ comparado aos modelos candidatos $\mathbf{w}'$ explorados com base no parâmetro δ. Quanto mais próximo o valor absoluto de γ está de um, maior é a variação introduzida sobre o melhor modelo corrente. No algoritmo proposto, o parâmetro γ também é utilizado para modelar a direção que o melhor modelo corrente é atualizado, a saber nos itens abaixo:

- Caso o *feedback* do usuário sobre a música selecionada pelo melhor modelo cor-

rente $\mathbf{w}^*$ seja positivo (*play*), o parâmetro mantém o seu valor absoluto, ou seja, $\gamma > 0$. Dessa forma, os modelos candidatos $\mathbf{w}'$ com maior eficácia em relação ao melhor modelo corrente $\mathbf{w}^*$, serão incorporados de forma positiva com um fator γ.

- Devido a observação do *feedback* sobre apenas uma música selecionada pelo melhor modelo corrente, é possível observar o *feedback* implícito negativo (*skip*) indicando abandono da música em execução e uma possível insatisfação do usuário. Introduzimos um conceito de inversão sobre o parâmetro γ, ou seja, o valor absoluto é convertido para negativo, onde $\gamma < 0$. O objetivo é induzir que o algoritmo utilize o *skip* considerando um conceito negativo sobre a música que recebeu o *feedback*. Portanto, ao atualizar o melhor modelo corrente $\mathbf{w}^*$, o algoritmo penaliza os modelos candidatos que recomendam, ou tem mais chance de recomendar, uma música irrelevante.

3.2.5 Múltiplos Duelos

Relembrando o cenário de recomendação *online* de músicas, a cada interação é possível observar o *feedback* apenas sobre uma música $s_{(t)}$, selecionada pelo melhor modelo corrente $w^*_{(t)}$ a cada tempo t, fato que torna inviável a adaptação dos algoritmos DBGD e do MGD apresentados como motivação desta abordagem (Seção 3.2). Sendo assim, esta seção descreve nossa proposta para reutilização do mesmo *feedback* para comparação em múltiplos duelos baseado nas seções anteriores.

Para cada duelo $i \leftarrow 1$ **to** m, é explorado um modelo candidato $\mathbf{w}' \leftarrow \mathbf{w}^* + \delta\mathbf{v}$ vizinho do melhor modelo corrente $\mathbf{w}^*$ conforme o parâmetro δ de exploração e de um vetor unitário $\mathbf{v}$. Baseado na música selecionada s, simulamos a avaliação da eficácia para o modelo candidato $\mathbf{w}'$ a fim de medir a eficácia conforme descrito nas seções anteriores (Seção 3.2.1). Caso a eficácia $e^{w'}$ de $\mathbf{w}'$ seja maior que a eficácia e^{w^*} do melhor modelo corrente $\mathbf{w}^*$, então $\mathbf{w}^* \leftarrow \mathbf{w}^* + \gamma r\mathbf{v}$ é atualizado, baseado no parâmetro γ de aproveitamento, na eficácia medida para o modelo candidato $e^{w'}$ e no vetor unitário $\mathbf{v}$.

3.3 Sumário

Neste capítulo apresentamos uma modelagem sobre o comportamento do usuário ao utilizar um serviço de *streaming* de música, com enfoque na tarefa de recomendar cada música da sessão e receber sequencialmente o *feedback* implícito a partir dos comandos

de *play* e *skip*. Destaque para o algoritmo Multi-DBGD, com vistas para as fontes de conhecimento que motivaram a proposta desse novo algoritmo, assim como para seus principais componentes, parâmetros e elementos funcionais descritos neste capítulo.

O seguinte capítulo apresenta as configurações experimentais realizadas nesta dissertação a fim avaliar a eficácia do Multi-DBGD. Ainda sobre os experimentos, são detalhados o conjunto de características utilizados na representação de cada música da coleção e os *baselines* utilizados na comparação dos resultados.

Capítulo 4

Metodologia Experimental

Neste capítulo são apresentados os detalhes sobre a metodologia experimental proposta para avaliação do algoritmo Multi-DBGD e os *baselines*. A Seção 4.1 descreve e caracteriza a base de dados utilizada nos experimentos. A Seção 4.2 descreve as características que serão utilizados para representar as músicas e as preferências dos usuários. A Seção 4.3 descreve as melhorias realizadas na base de dados. A Seção 4.4 descreve a configuração experimental sobre o modelo de treino, validação e teste. Por fim, a Seção 4.5 descreve o melhor algoritmo que encontramos na literatura com o objetivo de servir como *baseline*.

4.1 Caracterização da Base de Dados

A base de dados utilizada nos experimentos desta dissertação foi disponibilizada por Celma [2010] (Last.fm 1K).[1] A base de dados consiste de interações entre usuários e músicas ordenados temporalmente, contendo 992 usuários distintos observados no período de julho de 2005 a maio de 2009 na rede social para compartilhamento de músicas conhecida como Last.fm.[2] Conforme o funcionamento da rede social, os eventos são registrados a partir de mensagens chamadas de *scrobbles*,[3] enviadas ao sistema com as informações sobre a música que o usuário está escutando.

Na base de dados Last.fm 1K foram coletadas 19.150.868 interações contendo o histórico de músicas dos usuários (código do usuário, data e hora do evento, código do artista, nome do artista, código da música, título da música). Além disso, a base de dados é enriquecida com informações dos usuários (Sexo, Idade, País e Data de

[1] http://www.dtic.upf.edu/ ocelma/MusicRecommendationDataset/lastfm-1K.html
[2] http://www.last.fm/
[3] https://www.last.fm/about/trackmymusic

cadastro no sistema). A Tabela 4.1 descreve algumas estatísticas da base de dados completa.

Tabela 4.1. Estatísticas da base de dados Last.fm 1K.

Descrição	Estatísticas
Total de usuários	992
Total de músicas	1.084.865
Total de artistas	174.090
Total de interações	19.150.868
Total de sessões	1.254.262
Tamanho médio das sessões	15,00
Cobertura do enriquecimento	55,36%

4.2 Engenharia de Características

A noção de características (do inglês *features*) é um conceito comum em aprendizado de máquina e pode ser definido como vetores de valores que representam um item, ou seja, características interpretáveis por computadores. As características variam entre sistemas, cenários e até mesmo das necessidades de cada tarefa. O processo de definição de características é, normalmente, muito custoso devido na maioria das vezes exigir dedicação manual para definição, estruturação e validação dos valores para representar cada item. Esse processo é conhecido como engenharia de características e apesar de não ser um termo formalizado, compõe uma tarefa muito importante em trabalhos de aprendizado de máquina [Murphy, 2012].

Em sistemas de recomendação, qualquer modelo e/ou atributos são fontes para extração de características a serem utilizados na representação dos itens. Por exemplo: (*i*) informações colaborativas [Herlocker et al., 1999; Sarwar et al., 2001]; (*ii*) atributos de conteúdo [Pazzani & Billsus, 2007]; (*iii*) dados sociais [Basu et al., 1998]; entre outros. Voltando ao cenário de recomendação de músicas, é possível extrair características a partir dos atributos do áudio [Annesi et al., 2007; King & Imbrasaitė, 2015].

O conjunto de características tem como objetivo representar o conhecimento sobre o contexto, observações sobre o comportamento das transições entre os itens e sobre as preferências dos usuários. Esse conjunto é constituído por cinco grupos, descritos a seguir:

1. **Colaborativas:** A recomendação a partir da colaboração entre os usuários é baseada no conceito de "sabedoria das multidões", buscando conhecimento a partir

do que a maioria dos usuários apresentam interesse [Herlocker et al., 1999]. Para garantir a utilização do *feedback* implícito, utilizamos um processo de transformação sobre os dados binários (um para as músicas aceitas e zero para as músicas não aceitas pelos usuários) e treinamento de um modelo de recomendação seguindo a proposta de Volkovs & Wei Yu [2015].

Primeiramente, é construída uma matriz binária $R = U \times V$, onde U representa os usuários e V os itens (sejam eles músicas ou artistas). Dessa forma, aplicamos uma transformação sobre dados binários, a saber:

$$\textbf{item-item: } S(u,v) = \sum_{v' \in V(u)} R(:,v)^T R(:,v'), \tag{4.1}$$

onde v representa o item alvo e $S(u,v)$ a avaliação ao item v para o usuário u. Utilizamos o modelo de vizinhança entre itens, baseado nos resultados obtidos por Volkovs & Wei Yu [2015]. Nessa transformação, quanto mais frequente o usuário fornece *feedback* sobre um item, mais chances desse item ser relevante para o usuário.

Após o processo de transformação dos dados ser realizado para todos os usuários e itens, a matriz é normalizada conforme equação que se segue:

$$S(:,v) = \frac{S(:,v)}{\sqrt{\sum_{u \in U} S(u,v)^2}}. \tag{4.2}$$

Ainda sobre a proposta de Volkovs & Wei Yu [2015], a matriz normalizada S é dada como entrada para o método de Decomposição em Valores Singulares Truncado (ou em inglês Truncated Singular Value Decomposition) (SVD truncado). O SVD é uma técnica para fatoração de matrizes capaz de reconstruir a matriz original S a partir do produto de três matrizes menores. A extensão truncada do SVD propõe um tamanho limite de k fatores latentes para as três matrizes intermediárias. A reconstrução da matriz S é realizada conforme a seguinte equação:

$$S = U_k \Sigma_k V_k^T, \tag{4.3}$$

onde U_k é a matriz de k fatores latentes representando cada usuário, Σ_k define uma matriz diagonal de pesos com tamanho $k \times k$ e V_k^T a matriz transposta representando cada item por k fatores latentes. Propomos para esse modelo a utilização de $k = 100$.

2. **Sociais**: Este grupo é observado a partir de dados de colaboração social entre

os usuários, assim como a atribuição de *tags*. As *tags* são palavras, ou um conjunto de poucas palavras, atribuídas por usuários de alguma plataforma (normalmente na internet) para expressar informações que representem um determinado item [Eck et al., 2008]. Sobre esse conteúdo utilizamos a técnica de modelo de espaço vetorial [Salton et al., 1975], a fim de representar os conteúdos textuais em vetores de frequência conforme a ocorrência das palavras nas *tags*.

3. **Conteúdo**: Para cada música, as informações textuais (como o título da música ou o nome do artista) são concatenadas, representando cada música por conjunto de palavras. Assim como as características sociais, para processar o conteúdo textual das músicas utilizamos o modelo de espaço vetorial [Salton et al., 1975], transformando a representação em um vetor de frequências.

4. **Áudio**: As características de áudio compõe parte do enriquecimento proposto para a base de dados utilizando informações fornecidas pela Spotify. Nesse caso, foram utilizados todos os atributos de áudio[4] disponibilizados pela API da empresa, assim como a intensidade ou instrumentalidade de uma música.

5. **Histórico**: Por fim, as características deste grupo são obtidos a partir de informações históricas, sendo eles: (i) a predição de avaliação do usuário sobre uma música; (ii) a predição de avaliação do usuário sobre um artista; (iii) similaridade das representações entre as músicas da coleção e a música preferida de cada usuário, predita pelo SVD; (iv) similaridade das representações entre os artistas da coleção e o artista preferido de cada usuário, predito pelo SVD; (v) popularidade global da música; (vi) novidade da música personalizada para cada usuário.

   As quatro primeiras características históricas (i até iv) produzem um valor utilizando as predições do SVD e suas representações. A característica de novidade (vi) é calculada conforme proposta de Wang et al. [2014], a saber:

$$U_n = 1 - e^{-t/s}, \tag{4.4}$$

   onde t representa quanto tempo o usuário interagiu com cada música pela última vez e s o fator de suavização conforme recomendado pelos autores. Além disso, este é o único atributo do grupo que exige atualização a cada interação, devido a dependência sobre o tempo.

Com o objetivo de modelar coesão dentro das sessões de cada usuário, propomos uma transformação para as características dos grupos de (1) a (4). Essas características

[4]Disponível em: https://developer.spotify.com/web-api/get-several-audio-features/

são calculadas sobre dois níveis, música e artista. Os itens presentes em cada nível são representados por vetores de valores, sendo o nível de artista uma média sobre as características de suas músicas. Sobre cada nível, propomos o cálculo de similaridade entre as músicas candidatas em relação ao primeiro item (música ou artista) da sessão de cada usuário e ao último item escutado na sessão corrente. Dessa forma, esse conjunto de características sensíveis à sessão corrente do usuário (usando o cálculo da similaridade) possui 16 valores. No total cada música é representada por um vetor contendo 22 características.

Algumas características utilizam do cálculo da similaridade para obter um valor de relação entre dois vetores de atributos. Em todos os casos, utilizamos a medida similaridade do cosseno [Baeza-Yates & Ribeiro-Neto, 2011], a saber:

$$\text{similaridade}(\mathbf{x}, \mathbf{y}) = \text{cosseno}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||||\mathbf{y}||} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \mathbf{y}_i}{\sqrt{\sum_{i=1}^{n} {\mathbf{x}_i}^2} \sqrt{\sum_{i=1}^{n} {\mathbf{y}_i}^2}}. \quad (4.5)$$

Parte dessas características foram motivadas por Knees & Schedl [2015], trabalho que introduz conhecimentos gerais sobre a área de recuperação e recomendação de músicas. Por fim, a Tabela 4.2 resume as características de cada grupo que representam os itens nos níveis de música e artista (exceto para o grupo de características históricas, cujo valores estão discretizados). Na tabela, cos-SVD representa o cálculo da similaridade entre os vetores de fatores latentes calculados pelo SVD.

Tabela 4.2. Resumo das características separados em grupos.

Grupo	Descrição	Total
Colaborativos	cos-SVD relação ao primeiro e último item escutado	4
Sociais	Similaridade das *tags* dos itens	4
Conteúdo	Similaridade de conteúdos textual dos itens	4
Áudio	Similaridade dos atributos de áudio dos itens	4
Histórico	Predição do SVD para a música alvo	1
	Predição do SVD para o artista alvo	1
	cos-SVD em relação a música preferida do usuário	1
	cos-SVD em relação ao artista preferido do usuário	1
	Popularidade da música na base de dados	1
	Novidade: Intervalo entre as interações usuário e item	1

A base de dados original não possui todas as informações necessárias para extração das características descritos nesta seção. Tendo em vista ampliar as fontes de informação, a seguinte seção apresenta detalhes sobre a proposta de enriquecimento na base de dados.

4.3 Enriquecimento da Base de Dados

A fim de suportar a criação de características sobre diversas fontes, propomos o enriquecimento da base de dados Last.fm 1K utilizando dados externos. Foram consideradas duas importantes fontes na literatura para processamento de músicas.

Primeiramente, a base de dados Last.fm 1K possui um identificador único para as músicas e artistas que está relacionado com a enciclopédia de músicas pública conhecida como Music Brainz[5] de Swartz [2002]. A Music Brainz é uma coleção de meta-dados sobre música, ou seja, informações de conteúdo. Essas informações permitem extrair representações de conteúdo para as músicas utilizando, por exemplo, o título da música e do álbum, o nome do artista, entre outros. Além disso, a Music Brainz fornece dados de colaboração social, conforme os usuários da plataforma *online* atribuem *tags* (palavra, ou pequeno conjunto de palavras) que representam as várias entidades presentes na base, assim como as músicas, artistas, álbuns, entre outras, ambos não presentes na base original.

Segundo, foram utilizados dados públicos coletados da API disponibilizada pela Spotify.[6] O processo conduzido para recuperação dos dados foi separado em quatro etapas, sendo elas: (*i*) todos os títulos e artistas únicos contidas na base de dados Last.fm 1K foram coletados utilizando o serviço da API para busca de itens;[7] (*ii*) após a coleta, as músicas e os artistas da base de dados disponíveis na API foram mapeadas para os novos identificadores; (*iii*) utilizando os novos identificadores, foram coletadas mais informações utilizando do serviço de busca de artistas;[8] (*iv*) por fim, utilizando os novos identificadores das músicas, foram coletados os atributos de áudio fornecidos pela API.[9]

As melhorias propostas permitem uma cobertura de 60% do total de músicas da base de dados original Last.fm 1K. Destacamos que somente essas músicas são utilizadas para execução dos testes, com o objetivo de permitir a extração das características propostas para todas as músicas, a fim de garantir que os testes sejam igualmente executados.

[5] http://www.musicbrainz.org/
[6] https://developer.spotify.com/
[7] https://developer.spotify.com/web-api/console/get-search-item/
[8] https://developer.spotify.com/web-api/console/get-several-artists/
[9] https://developer.spotify.com/web-api/console/get-audio-features-several-tracks/

4.4 Configuração Experimental

De modo geral, o objetivo dos experimentos é garantir confiança sobre os resultados, espelhando o mais próximo de um ambiente de utilização real e não controlado. Entre as perspectivas como os experimentos *online*, estudo de usuário e experimentos *offline*, nesta dissertação propomos a execução de experimentos *offline* usando dados históricos, devido à facilidade de acesso as informações e aumento da reprodutibilidade.

Experimentos usando dados históricos não possuem limitações de execução, assim como as encontradas nos experimentos *online*, devido a exposição de usuários reais, acesso aos sistemas, tempo para coletar os resultados, entre outros fatores. Dessa forma, os testes podem ser executados em grandes massas de dados de forma rápida e sem exposição de usuários reais. Porém, o uso de dados históricos introduz outros problemas, como o viés e o ruído sobre o histórico coletado. Um exemplo de viés é chamado de observação parcial e refere-se ao fato de que apenas uma porção dos dados possuem rótulos coletados para identificá-los na tarefa, já que os usuários interagem somente sobre um conjunto reduzido de dados. O ruído pode ser observado quando a informação coletada é fruto de uma observação falha, por exemplo, quando um usuário executa involuntariamente um comando de *skip* em uma música que desejava escutar. Esses problemas produzem desafios para a adequação dos experimentos e para estimar métricas *online* sobre dados históricos [Hofmann, 2013].

Ainda sobre os experimentos *offline* usando dados históricos, podemos observar duas metodologias principais, sendo elas:

1. A primeira metodologia tem como objetivo inferir um modelo de comportamento dos usuários sobre o sistema, a partir de dados históricos. Em recuperação da informação por exemplo, existem vários modelos capazes de simular o comportamento dos usuários mesmo sobre resultados que não tenham sido apresentados na coleta. O trabalho de Chuklin et al. [2015] apresenta uma visão geral sobre diversos modelos existentes em recuperação da informação. No entanto, esses modelos não são adaptáveis ao cenário de recomendação *online* de músicas devido a maioria utilizarem variáveis específicas de recuperação da informação, assim como a ordem dos documentos clicados em uma lista (fato que não ocorre na recomendação *online* de músicas, já que apenas um item é consumido a cada interação).

2. A segunda metodologia intervém sobre os itens que são apresentados durante a coleta. Em sistemas de recomendação Li et al. [2011] apresentam uma abordagem capaz de executar experimentos *offline* garantindo resultados potencialmente

comparáveis aos resultados encontrados sobre experimentos *online*. Essa abordagem além de ser independente de simulação é direcionada aos dados e foi utilizada na avaliação de recomendações de notícias. Nesse caso, com o objetivo de mitigar o viés observado sobre os dados, Li et al. [2011] propõem o controle sobre a coleta dos dados, a fim de garantir que os itens apresentados aos usuários sejam aleatoriamente ordenados. Porém, essa característica dificilmente é observada em bases de dados publicamente disponíveis.

Dessa forma, propomos uma adaptação dessas duas metodologias seguindo algumas premissas, a saber: (*i*) selecionamos o melhor algoritmo apresentado por Li et al. [2011] com o objetivo de servir como *baseline*, já que em ambos os casos, somente um item pode ser consumido a cada interação e o comportamento dos usuários é influenciado pelo contexto; (*ii*) a consistência temporal presente nos dados históricos é conservada, a fim de manter a coesão sequencial conforme as sessões musicais são consumidas pelos usuários.

4.4.1 Observação do *Feedback*

Segundo Yang et al. [2012], o *feedback* fornece sinais sobre as preferências dos usuários na forma implícita ou explícita. Em serviços de *streaming* de música, esses sinais podem ser extraídos de diversas formas, como por exemplo sobre as interações dos usuários com os comandos de *play* e *skip*. Nesta dissertação, propomos a utilização do *feedback* implícito separados em três tipos, observados na base de dados conforme abaixo:

- O *play* explícito, ocorre quando o usuário utiliza o comando de *play* com a intenção de executar uma música específica. Em nossa tarefa por exemplo, consideramos que a primeira música da sessão do usuário é conhecida a priori, ou seja, um *play* explícito. Nesse caso, simulamos a ação de um usuário ao fornecer uma música semente para geração automática de uma *playlist*.

- O *play* de conclusão, ocorre quando o usuário escuta uma música sem executar o comando de *skip*. Consideramos um *play* de conclusão (ou uma música aceita), as interações na base de dados que possuem um intervalo de execução até a próxima música de pelo menos 50% do tempo da música [Bosteels et al., 2009; Yang et al., 2012].

- O *skip* explícito, ocorre quando o usuário utiliza o comando de *skip* para abandonar uma música em execução. Nesse caso, ao contrário do *play* de conclusão,

toda interação em que a música não é escutada completamente é considerada como *skip* explícito, ou seja, uma música não aceita [Yang et al., 2012].

Essa modelagem é utilizada para recuperar sobre o histórico dos usuários as interações representados com *play* e *skip* (músicas aceitas e não aceitas). Esse *feedback* observado para cada interação é utilizado como fonte de informação para extração das características e para filtrar as músicas utilizadas no procedimento de teste.

4.4.2 Organização da Base de Dados

A base de dados Last.fm 1K possui o histórico de 992 usuários contendo eventos organizados temporalmente e coletados durante os anos de 2005 e 2009. Cada evento representa uma interação entre um usuário e uma música durante milhares de sessões. Para execução do processo de treino, teste e validação, propomos a divisão desses eventos em oito partições igualmente espaçadas no tempo em intervalos seis meses, a fim de observar o comportamento sobre períodos distintos em diferentes estados da base e manter a sequência real das sessões. Esse processo de particionamento dos dados utilizados para treino, teste e validação é um modelo comum no processo experimental de sistemas de recomendação [Cremonesi et al., 2010; Hurley & Zhang, 2011].

Um dos métodos experimentais mais usados propõe a variação das partições para treino e teste e é chamado de validação cruzada (do inglês *cross-fold validation*) [Kohavi, 1995]. Na validação cruzada, os experimentos são executados repetidamente sobre múltiplas combinações entre as partições utilizadas no treino e teste. A execução repetida dos experimentos tem como objetivo estimar a eficácia a partir da média dos resultados sobre as varias combinações de execuções usando diferentes estados da base.

Entretanto, na recomendação *online* de músicas, os dados possuem uma ordem temporal, ou seja, os eventos estão sequencialmente organizados. Dessa forma propomos a execução dos experimentos sobre uma janela flutuante combinando duas partições sequenciais, sendo uma partição utilizada no treinamento dos modelos e outra para validação e teste. Esse processo foi inspirado nos trabalhos de Matsubara et al. [2012], Ahmed et al. [2013] e [Cheng et al., 2016a], com objetivo garantir a coerência dos dados respeitando a ordem natural com que foram observados no cenário real.

Esse particionamento sequencial permitiu observar algumas anomalias encontradas na base de dados. Com o objetivo de reduzir o ruído devido essas anomalias, consideramos apenas partições com pelo menos 500 usuários distintos. A Figura 4.1 apresenta em um gráfico a quantidade de usuários e o número médio de interações por usuário para cada partição referenciada pela tupla ano-semestre.

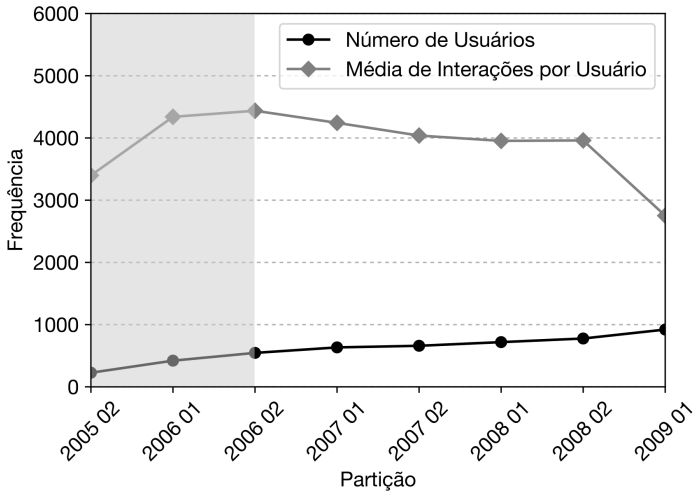

Figura 4.1. Gráfico comparando o número de usuários e a média de interações por usuário para cada partição. A área cinza representa os dados descartados a incluir a última partição.

Ainda sobre a Figura 4.1, a fim de reduzir possíveis impactos na inicialização da coleta, as duas primeiras partições foram descartadas, devido a quantidade de usuários ser menor que 500, ou seja, pelo menos 50% da quantidade total de usuários. Além disso, é possível observar na última partição que mesmo com um número de usuários superior comparado às demais partições, a média de interações é drasticamente inferior e por isso essa partição também foi descartada. Por fim, as cinco partições restantes, devido o comportamento mais estável, são utilizadas nesta dissertação para execução dos experimentos.

A Figura 4.2 apresenta uma divisão dos dados utilizados nas configurações dos experimentos sobre o modelo de treino, teste e validação, sendo nesta dissertação considerados dois tipos de testes, *online* e *offline* que serão descritos nas próximas seções. Na figura, cada caixa na parte superior da imagem representa uma partição em sequência referenciada pela tupla ano-semestre. Dentro de cada partição, os usuários são representados por uma lista de interações (músicas observadas a partir do histórico de cada usuário). Esses usuários são separados em validação e teste onde, dentre os usuários para teste, as interações são sub divididas entre teste *online* e teste *offline*, sendo 80% para o teste *online* e 20% para o teste *offline*. Além disso, para garantir o processamento de todas as características de forma similar, os testes são executados a partir da segunda partição, sendo a primeira utilizada para o estimar as características, processo descrito na próxima seção.

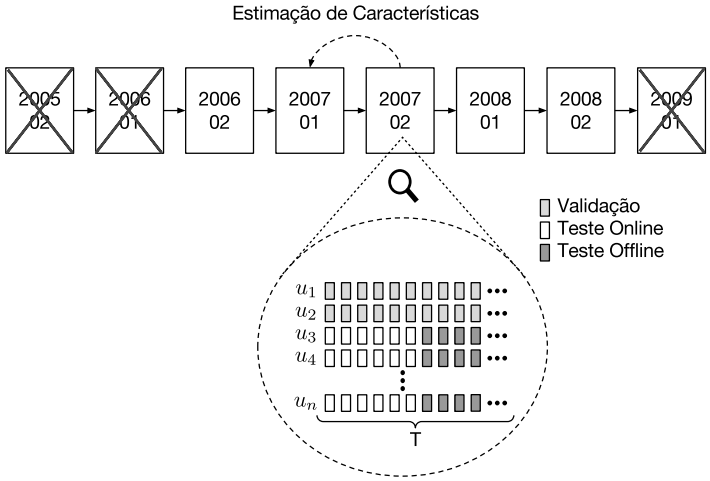

Figura 4.2. Particionamento da base de dados em treino, validação e teste.

4.4.3 Estimação de Características

Algumas características dependem de um conjunto de dados históricos para serem calculadas, por exemplo, a partir do histórico dos usuários. Chamamos esse processo de estimação de características. Nesse caso, todas as interações de cada partição são utilizadas, sendo as tratativas para cada tipo de evento (seja *play* ou *skip*) particulares de cada modelo utilizado para estimar as características.

Ainda sobre a Figura 4.2, esse processo é apresentado no topo da imagem. Com o objetivo de padronizar a extração para cada teste, essa estimação é realizada sobre as mesmas partições que estão temporalmente balanceadas. Sendo assim, as características são estimadas a partir de todos os dados da partição imediatamente anterior a partição de teste, ou seja, seis meses passados, conforme indicado pela seta curva no topo da figura. Nos casos onde os modelos exigem calibração parâmetros foram utilizados 10% dos usuários da mesma partição para validação da melhor configuração do modelo *offline*.

4.4.4 Configuração dos Testes

Uma alternativa para avaliação na tarefa de geração dinâmica de *playlist* é o trabalho de Bosteels et al. [2009]. Os autores propõem uma nova abordagem de avaliação baseada nos padrões dos usuários. Nesse trabalho, as sessões extraídas do histórico de músicas dos usuários contém até 22 músicas, onde das duas últimas somente uma música foi aceita. A ideia principal é avaliar a eficácia das heurísticas em incluir músicas sobre *playlists* geradas dinamicamente. A eficácia é medida a partir da capacidade dessas heurísticas em selecionar qual das duas últimas músicas foi aceita com base nas

20 músicas anteriores.

Apesar de também propormos a geração dinâmica de *playlists* a partir da recomendação de cada música que compõe a sessão do usuário, o objetivo desta dissertação é construir uma sessão interativa utilizando aprendizado *online* e o *feedback* contínuo do usuário a fim de aprimorar o modelo a cada interação. Dessa forma, a proposta de Bosteels et al. [2009] não se adéqua aos experimentos necessários para avaliar o algoritmo proposto, já que somente duas músicas de cada sessão são utilizadas no teste.

Ainda sobre a Figura 4.2, a simulação proposta para execução dos testes é formalizada conforme a seguir:

1. Cada partição selecionada para teste possui um conjunto $\mathbf{U}$ de usuários. Nesse caso, cada usuário $u \in \mathbf{U}$ é representado por uma lista de interações $\mathbf{R}^u$ contendo músicas escutadas sequencialmente.

2. As interações observadas na lista $\mathbf{R}^u$ são extraídas do histórico de cada usuário. Nesse processo são consideradas apenas as músicas aceitas no histórico dos usuários (escutadas por pelo menos 50% do tempo, assim como apresentado por Bosteels et al. [2009]), com o objetivo de selecionar uma sessão mais consistente em relação à intenção do usuário.

3. Para cada usuário, as músicas selecionadas sequencialmente são limitadas a um acumulado de sessões contendo até 500 músicas relevantes $\mathbf{R}^u$. Dessa forma, para cada tempo $t \leftarrow 1$ **to** 500, existe uma música relevante $r_t^u \in \mathbf{R}^u$. Além disso, a primeira música r_1^u dessa sessão é conhecida a priori, a fim de modelar o comportamento de um usuário ao fornecer uma música semente para geração automática de uma *playlist*.

   A seleção de 500 músicas por usuário permite recuperar em média 30 sessões por usuário. Dentro dessas 30 sessões, potencialmente ocorrem mudanças no comportamento do usuário devido à recuperação de músicas escutadas ao longo de várias sessões no tempo.

4. Para cada música relevante r_t^u, propomos a seleção de um conjunto de músicas aleatórias para compor o conjunto de irrelevantes para cada tempo t. Assim como em Cremonesi et al. [2010], a tarefa passa a ter como objetivo ordenar a música relevante no topo dessa lista de músicas candidatas. Entretanto, em vez de selecionarmos as músicas irrelevantes tendo como base todas as músicas desconhecidas da coleção seguindo a proposta de Cremonesi et al. [2010], propomos um novo conjunto menor e potencialmente mais relevante. A fim aumentar a

complexidade sobre a tarefa, para cada usuário $u \in \mathbf{U}$ é gerada uma amostra $\mathbf{A}^u$ utilizando o SVD, contendo 1000 músicas mais similares à primeira música da sessão r_1^u.

5. Para cada música relevante r_t^u no tempo t, um conjunto $\mathbf{A}_t^u$ contendo 99 músicas é recuperado aleatoriamente de $\mathbf{A}^u$ e adicionado aos candidatos em $\mathbf{M}_t^u$ junto a música relevante r_t^u. Ou seja:

$$\mathbf{M}_t^u \leftarrow r_t^u \| \mathbf{A}_t^u e \tag{4.6}$$

$$|\mathbf{M}_t^u| = 100, \tag{4.7}$$

visando transformar em um problema de ordenação.

6. Sequencialmente para cada tempo t, o algoritmo deve selecionar uma a única música s_t^u entre as candidatas na lista $\mathbf{M}_t^u$, compondo uma sessão $\mathbf{S} = \{s_1^u, s_2^u, ..., s_n^u\}$ de n itens. Para cada s_t^u é atribuído um *payoff* $= 0$ (simulando um *feedback* implícito negativo ou *skip*) caso seja selecionada uma música irrelevante ($s_t^u \neq r_t^u$), ou *payoff* $= 1$ (simulando um *feedback* implícito positivo ou *play*) caso a música selecionada seja a relevante ($s_t^u = r_t^u$). Em ambos os casos, os testes seguem para a próxima música até o final da sessão de teste.

Por fim, as configurações apresentadas nesta seção foram motivadas pelos trabalhos de Cremonesi et al. [2010] e Li et al. [2011]. Cada interação com o usuário é representada como um caso de teste onde a tarefa é selecionar uma música relevante dentre 99 músicas irrelevantes para cada tempo t. Baseado nos conceitos aqui apresentados, as seguintes seções descrevem as etapas do modelo experimental de validação e testes (*online* e *offline*).

Validação

Em aprendizado de máquina, alguns algoritmos possuem hiper parâmetros utilizados durante o aprendizado com o objetivo de calibrar a solução conforme observações sobre o cenário e os dados. Dessa forma, os hiper parâmetros influenciam diretamente na eficácia do algoritmo para uma determinada tarefa. Sendo assim, o processo de validação tem como objetivo buscar as melhores configurações dos hiper parâmetros para execução dos algoritmos sobre um cenário específico. Normalmente, esse processo ocorre sobre as mesmas configurações dos testes, porém utilizando conjuntos de dados distintos [Murphy, 2012].

Nossa proposta de validação é aplicada separadamente para cada partição de teste. Dentro de cada partição, existem usuários representados por uma sequência de músicas (extraídas dos dados históricos) dedicada para teste dos algoritmos. Desse conjunto de usuários, propomos selecionar 10% exclusivamente para tarefa de validação, ou seja, 10% dos usuários são excluídos dos testes. Sobre esse conjunto de usuários exclusivos é realizado o processo validação conhecido como busca em grade (do inglês *grid search*) [Bergstra et al., 2011], conforme procedimento a seguir:

- Um conjunto finito de valores possíveis é definido para cada parâmetro do algoritmo.
- Todas as combinações de valores possíveis são avaliadas sobre o mesmo conjunto de usuários separados exclusivamente para o processo de validação. Nesse caso, é usado o mesmo processo de teste *online* apresentado na próxima seção, a fim de avaliar a eficácia do algoritmo nessa amostra de usuários.
- A melhor combinação dos hiper parâmetros é selecionada a fim de prosseguir com a execução dos testes.

O conjunto finito de valores escolhido para execução do algoritmo Multi-DBGD e os *baselines* estão no apêndice desta dissertação na Tabela A.1.

Teste *Online*

O teste *online* tem como objetivo avaliar a capacidade dos algoritmos em recuperar itens relevantes conforme o *feedback* dos usuários são recebidos. Ou seja, como os modelos se adéquam às necessidades dos usuários utilizando a observação do *feedback* [Hofmann, 2013]. Nesta dissertação, propomos a execução do teste desta categoria em uma simulação sobre o comportamento dos usuários a partir de dados históricos. Os dados estão separados em partições igualmente espaçadas no tempo, contendo seis meses de dados cada uma.

Os usuários de cada partição são representados por uma sequência de músicas aceitas a partir de dados históricos. Dessa forma, propomos medir a qualidade dos modelos aprendidos sobre a tarefa de predizer o item relevante a cada tempo t. A Figura 4.3 apresenta o diagrama de execução do teste *online* para um usuário de exemplo. Na figura, para cada tempo t existe uma lista de músicas candidatas $\mathbf{M}_t^u$, onde apenas uma música é relevante. O modelo aprendido no tempo t é utilizado para selecionar a próxima música da sequência no tempo t_{+1}, essa seleção está representada na figura pelas setas que ligam um modelo a uma música. A música selecionada recebe

feedback, que é utilizado para aprimorar o modelo no tempo t_{+1}. No exemplo da figura, considerando o *feedback* implícito onde o *skip* é representado por *payoff* $= 0$ e o *play payoff* $= 1$, o *payoff online* acumulado do algoritmo para esse usuário é três, pois possui três acertos.

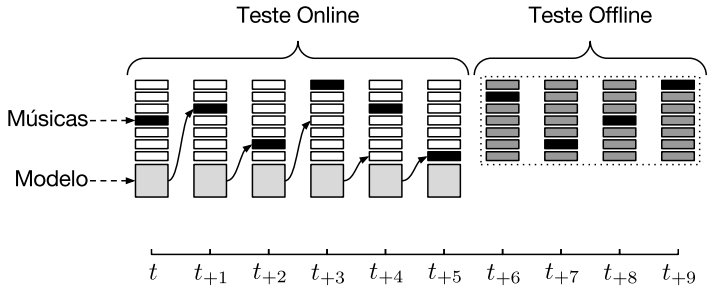

Figura 4.3. Teste *online* para a sessão de um usuário.

A fim de medir a eficácia *online* dos algoritmos, são utilizados todos os usuários presentes em cada partição, exceto aqueles utilizados no processo de validação. Além disso, parte dos casos de testes são separados para o teste *offline*, descrito na seguinte seção.

Teste *Offline*

Em paralelo ao teste *online* e apesar de também acompanhar a mesma modelagem, o teste *offline* tem como objetivo medir a taxa de aprendizado dos algoritmos conforme mais *feedback* pode ser observado pelo algoritmo [Hofmann, 2013]. Ou seja, o teste *offline* mede a evolução dos modelos aprendidos para cada interação com o usuário. No cenário de recomendação *online* de músicas, a qualidade e velocidade do aprendizado é avaliada no decorrer de uma mesma sessão, a cada *feedback* observado.

Para execução do teste *offline* em cada partição, são selecionadas 100 últimas músicas presentes na sessão de cada usuário utilizado para o teste *online*. Esse conjunto de músicas é utilizado exclusivamente para execução do teste *offline*. A Figura 4.4 apresenta o diagrama de execução dos teste *offline* para um usuário de exemplo. Para cada tempo t, existe uma lista de músicas candidatas $\mathbf{M}_t^u$, onde apenas uma música é relevante r_t^u, tanto para o conjunto de teste *online*, quanto no teste *offline*.

Ainda sobre a Figura 4.4, o modelo aprendido no teste *online* é fixado a cada tempo t e utilizado para selecionar a sequência de músicas do conjunto de teste *offline*. Esse processo de avaliação se repete a cada modelo aprendido durante o teste *online*. No exemplo da figura, o *payoff offline* acumulado do algoritmo para esse usuário sobre o modelo aprendido no tempo t é um. As músicas selecionadas no teste offline por

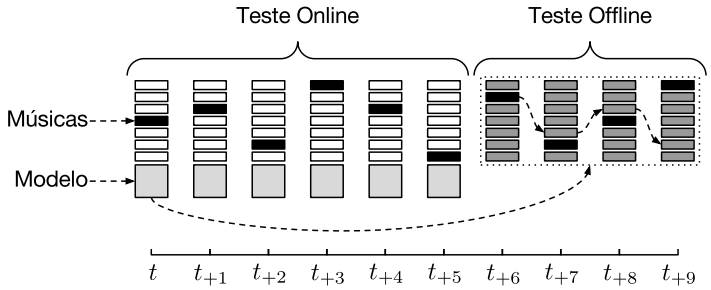

Figura 4.4. Teste *offline* para a sessão de um usuário.

cada modelo, estão representados na figura pelas setas ligando a sequência de músicas de cada caso de teste.

4.4.5 Métricas de Avaliação

Conforme as configurações dos testes apresentadas nas seções anteriores, a presente seção tem como objetivo definir as métricas utilizadas para medir os resultados dos algoritmos. De modo geral, representamos o *feedback* sobre o conceito de *payoff*, podendo assumir $payoff = 0$ em caso de *skip* ou $payoff = 1$ em caso de *play*.

Com o objetivo de medir os resultados do teste *online*, a seguinte equação define o cálculo para o *payoff online* acumulado para cada usuário da partição até o tempo t:

$$PayoffOnline@\mathrm{t} = \sum_{i=1}^{t} \sigma^{i-1} payoff(u, i, s_i^u), \tag{4.8}$$

onde u representa o usuário e s_i^u a música selecionada pelo modelo aprendido entre os tempos i até t, sendo t o tempo corrente e i os tempos anteriores desde o primeiro item da sessão, sendo a equação uma adaptação do cálculo de recompensa acumulada [Hofmann, 2013].

A tarefa de recomendação *online* de músicas pode ser formulada sobre um tarefa de horizonte infinito, ou seja, as músicas são consumidas continuamente pelos usuários indefinidamente sobre várias sessões, ocorrendo em vários casos interrupções temporais repentinas. Porém, um problema em horizontes infinitos é chamado de aproveitamento infinitamente postergado (do inglês, *infinitely delayed splurge*): uma vez que há exemplos indefinidamente, o agente sempre explora, confiante de que há tempo suficiente para obter todo o conhecimento sobre o ambiente. Para solucionar esse problema, ainda sobre a Equação 4.8, propomos a utilização do fator de desconto $\sigma = 0.995$,[10]

[10]Utilizamos σ no lugar de γ para não confundir com o hiper parâmetro γ do algoritmo proposto neste trabalho e em Yue & Joachims [2009]

valor comum nos estudos de aprendizado por reforço [Hofmann, 2013].

A fim de medir os resultados do teste *offline*, a equação abaixo define o cálculo para o *payoff offline* acumulado utilizando cada modelo aprendido fixado no teste *online* para cada usuário sobre todo tempo t:

$$PayoffOffline@\mathrm{t} = \sum_{i=T_{offline}}^{T} payoff(u, i, s^u_{t,i}), \tag{4.9}$$

onde, $T_{offline}$ representa o índice (ou o tempo) para o primeiro item do conjunto de músicas exclusivas para execução do teste *offline*, T o último item da sessão do usuário, u representada o usuário e $s^u_{t,i}$ define a música selecionada pelo modelo aprendido em t para cada índice i do conjunto de teste *offline*.

Por fim, o *payoff* utilizado nas Equações 4.8 e 4.9 modela o *feedback* simulado, a saber:

$$payoff(u, t, s^u_t) = \begin{cases} 1, & \text{se } s^u_t = r^u_t \\ 0, & \text{caso contrário} \end{cases}, \tag{4.10}$$

onde r^u_t representa a música relevante para o usuário u no tempo t. Dessa forma, o $payoff = 1$ representa o *feedback* implícito positivo (*play*) sobre a música s^u_t selecionada na lista de candidatas no tempo t. Caso contrário, o $payoff = 0$ representa o *feedback* implícito negativo (*skip*).

4.5 *Baselines*

O melhor algoritmo que encontramos na literatura com o objetivo de servir como *baseline* para o Multi-DBGD é o LINUCB [Li et al., 2010]. O LINUCB, é um algoritmo para recomendação de notícias sensível ao contexto, onde a tarefa é selecionar uma notícia em um conjunto de candidatos para cada ponto no tempo. Li et al. [2010] propõem a extração de características a partir das informações das notícias e dos usuários, por exemplo, características sobre o histórico de leitura, localidade, entre outras fontes. Voltando ao cenário de recomendação *online* de músicas, o LINUCB é facilmente adaptável, porém no lugar de notícias, o algoritmo deverá selecionar cada música de forma sequencial utilizando as características extraídas das músicas e o *feedback*.

O LINUCB é um algoritmo baseado em algumas premissas. Primeiramente, assume-se que o *payoff* estimado é uma função linear sobre as características e algum coeficiente desconhecido para cada ponto no tempo. Além disso, utiliza o algoritmo baseado *Upper Confidence Bound* (UCB) (ou em português intervalo de confiança supe-

rior) [Lai & Robbins, 1985; Auer et al., 2002], a fim de calcular o *payoff* médio estimado a partir de cada interação com um intervalo de confiança. Para cada ponto no tempo o item com maior probabilidade entre os candidatos é selecionado para recomendação. Por fim, o LINUCB possui apenas um hiper parâmetro (α) utilizado para calibrar a estimativa sobre o intervalo de confiança.

É importante destacar que utilizamos duas configurações para o LINUCB, sendo elas: (i) LINUCB Por Usuário: O modelo é inicializado e executado exclusivamente para cada usuário de cada partição; (ii) LINUCB Geral: O modelo é inicializado apenas uma vez por partição e executado sobre todos os usuários de forma continua, armazenando o conhecimento sobre o que já foi aprendido com usuários anteriores, ou seja, um modelo não personalizado.

4.6 Sumário

Apresentamos neste capítulo detalhes sobre a configuração experimental separada sobre o modelo de treino, teste e validação. Primeiramente introduzimos a coleção de dados utilizada nos experimentos desta dissertação. Apresentamos as características extraídas para representação das músicas e preferências dos usuários, assim como as melhorias propostas para a base de dados a fim de garantir atributos de diversas fontes. Formalizamos a utilização dos dados nos experimentos e modelamos a simulação do *feedback* utilizado para medir os resultados a partir de métricas de eficácia. Por fim, apresentamos os *baselines* que serão submetidos sobre a mesma configuração experimental a fim de comparar os resultados com o algoritmo Multi-DBGD.

A próxima seção apresenta os resultados coletados a partir dessa configuração experimental, assim como mostra a eficácia do algoritmo proposto em relação aos *baselines*, a fim de responder cada pergunta de pesquisa desta dissertação.

Capítulo 5

Resultados Experimentais

O presente capítulo descreve os principais resultados alcançados pelo algoritmo Multi-DBGD. O objetivo é medir a eficácia do algoritmo Multi-DBGD em relação aos *baselines*, assim como estimar o funcionamento do algoritmo proposto em cenários reais fornecendo embasamento para as perguntas de pesquisa a serem respondidas nesta dissertação, a saber:

- Q1: O algoritmo proposto produz melhores resultados que o melhor algoritmo encontrado na literatura?
- Q2: O algoritmo proposto é capaz de aprender mais rápido que o melhor algoritmo encontrado na literatura?
- Q3: Como o modelo de inicialização do algoritmo proposto afeta sua eficácia?
- Q4: Qual impacto do número de duelos na eficácia do algoritmo proposto?

Conforme detalhado na Seção 4.5, o melhor algoritmo que encontramos na literatura com o objetivo de servir como *baseline* para comparação com o nosso algoritmo no cenário de recomendação *online* de música é o LINUCB [Li et al., 2010]. Propomos duas configurações diferentes para o LINUCB, a saber:

- LINUCB Por Usuário: O modelo é inicializado e executado exclusivamente para cada usuário de cada partição;
- LINUCB Geral: O modelo é inicializado apenas uma vez por partição e executado sobre todos os usuários de forma contínua, armazenando o conhecimento sobre o que já foi aprendido com usuários anteriores, ou seja, um modelo não personalizado.

A configuração experimental foi separada em treino, teste (*online* e *offline*) e validação. Os experimentos foram executados sobre a base de dados Last.fm 1K [Celma, 2010], a qual contém o histórico de 992 usuários observados sobre o período de julho de 2005 a maio de 2009. O histórico é composto por interações (eventos de *play* ou *skip* sobre as músicas) sequencialmente coletados da rede social Last.fm. Para execução dos testes os dados foram separados em partições igualmente espaçadas no tempo, com duração de seis meses. A fim de evitar possíveis ruídos sobre a etapa inicial e final coleta, as duas partições iniciais e a última partição foram descartadas. Sobre as quatro partições restantes, referenciadas pela tupla ano-semestre, cada usuário é representado uma sequência de casos de teste, sendo cada caso de teste composto por um item relevante e 99 irrelevantes, onde a tarefa é selecionar a música relevante para cada tempo t. O valores são medidos a partir do resultado das médias de acertos entre todos os usuários com intervalos de 99% de confiança conforme a tabela *t-student*.

O algoritmo Multi-DBGD utiliza aleatoriedade para buscar modelos candidatos a serem comparados com o melhor modelo corrente. Para evitar algum tipo de tendência o algoritmo proposto é executado seis vezes sobre cada partição de teste, totalizando 24 execuções. Dessa forma, para o Multi-DBGD os valores de média e o intervalo de confiança de cada partição são calculados sobre todas as execuções.

As seções deste capítulo estão organizadas como se segue. A Seção 5.1 apresenta os resultados da eficácia do algoritmo proposto em relação aos *baselines* utilizando o teste *online* a fim de responder a primeira pergunta de pesquisa desta dissertação (Q1). A Seção 5.2 apresenta a taxa de aprendizado do algoritmo proposto em relação aos *baselines* utilizando o teste *offline* a fim de responder a segunda pergunta de pesquisa desta dissertação (Q2). A Seção 5.3 compara a influência dos modelos de inicialização sobre a qualidade dos resultados do Multi-DBGD, a fim de responder a terceira pergunta de pesquisa desta dissertação (Q3). A Seção 5.4 compara a influência do parâmetro número de duelos sobre a qualidade dos resultados do Multi-DBGD, a fim de responder a última pergunta de pesquisa desta dissertação (Q4).

5.1 Eficácia do Aprendizado

A primeira pergunta de pesquisa (Q1) a ser respondida é se o algoritmo proposto produz melhores resultados que o melhor algoritmo encontrado na literatura. Para isso, utilizamos o teste *online* com objetivo de comparar a eficácia *online* do Multi-DBGD em relação aos *baselines*.

A Figura 5.1 apresenta o resultado acumulado médio para todos os usuários

separado em quatro partições. Para todo tempo t de cada partição, o Multi-DBGD possui resultados no mínimo igual aos *baselines*. De modo geral, quando $t \geq 150$ a diferença de eficácia *online* entre os modelos é significativa para todas as partições, sendo o Multi-DBGD superior em todos os casos. Para fins de comparação, o Oráculo define a eficácia de um suposto algoritmo capaz de acertar todas as músicas relevantes e representa o limite da métrica acumulada. O comportamento curvo observado em todos os algoritmos é em decorrência do decaimento da métrica, que reduz o valor para acertos distantes do início da sessão, ou seja, valoriza acertos com poucas interações com o usuário.

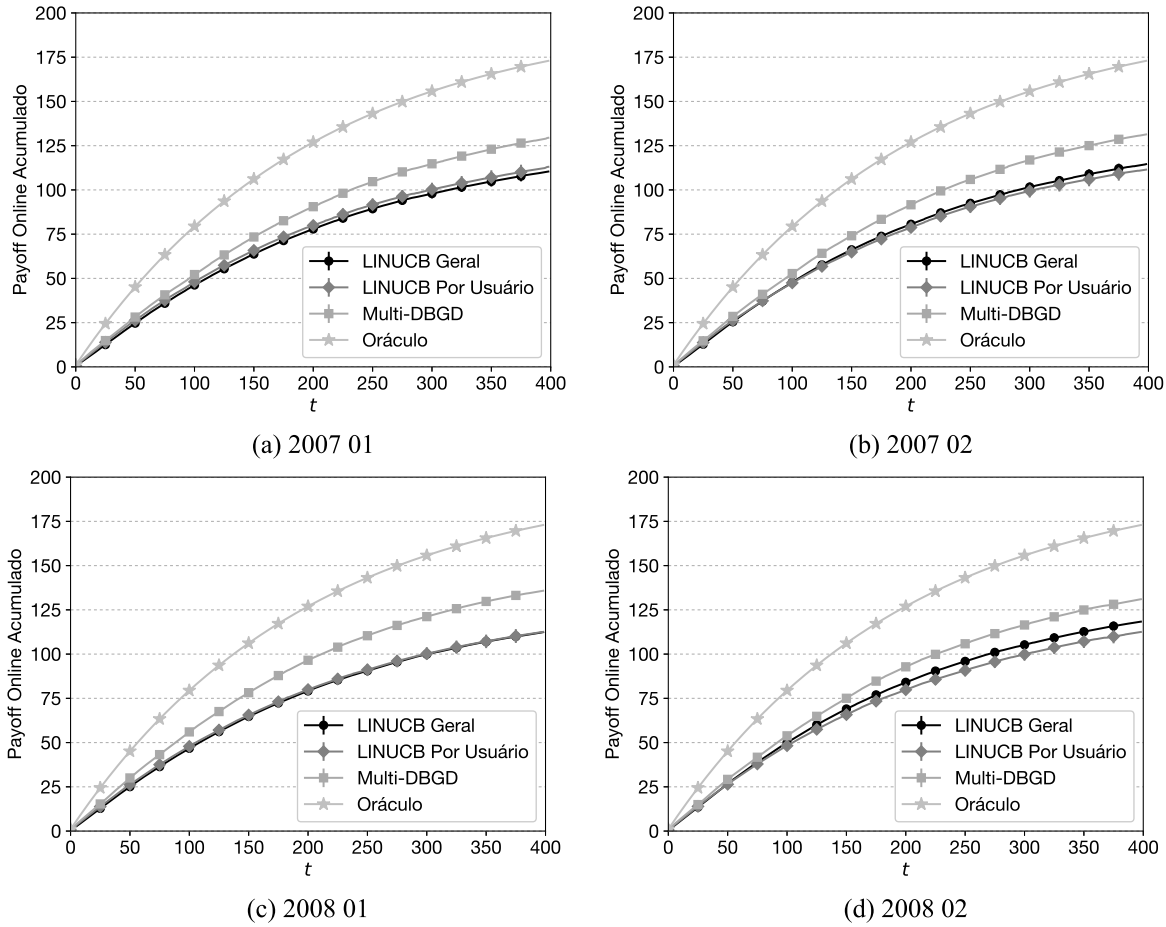

Figura 5.1. Resultados para o teste *online*.

A Tabela 5.1 apresenta o payoff médio para todos os usuários de cada partição e $t = 400$. Essa métrica se assemelha ao *Click Through Rating* (CTR) (ou em português, percentual de cliques certos) e mede quantidade média de acertos para todos os usuários em relação ao total de irrelevantes candidatos. Os resultados apresentados na Tabela 5.1 demonstram que a eficácia *online* média do Multi-DBGD é superior estatisticamente ($p < 0{,}01$) em relação aos *baselines* em todas as partições.

Tabela 5.1. Média final do *payoff* para teste *online* após $t = 400$ para todos os usuários de cada partição de teste. A significância está indicada pelo símbolo ▲ ($p < 0{,}01$) em relação a cada *baseline*.

Algoritmo	2007 01	2007 02	2008 01	2008 02
Multi-DBGD	$\mathbf{0{,}32}_{\pm 0{,}01}$▲▲	$\mathbf{0{,}33}_{\pm 0{,}01}$▲▲	$\mathbf{0{,}34}_{\pm 0{,}02}$▲▲	$\mathbf{0{,}33}_{\pm 0{,}01}$▲▲
LINUCB Geral	$0{,}28_{\pm 0{,}01}$	$0{,}29_{\pm 0{,}01}$	$0{,}28_{\pm 0{,}01}$	$0{,}30_{\pm 0{,}01}$
LINUCB Por Usuário	$0{,}28_{\pm 0{,}01}$	$0{,}28_{\pm 0{,}01}$	$0{,}28_{\pm 0{,}01}$	$0{,}28_{\pm 0{,}01}$

5.2 Taxa de Aprendizado

A segunda pergunta de pesquisa (Q2) a ser respondida nesta dissertação é se o algoritmo proposto é capaz de aprender mais rápido que o melhor algoritmo encontrado na literatura (LINUCB). O principal objetivo é avaliar a taxa de aprendizado dos algoritmos além de mensurar a eficácia dos modelos conforme novas evidências dos testes são observadas, ou seja, quanto mais interações são realizadas com o usuário e mais *feedback* pode ser observado. Para isso, utilizamos o teste *offline*, onde o modelo aprendido em cada tempo t da sessão de cada usuário é fixado no teste *online* e avaliado sobre um novo conjunto de teste. Esse novo conjunto de teste é formado por 100 casos de testes extraídos após $t = 400$ sobre a sessão do mesmo usuário.

Dessa forma, os modelos aprendidos a cada tempo t no teste *online* não são atualizados interativamente conforme o *feedback* pode ser observado no conjunto de teste *offline*. Esse modelo é utilizado apenas para selecionar cada música da sessão de teste *offline*. Nesse caso, o *payoff* também recebe um ou zero, que representa respectivamente os comandos de *play* ou *skip*. Devido o conjunto de teste offline possuir 100 casos de testes para cada usuário, a eficácia offline conforme a métrica de *payoff offline* é no máximo 100.

A Figura 5.2 apresenta a eficácia *offline* média para os modelos aprendidos a cada tempo t do teste online em quatro partições. A taxa de aprendizado observada para os modelos iniciais do Multi-DBGD, possuem resultados similares ao *baseline* LINUCB Geral e superiores ao *baseline* LINUCB Por Usuário para todos os casos. De modo geral, a partir de $t \geq 100$ (em algumas partições ocorre antes, por exemplo, a "2008 01") os modelos aprendidos com o Multi-DBGD possuem eficácia *offline* superior em todas as partições sobre ambos os *baselines*, mesmo apresentando maior variação, devido à exploração de vários modelos reutilizando o mesmo *feedback*. Dessa forma, o Multi-DBGD é capaz de explorar o *feedback* do usuário e encontrar mais rapidamente modelos com maior eficácia se comparados aos modelos aprendidos pelos *baselines*.

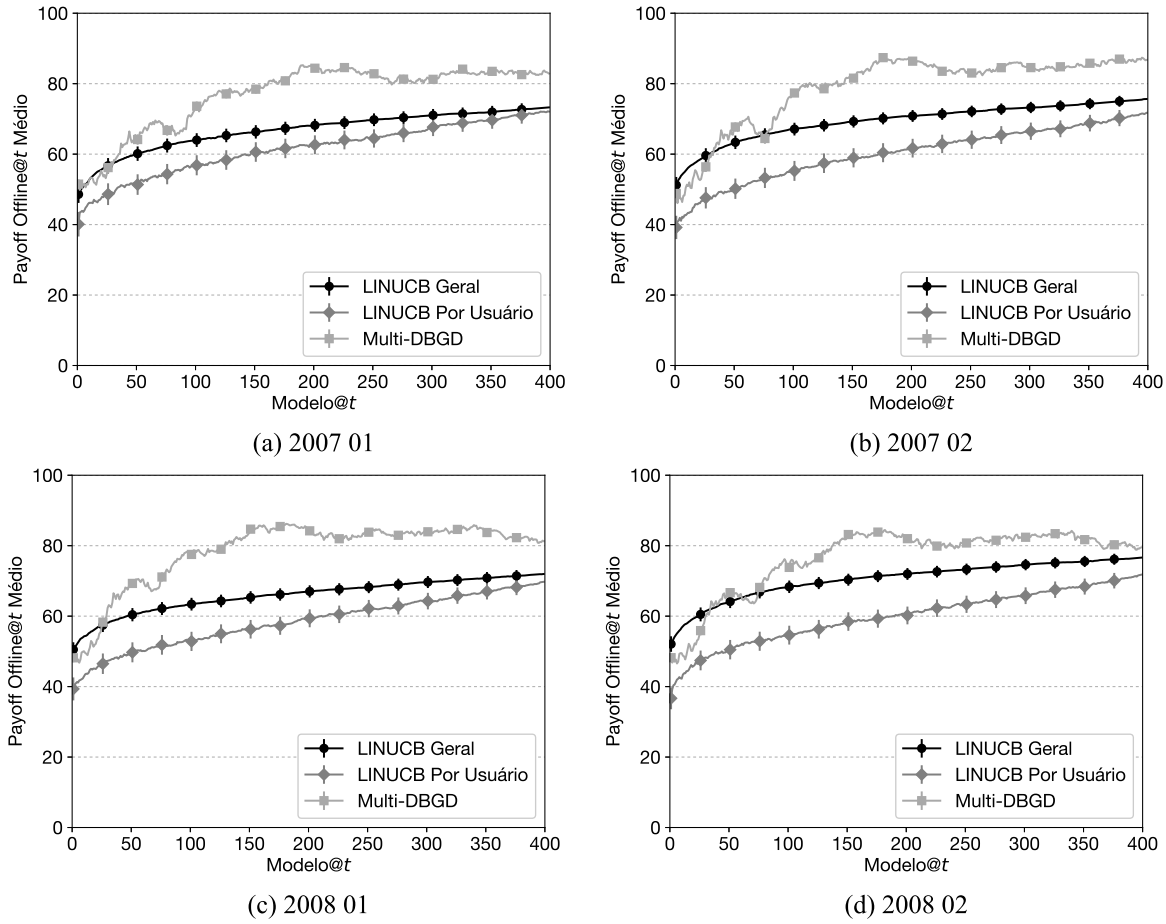

Figura 5.2. Resultados para o teste *offline*.

A Tabela 5.2 apresenta a média da eficácia *offline* para todos os modelos fixados a cada tempo t do teste *online*, ou seja, é uma média do resultados de 400 modelos. Os valores apresentados possuem significância estatística ($p < 0{,}01$), sendo os resultados do Multi-DBGD superiores em todas as partições.

Tabela 5.2. Média final do *payoff* para o teste *offline* sobre os modelos aprendidos a cada tempo t de todos usuários para cada partição de teste. A significância está indicada pelo símbolo ▲ ($p < 0{,}01$) em relação a cada *baseline*.

Algoritmo	2007 01	2007 02	2008 01	2008 02
Multi-DBGD	$\mathbf{76{,}79}_{\pm 0{,}91}$▲▲	$\mathbf{78{,}94}_{\pm 1.00}$▲▲	$\mathbf{78{,}54}_{\pm 0{,}92}$▲▲	$\mathbf{76{,}46}_{\pm 0{,}92}$▲▲
LINUCB Geral	$66{,}75_{\pm 0{,}53}$	$69{,}40_{\pm 0{,}50}$	$65{,}93_{\pm 0{,}45}$	$70{,}58_{\pm 0{,}51}$
LINUCB Por Usuário	$61{,}47_{\pm 0{,}71}$	$60{,}22_{\pm 0{,}74}$	$58{,}36_{\pm 0{,}70}$	$59{,}87_{\pm 0{,}74}$

5.3 Influência da Inicialização

A terceira pergunta de pesquisa (Q3) a ser respondida nesta dissertação é como o modelo de inicialização do algoritmo proposto afeta sua eficácia. Essa pergunta está relacionada apenas com as características do algoritmo Multi-DBGD. O algoritmo proposto aprende modelos para recomendação *online* de músicas a partir de um modelo inicial representado por um vetor de pesos. Propomos então avaliar a influência do modelo inicial sobre duas configurações: (*i*) Multi-DBGD Fixo, onde todas as posições do vetor de pesos são inicializadas com o valor do parâmetro γ; (*ii*) Multi-DBGD Aleatório, sendo o vetor de pesos inicializado a partir de um vetor unitário gerado aleatoriamente.

A Figura 5.3 apresenta a eficácia *online* acumulada para cada configuração proposta. A eficácia *online* acumulada do Multi-DBGD Fixo é pouco superior para qualquer t em todas as partições se comparado a configuração aleatória, sendo a diferença dos resultados para as duas últimas partições, maior que as duas primeiras.

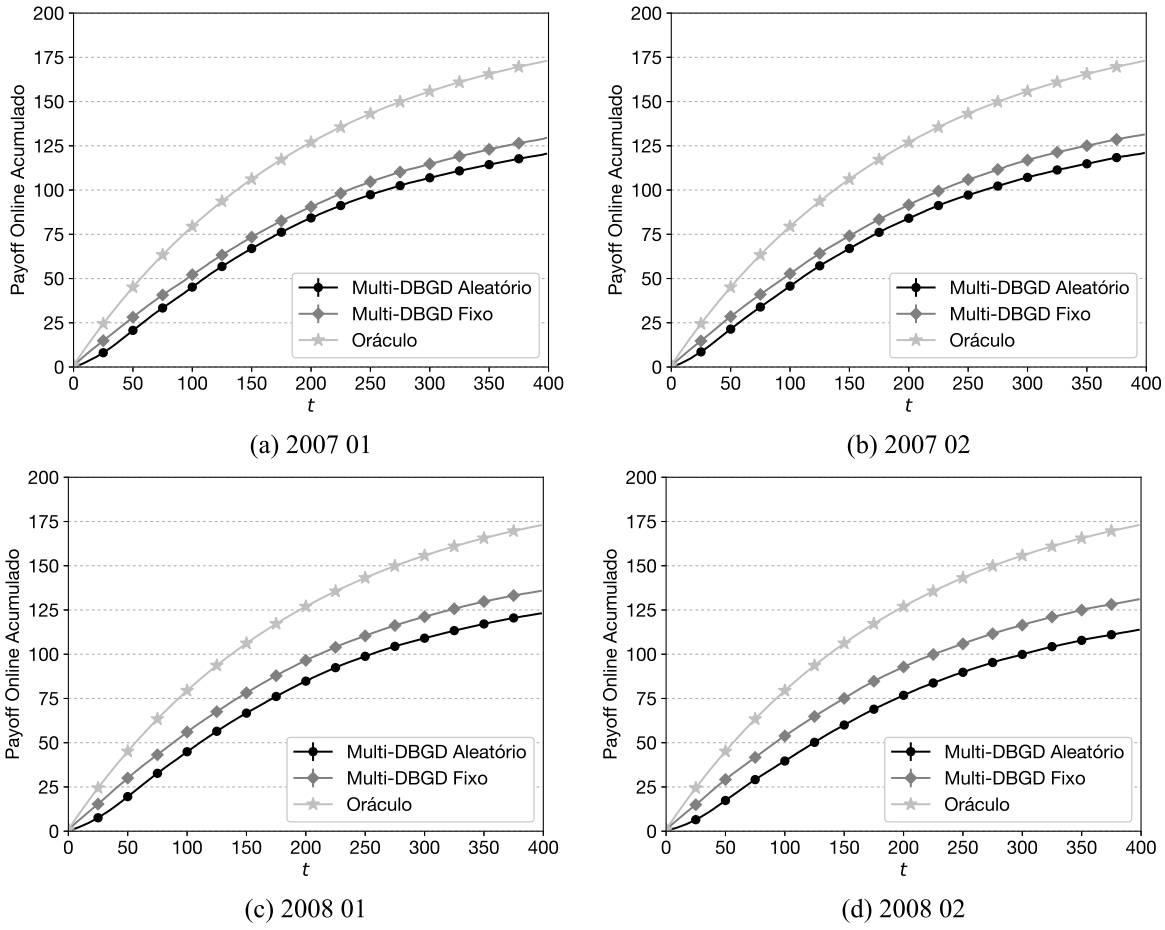

Figura 5.3. Resultados para o teste *online* em relação a inicialização.

Ainda sobre a Figura 5.3, apesar do modelo inicial afetar a eficácia do Multi-DBGD, o algoritmo proposto é capaz de aproximar os resultados mesmo partindo de um modelo totalmente desconhecido e aleatório. Essa análise permite levantar duas hipóteses para justificar os resultados similares, sendo elas: (i) a eficácia do modelo inicial aleatório é similar ao modelo fixo; (ii) o Multi-DBGD é capaz de convergir rapidamente partindo de um modelo totalmente desconhecido, para um modelo potencialmente superior e similar aos aprendidos partindo de um modelo conhecido.

A Figura 5.4 apresenta a eficácia *offline* média para cada configuração inicial, com objetivo de medir a qualidade dos modelos aprendidos a cada tempo t. Primeiramente, a eficácia dos modelos iniciais gerados com o Multi-DBGD Aleatório é drasticamente inferior comparado ao Multi-DBGD Fixo para todas as partições, refutando a primeira hipótese.

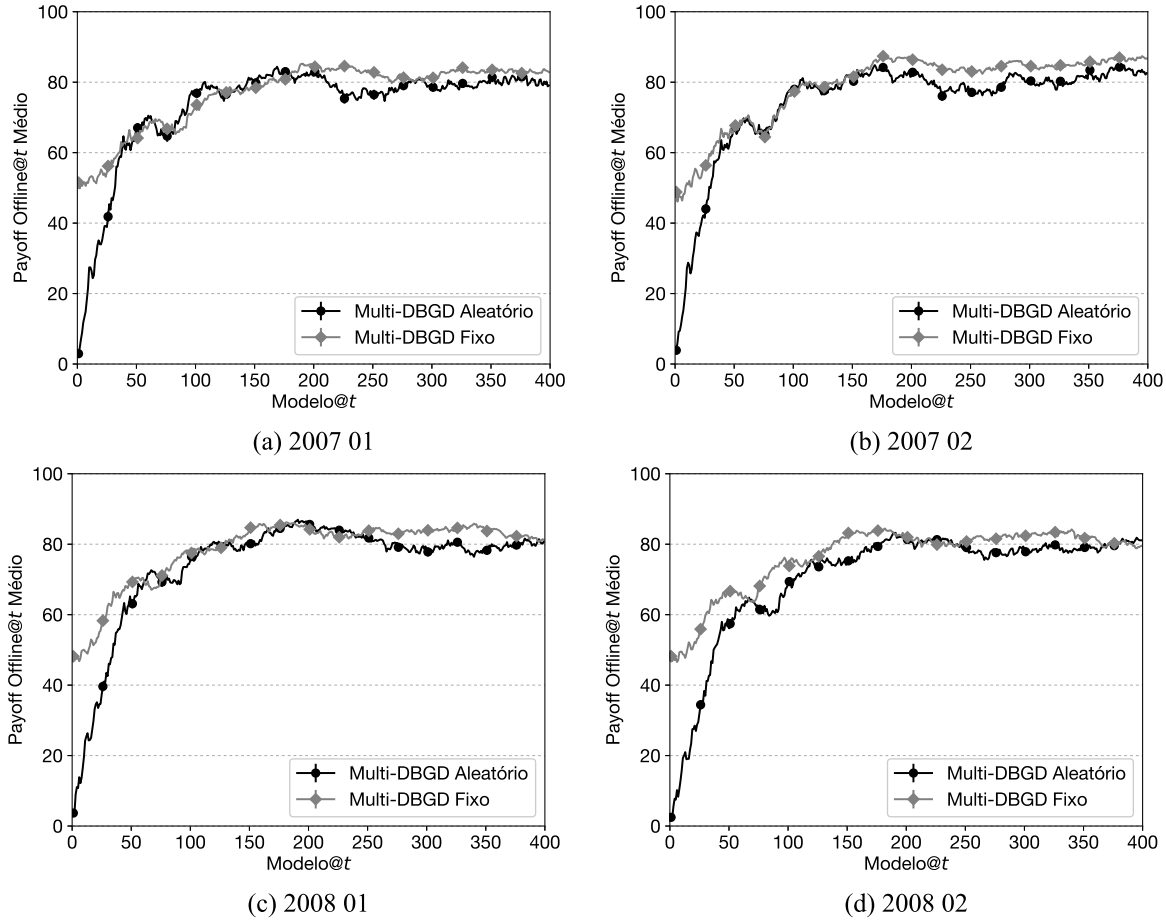

Figura 5.4. Resultados para o teste *offline* em relação a inicialização.

Ainda sobre a Figura 5.4, após poucas interações, o Multi-DBGD Aleatório é capaz de gerar modelos com eficácia dezena de vezes melhores além de similares aos Multi-DBGD Fixo. Esses resultados demonstram alta capacidade de convergência do

Multi-DBGD. Sendo assim, mesmo que nenhum conhecimento a piori seja aplicado sobre o modelo inicial, o algoritmo proposto é capaz de encontrar rapidamente modelos potencialmente mais eficazes com base somente nas interações junto ao usuário. Nos demais experimentos foi utilizada a abordagem de inicialização fixa, devido a melhor eficácia desde o início da sessão.

5.4 Influência do Parâmetro Número de Duelos

A quarta pergunta de pesquisa (Q4) a ser respondida nesta dissertação é qual impacto do número de duelos na eficácia do algoritmo proposto. O Multi-DBGD introduz um parâmetro m a fim de reutilizar o *feedback* sobre uma mesma música em múltiplas comparações entre o melhor modelo corrente e os modelos candidatos (duelos). Dessa forma, o parâmetro m indica o número de duelos realizados a cada *feedback* observado. A hipótese principal é que múltiplas comparações reutilizando o mesmo *feedback* permite acelerar o processo de aprendizado do algoritmo e consequentemente aprimorar os resultados dos modelos aprendidos a cada interação.

Os testes nesta seção são conduzidos conforme os seguintes passos: (i) definimos um espaço M de valores possíveis (entre um e dez) para o hiper parâmetro m, sendo $M \leftarrow [1, ..., 10]$; (ii) os valores de m são fixados $\forall m \in M$ e os demais hiper parâmetros do algoritmo Multi-DBBGD (γ e δ) são calibrados seguindo o processo de validação; (iii) a melhor configuração dos hiper parâmetros para cada m são submetidos aos testes *online* e *offline*. Os resultados são comparados com o resultado do melhor *baseline* para cada partição, sendo o mesmo fixo para qualquer m já que não depende deste hiper parâmetro.

A Figura 5.5 apresenta a eficácia *online* média sobre as variações do parâmetro m para todos os usuários de cada partição. Os resultados apresentam variação positiva conforme o aumento do parâmetro m, principalmente nos valores iniciais e se estabiliza quando m se aproxima do valor máximo proposto. Dessa forma, a introdução dos múltiplos duelos permite aprimorar o algoritmo proposto a ponto de recuperar resultados superiores comparados ao melhor *baseline* de cada partição. Mais ainda, a utilização de múltiplos duelos resulta em uma eficácia *online* superior em todas as partições e configurações se comparado ao mesmo algoritmo realizando apenas um duelo a cada *feedback* ($m = 1$).

A Figura 5.6 apresenta a eficácia *offline* média sobre a mesma variação do parâmetro m, a fim de medir a influência do parâmetro m sobre a taxa de aprendizado do algoritmo proposto. Assim como no teste *online*, os resultados apresentam variação

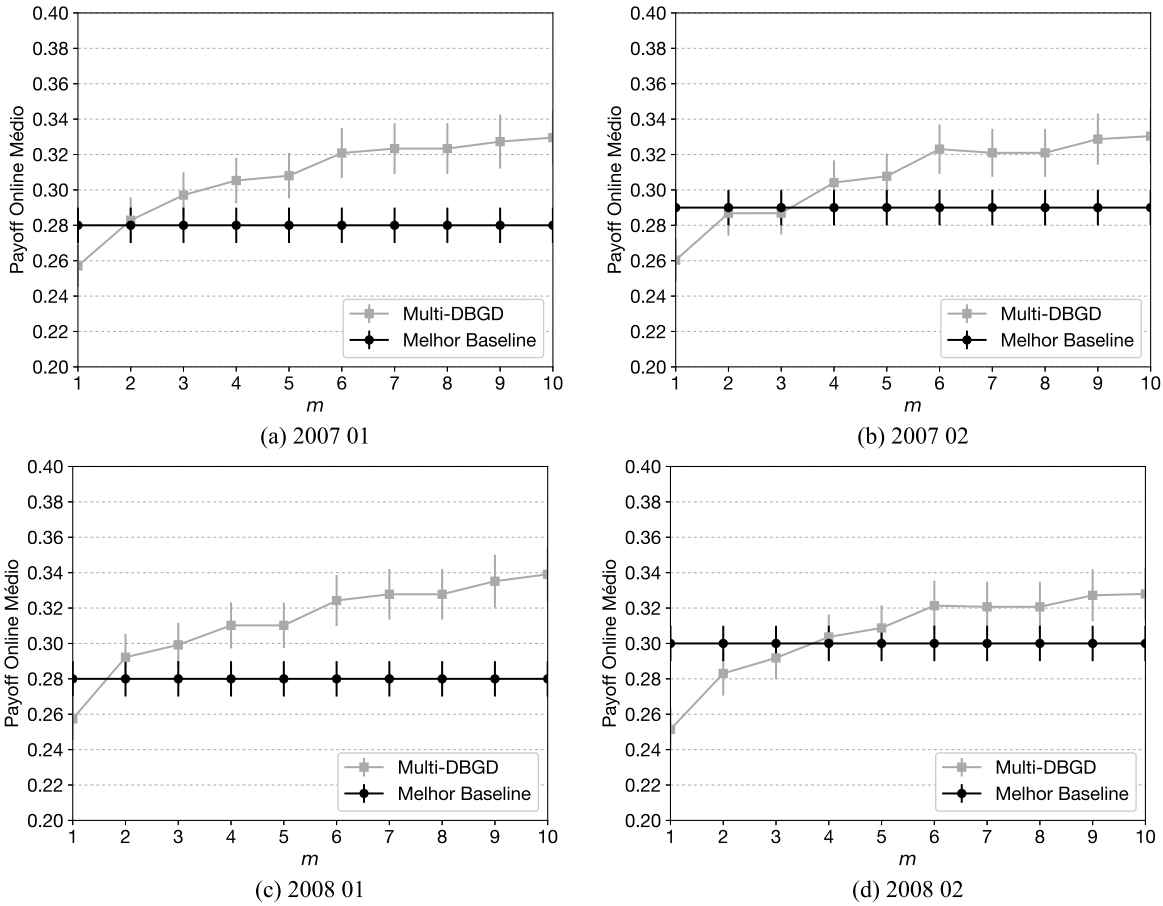

Figura 5.5. Resultados para o teste *online* variando o parâmetro m.

positiva conforme aumento do parâmetro m, se estabilizando quando m se aproxima do valor máximo proposto. O aumento do número de duelos permite recuperar valores superiores aos resultados obtidos pelo melhor *baseline* de cada partição. Além disso, a influência sobre a taxa de aprendizado é evidente, tendo em vista que a eficácia *offline* do algoritmo utilizando apenas um duelo a cada *feedback* ($m = 1$) é em média 30% inferior comparado aos melhores resultados observados para cada partição.

O impacto do número de duelos na eficácia do algoritmo proposto (Q4) é positivo e permite recuperar resultados superiores se comparados aos *baselines*. Dessa forma, a reutilização do *feedback* sobre um mesmo item a cada interação permite melhorar a eficácia dos modelos gerados e reduzir a quantidade de interações necessárias para recuperar modelos mais eficazes. Mais ainda, no cenário de recomendação *online* de músicas, as recomendações devem ser realizadas de forma instantânea devido as rápidas interações entre um novo *feedback* e a próxima música. Dessa forma, quanto menor o valor de m menos iterações são realizadas antes de uma nova recomendação, sendo importante destacar que o Multi-DBGD é capaz de superar os *baselines* em todos os

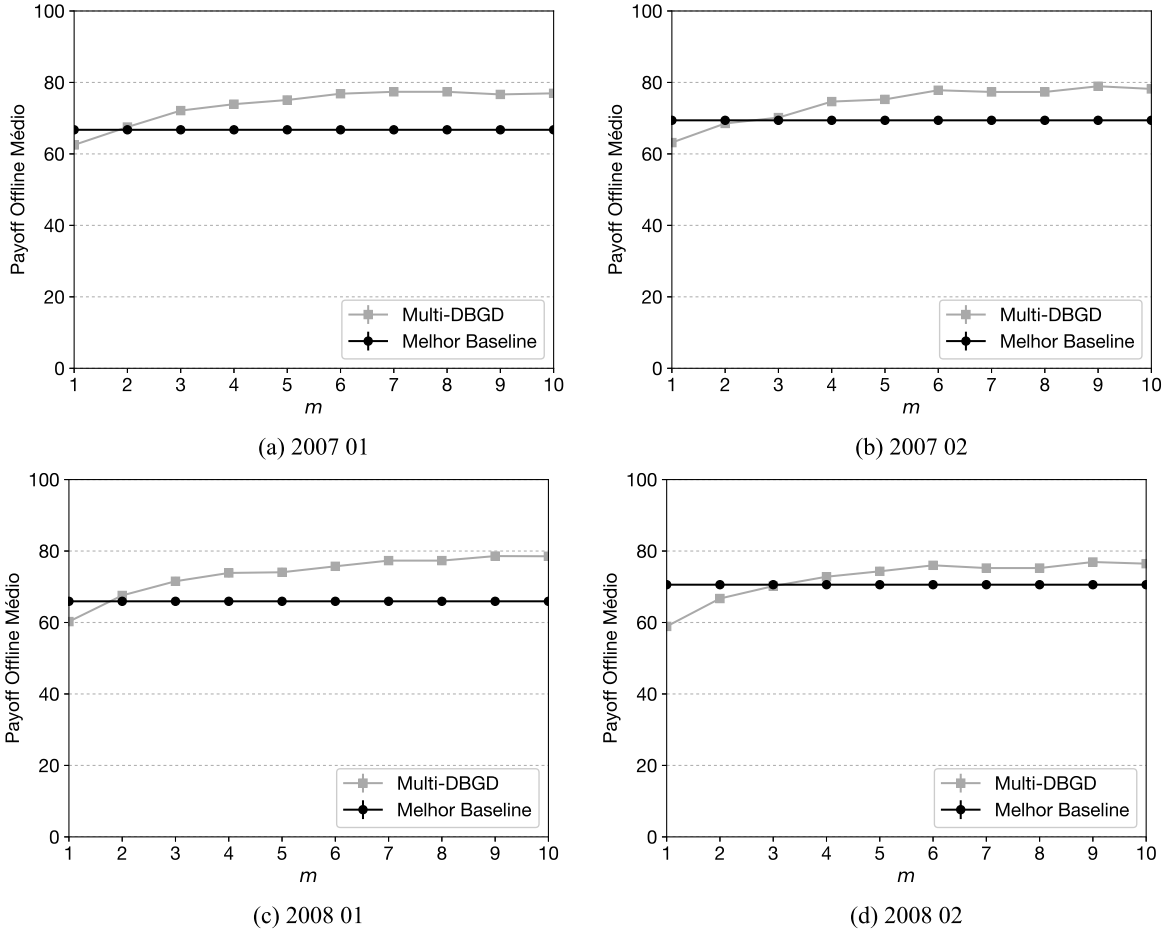

Figura 5.6. Resultados para o teste *offline* variando o parâmetro m.

casos com pequenos valores de m.

5.5 Sumário

Este capítulo descreve os resultados obtidos com uma configuração experimental separada em treino, teste e validação, utilizando sessões musicais de usuários reais coletadas da rede social Last.fm. O algoritmo Multi-DBGD é significantemente mais eficaz em todos os testes realizados se comparados aos *baselines*. Dessa forma, o algoritmo proposto é capaz de aprender melhores modelos de forma mais rápida se comparado aos *baselines*. Além disso, o algoritmo também é capaz de convergir rapidamente a partir de um modelo totalmente desconhecido e pouco efetivo para modelos potencialmente melhores. Por fim, o parâmetro m de número de duelos a cada interação com usuário é avaliado sobre um conjunto finito de opções demonstrando sua real contribuição na eficácia do algoritmo proposto ao reutilizar o *feedback* sobre uma mesma música.

A seguinte seção descreve as conclusões e contribuições baseadas nos resultados obtidos nesta seção. Por fim, são apresentadas direções futuras motivadas pela presente dissertação.

Capítulo 6

Conclusões e Trabalhos Futuros

Com a crescente demanda de informação, novos serviços têm surgido com o objetivo de suportar os usuários de diversas áreas. No caso do consumo de mídias por exemplo, os serviços de *streaming* tem se tornado essenciais no cotidiano de várias pessoas. Particularmente no cenário de músicas, Spotify, Google Play Music são exemplos de grandes serviços de *streaming* de música que fornecem uma biblioteca contendo milhares de músicas a fim de maximizar o alcance sobre variados gêneros e tipos de usuários.

Essa vasta biblioteca de opções exige métodos cada vez mais sofisticados a fim de aumentar a satisfação dos usuários ao interagirem com esses sistemas, tendo os sistemas de recomendação papel fundamental nesse aspecto. No cenário de recomendação de músicas, os desafios são vários partindo desde a sugestão de gêneros musicais até a recomendação de artistas ou músicas específicas. Nessa dissertação, o objetivo é a recomendação online de músicas, tarefa que envolve a interação continua com o usuário a fim de aprimorar os modelos de recomendação enquanto recomenda cada música de uma sessão e recebe *feedback*.

Em contraste as abordagens existentes para recomendação online de músicas, propomos a utilização do *feedback* implícito como único sinal de preferências dos usuários a cada ponto no tempo conforme as conclusões de Vall [2015]. Sobre esse cenário, encontramos apenas o trabalho de King & Imbrasaitė [2015] que propõem a utilização do *feedback* implícito e aprendizado por reforço na recomendação de músicas. Entretanto, esse processo é aplicado para escolha de grupos de músicas e não da próxima música a ser recomendada, conforme tratado pelo presente trabalho.

Nesta dissertação apresentamos o algoritmo Multi-DBGD (*Multi Dueling Bandits Gradient Descent*, ou em português, Gradiente Descendente de Multi Duelos entre Bandidos) um novo algoritmo de aprendizado online utilizando *feedback* implícito para recomendação de músicas. Nossa proposta é a recomendação de cada música a fim de

compor interativamente a sessão de um usuário, reutilizando o *feedback* observado sobre um mesmo item a cada interação para comparar múltiplos modelos de recomendação. A Seção 6.1 apresenta as principais contribuições desta dissertação. A Seção 6.2 apresenta as conclusões observadas sobre os resultados. A Seção 6.3 apresenta direções futuras motivadas por esta dissertação. Por fim, a Seção 6.4 apresenta nossas considerações finais.

6.1 Resumo das Contribuições

As contribuições desta dissertação são:

- Proposta de um novo algoritmo, chamado Multi-DBGD, capaz de aprender interativamente a importância sobre as características consideradas como representação de cada música para recomendação *online* usando o *feedback* implícito como único sinal sobre as preferências dos usuários a cada ponto no tempo.
- Extensa avaliação do algoritmo proposto quanto a sua eficácia e convergência.
- Resultados experimentais mostram que o algoritmo Multi-DBGD é capaz de superar o melhor algoritmo encontrado na literatura, utilizando dados reais e publicamente disponíveis.

6.2 Resumo das Conclusões

A presente seção apresenta as conclusões observadas a partir dos resultados obtidos pelo algoritmo Multi-DBGD e os *baselines*, apresentados no capítulo anterior. O objetivo é compreender os componentes do algoritmo proposto, assim como responder as perguntas de pesquisa apresentadas na Seção 1.1.

Na Seção 5.1 apresentamos os resultados com base na eficácia online dos algoritmos. A eficácia online é uma forma de simular o comportamento dos algoritmos sobre um cenário similar ao real. Sendo assim, quanto mais eficaz nesse teste, menor a quantidade de *feedback* negativo é observado resultando em sessões mais consistentes em relação as preferências dos usuários. O Multi-DBGD apresentou ganhos acumulados significantes para todas as partições, sendo a eficácia online do algoritmo proposto no mínimo igual à dos *baselines*. Os resultados demonstram que os modelos aprendidos interativamente usando o Multi-DBGD possuem maior taxa de acerto na seleção de músicas relevantes para as sessões dos usuários.

Na Seção 5.2 apresentamos a taxa de aprendizado a partir da eficácia offline, com objetivo de medir a velocidade de aprendizado dos algoritmos e avaliar a capacidade em aprimorar os modelos a cada novo *feedback* observado. A eficácia offline dos modelos gerados pelo Multi-DBGD é significantemente superior à dos *baselines* em todos as partições. Os resultados dos modelos iniciais gerados pelo Multi-DBGD na média são próximos ao melhor *baseline* para cada partição. Entretanto, após algumas interações o algoritmo proposto é capaz de gerar modelos efetivamente melhores. Apesar dos modelos aprendidos com o Multi-DBGD apresentarem maior variação na eficácia, em todos os casos ela permanece superior após poucas interações. Dessa forma, os resultados qualificam a capacidade do algoritmo proposto em encontrar melhores modelos mais rapidamente comparado aos resultados dos *baselines*.

Na Seção 5.3 foram propostas duas configurações iniciais (Multi-DBGD Aleatório e Multi-DBGD Fixo) para o algoritmo proposto a fim de verificar como o modelo inicial afeta a eficácia do algoritmo. Na eficácia online, os resultados apesar de similares, o Multi-DBGD Fixo é melhor se comparado ao modelo aleatório. Já a eficácia offline permite observar o comportamento dos modelos conforme são observados novos *feedback*. Os resultados dos primeiros modelos gerados pelo Multi-DBGD Aleatório são drasticamente piores se comparados ao modelo fixo. Apesar disso, em poucas interações a eficácia offline é similares em todas as partições, ou seja, mesmo partindo de modelos totalmente desconhecidos, o Multi-DBGD apresentou uma importante capacidade em convergir para modelos potencialmente melhores em poucas interações.

Por fim, na Seção 5.4 é estudada a influência do parâmetro m na eficácia online e offline do algoritmo Multi-DBGD. O parâmetro m indica o número de duelos realizados a cada interação com os usuários. Ou seja, define quantas vezes o algoritmo irá reutilizar o mesmo *feedback* e quantos modelos serão comparados a cada interação. Primeiramente, a eficácia online média apresenta uma variação pequena conforme os valores de m são acrescidos. Entretanto, essa variação é suficiente para recuperar resultados médios superiores aos melhores *baselines* de cada partição. A eficácia offline média permite observar maior variação nos resultados, sendo o acréscimo sobre o parâmetro m do Multi-DBGD recupera resultados em média 12% superiores se comparado aos melhores *baselines* de cada partição. Além disso, em relação ao modelo utilizando apenas um duelo (ou seja, $m = 1$) a cada interação, o ganho é em média 30%. Apesar de melhorar a eficácia se estabiliza mesmo enquanto o valor de m é acrescido.

6.3 Trabalhos Futuros

A partir dos estudos realizados nesta dissertação, observamos algumas direções a serem exploradas, assim como as seguintes:

- Motivados pelos resultados obtidos com o Multi-DBGD sobre o cenário de recomendação de música, propomos como trabalhos futuros a adaptação do Multi-DBGD para outros cenários, a fim de explorar a capacidade de generalização do algoritmo ao comparar os resultados em múltiplos cenários. Por exemplo, a recomendação de notícias assim como proposito inicial do *baseline* LINUCB.

- No caso do Multi-DBGD, os hiper parâmetros de exploração e aproveitamento são fixos para todo o teste, sendo estimados sobre um conjunto de usuários. Propomos então a adaptação desses hiper parâmetros durante o aprendizado *online*, ou seja, interativamente, também a partir do *feedback* implícito.

- Um difícil e importante problema em recomendação é conhecido como *cold-start* [Schein et al., 2002]. Nesse caso, itens, usuários ou ambos não possuem interações no sistema, ou seja, as preferências são desconhecidas a priori. Sobre esse problema, propomos estender o Multi-DBGD a fim de avaliar e aprimorar as soluções atuais utilizando o *feedback* implícito.

- Propomos estender a técnica de aprendizado online usando *feedback* implícito para o processo de recuperação de músicas candidatas. A diferença com o cenário tratado nesta dissertação está no objetivo de buscar um conjunto de músicas candidatas a uma *playlist*, sem preocupar com qual música dessa lista será escutada a cada interação e tendo a disposição todo o horizonte de músicas da base de dados. Nesse caso, sobre o conjunto de músicas selecionadas, o próprio Multi-DBGD poderia ser utilizado a fim de criar a sessão junto com o usuário. A utilização do comportamento online dos usuários para auxiliar o processo de recuperação de músicas foi utilizado no trabalho de Cheng et al. [2016b]. Entretanto, propomos a recuperação de músicas sem a necessidade de consultas dos usuários, dispondo apenas de informações da sessão corrente e do *feedback* implícito.

- Com o sucesso dos dispositivos móveis e a crescente utilização desses equipamentos no dia dia das pessoas, propomos como trabalhos futuros aprimorar o conjunto de características que representam as músicas a partir de informações contextuais como localização, movimentação, entre outros. Também sobre o conjunto de características, propomos estender a extração para múltiplos domínios, por exemplo, utilizando informações das redes sociais de cada usuário.

6.4 Considerações Finais

Nesta dissertação propomos um novo algoritmo para operar sobre um cenário pouco explorado em recomendação online de músicas. Em contraste as abordagens existentes na literatura, apresentamos o Multi-DBGD, um algoritmo de aprendizado *online* capaz de aprender interativamente e sensível às características consideradas na representação de cada música na recomendação online, utilizando o *feedback* implícito como único sinal sobre as preferências dos usuários a cada ponto no tempo. Comparado ao melhor algoritmo encontrado na literatura para servir de *baseline*, o Multi-DBGD apresentou ganhos significativos sobre experimentos utilizando dados reais e publicamente disponíveis.

Apêndice A

Validação de Parâmetros

Tabela A.1. Conjunto de parâmetros utilizados nesta dissertação para validação do algoritmo Multi-DBGD e dos baselines utilizando busca em grande.

Algoritmo	Parâmetro	Valores
Multi-DBGD	m	[1 2 3 4 5 6 7 8 9 10]
	δ	[0,1 0,2 0,3 0,4 0,5 0,6 0,7 0,8 0,9 1,0]
	γ	[0,1 0,2 0,3 0,4 0,5 0,6 0,7 0,8 0,9 1,0]
LINUCB	α	[0,1 0,2 0,3 0,4 0,5 0,6 0,7 0,8 0,9 1,0]

Referências Bibliográficas

Ahmed, M.; Spagna, S.; Huici, F. & Niccolini, S. (2013). A peek into the future: Predicting the evolution of popularity in user generated content. Em *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, pp. 607--616.

Annesi, P.; Basili, R.; Gitto, R.; Moschitti, A. & Petitti, R. (2007). Audio feature engineering for automatic music genre classification. Em *Large Scale Semantic Access to Content (Text, Image, Video, and Sound)*, pp. 702--711.

Auer, P.; Cesa-Bianchi, N. & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2/3):235--256.

Baeza-Yates, R. & Ribeiro-Neto, B. (2011). *Modern Information Retrieval – The Concepts and Technology Behind Search.* Pearson Education Ltd, 2nd edição.

Baltrunas, L. & Amatriain, X. (2009). Towards time-dependant recommendation based on implicit feedback. Em *Workshop on Context-Aware Recommender Systems.*

Basu, C.; Hirsh, H. & Cohen, W. (1998). Recommendation as alassification: Using social and content-based information in recommendation. Em *Proceedings of the 10th Conference on Innovative Applications of Artificial Intelligence*, pp. 714--720.

Ben-David, S.; Kushilevitz, E. & Mansour, Y. (1997). Online learning versus offline learning. *Machine Learning*, 29(1):45--63.

Bergstra, J. S.; Bardenet, R.; Bengio, Y. & Kégl, B. (2011). Algorithms for hyper-parameter optimization. Em *Proceedings of the 24th Advances in Neural Information Processing Systems*, pp. 2546--2554.

Bonnin, G. & Jannach, D. (2015). Automated generation of music playlists: Survey and experiments. *Computing Surveys*, 47(2):26.

Bosteels, K. & Kerre, E. E. (2009). A fuzzy framework for defining dynamic playlist generation heuristics. *Fuzzy Sets and Systems*, 160(23).

Bosteels, K.; Pampalk, E. & Kerre, E. E. (2009). Evaluating and analysing dynamic playlist generation heuristics using radio logs and fuzzy set theory. Em *Proceedings of the 10th International Conference on Music Information Retrieval*, pp. 351--356.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. Em *Proceedings of the 19th International Conference on Computational Statistics*, pp. 177--186.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331--370.

Celma, O. (2010). *Music recommendation and discovery: The long tail, long fail, and long play in the digital music space.* Springer, 1st edição.

Chapelle, O.; Schölkopf, B. & Zien, A. (2010). *Semi-Supervised Learning.* The MIT Press, Cambridge, Massachusetts, 1st edição.

Cheng, J.; Adamic, L. A.; Kleinberg, J. M. & Leskovec, J. (2016a). Do cascades recur? Em *Proceedings of the 25th International Conference on World Wide Web*, pp. 671--681.

Cheng, Z.; Jialie, S. & Hoi, S. C. (2016b). On effective personalized music retrieval by exploring online user behaviors. Em *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 125--134.

Chuklin, A.; Markov, I. & de Rijke, M. (2015). Click models for web search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 7(3):1--115.

Cremonesi, P.; Koren, Y. & Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. Em *Proceedings of the 4th ACM Conference on Recommender Systems*, pp. 39--46.

Eck, D.; Lamere, P.; Bertin-Mahieux, T. & Green, S. (2008). Automatic generation of social tags for music recommendation. Em *Proceedings of the 21th Advances in Neural Information Processing Systems*, pp. 385--392.

Flaxman, A. D.; Kalai, A. T. & McMahan, B. (2005). Online convex optimization in the bandit setting: Gradient descent without a gradient. Em *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 385--394.

Flexer, A.; Schnitzer, D.; Gasser, M. & Widmer, G. (2008). Playlist generation using start and end songs. Em *Proceedings of the 9th International Conference on Music Information Retrieval*, pp. 173--178.

Garivier, A. & Cappé, O. (2011). The kl-ucb algorithm for bounded stochastic bandits and beyond. Em *Proceedings of the 24th Annual Conference on Learning Theory*, pp. 359--376.

Gittins, J. C. (1979). Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 2(41):148--177.

Hariri, N.; Mobasher, B. & Burke, R. (2012). Context-aware music recommendation based on latent topic sequential patterns. Em *Proceedings of the 6th ACM Conference on Recommender Systems*, pp. 131--138.

Hariri, N.; Mobasher, B. & Burke, R. (2015). Adapting to user preference changes in interactive recommendation. Em *Proceedings of the 24th International Conference on Artificial Intelligence*, pp. 4268--4274.

Herlocker, J. L.; Konstan, J. A.; Borchers, A. & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. Em *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 230--237.

Hofmann, K. (2013). *Fast and Reliable Online Learning to Rank for Information Retrieval*. Tese de doutorado, University of Amsterdam.

Hou, Y.; Zhou, P.; Wang, T.; Yu, L.; Hu, Y. & Wu, D. (2016). Context-aware online learning for course recommendation of mooc big data. *CoRR*, abs/1610.03147.

Hu, R. & Pu, P. (2009). A comparative user study on rating vs. personality quiz based preference elicitation methods. Em *Proceedings of the 14th International Conference on Intelligent User Interfaces*, pp. 367--372.

Hu, R. & Pu, P. (2010). A study on user perception of personality-based recommender systems. Em *Proceedings of the 18th International Conference on User modeling, Adaptation, and Personalization*, pp. 291--302.

Hurley, N. & Zhang, M. (2011). Novelty and diversity in top-n recommendation: Analysis and evaluation. *ACM Transactions on Internet Technology (TOIT)*, 10(4):14.

Jawaheer, G.; Szomszor, M. & Kostkova, P. (2010). Comparison of implicit and explicit feedback from an online music recommendation service. Em *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, pp. 47--51.

Kelly, D. & Teevan, J. (2003). Implicit feedback for inferring user preference: A bibliography. Em *Proceedings of the 38th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 18--28.

King, J. & Imbrasaitė, V. (2015). Generating music playlists with hierarchical clustering and q-learning. Em *Advances in Information Retrieval*, pp. 315--326.

Knees, P. & Schedl, M. (2015). Music retrieval and recommendation: A tutorial overview. Em *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1133--1136.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. Em *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, volume 2, pp. 1137--1143.

Koren, Y.; Bell, R. & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30--37.

Lai, T. L. & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4--22.

Langford, J. & Zhang, T. (2008). The epoch-greedy algorithm for multi-armed bandits with side information. Em *Proceedings of the 21th Advances in Neural Information Processing Systems*, pp. 817--824.

Li, L.; Chu, W.; Langford, J. & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. Em *Proceedings of the 19th International Conference on World Wide Web*, pp. 661--670.

Li, L.; Chu, W.; Langford, J. & Wang, X. (2011). Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. Em *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, pp. 297--306.

Liebman, E.; Saar-Tsechansky, M. & Stone, P. (2015). Dj-mc: A reinforcement-learning agent for music playlist recommendation. Em *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 591--599.

Lops, P.; De Gemmis, M. & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. Em *Recommender Systems Handbook*, pp. 73--105.

March, J. G. (1991). Exploration and exploitation in organizational learning. *Organization Science*, 2(1):71--87.

Matsubara, Y.; Sakurai, Y.; Prakash, B. A.; Li, L. & Faloutsos, C. (2012). Rise and fall patterns of information diffusion: Model and implications. Em *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 6--14.

Murphy, K. P. (2012). *Machine Learning – A Probabilistic Perspective*. The MIT Press, Cambridge, Massachusetts.

Nguyen, M. N.; Pham, C.; Son, J. & Hong, C. S. (2016). Online learning-based clustering approach for news recommendation systems. Em *Proceedings of the 18th Asia-Pacific Network Operations and Management Symposium*, pp. 1--4.

Pampalk, E.; Pohle, T. & Widmer, G. (2005). Dynamic playlist generation based on skipping behavior. Em *Proceedings of the 6th International Conference on Music Information Retrieval*, pp. 634--637.

Pazzani, M. J. & Billsus, D. (2007). *Content-Based Recommendation Systems*, pp. 325--341.

Peng, Z.; Cui, D.; Ma, Y.; Xiong, J.; Xu, B. & Lin, W. (2016). R-learning and gaussian process regression algorithm for cloud job access control. Em *IEEE 3rd International Conference on Cyber Security and Cloud Computing*, pp. 163–166.

Ricci, F.; Rokach, L. & Shapira, B. (2015). *Introduction to Recommender Systems Handbook*. Springer, 2nd edição.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747.

Salton, G.; Wong, A. & Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613--620.

Sarwar, B.; Karypis, G.; Konstan, J. & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. Em *Proceedings of the 10th International Conference on World Wide Web*, pp. 285--295.

Schein, A. I.; Popescul, A.; Ungar, L. H. & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. Em *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 253--260.

Schuth, A.; Oosterhuis, H.; Whiteson, S. & de Rijke, M. (2016). Multileave gradient descent for fast online learning to rank. Em *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pp. 457--466.

Shalev-Shwartz, S. & Singer, Y. (2007). *Online Learning: Theory, Algorithms, and Applications*. Tese de doutorado.

Song, L.; Tekin, C. & Schaar, M. (2014a). Clustering based online learning in recommender systems: A bandit approach. Em *Proceedings of the 39th International Conference on Acoustics, Speech and Signal Processing*, pp. 4528--4532.

Song, L.; Tekin, C. & Schaar, M. (2014b). Online learning in large-scale contextual recommender systems. *IEEE Transactions on Services Computing*, 9(3):433–445.

Song, Y.; Dixon, S. & Pearce, M. (2012). A survey of music recommendation systems and future perspectives. Em *Proceedings of the 9th International Symposium on Computer Music Modeling and Retrieval*, pp. 395--410.

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*, volume 1. The MIT Press, Cambridge, Massachusetts.

Swartz, A. (2002). Musicbrainz: A semantic web service. *IEEE Intelligent Systems*, 17(1):76--77.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285--294.

Vall, A. (2015). Listener-inspired automated music playlist generation. Em *Proceedings of the 9th ACM Conference on Recommender Systems*, pp. 387--390.

Volkovs, M. N. & Wei Yu, G. (2015). Effective latent models for binary feedback in recommender systems. Em *Proceedings of the 38th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 313--322.

Wang, X.; Wang, Y.; Hsu, D. & Wang, Y. (2014). Exploration in interactive personalized music recommendation: A reinforcement learning approach. *Transactions on Multimedia Computing, Communications, and Applications*, 11(1):1--22.

Wiering, M. & Van Otterlo, M. (2012). *Reinforcement Learning – State-of-the-Art*, volume 12. Springer.

Xing, Z.; Wang, X. & Wang, Y. (2014). Enhancing collaborative filtering music recommendation by balancing exploration and exploitation. Em *Proceedings of the 15th Conference of the International Society for Music Information Retrieval*, pp. 445--450.

Yang, B.; Lee, S.; Park, S. & Lee, S.-g. (2012). Exploiting various implicit feedback for collaborative filtering. Em *Proceedings of the 21st International Conference on World Wide Web*, pp. 639--640.

Yue, Y. & Joachims, T. (2009). Interactively optimizing information retrieval systems as a dueling bandits problem. Em *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1201--1208.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3):338--353.