Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Ciência da Computação

Uma Estratégia para Predição da Taxa de Aprendizagem do Gradiente Descendente para Aceleração da Fatoração de Matrizes

Caio Santos Bezerra Nóbrega

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Metodologias e Técnicas da Computação

Leandro Balby Marinho
(Orientador)

Campina Grande, Paraíba, Brasil

Resumo

Sugerir os produtos mais apropriados aos diversos tipos de consumidores não é uma tarefa trivial, apesar de ser um fator chave para aumentar satisfação e lealdade destes. Devido a esse fato, sistemas de recomendação têm se tornado uma ferramenta importante para diversas aplicações, tais como, comércio eletrônico, sites personalizados e redes sociais. Recentemente, a fatoração de matrizes se tornou a técnica mais bem sucedida de implementação de sistemas de recomendação. Os parâmetros do modelo de fatoração de matrizes são tipicamente aprendidos por meio de métodos numéricos, tal como o gradiente descendente. O desempenho do gradiente descendente está diretamente relacionada à configuração da taxa de aprendizagem, a qual é tipicamente configurada para valores pequenos, com o objetivo de não perder um mínimo local. Consequentemente, o algoritmo pode levar várias iterações para convergir. Idealmente, é desejada uma taxa de aprendizagem que conduza a um mínimo local nas primeiras iterações, mas isto é muito difícil de ser realizado dada a alta complexidade do espaço de valores a serem pesquisados. Começando com um estudo exploratório em várias bases de dados de sistemas de recomendação, observamos que, para a maioria das bases, há um padrão linear entre a taxa de aprendizagem e o número de iterações necessárias para atingir a convergência. A partir disso, propomos utilizar modelos de regressão lineares simples para predizer, para uma base de dados desconhecida, um bom valor para a taxa de aprendizagem inicial. A ideia é estimar uma taxa de aprendizagem que conduza o gradiente descendente a um mínimo local nas primeiras iterações. Avaliamos nossa técnica em 8 bases de sistemas de recomendação reais e comparamos com o algoritmo padrão, o qual utiliza um valor fixo para a taxa de aprendizagem, e com técnicas que adaptam a taxa de aprendizagem extraídas da literatura. Nós mostramos que conseguimos reduzir o número de iterações até em 40% quando comparados à abordagem padrão.

Abstract

Suggesting the most suitable products to different types of consumers is not a trivial task, despite being a key factor for increasing their satisfaction and loyalty. Due to this fact, recommender systems have become an important tool for many applications, such as e-commerce, personalized websites and social networks. Recently, Matrix Factorization has become the most successful technique to implement recommendation systems. The parameters of this model are typically learned by means of numerical methods, like the gradient descent. The performance of the gradient descent is directly related to the configuration of the learning rate, which is typically set to small values, in order to do not miss a local minimum. As a consequence, the algorithm may take several iterations to converge. Ideally, one wants to find a learning rate that will lead to a local minimum in the early iterations, but this is very difficult to achieve given the high complexity of search space. Starting with an exploratory study on several recommendation systems datasets, we observed that there is an overall linear relationship between the learning rate and the number of iterations needed until convergence. From this, we propose to use simple linear regression models to predict, for a unknown dataset, a good value for an initial learning rate. The idea is to estimate a learning rate that drives the gradient descent as close as possible to a local minimum in the first iteration. We evaluate our technique on 8 real-world recommender datasets and compared it with the standard Matrix Factorization learning algorithm, which uses a fixed value for the learning rate over all iterations, and techniques from the literature that adapt the learning rate. We show that we can reduce the number of iterations until at 40% compared to the standard approach.

Agradecimentos

Primeiramente, a Deus.

Aos meus pais, Audálecio Nóbrega e Albanete Nóbrega, por me darem todas as condições para que eu pudesse ter a melhor educação possível, sem a qual não poderia chegar à conclusão do mestrado.

À minha namorada Angélica por sempre me apoiar e entender os momentos em que tive que me dedicar mais ao trabalho.

Ao meu orientador Leandro Balby pelos ensinamentos e discussões, que me incentivaram para a pesquisa científica e também me influenciaram como pessoa.

Aos amigos que fiz durante a graduação, especialmente, Andryw Marques, Diógenes Gondim e Iury Dewar, por me motivarem e tornarem a minha trajetória universitária mais divertida.

E, finalmente, a HP e a CAPES por financiarem o meu projeto.

Conteúdo

1 Introdução **1**

1.1 Contexto 1

1.2 Definição do Problema 5

1.3 Resultados e Contribuições 7

1.4 Estrutura da Dissertação 8

2 Fundamentação Teórica **10**

2.1 Principais Conceitos de Sistemas de Recomendação 10

2.2 Fatoração de Matrizes 14

2.3 Gradiente Descendente Estocástico 17

3 Trabalhos Relacionados **20**

3.1 Estratégias para a Determinação de Hiperparâmetros 20

3.1.1 Técnicas de Taxa de Aprendizagem Adaptativa 23

3.2 Gradiente Descendente Estocástico Distribuído 25

4 Um técnica para Estimar o Valor Inicial da Taxa de Aprendizagem **28**

4.1 Intuição 28

4.2 Predição da Taxa de Aprendizagem 29

5 Avaliação Experimental **34**

5.1 Descrição das Bases de Dados 34

5.2 Planejamento de Experimentos 35

5.2.1 Reprodutibilidade dos Experimentos 36

5.3 Predizendo a Taxa de Aprendizagem Inicial 37

5.4 Comparação com Estratégias de Taxa de Aprendizagem Adaptativa 42

6 Conclusões e Trabalhos Futuros **48**

A Características das Bases de Dados **56**

A.1 Distribuição das Avaliações 56

A.2 Distribuição dos Graus de Usuários e Itens 60

B Outras Abordagens Avaliadas **63**

B.1 Line Search 63

B.2 Método Barzilai-Borwein 65

Lista de Símbolos

RMSE:	Raiz Quadrada do Erro Médio (do inglês *Root-Mean-Square Error*).
SSE:	Soma dos Erros Quadráticos (do inglês *Sum of Squared Errors*).
U	Conjunto de usuários de uma base de dados.
I	Conjunto de itens de uma base de dados.
A	Conjunto de possíveis valores de avaliação, por exemplo $A := 1, 2, 3, 4, 5$.
U_i	Conjunto de usuários que avaliaram o item i.
I_u	Conjunto de itens avaliados pelo usuário u.
R	Matriz que contém o conjunto de avaliações do base de dados, tal que $R \in \mathbb{R}^{\lvert U\rvert \times \lvert I\rvert}$.
$r_{u,i}$	Avaliação/nota do usuário u ao item i.
$\hat{r}_{u,i}$	Predição da nota que o usuário u daria ao item i.
k	Número de fatores latentes da Fatoração de Matrizes.
α	Taxa de aprendizagem da Fatoração de Matrizes.
λ	Termo de regularização da Fatoração de Matrizes.
P	Matriz de fatores latentes dos usuários, tal que $P \in \mathbb{R}^{\lvert U\rvert \times k}$.
Q	Matriz de fatores latentes dos itens, tal que $Q \in \mathbb{R}^{\lvert I\rvert \times k}$.
p_u	Vetor de fatores latentes do usuário u, tal que $p_u \in \mathbb{R}^{k \times 1}$. O vetor representa a u-ésima linha do matriz P.
q_i	Vetor de fatores latentes do usuário i, tal que $q_i \in \mathbb{R}^{k \times 1}$. O vetor representa a i-ésima linha do matriz Q.

Lista de Figuras

1.1 Ilustração de uma lista de recomendações personalizadas no site Amazon.com. 2

1.2 Ilustração do gradiente descendente estocástico para a função $f(x) = x^2$: (a) usando $\alpha = 0,1$. (b) usando $\alpha = 0,01$ 6

2.1 Classificação resumida das estratégias de implementação de sistemas de recomendação, destacando a fatoração de matrizes. 13

2.2 Ilustração da projeção dos usuários e itens no espaço de fatores latentes. . . 14

2.3 Ilustração da Fatoração de Matrizes como solução para completar valores ausentes na matriz original. 16

4.1 Ilustração do gradiente descendente estocástico para a função $f(x) = x^2$: (a) usando $\alpha = 0,01$ (b) usando uma taxa de aprendizagem maior ($\alpha = 0,04$) para a primeira iteração. 29

4.2 Relação entre β e a porcentagem de iterações reduzidas. As linhas vermelhas mostram os modelos lineares ajustados a cada base de dados. 31

4.3 Relação entre β e a porcentagem de iterações reduzidas ao variar o número de fatores latentes. 32

5.1 Comportamento do RMSE na base Amazon com o passar das iterações do gradiente descendente estocástico usando a versão padrão e a nossa técnica. 40

5.2 Comparação da quantidade de iterações para atingir a convergência na base MovieLens-10M. 41

5.3 Comportamento do RMSE na a base Dating com o passar das iterações do gradiente descendente estocástico usando a versão padrão e a nossa técnica. 41

5.4 Comparação dos intervalos de confiança do RMSE nas 3 bases de teste. . . 44

5.5 Comparação da quantidade de iterações para atingir a convergência na base Amazon. 45
5.6 Comparação da quantidade de iterações para atingir a convergência na base MovieLens-10M. 45
5.7 Comparação da quantidade de iterações para atingir a convergência na base Dating. 46
5.8 Comparação de todas as técnicas na base Amazon. 46
5.9 Comparação de todas as técnicas na base MovieLens-10M. 47
5.10 Comparação de todas as técnicas na base Dating. 47

A.1 Histogramas das avaliações do usuários nas bases de dados MovieLens-100K e MovieLens-1M. 56
A.2 Histogramas das avaliações do usuários nas bases de dados MovieLens-10M e Yahoo-Movies. 57
A.3 Histogramas das avaliações do usuários nas bases de dados Netflix e Dating. 58
A.4 Histogramas das avaliações do usuários nas bases de dados Amazon e Epinions. 59
A.5 Distribuição dos graus da base de dados MovieLens-100K. 60
A.6 Distribuição dos graus da base de dados MovieLens-1M. 60
A.7 Distribuição dos graus da base de dados MovieLens-10M. 61
A.8 Distribuição dos graus da base de dados Yahoo-Movies. 61
A.9 Distribuição dos graus da base de dados Netflix. 61
A.10 Distribuição dos graus da base de dados Dating. 62
A.11 Distribuição dos graus da base de dados Amazon. 62
A.12 Distribuição dos graus da base de dados Epinions. 62

Lista de Tabelas

1.1 Exemplo de uma Matriz de Avaliações Usuário-Item de um Sistema de Recomendação de Filmes 3

4.1 Avaliação dos modelos lineares baseados na estatística R^2. Redução máxima indica a maior porcentagem de iterações reduzidas quando comparadas ao número de iterações usando a taxa de aprendizagem padrão. 33

5.1 Descrição das bases de dados utilizadas nos experimentos. A escala indica a faixa de valores que os usuários podem avaliar os itens. 35

5.2 Descrição dos metaparêmetros das técnicas de taxa de aprendizagem adaptativas utilizadas nos experimentos. 37

5.3 Predição da taxa de aprendizagem inicial para as bases de dados Amazon, MovieLens-10M e Dating, usando os modelos lineares obtidos das demais bases de dados. 39

5.4 Comparação da nossa técnica com as estratégias adaptativas. 43

B.1 Avaliação do *backtracking line search*. 64

Lista de Algoritmos

2.3.1 Gradiente Descendente para Fatoração de Matrizes. 18
B.1.1 Algoritmo dos métodos da estratégia *line search*. 64

Capítulo 1

Introdução

Neste capítulo, apresentamos o contexto e a motivação da pesquisa realizada nesta dissertação. Primeiramente, introduzimos os principais aspectos dos sistemas de recomendação. Em seguida, contextualizamos o modelo de fatoração de matrizes aplicado aos sistemas de recomendação e a motivação para investigação de métodos que acelerem a sua fase de treinamento. Então, a solução proposta nesta dissertação é apresentada e analisada em linhas gerais. Por fim, o capítulo é encerrado com uma descrição da organização dos demais capítulos que compõem esta dissertação.

1.1 Contexto

Diariamente, os consumidores deparam-se com inúmeras opções de produtos ou serviços. Sites de comércio eletrônico, redes sociais, entre outros serviços online, oferecem uma enorme quantidade de conteúdo, com o intuito de satisfazer as necessidades ou gostos dos diversos tipos de consumidores. Logo, sugerir opções relevantes tornou-se um ponto chave para aumentar a satisfação dos usuários, fidelizá-los ao serviço e, consequentemente, impulsionar os lucros nos negócios. Foi esse contexto que fomentou o surgimento dos sistemas de recomendação.

Sistemas de recomendação auxiliam os usuários, sugerindo itens que eles provavelmente gostariam de consumir. A finalidade é facilitar os processos de decisão que os usuários enfrentam, tais como, qual produto comprar, qual filme assistir ou qual música ouvir [21]. Geralmente, essas recomendações são personalizadas, isto é, usuários com preferências di-

ferentes receberão sugestões diferentes. Além disso, há também as recomendações não-personalizadas, que são mais simples de serem geradas, pois não levam em consideração as especificidades dos usuários. Frequentemente, as não-personalizadas são usadas em notícias online ou em jornais, por exemplo, a lista dos 10 melhores livros, CDs ou restaurantes de acordo com um algum critério predeterminado de seleção

Para as recomendações personalizadas, um perfil de preferências do usuário é criado baseado no seu comportamento passado, como seu histórico de compras, histórico de pesquisas ou de avaliações em produtos [41]. Elas constituem o centro das pesquisas em sistemas de recomendação e vêm sendo empregadas para as mais diversas aplicações. Empresas como Amazon.com [26], Google [10] e Yahoo! [33] adotaram sistemas de recomendação como parte fundamental nos seus negócios. Tipicamente, as recomendações personalizadas são apresentadas como ranking de itens, tal qual, a Figura 1.1 ilustra.

Figura 1.1: Ilustração de uma lista de recomendações personalizadas no site Amazon.com.

Um marco que concretizou a relevância dos sistemas de recomendação foi o desafio Netflix [6]. Visando aumentar seus lucros, a Netflix, serviço para assistir filmes via Internet, propôs um prêmio de 1 milhão de dólares a quem melhorasse seu sistema de recomendação em 10%. O desafio movimentou a comunidade científica, gerando avanços no que diz respeito à acurácia e à performance dos sistemas de recomendação.

Para traçar os perfis de preferências, os sites precisam registrar as atividades dos usuários. Geralmente, essas atividades podem ser representadas em uma grande tabela, na qual cada

linha representa o que o usuário clicou, comprou ou avaliou no site. Por exemplo, considere um serviço em que é possível avaliar filmes com notas de 1 a 5, como mostrado na Tabela 1.1. Nesse caso, as atividades são vistas como uma matriz, na qual as linhas representam os usuários, as colunas representam os filmes disponíveis no serviço e os valores das células representam as avaliações dos usuários nos filmes aos quais já assistiram.

	Matrix	**Titanic**	**Sr e Sra. Smith**	**Sexto Sentido**
Ana	?	5	?	3
Bruno	5	2	4	?
Carlos	3	?	?	5
Daniela	?	4	5	?

Tabela 1.1: Exemplo de uma Matriz de Avaliações Usuário-Item de um Sistema de Recomendação de Filmes

Note que um sistema de recomendação pode ser visto como um problema de completar os valores ausentes (representados pelo símbolo "?") numa matriz de avaliações Usuário-Item. Em outras palavras, predizer as notas que os usuários dariam aos itens se configura uma forma de recomendação. Por exemplo, predizer uma nota 5 a "Sr. e Sra. Smith" para Carlos seria uma forma de recomendar fortemente esse filme. Por outro lado, predizer uma nota 1 ou 2 indicaria que esse filme não está de acordo com o perfil de Carlos.

Na maioria das vezes, as notas/avaliações são tudo que o recomendador tem. Em outros casos, o recomendador utiliza dados externos relacionados aos itens avaliados (ex.: autor, diretor), mas esse não é o escopo dessa dissertação (mais detalhes no Capítulo 2). Contudo, um recomendador não se preocupa com as notas/avaliações propriamente ditas. Em vez disso, ele tenta capturar as preferências dos usuários de um modo geral, analisando características subentendidas dos itens avaliados [22].

Por exemplo, dado o cenário de filmes acima, um sistema de recomendação concluiria que Bruno gosta de filmes de fantasia e ação e não gosta de filmes de romance. Nesse caso, a característica subentendida foi o gênero, porém há outras características de filme, tais como, elenco, diretor ou se é baseado em fatos reais. Note que o número das características (ou dimensões) é muito menor do que o número de filmes existentes. Ao considerar essas

dimensões, um recomendador poderia rapidamente determinar se um usuário gostaria de um novo filme, bastaria comparar as dimensões do novo filme com o seu perfil. Essa ideia é chamada de redução de dimensionalidade.

Como dito anteriormente, alguns recomendadores coletam dados externos dos itens acessados para poder traçar o perfil do usuário. Um alternativa a essa abordagem é a chamada Filtragem Colaborativa, na qual são analisadas ações passadas, a fim de estabelecer conexões entre usuários e produtos, sem a necessidade de dados externos [20]. Os algoritmos de filtragem colaborativa se dividem basicamente em duas categorias: baseados em memória e baseados em modelo. Na primeira categoria, os dados da matriz de avaliações usuário-item 1.1 são usados diretamente para predizer avaliações, enquanto no segundo, a predição é feita por intermédio de um modelo abstrato aprendido dos dados da mesma matriz de avaliações (mais detalhes no Capítulo 2).

A fatoração de matrizes é uma técnica baseada em modelo que utiliza a ideia da redução de dimensionalidade. Como o próprio nome sugere, a técnica fatora a matriz original, exemplificada na matriz 1.1, em duas matrizes de dimensões menores, uma que relaciona usuários e as dimensões (também chamadas de fatores latentes) e outra que relaciona itens e as dimensões, de modo que o produto delas recrie uma versão densa da matriz original. Essas matrizes de dimensões menores são chamadas de matrizes de fatores latentes. Portanto, a fatoração de matrizes pode ser vista como uma forma de aproximação da matriz original (que, em casos reais, é bastante esparsa) a uma matriz densa, ou seja, com nenhum valor ausente.

Atualmente, a fatoração de matrizes é uma das formas de implementação de sistemas de recomendação mais bem sucedidas [24]. Ela combina acurácia, escalabilidade e implementação simples, fatores que justificam seu sucesso e popularidade. Além disso, o método vencedor do desafio Netflix supracitado utilizou a fatoração de matrizes como componente principal. A fatoração de matrizes também se destaca por ser flexível, dando margem à inúmeras variações [34; 17].

Assim como toda técnica baseada em modelo, um eventual obstáculo é a fase de treinamento do modelo, a qual pode demandar muito tempo. Deste modo, se o treinamento for muito custoso em termos de tempo, implementar sistemas de recomendação com a fatoração de matrizes deixa de ser uma alternativa atrativa, apesar das vantagens apresentadas.

1.2 Definição do Problema

Apesar das muitas vantagens da fatoração de matrizes, a fase de treinamento é o seu ponto fraco e uma das razões pela qual se torna menos atrativa. Suponha um sistema de recomendação real, cujo treinamento do modelo de fatoração de matrizes durou algumas horas e de forma *offline*, ou seja, ocorreu sem comprometer o serviço oferecido. No entanto, com a chegada de dados novos, tais como novos usuários, novos itens ou novas avaliações, o sistema tende a ficar desatualizado rapidamente. Logo, repetidos treinamentos devem ser realizados.

Esse problema se agrava em se tratando de cenários *big data*, condição das grande empresas da Internet, como a Amazon.com. Nesses cenários, a quantidade de dados novos é enorme e, consequentemente, a fase de treinamento seria ainda mais lenta e a necessidade de novos treinamentos é constante. Portanto, acelerar a fase de treinamento da fatoração de matrizes é um problema relevante, pois afeta muitas empresas/serviços, bem como seus usuários.

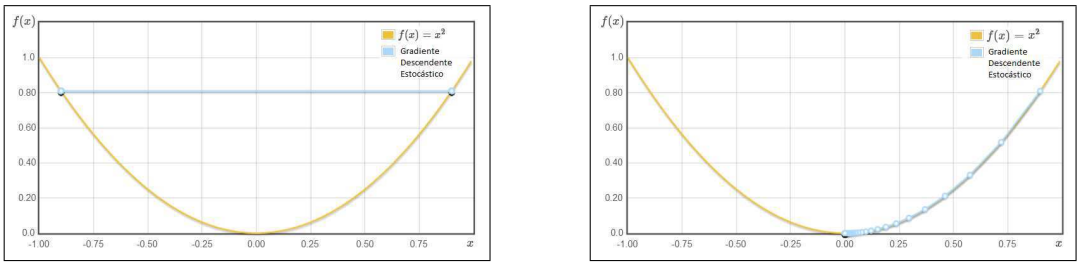
Como mencionado acima, a fatoração de matrizes pode ser considerada um problema de aproximação, no qual o objetivo é obter matrizes de fatores latentes, dada uma matriz de avaliações esparsa. Isso é obtido ao definir uma função objetivo (também chamada de função de perda) e treinar os fatores latentes das matrizes de forma a alcançar o mínimo global ou local da função. Esse treinamento é realizado com técnicas de otimização.

O gradiente descendente estocástico tornou-se o método padrão para o treinamento da fatoração de matrizes aplicada a sistemas de recomendação devido à sua eficiência para grande bases de dados e sua fácil implementação [24]. O gradiente descendente estocástico é um método de otimização de primeira ordem, ou seja, requer o cálculo das primeiras derivadas parciais (também chamadas de gradiente) da função objetivo a ser otimizada. Resumidamente, o método inicia com uma estimativa inicial e realiza atualizações em direção à negativa do gradiente, até atingir um mínimo local (convergência). A atualização da direção é proporcional à taxa de aprendizagem α, a qual indica o tamanho do passo que dever dado na direção calculada.

A convergência do gradiente descendente estocástico é sensível à escolha da taxa de aprendizagem [42]. Se o valor de α for grande, o método pode oscilar e nunca atingir a

convergência, ou seja, um mínimo local. Por outro lado, se o valor for muito pequeno, o método pode levar várias iterações, resultando em treinamento lento.

A Figura 1.2 mostra o comportamento do gradiente descendente estocástico para função convexa $f(x) = x^2$ para dois valores de α. Em ambos os casos, a estimativa inicial do mínimo local é a mesma $0,8$ (note que, por ser convexa, a função só tem um mínimo que é o global). Contudo, no primeiro caso (Figura 1.2a), o método diverge, pois o tamanho do passo é grande e sempre perde o mínimo global; enquanto no segundo (Figura 1.2b), o método converge após várias iterações, pois o tamanho do passo é pequeno (também chamado de conservador), garantindo a convergência para o mínimo global.



(a) Taxa de Aprendizagem Grande ($\alpha = 0,1$) (b) Taxa de Aprendizagem Pequena ($\alpha = 0,01$)

Figura 1.2: Ilustração do gradiente descendente estocástico para a função $f(x) = x^2$: (a) usando $\alpha = 0,1$. (b) usando $\alpha = 0,01$

Definir um valor adequado para a taxa de aprendizagem não é uma tarefa trivial, uma vez que valores grandes prejudicarão a acurácia do modelo, enquanto valores pequenos podem tornar o treinamento lento. No que diz respeito à fatoração de matrizes, a taxa de aprendizagem é configurada para valores pequenos (tipicamente $0,01$ ou $0,02$ [11]), a fim de garantir que o gradiente descendente estocástico não perca um mínimo local. Consequentemente, seu treinamento pode levar várias iterações. Por exemplo, Rendle e Schmidt-Thieme [37] reportaram que a fatoração de matrizes para a predição de ratings nos dados do desafio Netflix precisou de 200 iterações, usando uma taxa de aprendizagem de 0,01.

Geralmente, a taxa de aprendizagem é definida com um valor fixo por meio de uma busca em valores candidatos, usando a validação cruzada, que é uma técnica muito usada para o treinamento de parâmetros e hiperparâmetros na aprendizagem de máquina. Para cada valor candidato, os parâmetros do modelo são treinados - um processo que pode levar horas em problemas de larga escala. Depois de aprender vários modelos, o valor candidato que resulta

em um modelo mais acurado é escolhido [36].

Com o intuito de evitar essa situação, há trabalhos que adaptam o valor da taxa de aprendizagem durante o treinamento [27]. Uma taxa de aprendizagem adaptativa tentará manter o tamanho do passo tão grande quanto possível, porém mantendo a aprendizagem estável para evitar a divergência.

O cenário ideal seria definir uma taxa de aprendizagem que conduzisse à um mínimo local em poucas iterações. Como já foi dito, essa tarefa é difícil dada a grande quantidade de candidatos a serem pesquisados. No entanto, se após o primeiro passo, o algoritmo ficar próximo de um mínimo local, então restará poucas iterações até a convergência e, consequentemente, o treinamento será acelerado.

Não identificamos na literatura estudos sobre a identificação de bons valores iniciais para a taxa de aprendizagem com o objetivo de minimizar o número de iterações. Nesta dissertação, partimos do pressuposto que há uma relação intrínseca entre a taxa de aprendizagem e o número de iterações da fatoração de matrizes aplicada sistemas de recomendação. Portanto, nossa hipótese é: "A estimativa de um bom primeiro passo para o gradiente descendente estocástico resulta em menos iterações durante o treinamento da fatoração de matrizes para o problema da predição de notas em sistemas de recomendação.";

Em resumo, o problema que essa dissertação visa analisar tem os seguintes aspectos:

- **Problema de Negócio**: Acelerar o treinamento da fatoração de matrizes para o problema da predição de notas aplicada a sistemas de recomendação.
- **Problema Técnico**: Definir um valor inicial para a taxa de aprendizagem do gradiente descendente estocástico que minimize o número de iterações necessárias para atingir a convergência.

1.3 Resultados e Contribuições

Nesta dissertação, investigamos a relação entre o valor da taxa de aprendizagem do gradiente descendente estocástico e o número de iterações necessárias para atingir a convergência do modelo da fatoração de matrizes. Comparamos a acurácia dos modelos e o número de iterações durante o treinamento entre a nossa proposta, a abordagem tradicional, que utiliza

uma taxa de aprendizagem fixa e as abordagens adaptativas presentes na literatura. Utilizamos 8 bases de sistemas de recomendações reais de domínios diferentes para realizar as avaliações. A seguir destacamos as principais contribuições e resultados encontrados.

1. **Observação da relação linear entre a taxa de aprendizagem inicial e número de iterações**. Observamos que, para a maioria da bases de dados que foram utilizadas, há uma relação linear positiva entre a taxa de aprendizagem inicial e o número de iterações necessárias para a convergência.

2. **Uma técnica para estimar o valor inicial da taxa de aprendizagem para novas bases de dados**. A partir dessa observação, propomos um modelo linear para predizer a taxa de aprendizagem para novas bases de dados. Verificamos que podemos reduzir o número de iterações em 40% quando comparadas à abordagem tradicional.

3. **Avaliação de abordagens que propõem a taxa de aprendizagem adaptativa**. Estendemos o trabalho de Luo et al. [27] ao comparar suas abordagens com técnicas *annealing* e *bold driver* em diferentes bases de dados utilizados no seu trabalho original. Além disso, comparamos as 6 técnicas adaptativas com a técnica proposta nesta dissertação. Observamos que a nossa técnica também supera as técnicas adaptativas e que técnicas simples como o *bold driver* são competitivas quanto às técnicas presentes em Luo et al. [27].

As contribuições 1 e 2 também foram publicadas no artigo [30] e os resultados preliminares da contribuição 3 estão no artigo [31], o qual é uma extensão do artigo [30].

1.4 Estrutura da Dissertação

O conteúdo restante desta dissertação está organizado da seguinte maneira:

- **Capítulo 2 - Fundamentação Teórica.** Esse capítulo versa sobre os principais conceitos que embasam essa dissertação.

- **Capítulo 3 - Trabalhos Relacionados.** Esse capítulo aborda os trabalhos relacionados à aceleração do treinamento do modelo de fatoração de matrizes aplicada aos sistemas de recomendação.

- **Capítulo 4 - Uma Abordagem para Estimar o Valor Inicial da Taxa de Aprendizagem.** Nesse capítulo, aprensetamos uma técnica para a aceleração do treinamento do modelo de fatoração de matrizes.

- **Capítulo 5 - Avaliação Experimental.** Esse capítulo descreve as bases de dados utilizadas e a execução dos experimentos.

- **Capítulo 6 - Conclusão e Trabalhos Futuros.** Esse capítulo apresenta as conclusões e contribuições dessa dissertação e os pontos que permanecem como trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo, apresentamos a fundamentação teórica necessária para o entendimento do nosso trabalho. Inicialmente, mostramos os conceitos básicos relativos aos sistemas de recomendação e uma visão geral das estratégias existentes de implementação (Seção 2.1). Em seguida, apresentamos a fatoração de matrizes aplicada ao contexto de sistemas de recomendação (Seção 2.2) e, por fim, sua relação com o método de otimização do Gradiente Descendente Estocástico (Seção 2.3).

2.1 Principais Conceitos de Sistemas de Recomendação

Sistemas de recomendação são ferramentas que sugerem itens que provavelmente o usuário gostaria de consumir. Os mais populares envolvem a recomendação de filmes, músicas, notícias e produtos em geral [13].

Basicamente, os sistemas de recomendação abordam dois tipos de problemas: predição de avaliações e predição de itens. No primeiro caso, o qual é o escopo desta dissertação, é realizada uma predição de um valor numérico, que representaria a preferência do usuário por um determinado item. Por exemplo, em um serviço de filmes, no qual o usuário dá notas de 1 a 5 aos filmes vistos, o sistema irá predizer notas a filmes ainda não vistos pelo usuário, baseado no seu histórico de notas. Portanto, nesses casos, as recomendações são baseadas no *feedback* explícito dos usuários, expressado por suas avaliações/notas.

Para o segundo tipo de problema, há uma sugestão de itens que provavelmente teria utilidade para o usuário. Por exemplo, a recomendação de uma lista com os 10 filmes (top-10)

com maior chance de interesse, com base no histórico de filmes já assistidos pelo usuário. Nos casos em que as avaliações dos usuários não estão disponíveis, o sistema de recomendação infere as preferências por meio de *feedback* implícito, o qual é extraído a partir de iterações dos usuários com o sistema, tais como, itens acessados, histórico de compra, histórico de pesquisa, entre outros.

O escopo desta dissertação é sistemas de recomendação para o problema da predição de notas, o qual pode ser formalmente definido da seguinte forma. Sejam:

- U, o conjunto de usuários;
- I, o conjunto de itens;
- A, o conjunto de possíveis valores de avaliação, por exemplo $A = \{1, 2, 3, 4, 5\}$;
- $r_{u,i}$, avaliação/nota do usuário u ao item i, onde $u \in U$, $i \in I$;
- R, o conjunto de avaliações realizadas pelos usuários, tal que $R = \{r_{u,i} \in \mathbb{R}_{\geq 0} \mid u = 1, 2, ..., |U|,\ i = 1, 2, ..., |I|\}$
- Função $f : U \times I \rightarrow \mathbb{R}_{\geq 0}$, para a definição de uma predição $\hat{r}_{u,i}$;
- Função objetivo $l : R \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$, onde $l(r_{u,i}, \hat{r}_{u,i})$ quantifica quão ruim é a predição $\hat{r}_{u,i}$ comparada à real avaliação $r_{u,i}$.

Dadas partições de treino R^{treino} e teste R^{teste}, tal que $R^{\text{treino}} \cup R^{\text{treino}} = R$, podemos definir a seguinte função:

$$erro(f; R^{\text{teste}}) = \frac{1}{|R^{\text{teste}}|} \sum_{r_{u,i} \in R^{\text{teste}}} l(r_{u,i}, \hat{r}_{u,i}) \tag{2.1}$$

Logo, o problema da recomendação para a predição de notas consiste em minimizar a função erro, como descrito abaixo:

$$\min erro(f; R^{\text{teste}}) \tag{2.2}$$

Quanto às abordagens de implementação, os sistemas de recomendação são normalmente classificados em duas subclasses [1]:

- Baseada em Conteúdo
- Filtragem Colaborativa

Na abordagem baseada em conteúdo, o sistema de recomendação extrai os atributos (conteúdos) dos itens mais bem avaliados do histórico do usuário e constrói um perfil com suas preferências de conteúdo [2]. Então, sugere itens com atributos semelhantes ao seu perfil. Por exemplo, no contexto de filmes, os atributos são: atores, diretor, gênero, etc. Se o usuário dá notas altas a filmes de ação, mais filmes desse gênero serão recomendados. Essa abordagem não necessita de uma grande base de usuários para funcionar, pois basta que os atributos dos itens estejam disponíveis para gerar recomendações. Porém, dois problemas surgem: os atributos podem estar indisponíveis ou a extração pode não ser trivial; e o efeito colateral do sistema ficar muito especializado no perfil do usuário, de forma que falha ao não recomendar itens diferentes, mas que o usuário gostaria de ter.

Diferentemente da abordagem baseada em conteúdo, a abordagem de filtragem colaborativa baseia-se na avaliação dos demais usuários do sistema, identificando usuários com preferências similares ao usuário-alvo. O termo colaborativo decorre justamente do fato de não usar informações externas, apenas os itens avaliados/acessados pelos usuários. A ideia central é que usuários que gostaram de itens semelhantes no passado, gostarão de itens semelhantes no futuro [12]. Em outras palavras, se dois usuários deram notas similares aos mesmos itens, então provavelmente continuarão a dar notas similares a itens novos.

A filtragem colaborativa por sua vez pode ser dividida em duas sub-categorias: algoritmos baseados em memória e algoritmos baseados em modelos. Para a primeira categoria, são calculadas similaridades entre os usuários ou entre itens, com a finalidade de ponderar as avaliações no cálculo da predição. Nessa categoria, não há fase de treinamento, porém todos os dados devem estar na memória para realizar o cálculo da predição. Um ponto positivo é a facilidade que esta estratégia possibilita ao explicar as predições/recomendações.

Os algoritmos de filtragem colaborativa baseados em modelo necessitam de uma fase de treinamento, na qual um modelo é aprendido a partir de dados observados e as predições são feitas, usando os parâmetros que melhor ajustam o modelo. A vantagem de uma abordagem baseada em modelo é que uma vez o modelo é aprendido, as predições podem ser feitas até em tempo linear, porém o treinamento do modelo pode ser bastante custoso.

O ponto forte da filtragem colaborativa é a capacidade de sugerir itens que surpreendam os usuários. Por exemplo, ainda no contexto de filmes, um usuário que tem um perfil voltado a filmes de ação, pode receber uma recomendação de um filme de suspense. Isso se deve

ao fato de que outros usuários com perfil similar, ou seja, que gostam de filmes de ação, também gostaram do filme de suspense. Por outro lado, o grande número de usuários e itens gera problemas de escalabilidade, ou seja, a identificação de usuários com preferências similares fica muito custosa.

Um problema comum à maioria dos sistemas de recomendação é a esparsidade dos dados, devido ao fato de que os usuários normalmente avaliam ou acessam apenas uma pequena proporção dos itens disponíveis. Em sistemas reais, mesmo usuários ativos avaliam bem menos que 1% dos itens [40]. Outro problema relacionado à esparsidade dos dados é o chamado *cold-start*. Como as abordagens baseiam-se em preferências do passado, novos usuários que não consumiram/acessaram itens suficientes para que o sistema capture suas preferências (definição de um cenário *cold-start*), não receberão recomendações acuradas.

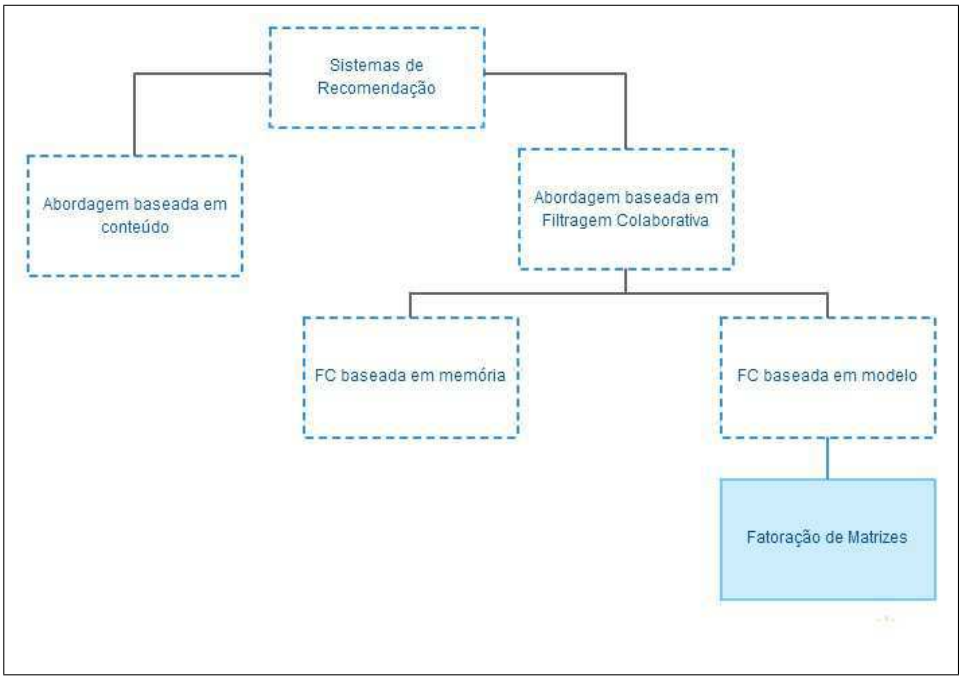


Figura 2.1: Classificação resumida das estratégias de implementação de sistemas de recomendação, destacando a fatoração de matrizes.

Por fim, a literatura ainda cita abordagens híbridas, formuladas com o intuito de unir as vantagens das abordagens baseada em conteúdo e filtragem colaborativa [5; 25]. Levando em consideração essa classificação, a fatoração de matrizes se encaixa como uma abordagem de filtragem colaborativa baseada em modelos, tal qual mostra a Figura 2.1, e será detalhada na próxima seção.

2.2 Fatoração de Matrizes

Como foi dito no Capítulo 1, um sistema de recomendação pode ser visto como um problema de completar os valores ausentes numa matriz de avaliações Usuário-Item, exemplificada na Tabela 1.1. Note que usuários e itens estão relacionados no espaço de avaliações.

Métodos de redução de dimensionalidade constituem uma das formas de analisar dados matriciais. A premissa implícita desses métodos é que os aspectos importantes dos dados podem ser capturados por meio de uma representação de menor dimensão. No nosso caso, a premissa é que apenas um pequeno número de fatores (características) dos itens influenciam as preferências dos usuários, de forma que a preferência de um usuário por um item é determinada por quanto cada fator se aplica/ajusta ao usuário e ao item. Para estabelecer esses ajustes de usuários e itens aos fatores, há uma projeção deles no espaço dos fatores.

A Figura 2.2 ilustra a projeção de usuários e itens da matriz do nosso exemplo (Tabela 1.1) para um espaço simplificado de duas dimensões, ou seja, dois fatores latentes. Como o exemplo trata de filmes, os fatores hipotéticos dos itens são: quantidade de romance no filme e quantidade de ação no filme. Logo, os fatores dos usuários são: grau de interesse em filmes com romance e grau de interesse em filmes com ação. Por exemplo, podemos perceber que Daniela poderá gostar de “Matrix”, enquanto que poderá não gostar de “Sexto Sentido”.

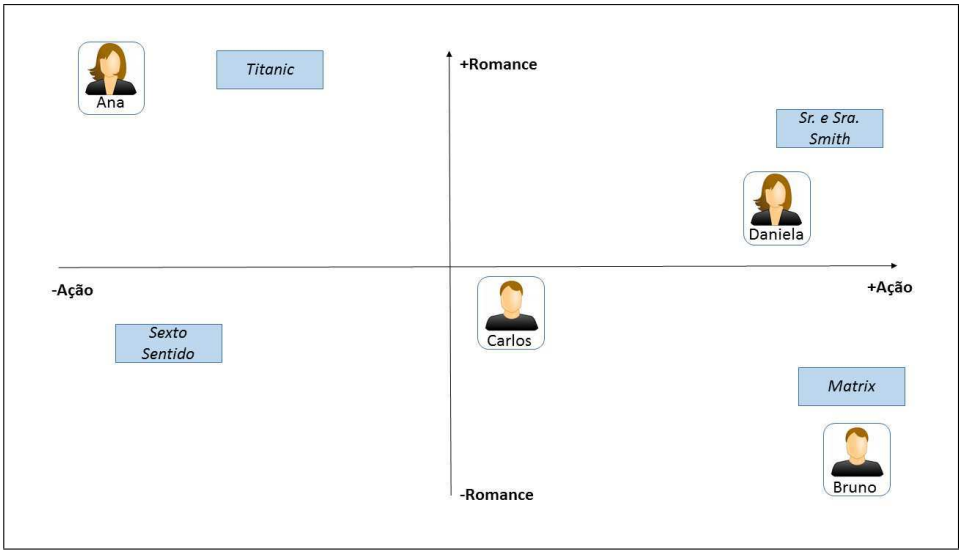


Figura 2.2: Ilustração da projeção dos usuários e itens no espaço de fatores latentes.

Os valores ausentes na matriz original indicam, no espaço de avaliações, onde não há

relação entre usuário e item. Com a projeção de usuários e itens no espaço denso de características latentes, novas interações entre usuários e itens podem ser estabelecidas, consequentemente, predições para os valores ausentes podem ser estimadas. Além disso, relações mais significativas podem ser descobertas, tais como, relações entre dois usuários, embora esses tenham avaliado itens diferentes [13].

De fato, a fatoração de matrizes é um caso especial da redução de dimensionalidade, no qual a matriz de dados observados R é decomposta no produto de duas matrizes densas $P \in \mathbb{R}^{|U| \times k}$ e $Q \in \mathbb{R}^{|I| \times k}$, onde k indica o número de fatores latentes e, tipicamente, $k \ll |U|$ e $k \ll |I|$. Como foi explicado acima, a matriz R indica o espaço de avaliações. Por consequência, as matrizes P e Q indicam a projeção no espaço de fatores latentes, de modo que as relações usuário-item são modeladas como produtos escalares nesse espaço.

Após a fatoração de R, cada usuário u estará associado a um vetor $p_u \in \mathbb{R}^k$ e cada item i estará associado a um vetor $q_i \in \mathbb{R}^k$. Note que cada elemento dos vetores corresponderá a um fator latente. O produto escalar dos vetores de fatores latentes p_u e q_i captura a interação entre usuário u e item i, de forma que $R \approx P \times Q^T$. A predição é dada da seguinte forma:

$$f(u,i) = \hat{r}_{u,i} = p_u \cdot q_i^T = \sum_{f=1}^{k} p_{u,f} \times q_{f,i} \tag{2.3}$$

A Figura 2.3 ilustra um exemplo simplificado da fatoração de matrizes para um espaço de fatores latentes de dimensão 3. A matriz original contém as avaliações dos usuários (representadas pelas células verdes) e em destaque está o valor que será predito (célula vermelha). As linhas em destaque representam como seria calcular a predição que usuário 1 (representado pela primeira linha da matriz P) daria ao item 3 (representado pela terceira coluna da matriz Q), ou seja, calcular o $\hat{r}_{1,3}$.

Dada a finalidade de aproximar a matriz original R à matriz resultante do produto de P e Q, a fatoração de matrizes pode ser formalizada como um problema de otimização da seguinte forma, reutilizando a Equação 2.1:

$$\underset{P,\ Q}{\operatorname{argmin}}\, erro(f; R) \tag{2.4}$$

A formulação acima responde a seguinte questão: "Quais são os melhores valores de P e Q, tal que o erro entre os valores observados R e suas respectivas predições calculadas

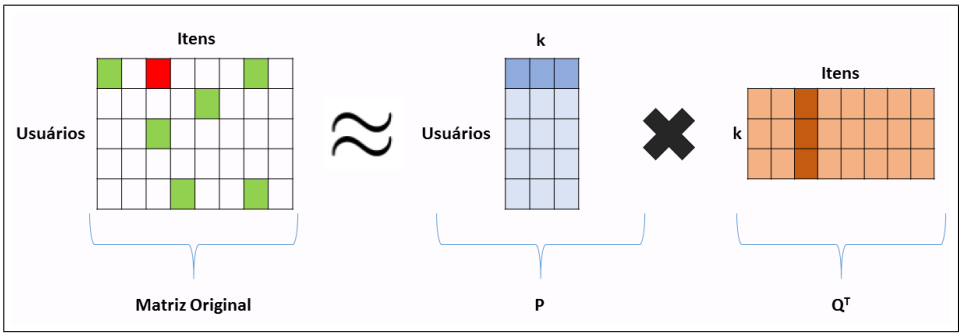


Figura 2.3: Ilustração da Fatoração de Matrizes como solução para completar valores ausentes na matriz original.

por $P \times Q^T$ seja o menor possível?". Portanto, o treinamento do modelo da fatoração de matrizes consiste em achar os parâmetros P e Q, de acordo com a Fórmula 2.4. Para o problema de predição de avaliações, a função objetivo l normalmente utilizada é a soma dos erros quadráticos (ou SSE, do inglês *Sum of Squared Errors*), descrita na Equação 2.5. Note que, como a matriz R é esparsa, o erro é calculado apenas em relação aos valores observados.

$$l(r_{u,i}, \hat{r}_{u,i}) = SSE = \sum_{r_{u,i} \in R} e_{u,i}^2 = \sum_{r_{u,i} \in R} (r_{u,i} - p_u \cdot q_i^T)^2 \tag{2.5}$$

onde $e_{u,i}$ definido abaixo:

$$e_{u,i} = r_{u,i} - \hat{r}_{u,i} \tag{2.6}$$

onde $r_{u,i} \in R$ e $\hat{r}_{u,i}$ é a predição calculada conforme 2.3.

O problema de otimização original 2.4 foi sofisticado por Simon Funk [15] com a incorporação de um termo de regularização. Em Estatística, o método de regularização incorporado por ele é chamado de regularização de Tikhonov. Resumidamente, a regularização visa evitar que o modelo fique super ajustado aos dados de treinos (*overfitting*), penalizando a magnitude dos parâmetros dos seus parâmetros. No nosso caso, os dados de treino são os valores observados da matriz R e os parâmetros do modelo são os fatores latentes das matrizes P e Q. A Equação 2.7 define a nova função objetivo ao incorporar o termo de regularização:

$$SSE\_Reg = \sum_{r_{u,i} \in R} e_{u,i}^2 + \lambda(||p_u||^2 + ||q_i||^2) \tag{2.7}$$

onde a constante λ controla o peso da regularização.

Para encontrar os parâmetros que otimizem a função objetivo 2.7, não há fórmula fechada (ou solução analítica). Consequentemente, é necessária a utilização de algoritmos de otimização iterativos. Dentre as classes de algoritmos de otimização, estão os algoritmos de primeira ordem, os quais requerem o cálculo das derivadas parciais (gradientes) da função objetivo que está sendo otimizada. Exemplos incluem Gradiente Descendente, Gradiente Descendente Estocástico e Gradiente Descendente Conjugado.

Simon Funk [15] popularizou o Gradiente Descendente Estocástico como o algoritmo padrão para otimização da função objetivo 2.7 e este tem sido aplicado com sucesso em vários outros trabalhos [23; 34; 39; 43], por isso ele será o padrão para o treinamento da fatoração de matrizes neste trabalho também.

2.3 Gradiente Descendente Estocástico

Como mencionado anteriormente, o gradiente descendente é uma forma iterativa de otimizar uma função objetivo. Em outras palavras, é uma forma de encontrar os mínimos locais de uma determinada função objetivo. Caso a função seja convexa, haverá um único mínimo global, caso contrário, haverá vários mínimos locais [29].

Suponha que nosso objetivo é minimizar uma função $f(x)$. O gradiente descendente inicia com uma estimativa de solução x_0, ou seja, um palpite para o mínimo local. Para descobrir qual é a melhor direção para otimizar $f(x)$, calculamos o gradiente nesse ponto. Intuitivamente, o gradiente vai dar a inclinação da função em x_0 e a sua direção irá apontar para onde a função cresce mais rapidamente. Visto que o nosso problema é de minimização, é tomado o negativo do gradiente. O algoritmo irá eventualmente convergir quando o gradiente for zero (ou aproximadamente zero), o que corresponde a um mínimo local. A Equação 2.8 indica a atualização durante a iteração t:

$$x^t = x^{t-1} - \alpha \frac{\partial}{\partial x} f(x) \tag{2.8}$$

onde α é taxa de aprendizagem do gradiente descendente, que corresponde ao tamanho do passo dado na direção gradiente.

É importante destacar a diferença entre o gradiente descendente padrão e o estocástico, visto que o último será o adotado nesse trabalho. Enquanto no gradiente descendente padrão

é preciso passar por todas as amostras do conjunto de treinamento para fazer uma única atualização de um parâmetro em uma iteração particular, o gradiente descente estocástico, por outro lado, precisa apenas de uma amostra de seu conjunto de treinamento para realizar a atualização em uma iteração particular. Isso faz com que o gradiente descendente estocástico convirja mais rapidamente que o gradiente descente, mantendo resultados aproximados. Por essas razões, em se tratando de grandes bases de dados, o gradiente descendente estocástico é o preferido.

No que diz respeito ao treinamento do modelo de fatoração de matrizes, o gradiente descendente estocástico funciona conforme o algoritmo 2.3.1. O primeiro passo do algoritmo é dar estimativas iniciais às matrizes P e Q. Então, iterações são realizadas para atualizar os vetores de fatores latentes que as compõem, no sentido de minimizar a função de perda 2.7. Tipicamente, o critério de parada do algoritmo é atingido quando a diferença entre os erros da função de perda de duas iterações consecutivas é desprezível, o que indica que o algoritmo convergiu.

Algoritmo 2.3.1 Gradiente Descendente para Fatoração de Matrizes.

1: **procedure** OTIMIZAÇÃO(R, P, Q, k, α, λ)
2: Inicializa P e Q
3: **repeat**
4: **for** $cada\ r_{u,i} \in R$ **do**
5: **for** $f = 1 \rightarrow k$ **do**
6: $p_{u,f} \leftarrow p_{u,f} + \alpha \cdot \frac{\partial}{\partial p_{u,f}}(SSE\_Reg)$
7: $q_{i,f} \leftarrow q_{i,f} + \alpha \cdot \frac{\partial}{\partial q_{i,f}}(SSE\_Reg)$
8: **until** critério de parada atingido

Para cada caso de treino $r_{u,i}$, calculam-se a predição e o erro associados conforme as equações 2.3 e 2.6, respectivamente. Então, os fatores latentes são atualizados proporcionalmente à α e na direção oposta dos gradientes descritos abaixo (demonstração presente em [15]):

$$\frac{\partial}{\partial p_{u,f}}(SSE\_Reg) = -e_{u,i} \cdot q_{i,f} + \lambda \cdot p_{u,f} \tag{2.9}$$

$$\frac{\partial}{\partial q_{i,f}}(SSE\_Reg) = -e_{u,i} \cdot p_{u,f} + \lambda \cdot q_{i,f} \tag{2.10}$$

Assintoticamente, o treinamento da fatoração de matrizes tem a seguinte ordem de complexidade temporal:

$$O(\#iter \cdot k(|U| + |I|)) \tag{2.11}$$

ou seja, depende da quantidade de iterações $\#iter$ para convergir, da dimensão do espaço de fatores latentes k e da quantidade de usuários e itens, respectivamente.

Note que é preciso definir os valores de k, α e λ antes do treinamento propriamente dito. Esses são chamados de hiperparâmetros e são tipicamente definidos por meio de *grid search* e validação cruzada. O *grid search* consiste numa técnica de busca exaustiva em valores candidatos, no nosso caso, valores candidatos de hiperparâmetros. O algoritmo de busca do *grid search* necessita ser guiado por alguma métrica, tipicamente medida por validação cruzada, que, por sua vez, é uma técnica para estimar o quão acurado é um modelo. O objetivo da validação cruzada é definir uma partição de dados de teste para avaliar o modelo na fase de treinamento (ou seja, utilizar um conjunto de dados de validação), a fim de limitar problemas como *overfitting* e dar uma ideia de como o modelo vai se comportar a um conjunto de dados desconhecido.

O gradiente descendente estocástico é altamente sensível à escolha desses hiperpâmetros, visto que escolhas inadequadas acarretarão em um modelo de baixa acurácia. De modo especial, está a taxa de aprendizagem α, foco deste trabalho. Como mencionado no Capítulo 1, a taxa de aprendizagem influencia a convergência do modelo, de modo que se for muito grande causará divergência, enquanto se for muito pequeno causará uma convergência lenta. O gradiente descendente estocástico padrão define um valor fixo para a taxa de aprendizagem que irá assegurar a convergência qualquer que seja a estimativa inicial.

Capítulo 3

Trabalhos Relacionados

Neste capítulo, são abordados diversos trabalhos que tratam do problema de acelerar o treinamento da Fatoração de Matrizes. A literatura apresenta várias formas de atacar esse problema. Podemos agrupar esses trabalhos nas seguintes categorias: métodos para encontrar valores ótimos para os hiperparâmetros (seção 3.1); e métodos para a criação de versões paralelizadas e distribuídas do gradiente descendente estocástico (seção 3.2). Analisamos esses trabalhos com o objetivo de destacar as semelhanças e diferenças em relação ao trabalho apresentado nesta dissertação.

Os trabahos apresentados neste capítulo foram encontrados por meio de pesquisas nas principais conferências de Aprendizagem de Máquina e Mineração de Dados (ICML, NIPS, ECML, AAAI, SIGIR, WWW), além da bibliotecas digitais da IEEE e da ACM e sites de busca como Google Acadêmico.

3.1 Estratégias para a Determinação de Hiperparâmetros

Recentemente, foi mostrado que o estado da arte em algumas áreas, tais como, classificação de imagens, pode ser aprimorado com uma configuração adequada dos hiperparâmetros dos modelos ao invés da criação de novos paradigmas [3; 7]. Esses resultados mostram a relevância e a tendência em pesquisar a otimização de hiperparâmetros.

Assim sendo, o sucesso dos modelos de fatoração de matrizes também depende da escolha adequada dos seus hiperparâmetros: o número de fatores latentes k, a taxa de aprendizagem α, e o termo de regularização λ. Na verdade, a taxa de aprendizagem é um hiperparâ-

metro associado ao gradiente descendente. Como este é um método de otimização utilizado como sub-rotina em várias técnicas de aprendizagem, tais como, redes neurais artificiais, a maior parte dos avanços no que diz respeito à algoritmos baseados em gradiente deriva de outras áreas de pesquisa diferentes de sistemas de recomendação [28].

Como explicado no capítulo 2, os hiperparâmetros são tipicamente determinados por intermédio de uma busca em valores candidatos predefinidos. Um das formas de busca é chamada de *grid search*. Foi discutido que essa busca é um processo que consome bastante tempo porque requer que vários modelos sejam treinados. Contudo, o *grid search* juntamente com a validação cruzada se mantêm como a abordagem *default* para definição de hiperparâmetros, pois sua implementação é simples e sua paralelização é trivial.

Para o treinamento de redes neurais no contexto de classificação de imagens, Bergstra e Bengio [8] mostram que uma forma de busca randômica de escolha de valores candidatos é mais eficiente que o *grid search*. Ou seja, os valores candidatos que geram modelos equivalentes ou melhores que o *grid search* são achados mais rapidamente. A vantagem desse método é a flexibilidade de aplicá-lo aos diversos hiperparâmetros, pois quaisquer hiperparâmetros necessitam de uma busca em valores candidatos. A necessidade de repetidos treinamentos ainda permanece como desvantagem. Além de ser de outra área, a principal diferença desse trabalho para o apresentado nesta dissertação é que nosso foco é dizer um valor inicial de um hiperparâmetro, especificamente, a taxa de aprendizagem sem treinar vários modelos.

Em se tratando do hiperparâmetro de regularização λ, Rendle [36] propôs uma variação no método do gradiente descendente estocástico para que o λ seja aprendido de forma adaptativa, acelerando assim o tempo de treinamento do modelo. Ou seja, a cada iteração, após a atualização dos parâmetros do modelo, é feita uma atualização de λ. A ideia é resolver um problema de otimização para determinar o valor de λ que minimize o erro na próxima iteração. Da mesma forma que foi utilizado para a otimização do modelo, o gradiente descendente estocástico foi utilizado para otimização do termo de regularização. A abordagem é flexível o bastante para funcionar com funções de perda diferentes e com qualquer quantidade de hiperparâmetros de regularização.

Com relação ao hiperparâmetro do número do fatores latentes k, Yu et al. [45] elabora uma estratégia para defini-lo a partir dos dados, de forma que quanto maior a quantidade de

dados maior será a dimensionalidade.

A principal diferença desses 2 últimos trabalhos e o presente nesta dissertação é que o nosso foco é a taxa de aprendizagem α. No entanto, estratégias que adaptam os hiperparâmetros durante o treinamento se tornaram uma solução típica, pois evitam o método da tentativa e erro, como o *grid-search* e resultam em modelos com acurácia equivalentes.

Estratégias adaptativas também foram propostas para a taxa de aprendizagem α. Um dos métodos mais simples e amplamente utilizado na área de redes neurais é método do *momentum* [38]. Esse método não altera a taxa de aprendizagem em si, mas a sua intuição foi utilizada por vários outros métodos de taxa de aprendizagem adaptativa. A principal ideia é de acelerar a otimização ao adicionar uma fração do gradiente da iteração anterior na atualização do gradiente da iteração atual. Logo, quando os gradientes apontam na mesma direção, maior será o valor final do gradiente, acelerando a aproximação ao mínimo local. Da mesma forma, quando o sinal dos gradiente se alterar, retardará a otimização naquela direção do gradiente. Isto é feito ao armazenar as atualizações dos gradientes da iteração anterior.

Outro método que tem mostrado bons resultados na área de redes neurais é o chamado ADAGRAD [14]. Semelhante ao método do *momentum*, o ADAGRAD utiliza os valores dos gradientes de todas as iterações passadas. Reiteramos que os métodos do *momentum* e ADAGRAD não são estratégias para adaptar a taxa de aprendizagem em si, uma vez que utilizam uma taxa de aprendizagem fixa e global. Eles modificam a forma de atualização do gradiente, contudo, estão relacionados, pois utilizam informações dos gradientes para alterar o processo de otimização, durante o treinamento (vide [19] para o um resumo de outras técnicas adaptativas).

Observando o sucesso da taxa de aprendizagem adaptativa para o contexto de redes neurais, Luo et al. [27] incorporou 3 técnicas oriundas dessa área (e inspiradas no método do *momentum*) e um método próprio ao gradiente descendente estocástico, com o objetivo de obter sucesso para o contexto sistemas de recomendação, especificamente o problema da predição de avaliações. As três técnicas escolhidas foram DSSA (do inglês *Deterministic Step Size Adaptation*), IDBD (do inglês *Incremental Delta Bar Delta*), SMD (do inglês *Stochastic Meta Descent*) e seu método foi o CGA (do inglês *Gradient Cosine Adaptation*). Essas técnicas e outras duas chamadas *annealing* [9] e *bold driver* [18] e são detalhadas na

Subseção 3.1.1, pois serão utilizadas como comparação na avaliação experimental descrita no Capítulo 5.

Exceto Luo et al. [27], nenhum dos trabalhos relacionados lidam com fatoração de matrizes aplicadas a sistemas de recomendação. Além disso, esses trabalhos seguem uma abordagem *bottom-up* ao adaptar a taxa de aprendizagem baseado em valores anteriores, enquanto seguimos uma abordagem *top-down*, tentando prever, desde o início, a taxa de aprendizado que vai levar a um número mínimo de iterações.

3.1.1 Técnicas de Taxa de Aprendizagem Adaptativa

Como dito anteriormente, as técnicas de taxa de aprendizagem adaptativa foram propostas originalmente para a área de redes neurais. Duas técnicas amplamente utilizadas por serem simples de implementar são *annealing* e *bold driver*.

A técnica *annealing* decrementa uma taxa de aprendizagem global, especificamente, transforma a taxa de aprendizagem em função do tempo. A variação utilizada no Capítulo 5 é chamada de *search-then-converge*. Basicamente, mantém a taxa de aprendizagem em torno do valor inicial durante as primeiras τ iterações (fase *search*), com o intuito de achar uma localização geral do mínimo. Em seguida, decrementa a taxa de aprendizagem (fase *converge*) para garantir a convergência. O τ é um metaparâmetro dessa técnica. Apesar de utilizar a mesma taxa de aprendizagem global nos cálculos, a técnicas *annealing* pode ser considerada adaptativa, pois a cada iteração a taxa de aprendizagem de fato utilizada é diferente.

A técnica *bold driver* é um pouco mais sofisticada que a técnica anterior. Ela compara o erro após duas iterações consecutivas (lógica semelhante ao *momentum*). Se o erro diminuir, ocorre um pequeno incremento na taxa de aprendizagem (tipicamente 1% a 5%). Se o erro aumentar ocorre um decremento drástico (normalmente 50%). Assim, a taxa de aprendizagem cresce lentamente até que ocorra um passo que claramente se distancia do mínimo da função objetivo. Isto significa que a otimização chegou em uma área complicada da superfície de erro, onde faz sentido reduzir o tamanho do passo neste momento, adotando uma abordagem conservadora.

As técnicas DSSA, IDBD e SMD também foram propostas originalmente para a área de redes neurais. O que as diferencia das técnicas *annealing*, *bold driver* são: utilizar uma taxa

de aprendizagem independente para cada parâmetro da fatoração de matrizes (no caso do método CGA, uma taxa de aprendizagem para cada avaliação) e atualizar a taxa de aprendizagem baseada no gradiente das iterações anteriores.

A estratégia DSSA atribui uma taxa de aprendizagem específica para cada parâmetro $p_{u,k} \in P$ e $q_{i,k} \in Q$, e atualiza cada taxa de aprendizagem de acordo com os sinais de duas sucessivas direções na atualização do respectivo parâmetro. Como explicado no Capítulo 2 as direções são a derivada parcial da função objetivo 2.7 descrita no Capítulo 2. A ideia é penalizar a taxa de aprendizagem, se o sinal das sucessivas direções são distintos, multiplicando-o por um valor ligeiramente inferior a 1, o que diminui o tamanho do passo, tornando o aprendizado deste parâmetro mais lento. Por outro lado, se as sucessivas atualizações dos gradientes são feitas na mesma direção, então, a taxa de aprendizagem é multiplicada por um valor ligeiramente maior do que 1, aumentando assim o tamanho do passo e tornando a aprendizagem deste parâmetro mais rápido. Assim, são necessárias duas outras matrizes auxiliares para manter os últimos gradientes de cada parâmetro (uma para os gradientes dos parâmetros de P e outra para os parâmetro de Q) a serem comparados com os gradientes da iteração atual. Os metaparâmetros dessa estratégia controlam quão maior ou menor que 1 deve ser o valor da penalidade supracitada e são mencionados no Capítulo 5.

Na estratégia IDBD, também existe uma taxa de aprendizagem específica para cada parâmetro. Mas todas as taxas de aprendizagem são na forma exponencial (com o número de Euler e como base), o que conduz a duas vantagens. Primeiro, essa forma garante que a taxa de aprendizagem será sempre positiva. Segundo, é um mecanismo para execução de passos exponenciais, o que é uma estratégia desejável, porque algumas taxas de aprendizagem devem ser muito pequenas, enquanto outras permanecem grandes. A ideia central do IDBD é considerar o efeito de todos os valores de taxas de aprendizagem passadas para a atualização dos parâmetros. Isto é feito por duas matrizes auxiliares que mantêm todas as últimas gradientes de cada parâmetro.

Assim como as técnicas DSSA e IDBD, na estratégia SMD existe uma taxa de aprendizagem específica para cada parâmetro. Todas as taxas de aprendizagem são ajustadas em escala logarítmica e são otimizadas com o passar de um decaimento exponencial dos gradientes, de forma semelhante à estratégia IDBD. No entanto, diferentemente do IDBD onde cada taxa de aprendizagem leva em conta os valores anteriores, SMD leva em consideração os efeitos

de uma taxa de aprendizagem especificado nos outros parâmetros relacionados (parâmetros relacionados são aqueles que compartilham o mesmo usuário ou mesmo item). Neste caso, há também duas matrizes auxiliares para manter os gradientes de cada um dos parâmetros e os seus respectivos parâmetros. Os metaparâmetros dessa estratégia denotam um fator de desconto para penalizar iterações anteriores.

Por fim, na estratégia GCA, existe uma taxa de aprendizagem específica para cada avaliação da matriz original R. A atualização da taxa de aprendizagem é baseada na cosseno do ângulo entre as direções (gradientes) de duas iterações sucessivas. Note que esta estratégia é semelhante à estratégia DSSA, mas aqui, se o cosseno está perto de um (gradientes que apontam para a mesma direção) o tamanho do passo aumenta, ao mesmo tempo que diminui para valores mais baixos de o cosseno (gradientes que apontam para diferentes direções). O meta-parâmetro controla o peso do cosseno na regra de atualização dos parâmetros.

Uma desvantagem dessas técnicas apresentadas em Luo et al. [27], e também a principal diferença para a nossa abordagem, é que elas consideram várias taxas de aprendizagem (um para cada parâmetro), o que eleva a complexidade de espaço, uma vez que esta informação precisa ser armazenada durante o processo de aprendizagem. De fato, enquanto a nossa abordagem tem complexidade de espaço $O((|U|+|I|) \times K + |T|)$, onde é $|R|$ é a quantidade de avaliações, as abordagens DSSA, IDBD, SMD são na ordem de $O((|U| + |I|) \times 3K + |T|)$. A abordagem GCA requer armazenamento extra para a armazenar valores de taxa de aprendizagem para cada avaliação, além dos valores dos últimos gradientes de cada fator dos vetores de fatores latentes dos usuários e itens. Então, GCA tem complexidade de espaço $O((|U| + |I|) \times K + |T| \times (3 + 2k))$

3.2 Gradiente Descendente Estocástico Distribuído

Outra linha de pesquisa que merece ser mencionada está relacionada com implementações paralelizadas e distribuídas do gradiente descendente estocástico. No cenário *Big Data*, a fatoração de matrizes pode envolver milhões de usuários e itens, e bilhões de avaliações. Dada essa escala gigantesca, algoritmos de paralelização e distribuídos se tornaram essenciais para que o treinamento da fatoração de matrizes não seja um gargalo, prejudicando sua adoção em sistemas reais [44].

Em geral, algoritmos paralelizados são utilizados quando a quantidade de dados é pequena o suficiente para caber na memória principal de uma máquina com vários processadores. Ou seja, há criação de *threads* para atuar numa memória compartilhada. Por exemplo, Recht e Ré [35] obtém uma versão paralelizada do gradiente descendente estocástico ao particionar os dados (matriz original R) igualmente entre as *threads*, de modo que cada uma realizará as atualizações nos parâmetros paralelamente. Ao final, os resultados de cada *thread* são combinados, isto é, os parâmetros das matrizes P e Q que foram aprendidos separadamente em cada *thread*, devido à partição dos dados, são combinados. É evidente que a desvantagem é a incapacidade em lidar com problemas de larga escala, uma vez que a grande limitação é a memória compartilhada, com a qual se torna inviável tratar grandes bases de dados.

Já as versões distribuídas podem manipular bases de dados maiores ao agregar a memória e as várias CPUs dos múltiplos computadores (nós) que podem ser utilizados. Nessa estratégia, os algoritmos particionam tanto a matriz de dados, quanto as matrizes de fatores latentes (diferença entre os algoritmos paralelizados). Gemulla et al. [18] expressa a matriz original como uma união de blocos, cada um contendo uma porção das avaliações do usuários. Para cada nó, é atribuído um bloco da matriz original R e blocos correspondentes (conjunto de linhas e colunas) da matrizes de fatores latentes P e Q . Então, em cada nó, é definida e computada uma função de perda. Devido à essa partição de dados em blocos, é possível paralelizar e distribuir o treinamento, de forma que cada nó realizará o treinamento independentemente. A implementação baseia-se no *framework* de processamento distribuído MapReduce[1].

Zhou et al. [46] também utilizou o *framework* MapReduce. No entanto, o objetivo do trabalho consistiu em criar uma versão distribuída para algoritmo ALS (do inglês *Alternating Least Squares*), o qual é uma alternativa ao gradiente descendente estocástico para a aprendizagem do modelo. Apesar dos ganhos em performance ao tratar grandes bases de dados, o principal desafio dos algoritmos distribuídos consiste em gerenciar a comunicação entre os nós, no que diz respeito à atualização das matrizes P e Q (eventualmente nós distintos podem estar atualizando os mesmos parâmetros).

Note que esses trabalhos estão relacionados com o desta dissertação, pois compartilham

[1] http://www.columbia.edu/ ak2834/mapreduce.html

o mesmo problema de negócio, no que diz respeito a acelerar o treinamento da fatoração de matrizes (mas não necessariamente para o problema da predição de notas aplicada a sistemas de recomendação). Ou seja, na perspectiva de quem vai usar a pesquisa, ambos visam acelerar a fatoração de matrizes.

Por outro lado, o que os diferencia do trabalho desta dissertação e os demais da seção 3.1 é a questão da determinação dos hiperparâmetros. Os trabalhos propõem versões distribuídas do gradiente descendente estocástico *default*, ou seja, todos os hiperparâmetros não se adaptam durante o treinamento. Portanto, as versões distribuídas ainda podem incorporar as estratégias de adaptação de hiperparâmetros.

Capítulo 4

Um técnica para Estimar o Valor Inicial da Taxa de Aprendizagem

Neste capítulo, apresentamos a principal contribuição dessa dissertação, uma técnica para estimar o valor inicial da taxa de aprendizagem do gradiente descendente aplicada a fatoração de matrizes. Inicialmente, comentamos a intuição que norteia a técnica (Seção 4.1). Em seguida, expomos a técnica propriamente dita (Seção 4.2).

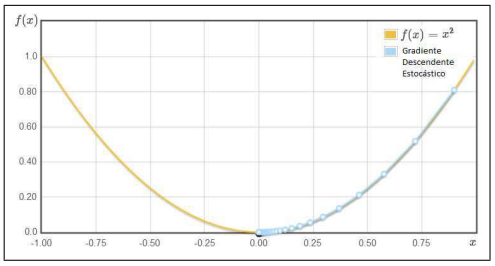
4.1 Intuição

Idealmente, gostaríamos de encontrar uma taxa de aprendizagem capaz de atingir um mínimo local (ponto de convergência) da função objetivo nas primeiras iterações do gradiente descendente estocástico. Dessa forma, ao poupar iterações, o processo de aprendizagem seria acelerado. Porém, essa não é uma tarefa trivial, dada a grande quantidade de valores de taxa de aprendizagem a serem testados. Além disso, cada base de dados pode necessitar de uma taxa de aprendizagem diferente [13].

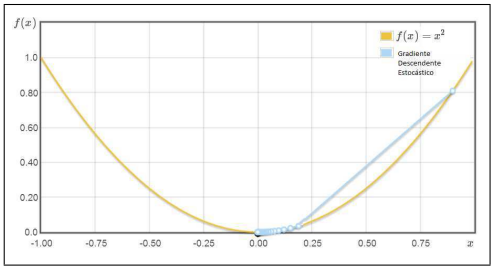
Dado o objetivo de diminuir o número total de iterações necessárias até a convergência, a ideia central da nossa técnica pode ser descrita da seguinte forma: determinar uma taxa de aprendizagem para ser usada apenas na primeira iteração e então, após essa primeira iteração, retornar a um valor conservador (ou padrão). Em outra palavras, na primeira iteração do processo de otimização, utilizamos uma taxa de aprendizagem para chegar o mais próximo possível de um mínimo local. Em seguida, voltamos a usar um tamanho de passo menor,

com o intuito de não ultrapassar o mínimo local.

A Figura 4.1 ilustra a intuição da nossa ideia de reduzir o número de iterações (representadas pelos pontos na curva) ao utilizar um bom valor inicial para a taxa de aprendizagem, ou seja, um valor que conduza ao mínimo local mais rapidamente. Comparamos com o mesmo exemplo de função $f(x) = x^2$ dado no Capítulo 1.



(a) α pequeno (padrão) para todas iterações.



(b) α maior para primeira iteração.

Figura 4.1: Ilustração do gradiente descendente estocástico para a função $f(x) = x^2$: (a) usando $\alpha = 0,01$ (b) usando uma taxa de aprendizagem maior ($\alpha = 0,04$) para a primeira iteração.

Perceba que ambos processos de otimização começam com a mesma estimativa inicial de $0,8$. Por estarem no mesmo ponto, ambos calculam o mesmo gradiente e, portanto, seguem na mesma direção rumo ao mínimo local. No entanto, ao dar um passo maior (taxa de aprendizagem maior), o processo da Figura 4.1b alcança, após a primeira iteração, uma determinada estimativa que o processo da Figura 4.1a só alcançará algumas iterações depois (daí em diante os comportamentos dos processos se assemelham por usarem a mesma taxa de aprendizagem). Logo, no processo da Figura 4.1b ocorreu uma economia de iterações e, consequentemente, um aceleramento do processo de otimização.

4.2 Predição da Taxa de Aprendizagem

O problema técnico dessa dissertação é predizer um estimativa de valor inicial da taxa de aprendizagem do gradiente descendente estocástico com o intuito de minimizar o número de iterações, conduzindo à rápida convergência em base de dados de recomendação ainda não exploradas (não treinadas).

Sejam β a taxa de aprendizagem usada na primeira iteração da Equação 2.8 e α a taxa

de aprendizagem usada nas demais iterações. Ademais, seja $iter(\beta, \alpha, P, Q)$ a função que retorna o número de iterações necessárias para aprender P e Q (parâmetros da fatoração de matrizes aprendidos através da função 2.4), dada uma determinada configuração de β e α. Logo, nosso problema pode ser formalizado da seguinte forma:

$$\underset{\beta}{\operatorname{argmin}}\, iter(\beta, \alpha, P, Q) \tag{4.1}$$

Para investigar o impacto dos diferentes valores de β na fase de treinamento da fatoração de matrizes, nós definimos uma métrica que indica a porcentagem de iterações reduzidas em comparação com a implementação padrão (descrita no Capítulo 2), a qual usa um valor fixo e pequeno para α.

Por exemplo, dado uma base de dados, suponha que a fatoração de matrizes necessitou de 20 iterações para atingir a convergência em um conjunto de validação, usando uma taxa de aprendizagem padrão, e necessitou de 12 ao utilizar um determinado valor de β como taxa de aprendizagem inicial. Nesse caso, a redução é de $0,4$ ou 40%, se compararmos com número original de iterações, e foi calculada como 1 - (12/20).

A fim de investigar a relação entre β e a porcentagem de iterações reduzidas, nós variamos o valor de β de 0,01 a 0,1 com incrementos de 0,005 e usamos $\alpha = 0,01$ para prosseguir até a convergência. A escolha por 0,01 para a taxa de aprendizagem α é baseada na literatura, na qual esta é usada em vários experimentos com diferentes bases de dados como um valor *default*. Além disso, escolhemos essa faixa de valores para β porque verificamos que, para a maioria das bases de dados utilizadas, quando $\beta > 0,1$ o gradiente descendente estocástico começa a divergir, ou seja, há um aumento no número de iterações e não uma redução. Esse fato pode indicar que o mínimo local foi ultrapassado nesses casos.

As bases utilizadas para realizar o treinamento da fatoração com os vários valores de β estão descritas na Seção 5.1. Resumidamente, foram 8 bases de dados de sistemas de recomendação reais de domínios e tamanhos variados. O objetivo foi obter mais evidência, já que são bases com aspectos (domínio e tamanho) diferentes. A Figura 4.2 mostra a porcentagem de iterações reduzidas em função dos valores de β para todas as bases de dados consideradas neste trabalho.

De fato, para todas essas bases observamos uma relação linear positiva entre β e a porcentagem de iterações reduzidas, ou seja, quanto maior o valor de β, maior é a economia de

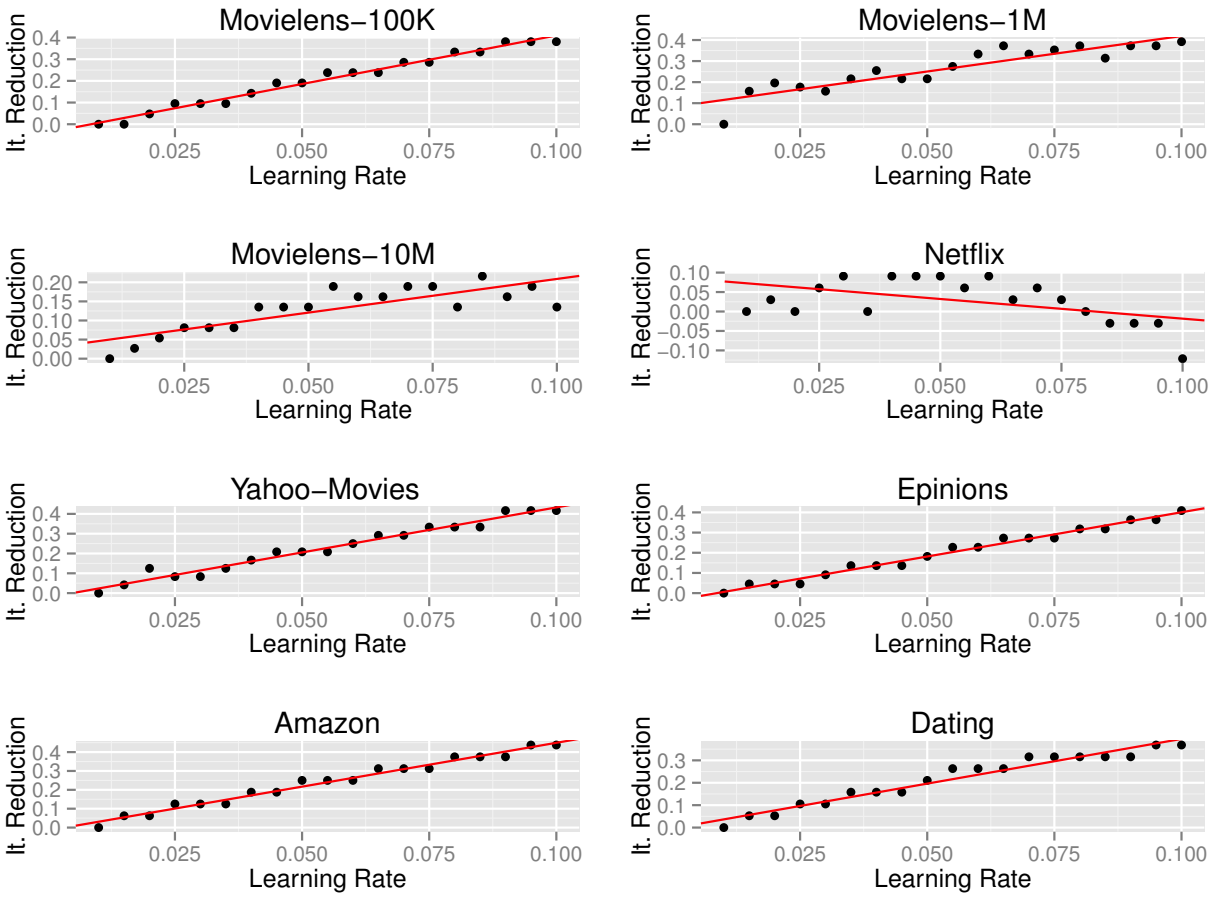


Figura 4.2: Relação entre β e a porcentagem de iterações reduzidas. As linhas vermelhas mostram os modelos lineares ajustados a cada base de dados.

iterações durante o treinamento.

Para confirmar essa relação, ajustamos um modelo de regressão linear simples para cada base e verificamos esse ajuste por meio da estatística R^2. O R^2 varia de 0 a 1, , indicando, em porcentagem, o quanto o modelo consegue explicar os valores observados. No nosso caso, R^2 explica a proporção da redução de iterações que podem ser explicadas pela variação do β. Quanto maior o R^2, mais ajustado aos dados está o modelo. Note que os valores de R^2 obtidos são próximos a 1, o que indica uma forte correlação linear (conforme Tabela 4.1) A última coluna da Tabela 4.1 mostra o número de iterações necessárias para fatoração de matrizes padrão ao usar valor padrão de $\alpha = 0,01$.

A exceção foi a base Netflix que não seguiu o mesmo padrão das demais bases. Apesar de não ter uma explicação concreta para isso ainda, um palpite é que $0,01$ já representa um bom

valor inicial para a taxa de aprendizagem, desta forma não havendo espaço para melhorias. A propósito, Koren [23] reporta uso de α com valores de $0,002$ e $0,005$ para esta base, o que reforça nossa hipótese de que $0,01$ já seria um valor grande para essa base de dados, especificamente. Outra hipótese diz respeito às distribuições das avaliações por usuários e por itens que são atípicas quando comparadas às demais bases (vide Apêndice A).

Ademais, o padrão linear permanece o mesmo quando o número de fatores latentes varia como mostra a Figura 4.3. Os resultados com 30 e 50 fatores latentes apresentam um percentual de redução um pouco melhor, por outro lado, o erro após a convergência também é ligeiramente mais elevado.

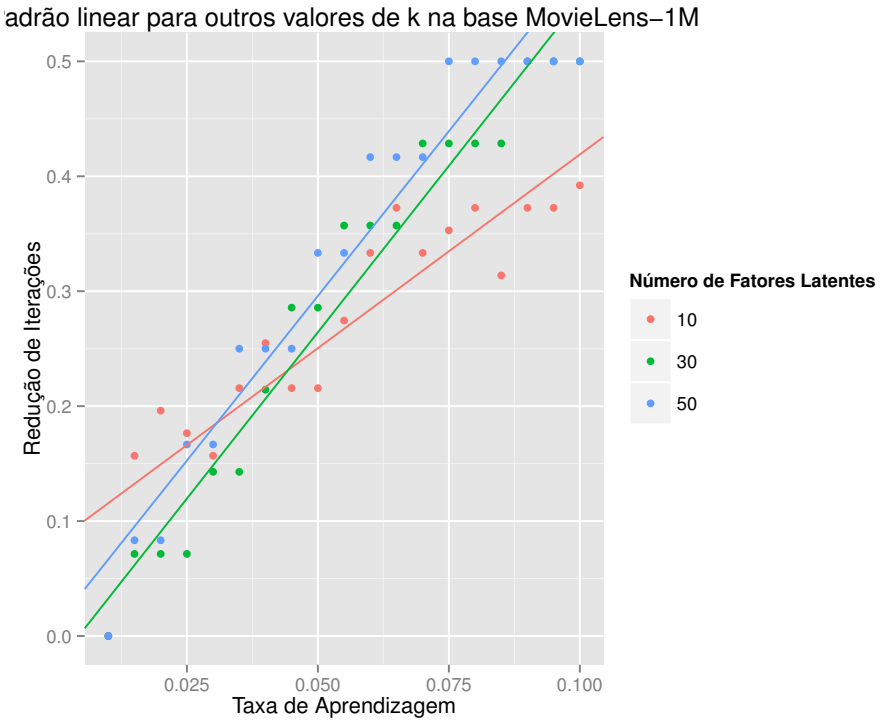


Figura 4.3: Relação entre β e a porcentagem de iterações reduzidas ao variar o número de fatores latentes.

Resumidamente, a fatoração de matrizes apresenta comportamento similar quando aplicada a bases de dados similares, isto é, bases que compartilham as mesmas entidades (usuário e itens), domínio (sistemas de recomendação) e faixa de avaliação (escala de 1 a 5 estrelas, por exemplo). A partir dessa observação, propomos a utilização do modelos de regressão linear simples para predizer, para uma base de dados desconhecida (não treinada), uma taxa

de aprendizagem inicial que conduza a um menor número de iterações.

Base de Dados	R^2	**% redução máxima**	**#iterações com** $\alpha = 0,01$
Movielens-100K	0.9801	38.1%	21
Movielens-1M	0.8184	39.2%	51
Movielens-10M	0.662	21.6%	37
Netflix	0.7525	9%	35
Yahoo-Movies	0.9694	41.6%	24
Epinions	0.9838	40.9%	22
Amazon	0.9779	43.7%	16
Dating	0.9514	36.8%	19

Tabela 4.1: Avaliação dos modelos lineares baseados na estatística R^2. Redução máxima indica a maior porcentagem de iterações reduzidas quando comparadas ao número de iterações usando a taxa de aprendizagem padrão.

Os detalhes das características das bases de dados e como utilizar os modelos lineares para a predição se encontram no Capítulo 5.

Capítulo 5

Avaliação Experimental

Neste capítulo, apresentamos os resultados das avaliações realizadas com o método descrito no Capítulo 4. Primeiramente, descrevemos as bases de dados (Seção 5.1). Em seguida, descrevemos como os experimentos foram realizados, inclusive informando os dados necessários para reprodução dos mesmos (Seção 5.2). Por fim, expomos como utilizar a técnica do Capítulo 4 e a comparamos com as técnicas de taxa de aprendizagem adaptativa (Seção 5.3).

5.1 Descrição das Bases de Dados

As características gerais das bases de dados de sistemas de recomendação utilizadas nos experimentos estão descritas na Tabela 5.1. Note que todas as bases são de *feedback* explícito, uma vez que nosso escopo é a fatoração de matrizes para o problema da predição de avaliações. Além disso, as bases são de tamanhos diferentes, variando de cem mil avaliações a cem milhões, e exemplificam 3 domínios diferentes, filmes, produtos e encontros.

Detalhes como a distribuição de avaliações e as distribuições da quantidade de itens acessados pelos usuários, bem como, quantidade de usuários que acessam cada item estão no Apêndice A.

Base de Dados	$\|U\|$	$\|I\|$	**#Avaliações**	**Escala**	**Domínio**
MovieLens-100k	943	1.682	100,000	[1, 5]	Filmes
MovieLens-1M	6.040	3.706	1.000,209	[1, 5]	Filmes
MovieLens-10M	69.878	10,677	10,000054	[1, 5]	Filmes
Netflix	480,189	17.770	100,480,507	[1, 5]	Filmes
Yahoo-Movies	7.642	11.916	221.364	[1, 5]	Filmes
Epinions	40,163	139.738	664.824	[1, 5]	Produtos
Amazon	2.146.276	1.231.018	5.744.088	[1, 5]	Produtos
Dating	135.359	168.791	17.359.346	[1, 10]	Encontros

Tabela 5.1: Descrição das bases de dados utilizadas nos experimentos. A escala indica a faixa de valores que os usuários podem avaliar os itens.

5.2 Planejamento de Experimentos

Os experimentos desta dissertação podem ser agrupados em duas fases. A primeira fase (Seção 5.3) descreve como utilizar a técnica do Capítulo 4 e a compara com o gradiente descendente estocástico padrão. A segunda fase (Seção 5.4) consiste em comparar 6 técnicas de taxa de aprendizagem adaptativa com, novamente, o gradiente descendente estocástico acrescido da técnica proposta nesta dissertação, portanto, inclui os resultados da primeira fase. (Os detalhes do funcionamento das estratégias estão no Capítulo 3)

Para a avaliação das duas fases, utilizamos 3 bases de dados como bases de teste, Amazon, MovieLens-10M e Dating. A escolha é baseada no fato de, apesar de serem bases de dados de sistemas de recomendação, elas são de domínios diferentes (conforme Tabela 5.1). A nossa intenção é, principalmente, ter mais evidência que a nossa técnica é consistente mesmo quando aplicada a domínios diferentes. Além disso, avaliamos em dois aspectos, a acurácia dos modelos e a rapidez da convergência do treinamento.

A métrica utilizada para avaliar a acurácia dos modelos é a RMSE (do inglês *Root-Mean-Square Error*), definida na Equação 5.1. Já a rapidez da convergência do treinamento é medida com o número de iterações para atingir a convergência, isto é, a iteração a partir da qual ocorre ou decréscimo insignificante do RMSE (quando o RMSE é menor que um ϵ

configurado antecipadamente) ou aumento do RMSE (o que indica início de divergência).

$$RMSE = \sqrt{\frac{\sum_{r_{u,i} \in R} (\hat{r_{u,i}} - r_{u,i})^2}{|R|}} \tag{5.1}$$

Com relação à segunda fase, 4 das 6 técnicas adaptativas são descritas em [27] e as outras 2 são técnicas mais simples que serviram como *baselines*. Portanto, estendemos a avaliação realizada por Luo et al. [27] ao comparar seus métodos com *baselines* e em outras bases de dados. Vale ressaltar que tentamos adaptar outros métodos aplicados a outros contextos diferentes de sistemas de recomendação, assim como Luo et al. [27] fez, no entanto, os resultados preliminares mostraram que não foram bem sucedidos (vide Apêndice B).

Ainda, para cada base de teste, executamos a fatoração de matrizes 10 vezes e comparamos o intervalo de confiança das métricas, o que também é uma lacuna no trabalho original. (Para o cálculo do intervalo de confiança foi utilizada a distribuição t de Student, uma vez que esta é a mais apropriada quando o número de amostras no experimento é pequeno).

A seção 5.2.1 indica os detalhes de implementação necessários para reprodução dos experimentos presentes nesta dissertação.

5.2.1 Reprodutibilidade dos Experimentos

Para fins de reprodutibilidade dos experimentos, foram escolhidas base de dados públicas e disponíveis online[1]. Quanto à implementação, aplicamos a validação-cruzada *5-fold* e utilizamos a configuração padrão da fatoração de matrizes da biblioteca *MyMediaLite* [16], na qual o número de fatores latentes k, o termo de regularização λ e o critério de parada são, respectivamente, 10, 0,015 e 0,001. Ainda, implementamos sobre o *MyMediaLite* as estratégias da taxa de aprendizagem adaptativa descritas em [27].

Cada técnica utilizada nos experimentos apresenta um conjunto de metaparâmetros, os quais são parâmetros para controlar alguma ação específica da técnica. A Tabela 5.2 descreve a configuração de metaparâmetros. O up e $down$ da técnica *bold driver*, representam as porcentagens de incremento ou decremento da taxa de aprendizagem; os metaparâmetros da

[1] A base de dados "Dating" está disponível em http://www.occamslab.com/petricek/data; a "Yahoo-Movies" está disponível em http://webscope.sandbox.yahoo.com/catalog.php; as demais base de dados estão disponíveis em http://konect.uni-koblenz.de

técnica *Annealing* estão descritas em [9] e os das demais técnicas estão em [27] (note que usamos aqui a mesma nomenclatura de metaparâmetros dos artigos originais).

Técnica	**Configuração de Metaparâmetros**
Bold Driver	$up = 0,05$, $down = 0,5$
Annealing	$c = 5$, $\tau = 5$
DSSA	$\alpha = 0,0005$, $\alpha = 0,0005$
IDBD	$\theta = 0,002$
SMD	$K = 0,09$, $\theta = 0,005$
GCA	$\omega = 0,05$

Tabela 5.2: Descrição dos metaparêmetros das técnicas de taxa de aprendizagem adaptativas utilizadas nos experimentos.

Exceto a base de dados Dating, cuja escala de avaliações é originalmente [1, 10], as demais bases se encontram no padrão [1, 5]. Similarmente a Luo et al. [27], normalizamos os dado da base Dating para a escala [1, 5], com o objetivo de que todos os algoritmos possam ser comparados adequadamente. A normalização consistiu em atribuir as avaliações 1 e 2 para 1, 3 e 4 para 2, 5 e 6 para 3, 7 e 8 para 4 e, finalmente, 9 e 10 para 5.

5.3 Predizendo a Taxa de Aprendizagem Inicial

Como discutido no Capítulo 4, a fatoração de matrizes apresenta comportamento similar em diferentes bases de dados de sistemas de recomendação com relação aos diferentes valores de β (taxa de aprendizagem para a primeira iteração) da faixa $[0,01, 0,1]$ (conforme Figura 4.2). Para a maioria das bases de dados, o comportamento é explicado por um modelo de regressão linear simples.

Novamente, a nossa hipótese é: "A estimativa de um bom primeiro passo para o gradiente descendente estocástico resulta em menos iterações durante o treinamento da fatoração de matrizes para o problema da predição de notas em sistemas de recomendação". Então o fato observado no Capítulo anterior sugere que podemos utilizar os modelos lineares ajustados aos dados para predizer a taxa de aprendizagem inicial, o primeiro passo citado na

nossa hipótese, para outras bases de dados ainda desconhecidas. Em outras palavras, dada uma base de dados desconhecida, queremos responder a seguinte questão: "Qual a taxa de aprendizagem inicial que devemos utilizar de forma a reduzir o número de iterações em $x\%$ comparado ao gradiente descendente estocástico padrão?"

Como dito na seção anterior, utilizamos as bases de dados Amazon, MovieLens-10M e Dating como bases de teste, representando as bases de dados desconhecidas. Para cada base de teste, utilizamos as demais, individualmente, para realizar a predição de β. Então, comparamos o número de iterações do gradiente descendente estocástico padrão e o número de iterações alcançadas com o β predito pelas demais bases. (Excluímos a base Netflix, uma vez que esta não segue o padrão da correlação linear positiva, como exposto no Capítulo 4.)

Para as bases de testes, tentamos predizer uma taxa de aprendizagem inicial β que alcançasse 40% de redução no número de iterações, o que é próximo aos limites superiores observados empiricamente e descritos na Tabela 4.1. Os limites superiores foram 43,7% para a base Amazon, 21,6% para MovieLens-10M e 36,8% para a base Dating. (Nesse caso, um limite superior indica a redução máxima de iterações ao variar o valor de β como mostra a Figura 4.2).

A Tabela 5.3 mostra os resultados. Para a base Amazon, a melhor predição foi da base MovieLens-10M. Surpreendentemente, esta previsão conduziu a uma redução de 62,5%, resultado maior do que a limite superior observado empiricamente. Isso pode ser considerado uma exceção, uma vez que valores acima de 0,1 geralmente levaram à divergência. Para o MovieLens-10M, a melhor predição veio da base MovieLens-1M. Neste caso, a redução foi de 21,6%, resultado esperado, uma vez que o limite superior empírico também foi de 21,6%. Para a base Dating, as melhores predições vieram das bases MovieLens-100k e Epinions, com redução de 36,8% (Novamente, o limite empiricamente observado foi o retornado).

As Figuras 5.1, 5.2 e 5.3 mostram o comportamento da fatoração de matrizes, expressado pelo RMSE ao longo das iterações, ao utilizar uma taxa de aprendizagem conservadora e ao utilizar uma taxa de aprendizagem predita com a nossa técnica, comprovando a redução de iterações nas 3 bases de teste. (Foram utilizados os melhores valores de β destacados em negrito na Tabela 5.3).

Vale ressaltar que nosso principal objetivo é a economia do número de iterações, mas outro aspecto diretamente relacionado é o tempo de execução do treinamento. Para medir

Base Teste	***Base Treino***	**β predito**	**%iterações reduzidas**	**#iterações com o β**
Amazon	Ml-100K	0,097	43,7%	9
	Ml-1M	0,087	37,5%	10
	Ml-10M	**0,159**	**62,5%**	**6**
	Yahoo-Movies	0,091	37,5%	10
	Epinions	0,099	43,7%	9
	Dating	0,098	43,7%	9
MovieLens-10M	Ml-100k	0,097	18,9%	30
	Ml-1M	**0,087**	**21,6%**	**29**
	Yahoo-Movies	0,091	16,2%	31
	Epinions	0,099	13,5%	32
	Amazon	0,088	16,2%	31
	Dating	0,098	13,5%	32
Dating	**Ml-100k**	**0,097**	**36,8%**	**12**
	Ml-1M	0,087	31,5%	13
	Ml-10M	0,159	21%	15
	Yahoo-Movies	0,091	31,5%	13
	Epinions	**0,099**	**36,8%**	**12**
	Amazon	0,088	31,5%	13

Tabela 5.3: Predição da taxa de aprendizagem inicial para as bases de dados Amazon, MovieLens-10M e Dating, usando os modelos lineares obtidos das demais bases de dados.

tempo é preciso um controlado bastante controlado para não haver interrupções no processo do treinamento, além de que a redução temporal pode variar, dependendo da configuração do computador que executa o treinamento. Para ilustrar a redução de tempo, conseguimos reduzir em 3 minutos, na base MovieLens-10M, quando comparado ao treinamento com método padrão, o qual durou 32 minutos.

Note que todos os resultados são muito similares. De fato, os intervalos de confiança (também calculados com a distribuição t de Student, devido ao mesmo motivo de pequeno número de amostras) da porcentagem de redução de iteração e do número de iterações são, respectivamente, [33,8 , 53,9] e [7,3 , 10,3] para a base de dados Amazon, [13,3 , 19,9] e [29,6 , 32] para a base MovieLens-10M, e [25,4 , 37,5] e [11,8 , 14,1] para base Dating, com nível de significância de 5%, o que comprova que todos os modelos apresentam predições

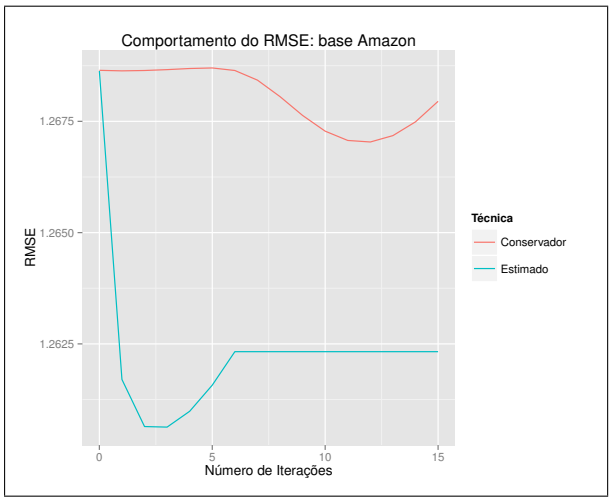


Figura 5.1: Comportamento do RMSE na base Amazon com o passar das iterações do gradiente descendente estocástico usando a versão padrão e a nossa técnica.

boas e similares. Apesar das predições semelhantes, as bases diferem quanto ao domínio dos itens. Este achado é interessante e bastante promissor porque essa observação tende a acontecer independentemente do domínio.

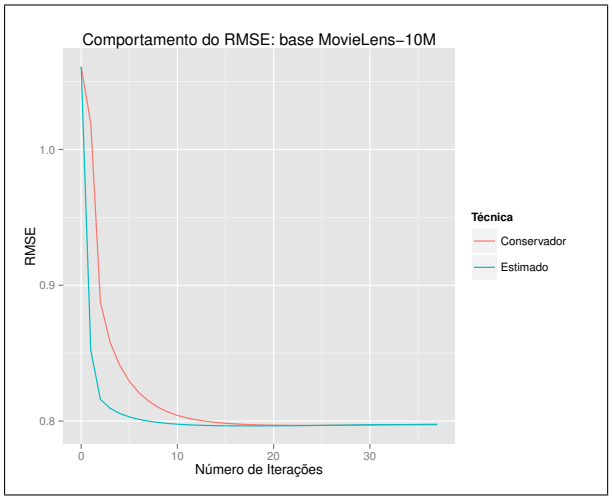


Figura 5.2: Comparação da quantidade de iterações para atingir a convergência na base MovieLens-10M.

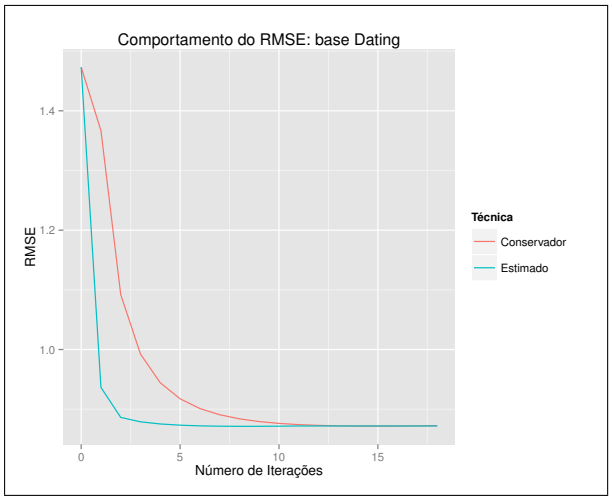


Figura 5.3: Comportamento do RMSE na a base Dating com o passar das iterações do gradiente descendente estocástico usando a versão padrão e a nossa técnica.

5.4 Comparação com Estratégias de Taxa de Aprendizagem Adaptativa

Como não achamos na literatura um trabalho diretamente relacionado, isto é, um trabalho que também proponha um passo inicial, decidimos comparar a nossa técnica, chamada aqui de Estimador de Taxa de Aprendizagem (ETA), com estratégias de taxa de aprendizagem adaptativas da literatura. As técnicas escolhidas foram o *bold driver*, *annealing* e as técnicas descritas por Luo et al. [27] chamadas, DSSA, IDBD, SMD e GCA.

Na avaliação, utilizamos a taxa de aprendizagem com maior porcentagem de redução de iterações, estimada na seção anterior, e definimos a taxa de aprendizagem padrão (0,01) para as estratégias adaptativas, a fim de tornar os resultados comparáveis com a abordagem tradicional e a nossa (ETA). A Tabela 5.4 mostra os resultados dos intervalos de confiança das médias do RMSE e da quantidade de iterações. Como o foco deste trabalho é a quantidade de iterações, ilustramos os resultados nas Figuras 5.5, 5.6 e 5.7 (a Figura).

O primeiro resultado a ser destacado é a equivalência da acurácia, nas 3 bases, de todas as abordagens adaptativas quando comparadas à acurácia da abordagem padrão da fatoração de matrizes, como mostra a Figura 5.4. Os intervalos de confiança da métrica RMSE se sobrepõem, confirmando que não há diferença significativa entre os métodos. Esse fato é interessante, pois não acontece o conflito que existe entre acurácia do modelo e rapidez no treinamento descrito no Capítulo 1, e pode ser explicado pela escolha de $0,01$ ser um valor adequado para a taxa de aprendizagem inicial.

Já a nossa abordagem, para quase todas as bases de dados, necessitou de menos iterações quando comparada com os métodos de taxa de aprendizagem adaptativa, e também mantendo RMSE semelhantes, ou seja, a nossa abordagem chegou a convergência mais rapidamente e manteve acurácia equivalente. As exceções foram os métodos *bold driver* e MF-GCA na base de dados MovieLens-10M, que atingem a convergência em 16 iterações, enquanto nosso alcança apenas com aproximadamente 31.

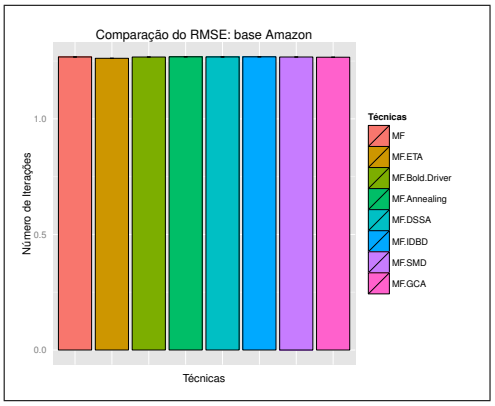
Também podemos concluir que os métodos mais complexos MF-DSSA, MF-IDBD e MF-SMD, além de serem equivalentes entre si, são equivalentes ao método *annealing*, que é bem mais simples. Como explicado no Capítulo 3, essas técnicas de taxa de aprendizagem adaptativas são mais complexas, pois têm de armazenar matrizes adicionais para manter

Base Teste	***Técnica***	**IC. RMSE**	**IC. #it**
Amazon	MF	[1,267544, 1,268107] ,	16
	MF-ETA	[1,261758, 1,262359]	[5,5 , 6,1]
	MF-Bold driver	[1,267385, 1,267905]	12
	MF-Annealing	[1,268110, 1,268926]	[15,35 , 16,05]
	MF-DSSA	[1,268000, 1,268000]	[14,95 , 15,65]
	MF-IDBD	[1,268292, 1,268588]	[15,35 , 16,05]
	MF-SMD	[1,267498, 1,267902]	[15,35 , 16,05]
	MF-GCA	[1,266368, 1,267092]	[15,03 , 15,77]
Movielens-10M	MF	[0,7968953, 0,7976134]	[35,73, 37,27]
	MF-ETA	[0,7973819, 0,7975636]	[31,90, 32,50]
	MF-Bold driver	[0,8011653, 0,8016800]	[16,12, 16,88]
	MF-Annealing	[0,7969451, 0,7975083]	[38,42, 39,58]
	MF-DSSA	[0,7973544, 0,7980456]	[34,95, 35,65]
	MF-IDBD	[0,7952172, 0,7961828]	[34,95, 35,65]
	MF-SMD	[0,7972665, 0,7984135]	[34,95, 35,65]
	MF-GCA	[0,8012360, 0,8025840]	[16,03, 16,77]
Dating	MF	[0,8721421, 0,8728850]	19
	MF-ETA	[0,8724769, 0,8727597]	12
	MF-Bold driver	[0,8746270, 0,8753452]	[13,12, 13,88]
	MF-Annealing	[0,8724831, 0,8726904]	[19,03, 19,77]
	MF-DSSA	[0,8723476, 0,8732524]	[18,23, 18,97]
	MF-IDBD	[0,8733529, 0,8747271]	[18,23, 18,97]
	MF-SMD	[0,8725904, 0,8735896]	[18,23, 18.97]
	MF-GCA	[0,8757814, 0,8764586]	[13,12, 13,88]

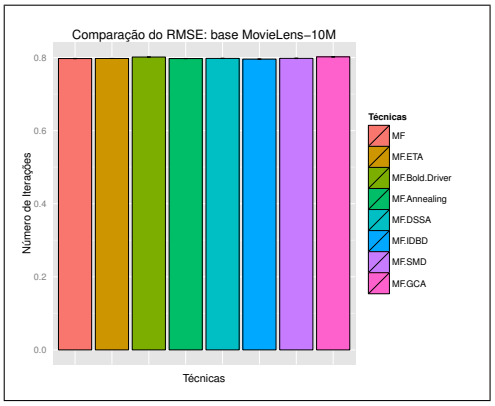
Tabela 5.4: Comparação da nossa técnica com as estratégias adaptativas.

as taxas de aprendizagem e os gradientes com o passar da iterações, enquanto o método *annealing* não passa de uma função que decrementa uma taxa de de aprendizagem global. Portanto, esses métodos não compensam no quesito acurácia nem no quesito rapidez de convergência.

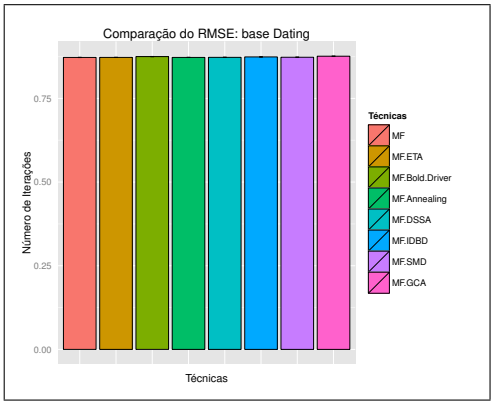
O método adaptativo descrito em [27] que apresentou ganho significativo quando comparado à abordagem tradicional foi o MF-GCA, principalmente, nas bases MovieLens-10M e Dating. Contudo, o seu desempenho é equivalente à técnica *bold driver*, mas esta realiza



(a) Comparação dos RMSE na base Amazon.



(b) Comparação dos RMSE na base MovieLens-10M.



(c) Comparação dos RMSE na base Dating.

Figura 5.4: Comparação dos intervalos de confiança do RMSE nas 3 bases de teste.

menos cálculos durante uma iteração.

Especulamos que os métodos adaptativos presentes em [27] só tenham ganhos efetivos quando a taxa de aprendizagem inicial for bastante pequena, pois, só nesse cenário a adaptação tornará o valor da taxa de aprendizagem bem maior quando comparado ao valor fixo e pequeno da abordagem padrão.

As figuras 5.8, 5.9 e 5.10 confirmam esses resultados. Não exibimos as técnicas MF-IDBD, MF-SMD e MF-GCA quando estas são estatisticamente equivalentes à MF-DSSA. Note que, apesar de apresentar valores de RMSE diferentes na base Amazon (Figura 5.8), as técnicas adaptativas são equivalentes, pois o que importa são intervalos de confiança descritos na Tabela 5.3.

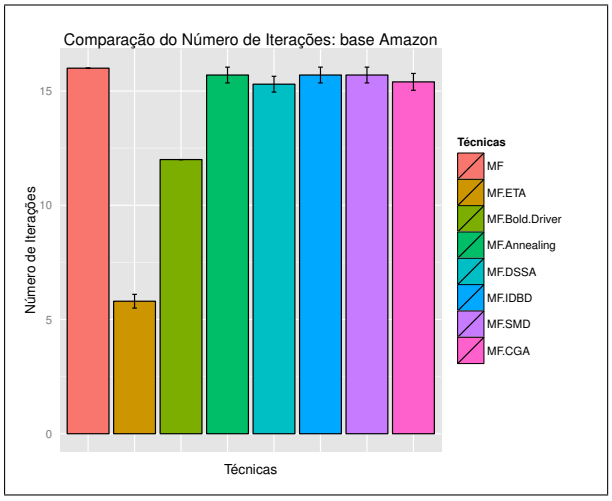


Figura 5.5: Comparação da quantidade de iterações para atingir a convergência na base Amazon.

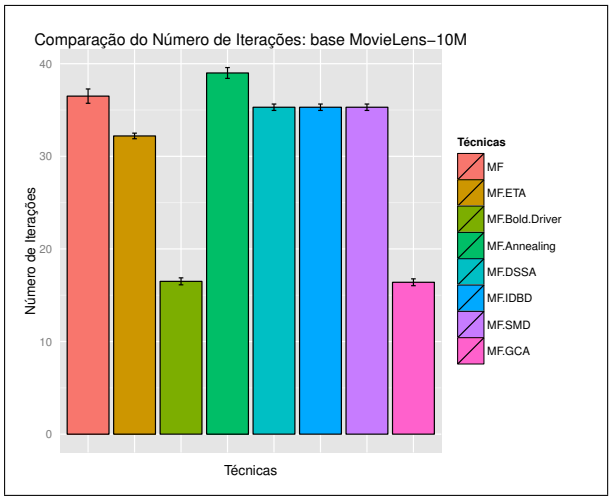


Figura 5.6: Comparação da quantidade de iterações para atingir a convergência na base MovieLens-10M.

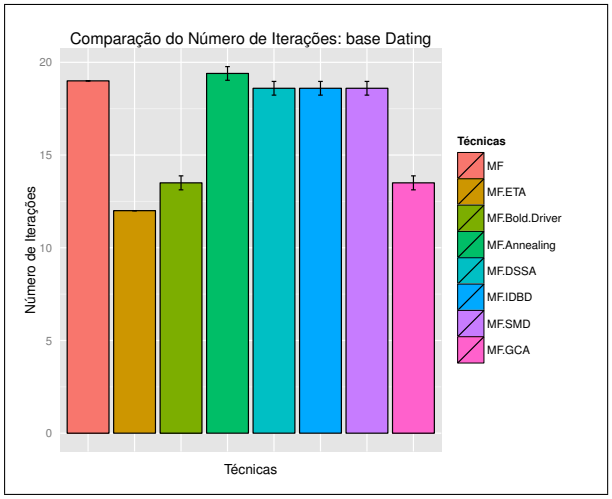


Figura 5.7: Comparação da quantidade de iterações para atingir a convergência na base Dating.

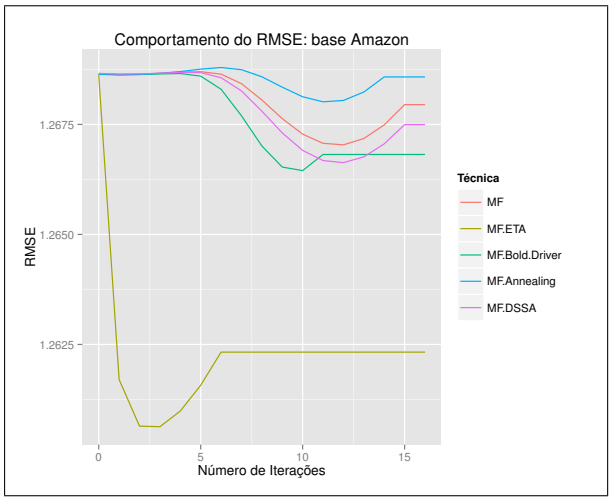


Figura 5.8: Comparação de todas as técnicas na base Amazon.

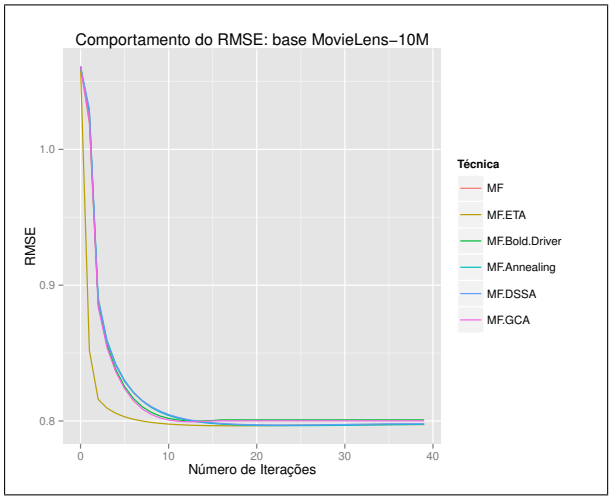


Figura 5.9: Comparação de todas as técnicas na base MovieLens-10M.

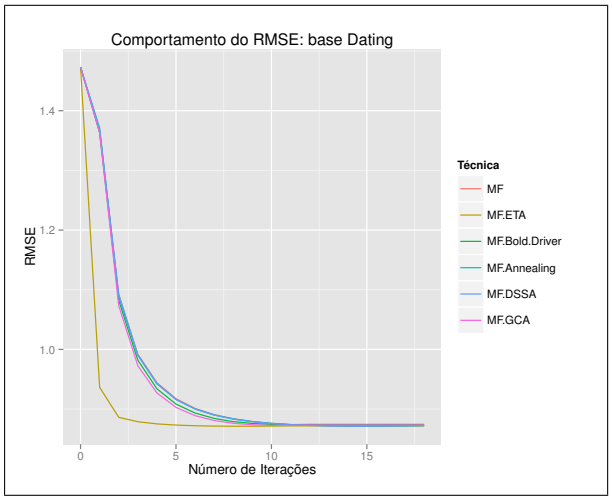


Figura 5.10: Comparação de todas as técnicas na base Dating.

Capítulo 6

Conclusões e Trabalhos Futuros

Neste trabalho, apresentamos uma abordagem para a predição da taxa de aprendizagem do gradiente descendente estocástico, algoritmo iterativo usado para a aprendizagem dos parâmetros da fatoração de matrizes aplicada a sistemas de recomendação. A ideia principal é utilizar esta predição para chegar o mais próximo possível a um mínimo local já na primeira iteração.

A partir de uma investigação exploratória em diferentes bases de dados de sistemas de recomendação, observamos que existe uma relação linear entre a taxa de aprendizagem e o número de iterações necessárias para atingir a convergência. A partir disso, propusemos a utilização de uma regressão linear simples para predizer, para uma base de dados desconhecido (ainda não treinada), a taxa de aprendizagem que minimizasse o número de iterações.

Nós testamos essa hipótese em 8 bases de dados de *feedback* explícito de serviços reais disponíveis online. Essas bases são de diferentes domínios, por exemplo, MovieLens e Netflix recomendam filmes, enquanto Epinions e Amazon recomendam produtos. Mostramos que, para quase todas as bases de dados, podemos reduzir o número de iterações em até 40% quando comparado com o método padrão.

Além disso, estendemos a avaliação realizada por Luo et al. [27] ao comparar os seus quatro métodos de abordagens de taxa de aprendizagem adaptativas com dois métodos também adaptativos mais simples. Concluímos que uma técnica simples apresenta resultados similares ao seu melhor método, ou seja, apresenta a mesma redução do número de iterações quando comparados ao método de gradiente descendente estocástico padrão. Portanto, fica evidente que a ideia de adaptar a taxa de aprendizagem ainda pode evoluir, uma vez que

métodos com heurísticas simples são equivalentes a métodos mais sofisticados.

Também comparamos nossa abordagem com essas seis estratégias de taxa de aprendizagem adaptativas presentes na literatura e mostramos que o nosso método supera quase todos eles em todos os conjuntos de dados avaliados. Adaptamos outros métodos chamados *line search* e *Barzilai-Borwein*, porém não houve redução do número de iterações.

Portanto, uma recomendação para ser utilizada na prática seria utilizar uma taxa de aprendizagem de 0,1 durante apenas a primeira iteração e, em seguida, voltar a utilizar um valor menor (o padrão de 0,01, por exemplo) para o treinamento da fatoração de matrizes aplicada a sistemas de recomendação.

É sabido que este trabalho apresenta algumas limitações, tais como o número reduzido de bases utilizadas na observação do padrão linear e o uso da mesma escala de avaliações. Essas limitações merecem ser investigadas como trabalho futuro, a fim de reforçar ainda mais a nossa técnica. Vale ressaltar que, apesar do número reduzido, utilizamos todas as bases de recomendação, com avaliações, disponíveis publicamente que conseguimos encontrar. Observamos também que a técnica não funciona para a base Netflix. Na verdade, o padrão linear positivo não ocorre nessa base, pois valores da taxa de aprendizagem inicial próximos ao maior valor testado ($\beta = 0,1$) não reduzem o número de iterações, mostrando que valores altos prejudicam a convergência. Novamente, o uso de mais bases realçaria qual o maior valor de taxa de aprendizagem deve ser utilizado.

Ainda como trabalho futuro, pretendemos combinar a nossa abordagem com algoritmos adaptativos, a fim de reduzir ainda mais o número de iterações necessárias até a convergência. Assim, adicionalmente ao valor inicial, teremos uma taxa de aprendizagem dinâmica, capaz de se adaptar durante o aprendizado do modelo. Finalmente, como dito anteriormente, pretendemos investigar a relação da taxa de aprendizagem e o número de iterações em outros tipos de conjuntos de dados, tais como conjuntos de dados de *feedback* implícitos.

Bibliografia

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[2] Marko Balabanović and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, March 1997.

[3] R. Bardenet, M. Brendel, B. Kégl, and M. Sebag. Collaborative hyperparameter tuning. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 199–207. JMLR Workshop and Conference Proceedings, May 2013.

[4] Jonathan Barzilai and Jonathan M. Borwein. Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, January 1988.

[5] Justin Basilico and Thomas Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 9–, New York, NY, USA, 2004. ACM.

[6] James Bennett, Stan Lanning, and Netflix Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[7] James Bergstra, Rémy Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *NIPS'2011*, 2011.

[8] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, February 2012.

[9] Christian Darken, Joseph Chang, Joseph Chang Z, and John Moody. Learning rate schedules for faster stochastic gradient search. IEEE Press, 1992.

[10] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA, 2007. ACM.

[11] Dennis DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 249–256, New York, NY, USA, 2006. ACM.

[12] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, January 2004.

[13] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 107–144. Springer, 2011.

[14] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[15] Simon Funk. Netflix update: Try this at home. Disponível em: `http://sifter.org/~simon/journal/20061211.html`. Último acesso em Abril de 2014.

[16] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. MyMediaLite: A free recommender system library. In *5th ACM International Conference on Recommender Systems (RecSys 2011)*, 2011.

[17] Zeno Gantner, Steffen Rendle, and Lars Schmidt-Thieme. Factorization models for context-/time-aware movie recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 14–19, New York, NY, USA, 2010. ACM.

[18] Rainer Gemulla, Erik Nijkamp, Peter J. Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 69–77, New York, NY, USA, 2011. ACM.

[19] Abraham P. George and Warren B. Powell. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Mach. Learn.*, 65(1):167–198, October 2006.

[20] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, December 1992.

[21] Paul B. Kantor. *Recommender systems handbook*. Springer, New York; London, 2009.

[22] Joseph A. Konstan and John Riedl. Deconstructing recommender systems. how amazon and netflix predict your preferences and prod you to purchase. Disponível em: `http://spectrum.ieee.org/computing/software/deconstructing-recommender-systems`. Último acesso em Abril de 2014.

[23] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[25] Qing Li and Byeong Man Kim. An approach for combining content-based and collaborative filters. In *Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages - Volume 11*, AsianIR '03, pages 17–24, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[26] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[27] Xin Luo, Yunni Xia, and Qingsheng Zhu. Applying the learning rate adaptation to the matrix factorization based collaborative filtering. *Know.-Based Syst.*, 37:154–164, January 2013.

[28] Miguel Moreira and Emile Fiesler. Neural networks with adaptive learning rate and momentum terms. Idiap-RR Idiap-RR-04-1995, IDIAP, Martigny, Switzerland, 10 1995.

[29] Nathan Srebro Nati and Tommi Jaakkola. Weighted low-rank approximations. In *In 20th International Conference on Machine Learning*, pages 720–727. AAAI Press, 2003.

[30] Caio Santos Bezerra Nóbrega and Leandro Balby Marinho. Predicting the learning rate of gradient descent for accelerating matrix factorization. In *Proceedings of KDMiLe - Symposium on Knowledge Discovery, Mining and Learning", ISSN 2318-1060*, 2013.

[31] Caio Nóbrega and Leandro Balby Marinho. Predicting the learning rate of gradient descent for accelerating matrix factorization. *JIDM*, 5(1):94–103, 2014.

[32] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer series in operations research and financial engineering. Springer, New York, NY, 2. ed. edition, 2006.

[33] Seung-Taek Park and David M. Pennock. Applying collaborative filtering techniques to movie search for better ranking and browsing. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 550–559, New York, NY, USA, 2007. ACM.

[34] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of the KDD Cup Workshop at the 13th ACM International Conference on Knowledge Discovery and Data Mining*, SIGKDD '07, pages 39–42, New York, NY, USA, 2007. ACM.

[35] Benjamin Recht and Christopher Ré. Parallel Stochastic Gradient Algorithms for Large-Scale Matrix Completion. *submitted*, 2011.

[36] Steffen Rendle. Learning recommender systems with adaptive regularization. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 133–142, New York, NY, USA, 2012. ACM.

[37] Steffen Rendle and Lars Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 251–258, New York, NY, USA, 2008. ACM.

[38] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[39] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 791–798, New York, NY, USA, 2007. ACM.

[40] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender systems: A case study. In *WebKDD Workshop at the ACM SIGKKD*, 2000.

[41] J. Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, pages 158–166, New York, NY, USA, 1999. ACM.

[42] Tom Schaul, Sixin Zhang, and Yann LeCun. No More Pesky Learning Rates. In *International Conference on Machine Learning (ICML)*, 2013.

[43] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Major components of the gravity recommendation system. *SIGKDD Explor. Newsl.*, 9(2):80–83, December 2007.

[44] Christina Teflioudi, Faraz Makari, and Rainer Gemulla. Distributed matrix completion. In Mohammed Javeed Zaki, Arno Siebes, Jeffrey Xu Yu, Bart Goethals, Geoffrey I. Webb, and Xindong Wu, editors, *ICDM*, pages 655–664. IEEE Computer Society, 2012.

[45] Kai Yu, Shenghuo Zhu, John Lafferty, and Yihong Gong. Fast nonparametric matrix factorization for large-scale collaborative filtering. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 211–218, New York, NY, USA, 2009. ACM.

[46] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management*, AAIM '08, pages 337–348, Berlin, Heidelberg, 2008. Springer-Verlag.

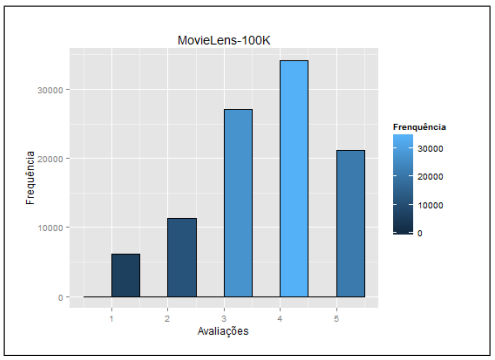
Apêndice A

Características das Bases de Dados

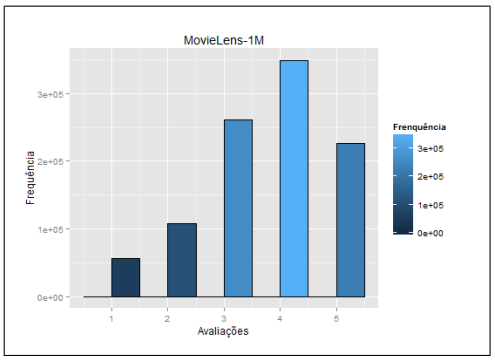
Este capítulo constitui um apêndice com mais detalhes das bases de dados utilizadas nos experimentos.

A.1 Distribuição das Avaliações

As figuras abaixo descrevem a distribuição das avaliações realizadas pelos usuários.

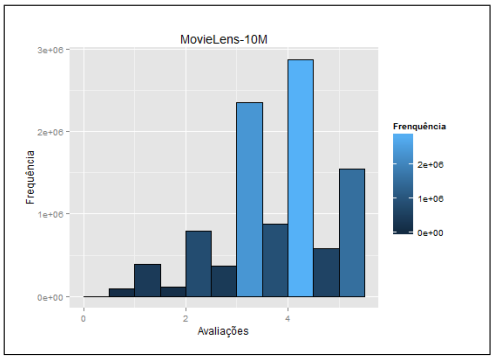


(a) Base de dados MovieLens-100K.

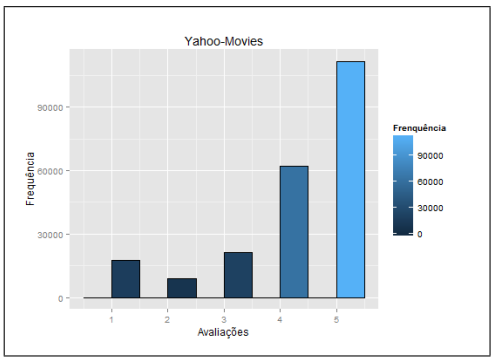


(b) Base de dados MovieLens-1M.

Figura A.1: Histogramas das avaliações do usuários nas bases de dados MovieLens-100K e MovieLens-1M.

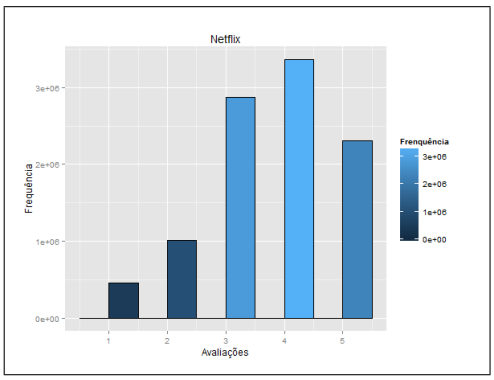


(a) Base de dados MovieLens-10M (avaliações variam de 0,5 em 0,5).

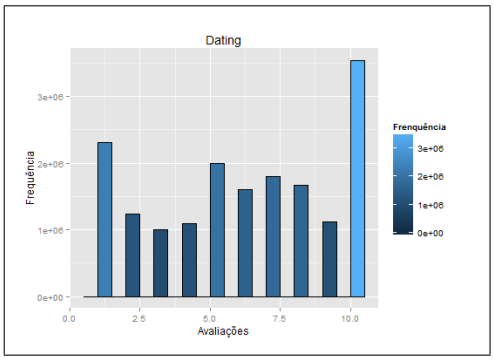


(b) Base de dados Yahoo-Movies

Figura A.2: Histogramas das avaliações do usuários nas bases de dados MovieLens-10M e Yahoo-Movies.

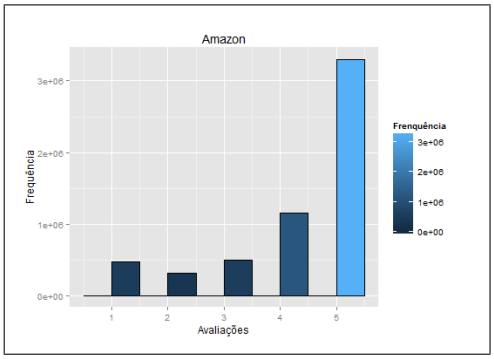


(a) Base de dados Netflix (amostra de 10% dos dados).

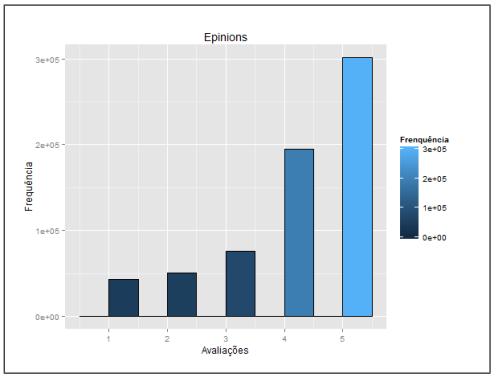


(b) Base de dados Dating (escala original de 0 a 10).

Figura A.3: Histogramas das avaliações do usuários nas bases de dados Netflix e Dating.



(a) Base de dados Amazon.



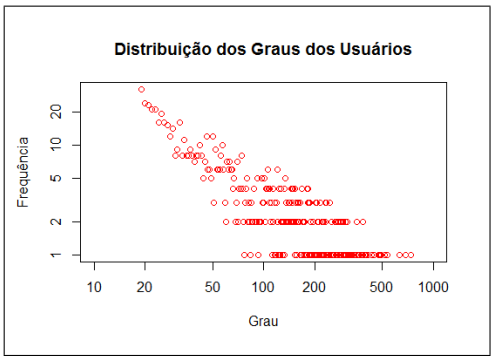
(b) Base de dados Epinions.

Figura A.4: Histogramas das avaliações do usuários nas bases de dados Amazon e Epinions.

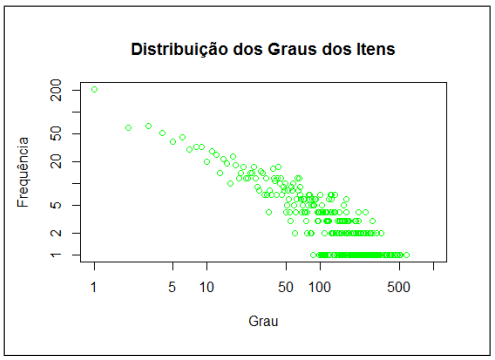
A.2 Distribuição dos Graus de Usuários e Itens

Um sistema de recomendação pode ser modelado como um grafo, no qual usuários e itens representam vértices e arestas representam a existência de avaliação do usuários em itens. Mais especificamente, os sistemas de recomendação das bases utilizadas nesta dissertação podem ser modelados como grafo bipartido, uma vez que usuários não avaliam usuários, nem itens avaliam itens. Logo, o grau de um usuário u indica quantos itens u avaliou, enquanto o grau de um item i indica quantos usuários avaliaram o item i.

As figuras desta seção descrevem a distribuição dos graus de usuários e itens das bases de dados utilizadas em escala logarítmica.

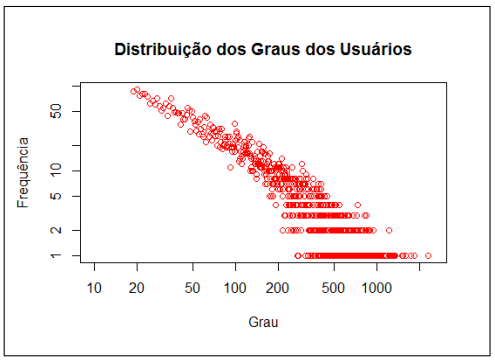


(a) Distribuição dos graus dos usuários.

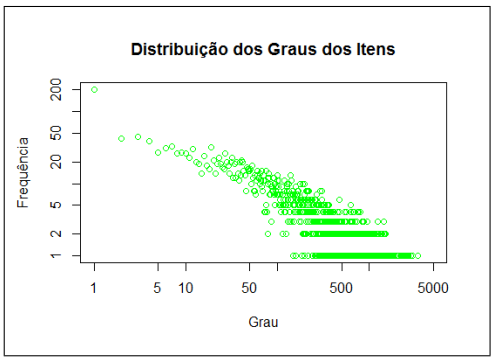


(b) Distribuição dos graus dos itens.

Figura A.5: Distribuição dos graus da base de dados MovieLens-100K.

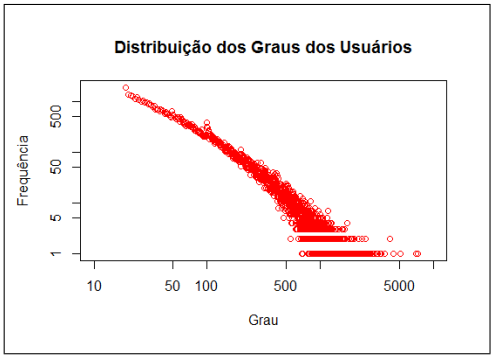


(a) Distribuição dos graus dos usuários.

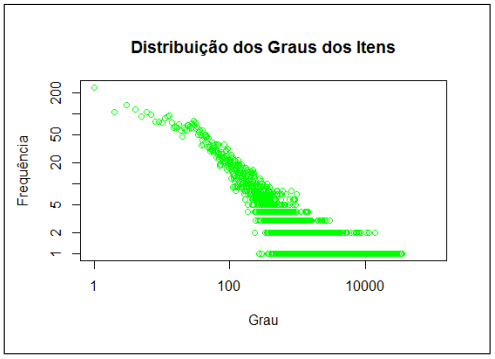


(b) Distribuição dos graus dos itens.

Figura A.6: Distribuição dos graus da base de dados MovieLens-1M.

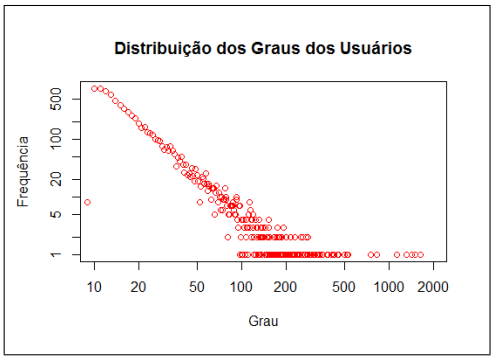


(a) Distribuição dos graus dos usuários.

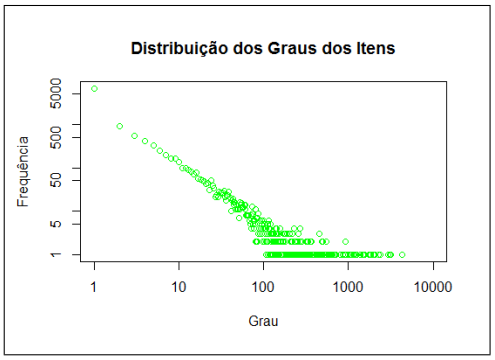


(b) Distribuição dos graus dos itens.

Figura A.7: Distribuição dos graus da base de dados MovieLens-10M.

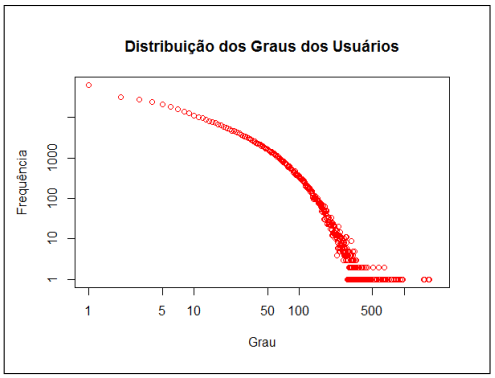


(a) Distribuição dos graus dos usuários.

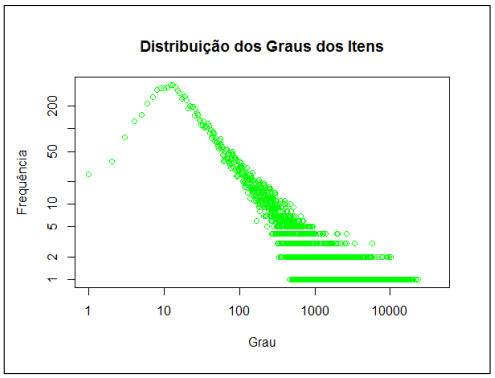


(b) Distribuição dos graus dos itens.

Figura A.8: Distribuição dos graus da base de dados Yahoo-Movies.

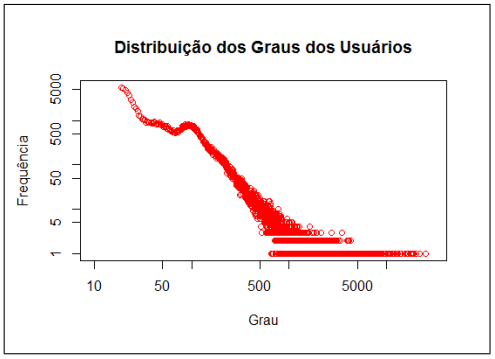


(a) Distribuição dos graus dos usuários.

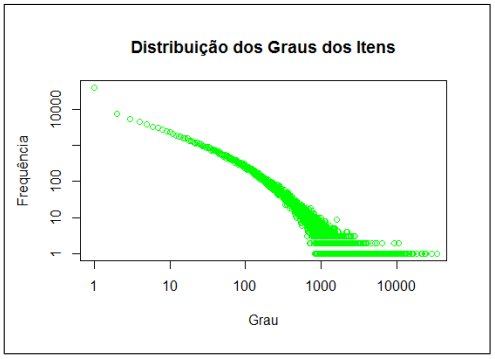


(b) Distribuição dos graus dos itens.

Figura A.9: Distribuição dos graus da base de dados Netflix.

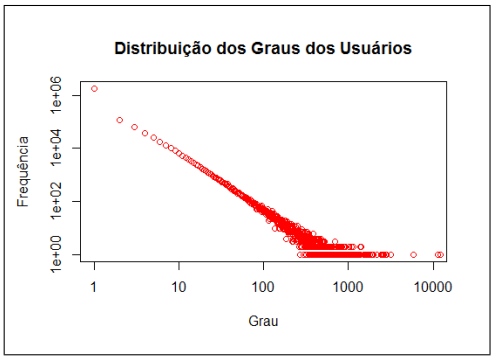


(a) Distribuição dos graus dos usuários.

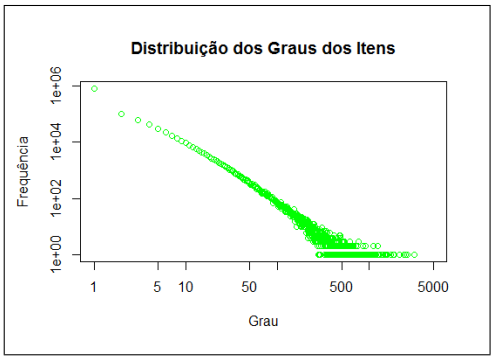


(b) Distribuição dos graus dos itens.

Figura A.10: Distribuição dos graus da base de dados Dating.

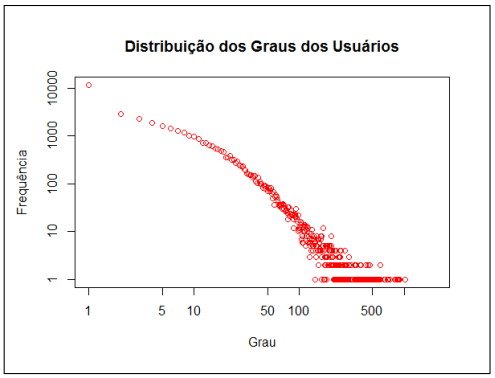


(a) Distribuição dos graus dos usuários.

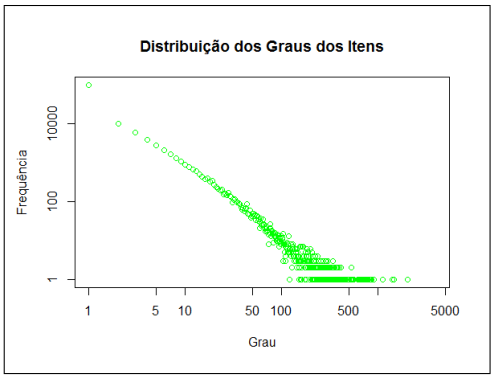


(b) Distribuição dos graus dos itens.

Figura A.11: Distribuição dos graus da base de dados Amazon.



(a) Distribuição dos graus dos usuários.



(b) Distribuição dos graus dos itens.

Figura A.12: Distribuição dos graus da base de dados Epinions.

Apêndice B

Outras Abordagens Avaliadas

Este apêndice descreve brevemente os métodos *line search* e Barzilai-Borwein e discute os seus resultados para o problema de predição de avaliações de sistemas de recomendação.

B.1 Line Search

Na área de otimização, o método do *line search* (em português, chamado de busca unidimensional, busca unidirecional ou busca linear) é uma estratégia iterativa para encontrar mínimos locais de uma função objetivo. Inicialmente, é feita uma estimativa de um mínimo local. Então, a cada iteração, o método calcula uma direção de descida e então computa um tamanho de passo para seguir ao longo desta direção, de forma que a função seja minimizada [32].

O sucesso deste método depende de uma escolha adequada tanto da direção quanto do tamanho do passo. A direção de descida pode ser computada por vários métodos, tais como gradiente descendente, métodos de Newton ou métodos Quasi-Newton. Já o tamanho do passo pode ser determinado de forma exata, quando é estimado de forma mais acurada, ou de forma inexata, quando há uma aproximação que melhore a otimização quando comparado à última iteração. Note que o gradiente descendente descrito no Capítulo 2 é um subtipo da estratégia *line search*, onde o tamanho do passo é a taxa de aprendizagem).

Sejam $f(x)$ uma função objetivo, p a direção de descida, α o tamanho do passo e x_0 a estimativa inicial. Assim, o procedimento seguido pelos métodos da estratégia *line search* é descrito no algoritmo B.1.1.

Algoritmo B.1.1 Algoritmo dos métodos da estratégia *line search*.

1: Inicializa $t = 0$.
2: **while** Critério de convergência não for atingido **do**
3: calcular direção de descida p^t
4: encontrar α que minimize $f(x + \alpha p^t)$
5: atualizar estimativa de mínimo local $x^{t+1} = x^t + \alpha^t x^t$
6: $t = t + 1$

A busca linear propriamente dita é feita no passo 4, onde temos uma função de uma variável. Nesse passo, há um subproblema de otimização: achar o melhor passo que minimize a função na próxima iteração. A nossa ideia foi incorporar esse subproblema ao gradiente descendente estocástico utilizado para a aprendizagem da fatoração de matrizes. A estratégia escolhida para solucionar esse subproblema foi a um tipo de busca linear inexata chamada *backtracking line search*(detalhes em [32]) por ser a de implementação mais fácil.

Apesar de fazer sentido na teoria, a aplicação do *backtracking line search* não ofereceu bons resultados para o nosso contexto. Utilizamos as mesmas métricas e valores padrões descritos nos experimentos do Capítulo 5. Em todas as iterações, o α retornado foi muito pequeno, fazendo com que a aprendizagem se tornasse lenta. Além disso, a inserção desse subproblema de otimização aumentou o tempo de processamento das iterações.

A tabela B.1 mostra os resultados para as duas menores bases utilizadas nos experimentos (o treinamento ficou tão lento que inviabilizou o uso das demais bases). A acurácia do modelo é equivalente, porém o número de iterações cresce em 10 vezes.

Base Teste	***Técnica***	**RMSE**	**#it**
MovieLens-100k	MF	0.9537	21
	MF-LineSearch	0.9572	202
Yahoo-Movies	MF	1.088	24
	MF-LineSearch	1.0853	217

Tabela B.1: Avaliação do *backtracking line search*.

A justificativa para esses resultados é baseada no fato de que o *backtracking line search* ou mantém ou decrementa o valor da taxa de aprendizagem atual. Logo, durante a atuali-

zação de um determinado parâmetro, a taxa de aprendizagem pode ter sido reduzida à um valor mínimo (no nosso caso, configurado para 0.001), que será utilizado na atualização dos parâmetros nas iterações futuras. Tentamos contornar esse problema ao utilizar uma taxa de aprendizagem para cada fator latente, mas o comportamento e os resultados foram semelhantes ao usar uma taxa de aprendizagem única.

Nossa conclusão é que há parâmetros (fatores latentes) que contribuem negativamente para o aprendizado da fatoração de matrizes. Portanto, a investigação desses parâmetros, bem como alternativas ao *backtracking line search* se configuram como trabalho futuro.

B.2 Método Barzilai-Borwein

O outro método presente na literatura para adaptação da taxa de aprendizagem foi proposto por Barzilai e Borwein [4]. A ideia central desse método é encontrar uma taxa de aprendizagem que minimize (usando a nomenclatura da seção anterior):

$$\alpha^t = \underset{\alpha}{\operatorname{argmin}} \, ||\Delta x - \alpha \Delta p||^2 \tag{B.1}$$

onde $\Delta x = x^t - x^{t-1}$ representa a diferença entre os parâmetros de duas iterações consecutivas, no nosso caso, diferença entre os fatores latentes, e $\Delta p = p^t - p^{t-1}$ representa a diferença entre os gradientes de duas iterações consecutivas.

A equação B.1 representa uma aproximação do método da secante, utilizado para encontrar as raízes de uma função e aplicado aos métodos de otimização de segunda ordem. E a solução para esse problema de minimização é achada ao aplicar a derivada com a relação a α:

$$\alpha \Delta p^T (\Delta x - \alpha \Delta p) = 0 \tag{B.2}$$

$$\alpha = \frac{\Delta p^T \Delta x}{\Delta p^T \Delta p} \tag{B.3}$$

A princípio, os resultados com o método Barzilai-Borwein levaram à divergência, pois o mesmo retornava, algumas vezes, valores negativos para a taxa de aprendizagem. Então, foi configurado um valor mínimo para evitar valores negativos. Com essa mudança, os resultados foram semelhantes aos da seção anterior. Com relação ao quesito acurácia, os modelos

treinados são equivalentes, porém o número de iterações não é reduzido, pelo contrário, cresce aproximadamente 10 vezes. O número de iterações semelhantes entre *backtracking line search* e o método Barzilai-Borwein é justificado por ambos compartilharem o mesmo valor mínimo configurado em 0.001.