APRENDIZADO ATIVO APLICADO À ELICITAÇÃO DE PREFERÊNCIAS PARA FINS DE INCENTIVO

Marden Braga Pasinato

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Zimbrão da Silva

Rio de Janeiro
Novembro de 2014

APRENDIZADO ATIVO APLICADO À ELICITAÇÃO DE PREFERÊNCIAS PARA FINS DE INCENTIVO

Marden Braga Pasinato

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Geraldo Zimbrão da Silva, D.Sc.

Prof. Geraldo Bonorino Xexéo, D.Sc.

Prof. Leandro Guimarães Marques Alvim, D.Sc.

Prof. Carlos Eduardo Ribeiro de Mello, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
NOVEMBRO DE 2014

Pasinato, Marden Braga

Aprendizado Ativo aplicado à Elicitação de Preferências para fins de Incentivo/Marden Braga Pasinato. – Rio de Janeiro: UFRJ/COPPE, 2014.

XIV, 80 p.: il.; 29, 7cm.

Orientador: Geraldo Zimbrão da Silva

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2014.

Referências Bibliográficas: p. 73 – 80.

1. Sistemas de Recomendação. 2. Aprendizado Ativo. 3. Elicitação de Preferências. I. Silva, Geraldo Zimbrão da. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Soli Deo Gloria

Agradecimentos

Em toda caminhada longa e árdua, há momentos de tropeço e desânimo que só são superados quando alguém lhe estende a mão e lhe coloca novamente no caminho. No meu caso não foi diferente, portanto, tenho muitas mãos a agradecer.

Agradeço ao meu orientador, prof. Geraldo Zimbrão, por acreditar no meu potencial e pela orientação deste trabalho. Minha gratidão aos membros da banca pelo tempo e atenção devotados à análise desta dissertação. Em especial, ao prof. Carlos Mello, cujas ideias e conselhos foram fundamentais, não apenas em minha vida acadêmica, mas também em minha vida pessoal. Cadu, se assim me permite, meu muito obrigado.

Quero agradecer aos meus amigos do PESC, que, entre um café e outro, me deram dicas de leitura; apontaram meus erros; aplaudiram meus acertos; me deram caronas; me contaram piadas; escutaram pacientemente minhas angústias e alegrias. Braida, Duarte, Horta, Luis e Pedro, não sei se aguentaria sem vocês.

Não posso deixar de agradecer ao prof. Daniel Figueiredo que, desde da graduação, me motivou a entrar no mundo acadêmico e, durante o mestrado, sempre se mostrou solicito a escutar meus problemas e pensar em soluções. Agradeço também a todos os funcionários do PESC e da CAPES por proverem uma infraestrutura onde pude me apoiar.

Sem dúvida, o maior auxílio que tive foi o amor e suporte da minha família. Não só agradeço, como também dedico este trabalho a eles: meus pais, minha irmã, meus avós, meus tios e primos. Meu muito obrigado pelo carinho; pelas palavras de ternura; pela compreensão e pelos tantos "vai dar tudo certo".

Por último, mas não menos importante, agradeço à mão que me pôs de pé após as quedas mais dolorosas e angustiantes. Obrigado Deus, por tudo!

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

APRENDIZADO ATIVO APLICADO À ELICITAÇÃO DE PREFERÊNCIAS PARA FINS DE INCENTIVO

Marden Braga Pasinato

Novembro/2014

Orientador: Geraldo Zimbrão da Silva

Programa: Engenharia de Sistemas e Computação

Na literatura de Sistemas de Recomendação, estratégias de Aprendizado Ativo foram extensamente aplicadas à elicitação de preferências para amenizar os efeitos do Arranque Frio, tarefa conhecida como *Elicitação de Preferências para fins de Arranque Frio*. Contudo, se soubéssemos quais são os N itens, dentre os já adquiridos mas não avaliados, que, se avaliados, trarão o maior ganho para o sistema em termos de acurácia, valeria a pena oferecer um incentivo para que o usuário avalie tais itens. A esta tarefa damos o nome de *Elicitação de Preferências para fins de Incentivo*.

Este trabalho propõe uma nova estratégia de Aprendizado Ativo, que seleciona os itens com base na distribuição de probabilidade dos mesmos, gerando um conjunto de treinamento sem viés. A batizada *Estratégia Livre de Viés* foi comparada com outras 16 estratégias populares dentro da literatura no que diz respeito a *Elicitação de Preferências para fins de Incentivo*. A *Estratégia Livre de Viés* mostrou desempenho superior às demais em termos de acurácia global do sistema. Além disso, apresentamos uma análise do desempenho de cada estratégia, levando em consideração os possíveis motivos de seu sucesso ou fracasso.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ACTIVE LEARNING APPLIED TO RATING ELICITATION FOR INCENTIVES PURPOSES

Marden Braga Pasinato

November/2014

Advisor: Geraldo Zimbrão da Silva

Department: Systems Engineering and Computer Science

In the literature of Recommender Systems, Active Learning strategies have been extensively applied to rating elicitation in order to mitigate Cold Start effects, namely *Rating Elicitation for Cold Start Purposes*. However, if we knew the best N items among those already acquired but not evaluated which, if evaluated, would result in the greatest improvement of the overall accuracy, it would be worthwhile to give users an incentive for evaluating them. We call this task *Rating Elicitation for Incentives Purposes*.

This work proposes a novel Active Learning strategy that selects items based on their probability distribution, creating an unbiased training set. The *Unbiased Strategy* was compared with other 16 popular strategies in the literature concerning *Rating Elicitation for Incentives Purposes* and it has outperformed all of them in terms of overall accuracy. Moreover, we present an analysis of each strategy performance, taking into account the possible reasons for their success or failure.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

1.1 Motivação

Há uma frase, de autor desconhecido, mas que ficou muito famosa, com o dizer: "informação é o novo petróleo". Apesar do conteúdo sensacionalista, não é possível negar que a veracidade da mesma vem se confirmando nas últimas décadas. De fato, nunca se produziu tanta informação em toda história e, ao mesmo tempo, nunca se dependeu tanto da mesma. As maneiras habituais de trabalho, estudo, locomoção, relacionamento e convívio foram completamente remodeladas, passando a ser fortemente informatizadas. Ou seja, os indivíduos estão cada vez mais dependentes de aparatos tecnológicos para executar tarefas antes consideradas básicas e triviais. Tal fenômeno chamou a atenção dos pesquisadores que o batizaram de *Era da Informação* [1].

Existem muitas vantagens e desvantagens associadas à tal Era da Informação, assim como em tudo o mais. No entanto, uma de suas mais aclamadas vantagens, a abundância de informação, está se transformando em uma desvantagem. Boa parte desta abundância ocorre na Internet, criando uma enorme oferta de conteúdo para seus usuários. O senso comum normalmente associa variedade à satisfação, isto é, quanto mais opções de conteúdo um usuário tiver, mais satisfeito ele estará. Contudo, o que se observa na prática é que, frente a um demasiado número opções, os usuários acabam por se angustiar, uma vez que o risco de fazer a escolha errada aumenta consideravelmente. Portanto, a exacerbada variedade de opções vem causando descontentamento ao invés de satisfação [2].

Este paradoxo torna-se mais visível quando consideramos o mundo do comércio eletrônico, também conhecido como *e-commerce*. Uma das esferas da vivência humana mais afetada pela Era da Informação foi, sem dúvida, o comércio. Chega a ser difícil encontrar uma empresa que não oferte seus produtos *online*. Por outro lado, a Internet deixou de ser apenas mais um canal de comunicação com o cliente e passou

a ser o habitat natural de novos negócios, comercializando inclusive bens intangíveis (e.g., *software*, áudio, vídeo, etc.), que são facilmente armazenáveis. Some-se a isto o fato de que lojas virtuais, quando ofertam bens tangíveis, isto é, produtos físicos em geral, não precisam tê-los em estoque no momento da venda. Tais fatores conferem ao *e-commerce* uma enorme flexibilidade para montar e oferecer seus catálogos, que podem atingir facilmente a ordem de milhões de itens.

Comprar, por sua vez, nunca foi uma ação tão simples e, ao mesmo tempo, tão complexa. Simples no sentido de que a revolução informacional acabou criando formas mais eficientes de pagamentos e, hoje, é possível literalmente comprar uma grande quantidade de itens com apenas poucos *clicks*. Entretanto, escolher o que comprar se tornou uma tarefa complexa tendo em vista as inúmeras possibilidades oferecidas *online*. O usuário comum se depara com uma infinidade de lojas virtuais, de toda parte do globo, cada uma com um catálogo praticamente ilimitado de itens. Não é à toa que estão surgindo diversos serviços voltados somente para auxiliar o usuário a encontrar o que deseja, como, por exemplo, buscadores específicos para produtos, comparadores de preço, atendimento virtual, entre outros [3].

Dentre as ferramentas que visam auxiliar o usuário durante a compra, nenhuma chamou tanto a atenção da indústria quanto os Sistemas de Recomendação (SR) [4]. Durante as últimas duas décadas, as grandes empresas da Internet investiram fortemente em tais sistemas, pois perceberam que o usuário se sente perdido em meio a tantas opções e, no fim, acaba por desistir da compra. Ao oferecer recomendações, o sistema pretende amenizar a dúvida do usuário, reduzindo o número de opções a se considerar. Ademais, a loja virtual passa a sensação de que compreende o usuário, ou seja, que sabe distinguir entre os itens que ele gosta e os que ele não gosta [5].

Além de, claro, aumentar a probabilidade de venda ao recomendar os itens que o usuário gosta (ou pelo menos que o site acredita que ele gosta), a sensação de ser atendido de forma personalizada estabelece, em muitos usuários, uma relação de confiança. Em outras palavras, SR não servem apenas para alavancar vendas, podem ser concebidos como uma forma de se estreitar o relacionamento com o cliente [5].

A academia, por sua vez, também investiu com veemência no aprimoramento das técnicas de recomendação. Em especial, a competição organizada pelo *Netflix* trouxe muita atenção para este tema, uma vez que levou pesquisadores de todo o mundo a trabalharem em um algoritmo que fosse mais eficiente que o do próprio *Netflix* [6]. Como resultado, diversos algoritmos foram propostos, a grande maioria se baseando em métodos de Aprendizado de Máquina (AM) e Data Mining (DM), o que trouxe importantes avanços para a área [4, 7].

Em geral, as técnicas de recomendação podem ser divididas em três abrangentes categorias: Baseadas em Contexto (BC), Filtragem Colaborativa (FC) e Híbridas.

No capítulo 2, iremos descrever em detalhes cada uma dessas categorias e as principais técnicas de cada uma. Por hora, é suficiente mencionar que as técnicas de FC se sobressaem às demais, principalmente em termos de desempenho e aplicabilidade. No entanto, elas sofrem com o problema do *Arranque Frio* (AF), ou problema do *Novo Usuário/Item*. Como muitas delas se baseiam em noções de similaridade entre usuários/itens, extraídas a partir das preferências dos usuários, não é possível calcular nenhuma recomendação para o novo usuário/item, visto que não se pode medir sua similaridade para com os demais usuários/itens [8].

Por esta razão, muitos trabalhos foram propostos na tentativa de amenizar o problema do AF em FC, sobretudo no tocante ao novo usuário [9–13]. Tais propostas buscam aumentar o conhecimento do sistema sobre os novos usuários, solicitando aos mesmos que deem suas preferências (na forma de avaliações) para alguns itens previamente definidos. A este processo, que precede o cálculo das recomendações, damos o nome de *Elicitação de Preferências* [11].

É importante delinear que a Elicitação de Preferências pode ser utilizada em contextos que não são necessariamente relacionados ao problema de AF. Infelizmente, na literatura, esses termos podem parecer sinônimos, uma vez que a grande maioria dos trabalhos envolvendo Elicitação de Preferências são voltados para contornar as deficiências geradas pelo AF. A fim de distinguir os cenários onde o processo de elicitação pode ser aplicado, iremos denominar o caso referente ao AF como *Elicitação de Preferências para fins de Arranque Frio*.

1.2 Proposta

Avaliar não é um hábito comum para boa parte dos usuários, de forma que, na maioria das lojas virtuais, sabe-se apenas quem comprou o que. Informações sobre a preferência dos usuários é difícil de se obter, visto que não há um incentivo direto para que eles avaliem os itens que compraram. Claro que muitos contribuem com suas preferências na esperança de poder ajudar os que estão em dúvida sobre um determinado item; outros simplesmente porque gostam de deixar seu contentamento (ou seu desprezo) registrados como forma de gratidão (ou revolta) para com o fornecedor; e há ainda aqueles que entendem que o sistema depende das preferências para calcular as recomendações e, por isto, contribuem na esperança de melhorar a qualidade das mesmas.

Em todo o caso, a motivação para contribuir com as suas preferências possui sempre aspectos colaborativos, visando o ganho do sistema, ou de outros usuários, e nunca o ganho individual. Embora tenham ocorrido mudanças significativas, sobretudo na *Web*, em direção a maior colaboração entre os usuários, o simples espírito "altruísta" parece não ser suficiente para mobilizar a maioria, que continua não

contribuindo assiduamente. Enquanto boa parte dos trabalhos que se propõem a motivar os usuários acabam por enveredar pela via colaborativa, tentando maximizá-la [14–16], outros foram além e procuraram pensar em um modelo econômico que capturasse a maneira como os usuários dão suas preferências [17, 18]. Em [19], os autores procuram elicitar preferências da multidão, usando incentivos, na esperança de que isto melhore o desempenho das técnicas de FC. Como não é possível saber quais itens são avaliáveis, [19] acaba tratando da *Elicitação de Preferências para fins de Arranque Frio* no contexto de multidão.

Uma vez que tais sistemas possuem a informação de compra, isto é, sabem quem comprou o que, é razoável assumir que, se um usuário comprou um determinado item, logo, ele é capaz de avaliar este item. Esta suposição não se dá necessariamente de forma imediata, pois certos itens requerem uma quantidade significativa de tempo para serem "consumidos" (e.g., livros e filmes). Contudo, o que podemos assumir é que, dado que um usuário efetuou a compra de um item, passado um determinado intervalo de tempo, este usuário estará apto a dar sua preferência sobre o item em questão, seja ela boa ou ruim.

Ao solicitar que o usuário avalie um item que ele já comprou, a probabilidade de se obter uma preferência, em forma de avaliação, é muito maior do que quando o mesmo é solicitado a avaliar um item a esmo. Portanto, temos aqui o outro contexto onde o processo de Elicitação de Preferências se encaixa bem, isto é, elicitar as preferências sobre itens já comprados. É importante ressalvar que nada impede que o processo de elicitação seja aplicado concomitantemente tanto neste caso quanto no caso voltado para o AF. O que chama a atenção sobre este contexto é o fato dele ser até mais eficiente na aquisição de preferências do que no primeiro caso. A *Amazon*, por exemplo, tenta persuadir seus clientes a contribuírem com suas preferências enviando *emails*, tempos após a compra, contendo os itens comprados e a opção de avaliá-los [20].

Não só a *Amazon*, mas praticamente todas lojas virtuais procuram elicitar preferências a respeito dos itens comprados pelo usuário, este fato, em si, não é nenhuma novidade. Geralmente, o processo de elicitação se dá através do envio de *emails* ou alguma outra forma de notificação. Todavia, se os *e-commerces* solicitarem que cada usuário avalie todos os itens já comprados, poderão gerar um excesso de notificações que causará incômodo aos usuários. Infelizmente, na prática, é exatamente isto o que acontece e, o pior, os usuários mais incomodados são justamente aqueles que mais compram, ou seja, os melhores clientes. Este erro ocorre devido a uma suposição muito questionável, a de que todas as preferências contribuem da mesma forma para o SR. É muito plausível que certas preferências possuam maior impacto no desempenho do SR que outras, assim poderíamos obter um ganho equivalente, ou até mesmo superior, solicitando menos notificações aos clientes, o que lhes pouparia

paciência.

Tanto neste caso como no caso referente ao AF, há um problema em comum no tocante a como selecionar os itens que devem ser avaliados por cada usuário. Nos trabalhos voltados para a *Elicitação de Preferências para fins de Arranque Frio*, os itens a se avaliar são escolhidos através de estratégias de Aprendizado Ativo (AA) [13]. Tais técnicas surgiram como uma subárea dentro de AM e visam selecionar as "melhores" instâncias de modo a construir o menor e mais eficaz conjunto de treinamento possível. Ou seja, estratégias de AA possibilitam que os modelos de AM sejam treinados de maneira mais rápida, tornando-os mais precisos com menos esforço de treinamento. No cenário referente a SR, a técnica de recomendação e as preferências podem ser considerados modelo de AM e instâncias de treinamento, respectivamente. Um apanhado geral sobre as diversas estratégias de AA pode ser encontrado no capítulo 2.

Supondo que o *e-commerce* saiba quais são os N itens que, se avaliados pelo usuário, irão trazer o maior ganho possível para o SR, em termos de desempenho, o mesmo pode oferecer ao usuário um incentivo individual a fim de garantir que este contribua com suas preferências. Como exemplo de incentivo pessoal, podemos citar desconto em futuras compras; pontos a serem trocados por brindes ou vales; *upgrade* de conta; entre outros. É importante frisar que a escolha dos incentivos que serão implantados e como os mesmos serão implantá-los foge a nossa *expertise*, visto que se trata de um problema de administração ou de economia. Entretanto, a tarefa de encontrar os N itens a serem avaliados por cada usuário pode ser atacada através de estratégias de AA.

Este trabalho propõe portanto o emprego de tais estratégias a fim de se encontrar os N itens mais importantes para cada usuário avaliar. A importância dessas preferências para o desempenho do SR deve ser tamanha a ponto do sistema estar disposto a oferecer incentivos individuais ao usuário pelas mesmas. Convencionamos chamar este problema de *Elicitação de Preferências para fins de Incentivo*.

Mais ainda, uma das principais deficiências das estratégias de AA é o fato de que todas se baseiam em suposições sobre os dados, i.e., heurísticas. Quando estas suposições não se verificam na prática, as estratégias acabam por gerar um conjunto de treinamento enviesado, algo que, ao invés de impulsionar, prejudica o desempenho do sistema. Em suma, a qualidade das recomendações depende diretamente da estratégia de AA empregada pelo SR, logo é vital que os projetistas saibam escolher aquela que trará o maior ganho para o sistema.

Para não cairmos na problemática relacionada ao viés, propomos o emprego de uma nova estratégia. A batizada *Estratégia Livre de Viés* ainda é desconhecida pela literatura relacionada a SR, sendo este trabalho sua primeira utilização em tal contexto. Como veremos no capítulo 3, esta estratégia busca selecionar os itens

de forma a aproximar a Função de Densidade de Probabilidade (FDP) empírica da população, garantindo que o conjunto de treinamento gerado é o que melhor representa a população de itens, i.e., um conjunto sem viés.

Realizamos uma extensa comparação entre diversas estratégias de AA em duas bases de dados consideradas clássicas na literatura. Ao todo, foram comparadas 17 estratégias, incluindo a *Estratégia Livre de Viés*, e verificou-se que esta última, em ambas as bases, supera as demais. O desempenho de cada estratégia foi analisado no capítulo 5, onde explicamos o motivo do sucesso ou fracasso das mesmas considerando as características das bases de dados. Mesmo havendo estratégias que possuem desempenho próximo à *Estratégia Livre de Viés*, acreditamos ter encontrado uma excelente opção para todos os tipos de bases.

1.3 Contribuições

As contribuições deste trabalhos podem ser resumidas aos itens abaixo:

- Introdução do problema referente à *Elicitação de Preferências para fins de Incentivo*, que não foi devidamente explorado na literatura até então.
- Emprego de uma estratégia nunca antes utilizada no contexto de SR, a *Estratégia Livre de Viés*, que garante produzir um conjunto de treinamento sem viés.
- Realização do experimento mais completo da literatura, envolvendo ao todo 17 estratégias, incluindo a *Estratégia Livre de Viés*.
- Análise do desempenho, seja ele bom ou ruim, de cada estratégia, levando-se em consideração as características da base de dados.

1.4 Organização

Esta dissertação está organizada em 6 capítulos, dos quais este é o primeiro. No capítulo 2, iremos descrever os principais conceitos teóricos envolvidos, além de mencionar os trabalhos relacionados ao tema. No capítulo 3, detalharemos como funciona a *Estratégia Livre de Viés* e como a mesma foi adaptada ao problema de SR. No capítulo 4, apresentamos a metodologia que guiou nossos experimentos, bem como as demais estratégias comparadas. No capítulo 5, expomos os resultados obtidos e iniciamos uma investigação tentando encontrar os motivos por trás do sucesso ou fracasso de cada estratégia. Finalmente, no capítulo 6, tiramos as devidas conclusões e apontamos possíveis trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Sistemas de Recomendação

Sistemas de Recomendação (SR) surgiram como uma área de pesquisa independente em meados da década de 90, quando foram desenvolvidos os primeiros sistemas de filtragem baseados nas preferências dos usuários como o *Tapestry* [21], o *Ringo* [22] e o *GroupLens* [23]. Até este momento, o problema de recomendação era considerado como um caso específico de filtragem de informação e, portanto, era tratado dentro da área de Recuperação de Informação (RI) [24]. A partir de então, a área devotada a SR cresceu de maneira considerável, devido a enorme aplicabilidade que esses sistemas possuem no cenário econômico atual.

Um dos trabalhos mais completos desta área, [25] define SR através de uma função de utilidade $u(i, j)$ que exprime a importância, ou o valor, que o item j possui para o usuário i. As avaliações dos usuários são consideradas observações da função de utilidade no espaço de usuários e itens. Assim, cabe ao SR tentar estimar artificialmente u para os pontos do espaço onde não há observação.

Em termos formais, seja U o conjunto de todos os usuários do sistema e I o conjunto de todos os itens do sistema. Em aplicações atuais de *e-commerce*, ambos podem chegar facilmente a ordem de milhões elementos. A função de utilidade a ser estimada é tal que $u : U \times I \to \mathbb{N}$ ou $u : U \times I \to \mathbb{R}$, dependendo do tipo de avaliação dada pelos usuários. Desta forma, um SR visa encontrar, para cada usuário i, o item j' tal que:

$$\forall i \in U, \quad j' = \operatorname*{argmax}_{j \in I} u(i, j) \tag{2.1}$$

Uma vez encontrado o item, ou conjunto de itens, que maximiza a função de utilidade para um determinado usuário, o SR pode então recomendá-lo ao usuário, na esperança de que o mesmo o adquira. Ou seja, no caso de um *e-commerce*, o sistema procura os itens que maximizam a probabilidade do usuário efetuar a

	Titanic	Shrek	Guerra nas Estrelas	Forrest Gump	Batman	Pânico
David	∅	5	4	3	∅	∅
Julia	4	3	∅	4	1	∅
Matheus	∅	∅	5	∅	∅	3
Aline	5	∅	∅	2	∅	1

Tabela 2.1: Matriz de Preferências

compra.

O espaço de usuário e itens $U \times I$ é, na prática, representado por uma matriz conhecida como matriz de preferências. Um exemplo de tal matriz pode ser visto na tabela 2.1, onde os usuários estão dispostos em linhas e os itens em colunas. Cada posição r_{ij}, dada pela interseção da linha i com a coluna j, contem a preferência do usuário i pelo item j em forma de nota. Vale ressaltar que, quando o usuário não explicita sua preferência a respeito do item, a posição r_{ij} da matriz fica em aberto, o que é representado pelo símbolo ∅. Como normalmente cada usuário avalia um subconjunto muito pequeno de itens, a grande maioria das posições na matriz de preferências são do tipo ∅. Portanto, dizemos que a matriz de preferências é uma matriz *esparsa*.

Muitas técnicas de recomendação acabam por preencher as posições em aberto com zeros, o que, na prática, significa dizer que a grande maioria das avaliações são do tipo mais baixo possível. Obviamente, isto é uma suposição implausível e pode introduzir viés na estimativa da função de utilidade. Para lidar com este problema, algumas soluções procuram introduzir, ao invés de zero, a média das avaliações do usuário; a média das avaliações do item; ou a média global de todas as avaliações presentes na matriz. Todavia, as críticas a tais soluções se apoiam no fato de que as mesmas continuam por introduzir informação na matriz que pode não condizer com a real preferência dos usuários. Ou seja, todas tentativas de preencher as posições em aberto correm o risco de introduzir viés na estimativa de u.

Uma abordagem alternativa é utilizar a matriz binária referente a matriz de preferência. Dada uma matriz de preferência R, a matriz binária B da mesma é construída da seguinte forma: se a posição $r_{ij} \in R$ contem uma preferência, então a posição correspondente $b_{ij} \in B$ é igual a 1; caso contrário b_{ij} é igual a zero. A matriz binária referente a matriz de preferências da tabela 2.1 é apresentada na tabela 2.2. Desta forma, na matriz binária, apesar de se perder a gradação das preferências, sabe-se que os valores ali presentes correspondem a um fato, ou seja, ou o usuário avaliou ou ele não avaliou o item.

Além disso, [25] classifica as técnicas utilizadas nos SR em três abrangentes categorias: Baseadas em Conteúdo (BC), Filtragem Colaborativa (FC) e Híbridas. Nas próximas seções, exploraremos em detalhes tais categorias, expondo as ideias

	Titanic	Shrek	Guerra nas Estrelas	Forrest Gump	Batman	Pânico
David	0	1	1	1	0	0
Julia	1	1	0	1	1	0
Matheus	0	0	1	0	0	1
Aline	1	0	0	1	0	1

Tabela 2.2: Matriz Binária

por trás de cada uma delas e indicando as técnicas mais famosas em cada caso.

2.1.1 Baseadas em Conteúdo

Técnicas BC derivam da forte influência que a área de RI teve sobre SR, sendo considerada por muitos como sua progenitora. Tais técnicas entendem que o resultado de $u(i, j)$ retrata a *compatibilidade* entre o usuário i e o item j e, portanto, para estimar corretamente a função u é preciso compreender como se dá tal compatibilidade.

Primeiramente, tais técnicas precisam criar um *perfil* para cada usuário e item. De posse de tais perfis, analisa-se o nível de compatibilidade entre um usuário e um item através da compatibilidade entre seus perfis, o que significa em termos formais:

$$u(i, j) = u(\text{perfil}(i), \text{perfil}(j)) \tag{2.2}$$

É interessante notar que este processo é extremamente intuitivo e que, muito provavelmente, o leitor já o pôs em prática de maneira inconsciente. Ao comprar um presente para alguém, é natural que o comprador tenha em mente um perfil do presentado e, ao analisar as opções de presente, o comprador está analisando se o perfil do presenteado é compatível com o perfil do item. Por exemplo, ao procurar um presente para uma pessoa jovem, busca-se itens com perfil jovial (e.g., artigos de aspecto moderno e informais). No caso de se presentear uma pessoa idosa, o perfil dos itens a serem considerados muda drasticamente (e.g., artigos considerados clássicos e formais).

Todos sabemos disso e colocamos isso em prática em nosso dia-a-dia, porém a medida de compatibilidade entre os perfis é feita de maneira subjetiva e até mesmo tácita, sendo difícil explicar como este processo é executado. Todavia, é esta maneira de se analisar a compatibilidade de perfis que as técnicas BC almejam aprender.

A criação de perfis, por si só, já é um desafio a parte, especialmente quando os itens são arquivos multimídia (e.g., imagens, áudio, vídeo). No caso de itens textuais, como páginas *Web*, a criação do perfil pode ser realizada através de métodos de representação vetorial de texto, como, por exemplo, Frequência de Termos, em inglês *Term Frequency* (TF), ou Frequência de Termos/Inverso Frequência de Do-

cumentos, em inglês *Term Frequency/Inverse Document Frequency* (TF-IDF) [24], muito utilizados dentro da área de RI. O perfil do item será então um vetor de tamanho n, onde cada dimensão representa o valor do TF, ou do TF-IDF, aplicado a cada uma das n palavras-chaves.

Em contrapartida, o perfil do usuário pode ser definido através de um vetor, também de tamanho n, onde cada dimensão representa o interesse do usuário sobre cada uma das n palavras-chaves. A compatibilidade entre os perfis pode ser definida como sendo o alinhamento entre esses dois vetores, dado pelo cosseno do ângulo entre os mesmos. Assim, se o cosseno for próximo de 1, significa que os vetores estão alinhados e na mesma direção, o que aponta para uma forte compatibilidade entre o usuário e o documento. Caso contrário, os vetores são ortogonais ou estão alinhados em direções opostas, o que representa uma baixa compatibilidade entre o usuário e o documento.

$$u(i, j) = \cos(\text{perfil}(i), \text{perfil}(j)) \tag{2.3}$$

Obviamente, o uso de tais vetores como perfis e da função cosseno como medida de compatibilidade foi apenas um exemplo de como poderíamos implementar um SR baseado em conteúdo. Outros métodos de representação vetorial e outras heurísticas de compatibilidade poderiam ser empregadas. No entanto, este exemplo já é suficiente para percebermos as deficiências das técnicas BC.

Primeiramente, como já foi apontado, extrair um perfil (em forma de vetor) de itens multimídia não é uma tarefa trivial. Normalmente a extração é realizada sobre os metadados desses itens, porém, em caso de haver poucos metadados ou nenhum, é praticamente inviável criar um perfil para o item, o que restringe as técnicas BC para trabalharem apenas com itens textuais.

Mais ainda, tais técnicas são incapazes de distinguir a qualidade dos itens. Ou seja, se há dois documentos sobre um determinado tema, um deles considerado bom e outro ruim, eles terão o mesmo valor de compatibilidade para o usuário interessado naquele tema, desde que usem as mesmas palavras-chaves. Por fim, as recomendações dadas por técnicas BC acabam ficando confinadas aos temas ou palavras-chaves sobre os quais os usuários demonstraram interesse explícito, tolhendo a capacidade do sistema surpreender o usuário com uma recomendação inusitada que foge à sua área de interesse pré-definida.

2.1.2 Filtragem Colaborativa

Técnicas FC foram desenvolvidas com o propósito de superar as deficiências das técnicas BC. A ideia que molda FC é utilizar as observações da função u para estimá-la, ao invés de recorrer aos perfis de usuários e itens, que podem ser criados de

maneira enganosa, acarretando em estimativas enviesadas de u. Em outras palavras, a fim de se calcular a estimativa da preferência de um usuário em relação a um item, $\hat{u}(i,j)$, técnicas de FC recorrem às preferências que os demais usuários forneceram explicitamente ao sistema, i.e., posições preenchidas da matriz de preferências. Tais técnicas podem ser divididas em duas subcategorias: FC baseada em Memória e FC baseada em Modelo.

2.1.3 FC baseada em Memória

Técnicas baseadas em Memória calculam a estimativa $\hat{u}(i,j)$ agregando as avaliações de j que foram fornecidas por usuários similares a i. Primeiramente, é preciso definir uma medida de *similaridade* entre usuários $sim(i,k)$, que pode ter várias formas, mas, essencialmente, todas visam mensurar o quão distante o usuário i está do usuário k. Entretanto, para se calcular *distância* entre dois elementos quaisquer, é necessário que ambos estejam mapeados em um espaço.

A representação dos usuários em um espaço vetorial, pode ser obtida tomando a linha da matriz de preferência, ou da matriz binária, correspondente a cada usuário. Caso o número de itens da matriz seja elevado, esta representação se dará em um espaço vetorial demasiadamente esparso, o que pode dificultar o cálculo de $sim(i,k)$. Desta forma, para fins de simplicidade, pode-se limitar a representação vetorial de i e k à interseção dos itens que ambos avaliaram.

A partir de então, basta escolher uma, dentre as diversas medidas de distância existentes na literatura, e usá-la como sendo a medida de similaridade entre os usuários. Entre as possíveis opções para $sim(i,k)$ estão a função cosseno, a correlação de Pearson, a distância Euclidiana, a distância Manhattan, entre outras.

De posse da similaridade entre os usuários, a estimativa $\hat{u}(i,j)$ é calculada através de uma *agregação* das preferências a respeito do item j, fornecidas explicitamente pelos N usuários mais similares a i. Seja C o conjunto que contém os N usuários mais similares a i e $\overline{u}(i)$ a média simples de todas as avaliações fornecidas pelo usuário i. Assim, podemos citar, como exemplos de agregação, as equações 2.4 e 2.5.

$$\hat{u}(i,j) = \frac{1}{N} \sum_{k \in C} sim(i,k) \times u(k,j) \tag{2.4}$$

$$\hat{u}(i,j) = \overline{u}(i) + \frac{1}{N} \sum_{k \in C} sim(i,k) \times [u(k,j) - \overline{u}(k)] \tag{2.5}$$

2.1.4 FC baseada em Modelo

Diferentemente das técnicas baseadas em Memória, as baseadas em Modelo procuram modelar matematicamente o comportamento do usuário através de *modelos* treinados com observações da função u. O termo *modelo* faz referência aos métodos de Aprendizado de Máquina (AM), todavia, como as observação de u estão no formato matricial (matriz de preferências), não é possível aplicar tais métodos diretamente [26].

Problemas de aprendizado supervisionado requerem um conjunto de dados devidamente rotulados no formato $\langle x, y \rangle$, onde x é uma instância em $\mathbb{R}^n$ e y sua respectiva classe ou valor associado. O fato de que, em problemas de recomendação, os dados estão dispostos em formato matricial impede que tais problemas sejam encarados como problemas tradicionais de aprendizado supervisionado.

Portanto, alguns modelos foram desenvolvidos para atuarem especificamente sobre a matriz de preferência, já que transformar os dados do formato matricial para o formato tradicional de aprendizado supervisionado não é uma tarefa trivial [26]. A enorme maioria desses modelos se baseia na decomposição da matriz de preferências. Em particular, uma decomposição ganhou grande notoriedade na literatura de SR devido a sua aplicabilidade e praticidade, a Decomposição em Valores Singulares.

2.1.5 Decomposição em Valores Singulares

A Decomposição em Valores Singulares, em inglês *Singular Value Decomposition* (SVD), se destacou dentro da literatura de SR, pois é uma maneira de se criar espaços vetoriais, de tamanho arbitrário, onde usuários e itens podem ser mapeados. A fim de apresentarmos esta decomposição em maiores detalhes, se faz necessário definir alguns conceitos importantes de álgebra linear.

Suponha que a matriz de preferências R seja quadrada ($R_{n \times n}$) e positiva. Neste caso, é possível decompor R utilizando os seus n autovalores e autovetores, λ_i e x_i, respectivamente. Tal decomposição é expressa através da equação 2.6 e, ao longo deste trabalho, iremos nos referir a mesma como *Decomposição em Autovalores*.

$$R = Q \Lambda Q^{-1} \tag{2.6}$$

A matriz Q contém os n autovetores de R em suas colunas e a matriz Λ contém os n autovalores de R em sua diagonal. Além disso, se R for simétrica, o que é pouco provável para uma matriz de preferências, sua matriz de autovetores Q apresenta uma interessante propriedade: ela é ortogonal, isto é, $Q^{-1} = Q^T$ [27].

$$Q = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}$$

$$\Lambda = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}$$

A Decomposição em Autovalores possui implicações muito importantes, principalmente nas abordagens numéricas a problemas envolvendo matrizes de dimensões elevadas. Por exemplo, suponha que é preciso calcular R^{100}. A primeira vista, o leitor pode até considerar efetuar sucessivas multiplicações de matrizes, porém, caso R possua dimensões elevadas, esta já não seria mais uma opção viável. No entanto, se fizermos uso do resultado apresentado na equação 2.6, temos:

$$R^{100} = Q\Lambda \underbrace{Q^{-1} \times Q}_{I} \Lambda \underbrace{Q^{-1} \times \ldots Q}_{\Lambda^{97}} \Lambda Q^{-1}$$

$$R^{100} = Q\Lambda^{100}Q^{-1}$$

Ou seja, com este resultado podemos substituir uma série de operações de multiplicação de matrizes (extremamente custosas) pela simples operação de potenciação da matriz Λ, que, por sua vez, se resume a potenciação dos valores em sua diagonal.

Contudo, é muito difícil que R seja quadrada em situações reais. Consideremos então R como sendo uma matriz retangular $m \times n$ de *rank* igual k, isto é, R possui k linhas e colunas linearmente independentes. Desejamos encontrar duas matrizes ortogonais U e V, que sejam bases para o espaço de linhas (*row space*) e de colunas (*column space*) de R, respectivamente. Mais ainda, desejamos que para cada vetor $v_i \in V$ e $u_i \in U$ a relação expressa pela equação 2.7 se mantenha.

$$R \underbrace{\begin{pmatrix} v_1 & v_2 & \cdots & v_k \end{pmatrix}}_{V} = \underbrace{\begin{pmatrix} u_1 & u_2 & \cdots & u_k \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_k \end{pmatrix}}_{\Sigma} \tag{2.7}$$

Como U e V são ortogonais, suas inversas são iguais às suas transpostas. Assim, podemos multiplicar ambos os lados por V^T, o que resultará na equação 2.8.

$$R = U\Sigma V^T \tag{2.8}$$

Os vetores v_i e u_i são chamados de *vetores singulares* enquanto que os escalares σ_i são chamados de *valores singulares*. Note que a relação apresentada na equação 2.8 lembra muito a relação da equação 2.6, porém a primeira é ainda mais genérica que a segunda, uma vez que R pode ter quaisquer dimensões. Nos resta ainda descobrir possíveis candidatos para V e U. Se utilizarmos a expressão dada pela equação 2.8 para escrever RR^T e R^TR, teremos:

$$RR^T = U\Sigma V^T(U\Sigma V^T)^T = U\Sigma V^T V\Sigma^T U^T = U\Sigma\Sigma^T U^T = U\Sigma^2 U^T$$
$$R^TR = (U\Sigma V^T)^T U\Sigma V^T = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T\Sigma V^T = V\Sigma^2 V^T$$

Ou seja, a matriz U nada mais é do que a matriz dos autovetores de RR^T, enquanto que a matriz V é a matriz dos autovetores de R^TR. Tanto RR^T quanto R^TR serão sempre positivas e simétricas, o que implica que sua Decomposição em Autovalores resultará em matrizes de autovetores ortogonais. Logo, temos a garantia de que tanto V quanto U serão sempre ortogonais. É interessante notar também RR^T e R^TR possuem os mesmos autovalores, o que nos garante que os valores singulares serão sempre a raiz quadrada de tais autovalores, em ambos os casos.

Em termos práticos, dada a matriz de preferências $R_{m\times n}$, com m usuários e n itens, a Decomposição em Valores Singulares retornará três matrizes: $U_{m\times k}$, $\Sigma_{k\times k}$ e $V_{n\times k}$. Apesar de termos utilizado o parâmetro k como sendo o *rank* de R, na realidade o *rank* é simplesmente um limite superior para k. Normalmente a matriz Σ possui os valores singulares dispostos em ordem decrescente em sua diagonal, deste modo pode-se verificar quais são os k ($k < rank$) valores singulares mais significativos e descartar os demais, reduzindo assim as dimensões de U e V.

As matrizes U e V possuem a representação dos usuários e itens em $\mathbb{R}^k$, respectivamente. As dimensões de tal espaço são ditas dimensões *latentes*, pois não é possível determinar uma interpretação definitiva para cada uma, pelo contrário, várias interpretações são cabíveis [28]. Esta é, sem dúvida, a grande vantagem da decomposição SVD, já que representar um usuário ou item no formato vetorial era algo considerado complexo devido a estrutura de dados utilizada (matriz de preferências). As opções que haviam eram representações em vetores esparsos com uma dimensão para cada usuário ou item. Contudo, agora é possível representá-los em vetores o quão pequenos quanto se queira, o que abre caminhos para diversos tipos de análises no espaço latente. Por exemplo, é possível agrupar usuários/itens neste espaço; calcular a similaridade entre os mesmos usando alguma noção de distância entre suas representações latentes; e, obviamente, este tipo de representação potencializa a construção de modelos genéricos, uma vez que os valores presentes na representação vetorial não estão mais associados a grandezas físicas ou conceituais.

Regularized SVD

A técnica *Regularized SVD* talvez seja a mais famosa dentre as que se utilizam da decomposição SVD. Proposta em [29], esta técnica chamou a atenção pelo bom desempenho obtido na competição do *Netflix*, além de mostrar simplicidade e elegância. Apesar de ter sido proposta durante a competição, o nome *Regularized SVD* só foi cunhado algum tempo mais tarde por [30].

O *Regularized SVD* utiliza a decomposição SVD para encontrar uma representação vetorial inicial de usuários e itens em k dimensões latentes. Seja $user_i$ o vetor de tamanho k que representa o usuário i, e $item_j$ o vetor que representa o item j também com tamanho k. Como a competição organizada pelo *Netflix* era voltada para recomendação de filmes, mais especificamente os filmes do próprio *Netflix*, [29] comenta que as k dimensões latentes poderiam corresponder a k gêneros cinematográficos e, portanto, o valor em cada dimensão aponta para a intensidade com que aquele gênero está associado ao filme. Por exemplo, se $k = 5$, poderíamos considerar que essas cinco dimensões correspondem aos gêneros ação, romance, comédia, suspense e terror. Obviamente, na prática, não é possível saber quais são exatamente os gêneros representados, mas a analogia nos ajudar a entender melhor a dinâmica do algoritmo.

Por outro lado, no caso da representação vetorial de usuários, as k dimensões representariam a inclinação, ou gosto, que o usuário possui pelo respectivo gênero. Aproveitando o exemplo anterior onde $k = 5$, poderíamos considerar estas cinco dimensões como sendo gosto por ação, gosto por romance, gosto por comédia, gosto por suspense e gosto por terror. Desta forma, assume-se que a previsão da preferência será dada pelo produto escalar dos vetores que representam o usuário e o filme, conforme mostra a equação 2.9. Ou seja, se o usuário gosta muito de filmes de ação e o filme em questão possui uma componente de ação elevada, o produto desses fatores contribuirá para aumentar o valor da previsão. No caso oposto, se o usuário detesta filmes de terror e o filme em questão possui uma componente elevada de terror, então o produto desses fatores contribuirá para a redução do valor da previsão.

$$\hat{u}(i,j) = user_i^T item_j \tag{2.9}$$

Durante a competição, as técnicas propostas foram avaliadas e ranqueadas de acordo com a Raiz do Erro Médio Quadrático, ou, em inglês, *Root Mean Square Error* (RMSE). Esta métrica calcula o Erro Quadrático (EQ) para todas a previsões feitas, toma seu valor médio e depois extrai sua raiz quadrada. Como o EQ é uma função quadrática, conforme nos mostra a equação 2.10, há um ponto de mínimo que pode ser alcançado "caminhando" na direção do gradiente da função,

método popularmente conhecido como *Gradiente Descendente*. Portanto, [29] decide retroalimentar suas estimativas $user_i$ e $item_j$, fornecidas primeiramente pela decomposição SVD, com o gradiente de EQ, descrito pelas equações 2.11 e 2.12.

$$e_{ij} = (u(i,j) - \hat{u}(i,j))^2 = \left(u(i,j) - user_i^T item_j\right)^2 \tag{2.10}$$

$$\frac{\partial e_{ij}}{\partial user_i} = \left(u(i,j) - user_i^T item_j\right) item_j = e_{ij} item_j \tag{2.11}$$

$$\frac{\partial e_{ij}}{\partial item_j} = \left(u(i,j) - user_i^T item_j\right) user_i = e_{ij} user_i \tag{2.12}$$

As constantes foram desconsideradas no cálculo das derivadas parciais, pois elas apenas quantificam a velocidade com que se "caminha" na direção do gradiente, também conhecida como taxa de aprendizado. Esta taxa será definida como ρ e será um dos parâmetros do algoritmo, devendo ser ajustada de acordo com os dados.

Uma vez que a retroalimentação com o gradiente procura levar as previsões $\hat{u}(i,j)$ o mais próximo possível das avaliações reais $u(i,j)$, presentes na matriz de preferência, isto pode gerar estimativas ruins de $user_i$ e $item_j$ para usuários e itens com poucas avaliações. Para evitar esta *superespecialização* em casos onde há pouca informação sobre usuários e itens, [29] insere o fator regularizador λ na retroalimentação. Desta maneira, a retroalimentação de $user_i$ e $item_j$ se dá de acordo com as equações 2.13 e 2.14 e se repete até que um valor limite ε de RMSE seja atingido.

$$user_i = user_i + \rho(e_{ij} item_j - \lambda user_i) \tag{2.13}$$

$$item_j = item_j + \rho(e_{ij} user_i - \lambda item_j) \tag{2.14}$$

Em suma, o *Regularized SVD* possui quatro parâmetros a se ajustar: a taxa de aprendizado ρ; o fator de regularização λ; o critério de parada ε; e o número de dimensões latentes k a serem extraídas da decomposição SVD. Segundo [30], os valores que apresentaram melhor desempenho foram: $\rho = 0,001$; $\lambda = 0,02$; $\varepsilon = 0,1$; e $k = 96$.

Improved Regularized SVD

Numa tentativa de se melhorar o *Regularized SVD*, [30] propôs acrescentar à previsão o viés do usuário $bias_i$ e o viés do item $bias_j$, conforme mostrado na equação 2.15. Por seu carácter incremental, esta técnica foi batizada de *Improved Regularized SVD*.

$$\hat{u}(i,j) = bias_i + bias_j + user_i^T item_j \tag{2.15}$$

A retroalimentação de $user_i$ e $item_j$ se dá conforme as equações 2.13 e 2.14, respectivamente. Entretanto, os novos parâmetros $bias_i$ e $bias_j$ também são recalculados a cada iteração, sendo sua retroalimentação dada pelas equações 2.16 e 2.17, respectivamente, onde μ é a média global das avaliações contidas na matriz de preferências.

$$bias_i = bias_i + \rho(e_{ij} - \lambda(bias_i + bias_j - \mu)) \tag{2.16}$$

$$bias_j = bias_j + \rho(e_{ij} - \lambda(bias_i + bias_j - \mu)) \tag{2.17}$$

2.1.6 Híbridas

Técnicas híbridas, como o nome indica, procuram mesclar características das técnicas de FC e das BC. Segundo [25], há basicamente quatro maneiras de se efetuar essa mesclagem: implementar técnicas de FC e BC de forma independente e combinar seus resultados; incorporar aspectos de conteúdo nas técnicas de FC; incorporar aspectos colaborativos nas técnicas BC; criar modelos que considerem tanto o aspecto colaborativo quanto o aspecto de conteúdo.

Como exemplo da primeira maneira, [31] calcula as recomendações finais através de uma média ponderada entre o resultado dado por uma técnica de FC e o resultado dado por uma técnica BC. A alocação dos pesos segue a dinâmica do sistema, isto é, se a proporção de avaliações aumenta (esparsidade da matriz de preferência diminui), é sinal que o lado colaborativo pode ser mais explorado e, portanto, o sistema atribui maior peso para o resultado da técnica de FC. Caso contrário, atribui-se peso maior ao resultado da técnica BC.

Aspectos de conteúdo foram explorados em técnicas de FC, sobretudo na tentativa de se amenizar os efeitos da esparsidade na matriz de preferências. Não é raro se deparar com pares de usuários cuja interseção de itens avaliados é demasiadamente pequena, nesses casos o cálculo da similaridade usando as avaliações em comum pode ser enganoso. Portanto, diversos trabalhos, como [32] e [33], propõem calcular a similaridade com base nos perfis dos usuários. Além disso, [34] propôs utilizar a informação dos perfis para preencher a matriz de preferências, método que batizou de *Filtragem Colaborativa impulsionada por Conteúdo*.

O caso contrário, isto é, a incorporação de aspectos colaborativos em técnicas BC, já não é tão usual. Como exemplo desse tipo de abordagem, podemos citar [35] que faz uso do método conhecido como Indexação Semântica Latente, em inglês,

Latent Semantic Indexing (LSI) [36], a fim de obter uma relação de similaridade entre os perfis dos usuários.

Há diversas maneira de se implementar a última abordagem, que consiste em criar um modelo abrangendo tanto os aspectos colaborativos quanto os de conteúdo. Em particular, métodos de Aprendizado de Máquina (AM) foram extensamente explorados, pois permitem que as instâncias do conjunto de treinamento sejam descritas considerando diversos aspectos diferentes, entre os quais os colaborativos e de conteúdo. Como exemplo deste tipo de abordagem podemos citar [37], [38] e [39].

2.2 Aprendizado Ativo

Modelos de Aprendizado de Máquina (AM) buscam identificar padrões "escondidos" nos dados. Para isto, tais modelos são expostos a um conjunto limitado de dados, conhecido como *conjunto de treinamento*, o qual é utilizado para refinar o modelo até que o mesmo seja capaz de identificar, com precisão, os padrões ali presentes. A esta etapa damos o nome de *treinamento*.

A fim de avaliarmos se a etapa de treinamento foi bem sucedida, o modelo é novamente utilizado, desta vez em um outro conjunto de dados, que não possui interseção com o primeiro, chamado de *conjunto de teste*. Como o nome indica, o modelo é solicitado a prever as classes ou valores associados às instâncias desse conjunto e as previsões, por sua vez, são averiguadas junto às verdadeiras classes ou valores associados. A esta etapa damos o nome de *teste* e, como resultado final, obtemos o valor de acurácia do modelo (número de previsões corretas dividido pelo total de previsões feitas).

É evidente que, se desejarmos obter o modelo mais acurado possível, o conjunto de treinamento deve ser o que melhor *representa* a população dos dados, no entanto, em estatística, é difícil mensurar o quão bem uma amostra representa sua população. Em termos práticos, julga-se que quanto maior o tamanho da amostra mais representativa ela é. O mesmo ocorre com conjuntos de treinamento, normalmente procura-se treinar o modelo com o maior e mais diversificado conjunto possível. Porém, ao contrário de amostras, que são simplesmente coletadas ou extraídas, um conjunto de treinamento requer esforço maior para ser construído, pois, além da coleta ou extração das instância, é necessário *rotular* cada uma delas.

Um conjunto de treinamento típico possui o formato $\langle x, y \rangle$, onde x é uma instância em $\mathbb{R}^n$ e y sua respectiva classe - em problemas de classificação - ou valor associado - em problemas de regressão. Ao processo de associar classes ou valores a cada uma das instâncias damos o nome de *rotulagem* e há determinados tipos de problemas onde tal processo pode ser muito custoso.

Por exemplo, no caso de classificação de notícias, é necessário que um rotula-

dor humano leia cada uma das notícias e escolha a classe que melhor lhe convém (e.g., esporte, economia, política, cultura, etc.). Supondo que cada notícia tenha alguns parágrafos de extensão, o tempo de leitura e rotulagem de cada instância é considerável. Para se construir um conjunto de treinamento grande o suficiente, ou gasta-se muito contratando diversos rotuladores que realizarão a rotulagem em pouco tempo; ou contrata-se poucos rotuladores que demorão bastante tempo para terminal a rotulagem. Em ambos os casos há um alto custo envolvido, seja em relação a tempo ou em relação a dinheiro.

Há casos ainda onde se necessita de rotuladores devidamente capacitados, o que acarreta em custos de contratação mais elevados. É o caso, por exemplo, da construção de conjuntos de treinamento para reconhecimento de voz. Cada discurso gravado deve ser destrinchado em fonemas e, para tal, é preciso ter linguistas como rotuladores. Além disso, mesmo com o auxílio de especialistas o processo de rotulagem em fonemas é, por natureza, complexo e pode demorar muito tempo para ser concluído.

Aprendizado Ativo (AA) surgiu como uma área de pesquisa dentro de AM que busca otimizar a construção do conjunto de treinamento. Normalmente, a seleção das instâncias que serão rotuladas se dá de forma aleatória. Todavia, é plausível que haja uma maneira inteligente de se escolher as instâncias a serem rotuladas de forma que possamos construir um conjunto de treinamento mais eficaz, isto é, um conjunto onde o modelo poderá obter bons resultados de acurácia ainda que possuindo poucas instâncias rotuladas.

As técnicas de AA, comumente chamadas de estratégias de AA, foram genericamente descritas e classificadas em [40]. Contudo, neste trabalho, optamos pela classificação realizada em [41], que aborda as estratégias especificamente sob o ponto de vista de SR. Os autores em [41] analisam as estratégias por diversos ângulos e apontam os vários critérios que devem ser considerados ao se escolher uma estratégia para o SR. Mais ainda, [41] classificam as estratégias de AA em duas abrangentes categorias que analisaremos nas próximas seções, são elas: Redução de Incerteza e Redução de Erro.

2.2.1 Redução de Incerteza

O conceito de *incerteza* é extremamente importante para se entender AA como um todo. Há instâncias para as quais a rotulagem será fácil e há aquelas para as quais a rotulagem será difícil, ou melhor, duvidosa. A incerteza associada a uma instância mede o quão difícil, ou duvidoso, será rotular esta instância. Estratégias baseadas na redução da incerteza assumem que ao se rotular as instâncias com maior valor de incerteza, isto é, acrescentá-las no conjunto de treinamento, estamos

reduzindo o erro do modelo que ali será treinado.

Obviamente, reduzir a incerteza sobre os dados não implica necessariamente em redução do erro do modelo. Se estivermos utilizando um modelo inadequado, ao reduzir a incerteza podemos simplesmente ficar mais certos sobre o modelo errado. Portanto, a medida que o conjunto de treinamento cresce, deve-se verificar se o modelo continua adequado, caso contrário, o simples emprego de tais estratégias acaba sendo enganoso e prejudica mais do que auxilia.

A incerteza associada a cada instância, por sua vez, pode ser medida sob diversos aspectos, como, por exemplo, a incerteza que o modelo possui sobre a instância; a incerteza associada à região de decisão; a incerteza associada à posição da instância no espaço; entre outros. Analisaremos os principais aspectos apresentando como se dá o cálculo da incerteza em cada caso.

2.2.2 Incerteza do Modelo

Quando nos referimos à *Incerteza do Modelo*, estamos afirmando que a incerteza associada a uma instância é medida pela incerteza que o modelo possui sobre esta instância. Este caso é bem nítido quando tratamos de modelos probabilísticos, isto é, modelos cujas previsões são dadas com base na probabilidade de uma instância pertencer a uma classe em detrimento das outras. Por exemplo, [40] apresenta um modelo probabilístico bem simples θ, onde só há duas possíveis classes y_1 e y_2.

$$\theta(x) = \begin{cases} y_1, & \text{se } P(y_1|x) > P(y_2|x) \\ y_2, & \text{se } P(y_2|x) > P(y_1|x) \end{cases} \tag{2.18}$$

Assim, uma estratégia baseada na incerteza do modelo selecionaria para rotulagem a instância x' cujas probabilidades posteriores $P(y_i|x')$ fossem as mais próximas possíveis. Como o modelo é binário, se uma das probabilidades for alta, obrigatoriamente a outra será baixa, isto é sinal de que o modelo está certo sobre a respectiva classe da instância. Caso contrário, se ambas forem próximas de $^1/_2$, é sinal de que o modelo está em dúvida quanto a classe da instância e, portanto, esta instância possui alta incerteza. A formalização da estratégia pode ser encontrada na equação 2.19, onde $y' = \text{argmax}_i\, P(y_i|x)$.

$$x' = \underset{x}{\text{argmax}}\, 1 - P(y'|x) \tag{2.19}$$

Caso haja mais de duas classes, o que, na prática, é muito mais realista, há uma adaptação da estratégia anterior, conhecida como *margem de amostragem*, que se baseia apenas na distância entre as duas maiores probabilidades posteriores. Sejam y_1 e y_2 as classes com maior probabilidade para a instância x, logo a instância de maior incerteza x' é aquela cuja distância entre $P(y_1|x)$ e $P(y_2|x)$ é a menor.

$$x' = \underset{x}{\operatorname{argmin}} \, |P(y_1|x) - P(y_2|x)| \tag{2.20}$$

Uma das críticas levantas por [40] a respeito da margem de amostragem é que ela desconsidera completamente a certeza (negativa) que o modelo possui quanto às outras classes. Por exemplo, no caso de três classes, se tivermos, para uma instância x_a, as probabilidades referentes às três classes muitos próximas, enquanto que, para outra instância x_b, há duas probabilidades altas que se sobressaem a terceira. Pela margem de amostragem, ambas possuirão o mesmo valor de incerteza, desde que a diferença entre as duas maiores probabilidades seja igual. Entretanto, a incerteza relacionada a x_a deveria ser muito maior do que a relacionada a x_b, pois, no caso desta última, o modelo está certo de que a terceira classe não é adequada para a instância, certeza esta que não temos no primeiro caso.

A fim de contornar este problema, [40] sugere o uso da entropia para se calcular a incerteza de uma instância. O conceito da entropia é muito utilizado na física e mede o quão "agitado" um sistema está. Esta agitação física, por sua vez, pode ser traduzida em termos probabilísticos como incerteza. Por exemplo, se um determinado sistema de partículas está estável, então a probabilidade de se encontrar uma partícula em um determinado local é mais alta do que nos demais locais. Uma vez que a agitação das partículas é baixa, ou seja, entropia é baixa, elas tendem a não se mover muito. Caso contrário, se a agitação das mesmas for alta, significa que a probabilidade de se encontrar a partícula em qualquer local do sistema é a mesma, dado que ela se move muito (a entropia é alta). Como a entropia consegue captar o quão equiprovável as previsões para uma instância são, ela é capaz de medir a incerteza da instância considerando todas as possíveis classes em que esta pode ser classificada.

$$x' = \underset{x}{\operatorname{argmax}} - \sum_{i=1}^{N} P(y_i|x) \log P(y_i|x) \tag{2.21}$$

Existem ainda muitas outras maneiras de se calcular a incerteza de uma instância com base no modelo. Considerando trabalhos voltados para SR, [42], por exemplo, calcula a incerteza associada a um item através do Valor Esperado da Informação Perfeita, em inglês, *Expected Value of Perfect Information* (EVPI). Já [43] chega conclusão que, quando se trata de modelos probabilísticos latentes (e.g., *Aspect Model*, *Flexible Mixture Model*), utilizar a entropia como medida de incerteza pode não ser uma boa opção, pois pode acabar enviesando a estimativa do modelo. Desta forma, [43] propõe o uso do divergente Kullback–Leibler (ver seção 3.2) entre as probabilidades estimadas pelo modelo e as reais. Todavia, este trabalho assume que o usuário sempre será capaz de avaliar o item de maior incerteza, suposição que não

é verdadeira para o problema de *Elicitação de Preferências para fins de Arranque Frio*. Logo, [44] adiciona ao divergente a probabilidade do usuário avaliar o item, que, em termos práticos, se resume a popularidade do mesmo.

Todos esses trabalhos utilizaram modelo probabilísticos, porém é possível desenvolver estratégias baseadas na incerteza do modelo quando o mesmo não é probabilístico. Por exemplo, no caso de classificação por vizinhos mais próximos, a proporção de vizinhos de cada classe pode ser tomada como sendo a probabilidade da instância pertencer àquela classe. No caso de modelos que traçam funções discriminantes no espaço, a distância da instância para a fronteira criada pela função pode ser tomada como medida de incerteza. No caso de problemas de regressão, a variância da previsão pode ser utilizada como incerteza [40], ou ainda pode-se supor que os maiores, ou menores valores, fornecidos pelo modelo são os de maior incerteza [13].

2.2.3 Incerteza do Comitê

Estratégias baseadas na incerteza do modelo dependem fortemente da aptidão do modelo. Ou seja, caso o modelo escolhido não seja adequado aos dados, o erro do modelo se propagará à estratégia que, por sua vez, construirá o conjunto de treinamento de maneira enviesada, distorcendo ainda mais o modelo. Tendo esse ciclo desastroso em mente, desenvolveu-se uma medida de incerteza que considera vários modelos distintos para, assim, minimizar a possibilidade de se ter um modelo errado.

Os diversos modelos formam um *comitê* onde cada um faz previsões para as instância não rotuladas de acordo com sua lógica. A partir de então, verifica-se, para cada instância, a discordância entre as previsões fornecidas pelo comitê. A instância de maior incerteza é aquela onde há maior discordância entre os modelos, enquanto que, em contrapartida, a de menor incerteza é aquela onde há maior concordância, sendo o ápice da "certeza" a unanimidade das previsões fornecidas pelo comitê [45]. Este modo de se calcular a incerteza é conhecido como *Incerteza do Comitê*.

Quanto mais modelos tivermos no comitê, supostamente melhor será a construção do conjunto de treinamento. Entretanto, isto significa que os n modelos devem ser treinados toda vez que se deseja selecionar uma instância para rotulagem, o que pode ser extremamente custoso. Uma solução para reduzir o tempo de treinamento seria treinar os n modelos em paralelo, porém isto acarretaria em um alto custo de infraestrutura. Como alternativa viável, [13] propôs um comitê formado não por modelos, mas por estratégias baseadas na incerteza do modelo e na incerteza da instância. Cada uma delas elege as instâncias de maior incerteza, conforme sua própria lógica. Ao final, as instâncias que receberam mais "votos" são

escolhidas como sendo as que possuem maior incerteza global, por este motivo esta estratégia se chama *votação*. É interessante notar que a votação diverge da ideia original de incerteza do comitê, escolhendo as instâncias com base na concordância ao invés da discordância.

2.2.4 Incerteza da Instância

O conceito de *Incerteza da Instância* assume que a medida de incerteza deve ser calculada com base na instância apenas e não depender de nenhum modelo de previsão, afinal todo modelo se baseia em alguma suposição sobre os dados, que, por sua vez, pode ser enganosa.

Dentro do contexto relacionado a SR, a incerteza de uma instância pode ser dada através de alguma medida de dispersão, e.g., entropia ou variância. Uma vez que as instâncias são os itens do sistema que serão selecionados para serem avaliados por um usuário, é provável que tais itens já tenham recebido avaliações de outros usuários. Assim, a dispersão das avaliações pode ser considerada como medida de incerteza, que, em termos práticos, representa o nível de consenso que os usuários possuem sobre o item em questão (quanto maior a incerteza, menor o consenso). Obviamente, esta suposição, ou heurística, pode não se verificar na prática, acarretando na construção de um conjunto de treinamento enviesado.

Um dos primeiros trabalhos a abordar *Elicitação de Preferência para fins de Arranque Frio*, [9] combina a entropia das avaliações recebidas por um item com sua popularidade, numa tentativa de combinar a dispersão do item como a probabilidade dele ser avaliado por um usuário qualquer. Em [10] os autores investigam outras maneiras de se calcular a entropia de um item (e.g., utilizar o ganho da informação ao invés da entropia simples) e [46] combina a popularidade com a variância das avaliações.

Outras abordagens incluem selecionar o item que possui maior similaridade com os demais [16] e selecionar aqueles que, se avaliados, possibilitarão o refinamento do cálculo da similaridade entre os usuários, melhorando assim as previsões calculadas com base nas mesmas [47]. Infelizmente, ambas estratégias são especificas para técnicas de FC baseadas em Memória, o que as torna dependentes desse modelo.

2.2.5 Redução de Erro

Como mencionado anteriormente, estratégias baseadas na redução da incerteza supõem que, ao reduzir a incerteza no conjunto de treinamento, o erro do modelo necessariamente reduzirá. Esta suposição é enganosa, uma vez que o erro do modelo pode depender tanto do conjunto de treinamento quanto da aptidão do próprio. Tais

estratégias combatem apenas o primeiro fator e negligenciam o segundo, portanto estratégias focadas em reduzir o erro do modelo como um todo foram desenvolvidas.

Seja Tr o conjunto de treinamento; T o conjunto de testes; $\theta^{Tr}(x)$ a previsão para a instância x dada pelo modelo treinado em Tr; e $u(x)$ a verdadeira classe da instância x. Uma das estratégias mais intuitivas, com base no redução do erro do modelo, é o valor esperado do erro do modelo [41]. Nesta estratégia, cada instância não rotulada é inserida em Tr como pertencendo a uma das possíveis N_y classes. A partir de então, treina-se o modelo com $Tr \cup \langle x, y_i \rangle$ e verifica-se seu desempenho em T através da uma métrica de avaliação Γ, como, por exemplo, MAE ou RMSE (ver seção 4.3). Assumindo que x possui a mesma probabilidade de pertencer a cada uma das classes y_i, a melhor instância a ser rotulada é x' dada conforme a equação 2.22.

$$x' = \underset{x}{\operatorname{argmin}} \frac{1}{N_y} \sum_{i=1}^{N_y} \sum_{x_a \in T} \Gamma \left[u(x_a), \theta^{Tr \cup \langle x, y_i \rangle}(x_a) \right] \tag{2.22}$$

Um dos problemas de se basear no erro fornecido pelo conjunto de testes é que a seleção da instância pode ficar demasiadamente condicionada ao conjunto em si, ou seja, estaríamos selecionando instâncias que favorecem a superespecialização do modelo. Como desejamos que o mesmo seja genérico o suficiente para ser utilizado com outros conjuntos de testes, uma melhor opção seria aplicar a mesma abordagem na validação cruzada (do conjunto de treinamento) e deixar o conjunto de testes apenas para verificação final. Seja N_P o número de partições de Tr, P_k a partição utilizada para testes, logo, a partição utilizada para treinamento será $P_{Tr} = Tr \setminus P_k$. Desta maneira, x' é dado através da equação 2.23.

$$x' = \underset{x}{\operatorname{argmin}} \frac{1}{N_y N_P} \sum_{i=1}^{N_y} \sum_{k=1}^{N_P} \sum_{x_a \in P_k} \Gamma \left[u(x_a), \theta^{P_{Tr} \cup \langle x, y_i \rangle}(x_a) \right] \tag{2.23}$$

Uma estratégia baseada na redução do erro foi proposta em [46] para o problema do AF em SR. Neste trabalho, utiliza-se um modelo de recomendação para prever as preferências dos usuários em relação aos itens. De posse das mesmas, cada item é inserido no conjunto de treinamento como tendo recebido a avaliação prevista. O modelo é então treinado e avaliado no conjunto de testes através do RMSE e os itens a serem solicitados ao usuário são aqueles que resultam no menor valor de RMSE. Esta estratégia foi batizada por [46] como *estendimento guloso*, uma vez que ela é gulosa em relação ao RMSE, i.e., sempre busca o item que mais minimiza o RMSE no momento.

Seguindo esta mesma linha, [48] propõe uma estratégia baseada na redução do erro utilizando uma árvore de decisão. Cada nó da árvore representa um item e,

de acordo com a avaliação dada para o item em questão, o usuário é direcionado a avaliar outro item, seguindo um caminho na árvore até chegar a um nó folha. Ao contrário de árvores de decisão típicas, onde a hierarquia dos nós é dada pela entropia ou pelo ganho de informação, neste caso a hierarquia dos nós é definida pelo valor de RMSE associado ao item.

O principal problema das estratégias baseadas na redução de erro é o esforço computacional de se calcular o erro associado a cada instância. Como [46] comenta, o estendimento guloso supera as demais estratégias em termos de acurácia, porém, em termos de tempo, ele é indiscutivelmente inferior às demais. Se o SR tiver um número considerável de usuários e itens, treinar e avaliar o modelo de recomendação para descobrir o erro associado a cada item é uma tarefa extremamente árdua, podendo levar horas. Além disso, tais estratégias ficam extremamente condicionadas ao conjunto de testes o que pode levar a superespecialização do modelo.

Capítulo 3

A Estratégia Livre de Viés

3.1 Desafios atuais

Conforme visto na seção 2.2, as estratégias de Aprendizado Ativo (AA) sugiram face a dificuldade de se rotular de uma enorme quantidade de instâncias. As aplicações atuais estão gerando dados em velocidade e volume que seriam inimagináveis alguns poucos anos atrás. Este fenômeno, batizado de *Big Data*, tem atraído a atenção da indústria, por seu valor econômico, e da academia, pelo desafio tecnológico que apresenta [49].

Técnicas usuais de Aprendizado de Máquina (AM) supervisionado requerem um conjunto de treinamento devidamente rotulado em classes para que seja possível "ensinar" um modelo a reconhecer o padrão que distingue tais classes. Obviamente, quanto maior, mais balanceado e mais diversificado for o conjunto de treinamento, melhor será a acurácia do modelo. Entretanto, quando o volume de dados atinge a ordem de *gigabytes* ou *terabytes*, rotular tantas instâncias é um procedimento extremamente complicado, se não impossível.

Desta forma, uma possível solução é extrair uma amostra de tamanho muito menor, porém mantendo as principais propriedades estatísticas da base, e usá-la para o treinamento do modelo. Estratégias de AA poderiam ser facilmente empregadas a fim de se construir tal amostra, todavia, como as mesmas, em sua maioria, são baseadas em heurísticas, há um alto risco de se gerar amostras que não representam a base de maneira apropriada, ou seja, amostras enviesadas.

Com isto em mente, [50] propõe um método que busca extrair a amostra mais representativa possível a partir de uma enorme base de dados. Neste trabalho, adaptamos o método de [50] para atuar como uma estratégia de AA em Sistemas de Recomendação (SR) e a batizamos de *Estratégia Livre de Viés*.

3.2 O método genérico

Seja P um conjunto demasiadamente grande de instâncias pertencentes a um espaço vetorial $\mathbb{R}^k$. Assim, podemos assumir que a Função Densidade de Probabilidade (FDP) $\hat{p}$, estimada a partir de P, é uma boa estimativa da FDP populacional. Desejamos encontrar uma amostra Q, de tamanho muito menor que P ($N_Q \ll N_P$), tal que a estimativa da FDP $\hat{q}$, feita a partir de Q, seja a mais próxima possível de $\hat{p}$.

Existem dois tipos de estimadores de FDP: paramétricos e não-paramétricos. Os primeiros buscam ajustar uma função paramétrica que representa a densidade dos dados, normalmente utilizando o Erro Médio Quadrático (EMQ) como indicador de aptidão [51]. Ou seja, quanto menor for o EMQ, mais apta é a função para representar a densidade dos dados. Obviamente, pode-se fazer o EMQ chegar a zero simplesmente ajustando os parâmetros para que a função passe por todos os pontos, caso conhecido com superespecialização. Além disso, caso não haja muitos dados disponíveis, pode-se acabar escolhendo uma função que destoa significativamente da FDP populacional.

Estimadores não-paramétricos, por sua vez, são mais apropriados quando há escassez de dados (propriedade que se espera para Q). Portanto, [50] faz a opção por estimar as FDPs através do estimador não-paramétrico conhecido como *Kernel Density Estimator* (KDE). Tal estimador consiste basicamente em aplicar uma função *kernel*, centrada em cada uma das instâncias da amostra, e tomar a média simples das mesmas como sendo a função que descreve a densidade dos dados [52].

Uma função *kernel* é qualquer função cuja integral de menos infinito até infinito, em todas as dimensões, é igual a 1. Podemos citar como exemplo de função *kernel* a função Gaussiana multivariada, dada através da equação 3.1, onde $x \in \mathbb{R}^k$, $\mu \in \mathbb{R}^k$, $det(\sigma)$ é o determinante da matriz de covariância e σ^{-1} é a sua inversa. Em [50], o autor faz a opção de usar, para o KDE, a função Gaussiana multivariada, assim $\hat{p}$ e $\hat{q}$ são dadas pelas equações 3.3 e 3.2, respectivamente.

$$G(x, \mu, \sigma) = \frac{1}{\sqrt{det(\sigma)(2\pi)^k}} \exp\left(-\frac{1}{2}(x-\mu)^T \sigma^{-1}(x-\mu) \right) \tag{3.1}$$

$$\hat{p}(x) = KDE(P) = \frac{1}{N_P} \sum_{i \in P} G(x, i, \sigma) \tag{3.2}$$

$$\hat{q}(x) = KDE(Q) = \frac{1}{N_Q} \sum_{i \in Q} G(x, i, \sigma) \tag{3.3}$$

Uma vez que se tenha em mãos as estimativas de FDP para ambas as amostras, é necessário saber o quão distante $\hat{q}$ está de $\hat{p}$. Na literatura, há diversas formas de

mensurar a distância entre duas FDPs, conhecidas como *divergentes* e habitualmente representadas através do símbolo Δ. Como exemplo, podemos citar os divergentes Kullback–Leibler (equação 3.4), Cauchy-Schwarz (equação 3.5), *Integrated Squared Error* (equação 3.6), entre outros. Calcula-se então a utilidade de cada instância em P, isto é, o quanto tal instância, se fosse inserida em Q, reduziria o divergente entre $\hat{p}$ e $\hat{q}$. A instância que deve ser inserida em Q é aquela que mais promove a redução do divergente, ou seja, aquela com menor valor de utilidade. O algoritmo 1 apresenta o procedimento genérico para formação do conjunto Q, onde o critério de parada foi definido como sendo o número de instâncias ε em Q, mas também poderia ser o valor do divergente.

$$KL(\hat{p}, \hat{q}) = \int_{-\infty}^{\infty} \hat{p(x)} \log \frac{\hat{p}(x)}{\hat{q}(x)} \, \mathrm{d}x \tag{3.4}$$

$$CS(\hat{p}, \hat{q}) = -\log \frac{\int_{-\infty}^{\infty} \hat{p}(x)\hat{q}(x) \, \mathrm{d}x}{\sqrt{\int_{-\infty}^{\infty} [\hat{p}(x)]^2 \int_{-\infty}^{\infty} [\hat{q}(x)]^2 \, \mathrm{d}x}} \tag{3.5}$$

$$ISE(\hat{p}, \hat{q}) = \int_{-\infty}^{\infty} [\hat{p}(x) - \hat{q}(x)]^2 \, \mathrm{d}x \tag{3.6}$$

Algoritmo 1 Método genérico para formação do conjunto Q

1: $\hat{p} \leftarrow KDE(P)$
2: $Q \leftarrow \emptyset$
3: **enquanto** $N_Q < \varepsilon$ **faça**
4: **para cada** $i \in P \setminus Q$ **faça**
5: $\hat{q} \leftarrow KDE(Q \cup \{i\})$
6: $util(i) \leftarrow \Delta(\hat{p}, \hat{q})$
7: **fim para**
8: $i' \leftarrow \mathrm{argmin}_i \, util(i)$
9: $Q \leftarrow Q \cup \{i'\}$
10: **fim enquanto**

3.3 Otimização do método

Como é possível perceber, o procedimento descrito pelo algoritmo 1 funciona para qualquer tipo de divergente. Além disso, este procedimento também funcionaria para qualquer estimador de FDP, paramétrico ou não-paramétrico. Todavia, [50] mostra que, quando o KDE (com *kernel* Gaussiano) é utilizado em conjunto com o divergente *Integrated Squared Error* (ISE), isto acarreta num significativo ganho de desempenho.

Para tornarmos isto nítido ao leitor, substituiremos as fórmulas que descrevem $\hat{p}$ e $\hat{q}$ (equações 3.2 e 3.3, respectivamente) na fórmula do ISE (equação 3.6). O

resultado pode ser encontrado na equação 3.7.

$$ISE(\hat{p},\hat{q}) = \int_{-\infty}^{\infty} \left[\frac{1}{N_P} \sum_{i \in P} G(x,i,\sigma) - \frac{1}{N_Q} \sum_{j \in Q} G(x,j,\sigma) \right]^2 \mathrm{d}x \tag{3.7}$$

Expandindo o produto notável, temos:

$$ISE(\hat{p},\hat{q}) = \int_{-\infty}^{\infty} \left[\left(\frac{1}{N_P} \sum_{i \in P} G(x,i,\sigma) \right)^2 - 2 \left(\frac{1}{N_P} \sum_{i \in P} G(x,i,\sigma) \right) \left(\frac{1}{N_Q} \sum_{j \in Q} G(x,j,\sigma) \right) + \left(\frac{1}{N_Q} \sum_{j \in Q} G(x,j,\sigma) \right)^2 \right] \mathrm{d}x$$

$$ISE(\hat{p},\hat{q}) = \int_{-\infty}^{\infty} \left[\frac{1}{{N_P}^2} \sum_{i \in P} \sum_{i' \in P} G(x,i,\sigma) G(x,i',\sigma) - \frac{2}{N_P N_Q} \sum_{i \in P} \sum_{j \in Q} G(x,i,\sigma) G(x,j,\sigma) + \frac{1}{{N_Q}^2} \sum_{j \in Q} \sum_{j' \in Q} G(x,j,\sigma) G(x,j',\sigma) \right] \mathrm{d}x$$

Aplicando o teorema da convolução de Gaussianas, que nos diz que $\int G(x,\alpha,\sigma)G(x,\beta,\sigma) = G(\alpha,\beta,2\sigma)$ [53], temos:

$$ISE(\hat{p},\hat{q}) = \frac{1}{{N_P}^2} \sum_{\substack{i \in P \\ i' \in P}} \left[\int_{-\infty}^{\infty} G(x,i,\sigma) G(x,i',\sigma) \, \mathrm{d}x \right] - \frac{2}{N_P N_Q} \sum_{\substack{i \in P \\ j \in Q}} \left[\int_{-\infty}^{\infty} G(x,i,\sigma) G(x,j,\sigma) \, \mathrm{d}x \right] + \frac{1}{{N_Q}^2} \sum_{\substack{j \in Q \\ j' \in Q}} \left[\int_{-\infty}^{\infty} G(x,j,\sigma) G(x,j',\sigma) \, \mathrm{d}x \right]$$

$$ISE(\hat{p},\hat{q}) = \frac{1}{{N_P}^2} \sum_{\substack{i \in P \\ i' \in P}} G(i,i',2\sigma) - \frac{2}{N_P N_Q} \sum_{\substack{i \in P \\ j \in Q}} G(i,j,2\sigma) + \frac{1}{{N_Q}^2} \sum_{\substack{j \in Q \\ j' \in Q}} G(j,j',2\sigma)$$

Seja $\hat{q}^{(N_Q)}$ a FDP calculada a partir do conjunto Q contendo N_Q instâncias, ou seja, $\hat{q}^{(N_Q)}$ é dada através da equação 3.3. Seja γ a melhor instância a se inserir em Q, isto é, aquela que resulta no menor ISE entre as duas FDPs. Ao inserirmos γ em Q e recalcularmos $\hat{q}$ agora com $N_Q + 1$ instâncias, isto é, $\hat{q}^{(N_Q+1)}$, encontramos para $\hat{q}^{(N_Q+1)}$ e para $ISE(\hat{p},\hat{q}^{(N_Q+1)})$ as equações 3.8 e 3.9, respectivamente.

$$\hat{q}^{(N_Q+1)} = \frac{1}{N_Q + 1} \left[\left(\sum_{j \in Q \setminus \{\gamma\}} G(x,j,\sigma) \right) + G(x,\gamma,\sigma) \right] \tag{3.8}$$

$$ISE(\hat{p}, \hat{q}^{(N_Q+1)}) = \frac{1}{{N_P}^2} \sum_{\substack{i \in P \\ i' \in P}} G(i, i', 2\sigma) - 2\frac{\sum_{i \in P} G(x, i, \sigma)}{N_P(N_Q+1)} \left[\sum_{j \in Q \setminus \{\gamma\}} G(x, j, \sigma) + G(x, \gamma, \sigma) \right] + \frac{1}{(N_Q+1)^2} \left[\sum_{j \in Q \setminus \{\gamma\}} G(x, j, \sigma) + G(x, \gamma, \sigma) \right]^2 \tag{3.9}$$

Expandindo o produto notável presente na equação 3.9, temos:

$$\begin{aligned} ISE(\hat{p}, \hat{q}^{(N_Q+1)}) = {} & \frac{1}{{N_P}^2} \sum_{\substack{i \in P \\ i' \in P}} G(i, i', 2\sigma) - \frac{2}{N_P(N_Q+1)} \sum_{\substack{i \in P \\ j \in Q \setminus \{\gamma\}}} G(i, j, 2\sigma) \\ & - \frac{2}{N_P(N_Q+1)} \sum_{i \in P} G(i, \gamma, 2\sigma) + \frac{1}{(N_Q+1)^2} \sum_{\substack{j \in Q \setminus \{\gamma\} \\ j' \in Q \setminus \{\gamma\}}} G(j, j', 2\sigma) \\ & + \frac{2}{(N_Q+1)^2} \sum_{j \in Q \setminus \{\gamma\}} G(j, \gamma, 2\sigma) + \frac{1}{(N_Q+1)^2} G(\gamma, \gamma, 2\sigma) \end{aligned}$$

Seja $V(\hat{p}, \hat{q})$ o Potencial de Informação [53] entre $\hat{p}$ e $\hat{q}$ dado através da equação 3.10. Seja também π a constante definida através da equação 3.11. Desta forma, é possível reescrever $ISE(\hat{p}, \hat{q}^{(N_Q+1)})$ em termos de $V(\hat{p}, \hat{q})$ e π, conforme nos mostra a equação 3.12.

$$V(\hat{p}, \hat{q}) = \frac{1}{N_P N_Q} \sum_{\substack{i \in P \\ j \in Q \setminus \{\gamma\}}} G(i, j, 2\sigma) \tag{3.10}$$

$$\pi = \frac{N_Q}{N_Q+1} \tag{3.11}$$

$$\begin{aligned} & ISE(\hat{p}, \hat{q}^{(N_Q+1)}) = V(\hat{p}, \hat{p}) - 2\pi V(\hat{p}, \hat{q}) - 2(1-\pi)\frac{1}{N_P} \left(\sum_{i \in P} G(i, \gamma, 2\sigma) \right) \\ & + \pi^2 V(\hat{q}, \hat{q}) + 2\pi(1-\pi)\frac{1}{N_Q} \left(\sum_{j \in Q \setminus \{\gamma\}} G(j, \gamma, 2\sigma) \right) + (1-\pi)^2 G(\gamma, \gamma, 2\sigma) \end{aligned} \tag{3.12}$$

De acordo com a definição do KDE, apresentada nas equações 3.2 e 3.3, $\hat{p}(\gamma) = \frac{1}{N_P} \sum_{i \in P} G(i, \gamma, 2\sigma)$ e $\hat{q}(\gamma) = \frac{1}{N_Q} \sum_{j \in Q \setminus \{\gamma\}} G(j, \gamma, 2\sigma)$. Além disso, $G(\gamma, \gamma, 2\sigma)$ é uma constante a qual nos referiremos como $\mathcal{K}$. Assim, renomeando e reorganizando as parcelas da equação 3.12, temos:

$$ISE(\hat{p}, \hat{q}^{(N_Q+1)}) = V(\hat{p}, \hat{p}) - 2\pi V(\hat{p}, \hat{q}) + \pi^2 V(\hat{q}, \hat{q}) + (1-\pi)^2 \mathcal{K} + 2(1-\pi)\left[\pi\hat{q}(\gamma) - \hat{p}(\gamma)\right] \tag{3.13}$$

Ou seja, para minimizar o ISE entre $\hat{p}$ e $\hat{q}$, basta escolher a instância γ que minimiza a parcela $\pi\hat{q}(\gamma) - \hat{p}(\gamma)$, visto que as outras parcelas da equação 3.13 são constantes. Este resultado representa um ganho computacional considerável, pois pode-se agora substituir o cálculo de $\hat{q}$, para cada instância, pela simples avaliação do valor de $\hat{q}$. Além disso, a atualização de $\hat{q}$ pode ser realizada através da equação 3.14, o que nos permite utilizar programação dinâmica.

$$\hat{q}^{(N_Q+1)} = \pi\hat{q}^{(N_Q)} + (1-\pi)G(x, \gamma, \sigma) \tag{3.14}$$

3.4 Adaptação do método para SR

Mesmo com a otimização apresentada na seção anterior, ainda há uma enorme barreira na adaptação deste método para atuar como uma estratégia de AA em SR. Como estamos interessados em selecionar os itens que, se avaliados, trarão o maior ganho de desempenho para o sistema como um todo, as instâncias sobre as quais o método de [50] se baseia devem ser itens do SR.

Contudo, para que esses itens possam ser considerados instâncias, se faz necessário ter uma representação dos mesmos em um espaço vetorial $\mathbb{R}^k$, para que, a partir deste, seja possível calcular as devidas FDPs. Todavia, como a única informação que se tem sobre usuários e itens está presente na matriz de preferências, obter a representação vetorial dos mesmos não é uma tarefa trivial.

Conforme comentado na seção 2.1.5, a Decomposição em Valores Singulares (SVD) é uma ferramenta de extrema utilidade em SR, pois permite extrair, a partir da matriz de preferências, uma representação de usuários e itens em k variáveis latentes, sendo o parâmetro k arbitrário. Ou seja, aplicando o SVD na matriz de preferências, conseguimos representar os itens em um espaço $\mathbb{R}^k$ e, com isto, é possível identificar as regiões de maior densidade neste espaço através do KDE. No entanto, aplicar a decomposição SVD diretamente na matriz de preferências pode resultar uma representação que não é fiel a realidade do problema.

É razoável assumir que uma avaliação presente na matriz de preferências retrata fielmente a preferência daquele usuário pelo item em questão. Entretanto, a ausência de uma avaliação levanta dúvidas quanto a preferência do usuário a respeito daquele item. A decomposição SVD, aplicada diretamente à matriz de preferências, assume que tais ausências correspondem a avaliações de valor zero, pois, como é um método numérico, necessita de uma matriz completamente preenchida. No entanto, esta

suposição é perigosa, pois subentende que a grande maioria das preferências são do tipo mais baixo possível e isto pode se refletir na representação vetorial de usuários e itens.

Algumas abordagens foram propostas a fim de se realizar um pré-processamento na matriz de preferências antes que ela seja decomposta. Por exemplo, sugeriu-se preencher os valores faltantes com a média das notas do respectivo usuário; do respectivo item; ou ainda a média global do sistema. Porém, em todos esses casos, estamos introduzindo informações que podem destoar significativamente das preferências reais. Se, ao contrário, aplicarmos a decomposição na matriz binária, ao invés da matriz de preferências, temos certeza que os valores ali presentes realmente correspondem com a realidade do sistema. Ou seja, se $b_{ij} = 1$, temos certeza que o usuário i adquiriu o item j. Caso contrário, temos certeza que i não adquiriu o item j. Em suma, trabalhar com a matriz binária pode acarretar em perda das informações sobre as preferências, mas, em contrapartida, nos fornece a segurança de que não estamos introduzindo nenhum viés na representação vetorial.

Ainda sim, há restrições quanto aos itens que podem ser selecionados. Quando tratamos de *Elicitação de Preferências para fins de Incentivo*, devemos ter em mente que há três tipos de itens: os que o usuário adquiriu e avaliou; os que o usuário adquiriu, porém não avaliou; e os que o usuário ainda não adquiriu. O método de [50] assume que todas as instâncias são passíveis de serem escolhidas, porém isto não é verdade em nosso contexto. Devemos restringir o método para selecionar somente os itens que o usuário adquiriu, mas não avaliou, pois não faz sentido selecionar itens de outros tipos neste contexto.

Desta forma, a *Estratégia Livre de Viés* é dada através do algoritmo 2. Primeiramente, aplicamos a decomposição SVD na matriz binária B e obtemos a matriz U contento a representação dos usuários em $\mathbb{R}^k$; a matriz V contento a representação dos itens em $\mathbb{R}^k$; e a matriz Σ contento os valores singulares de B (esta última não será utilizada, porém é um dos produtos da decomposição). Dado um usuário i, seja K o grupo abrangendo os itens que i adquiriu e avaliou e X o grupo abrangendo os itens que i adquiriu, porém não avaliou.

A estimativa $\hat{p}$ é dada através do KDE calculado sobre todos os itens e permanece inalterada durante todo o processo. Para cada usuário i, inicializamos a variável L, que armazenará os N itens a serem solicitados, e calculamos a estimativa $\hat{q}$ com base no conjunto K. A partir de então, para cada item $j \in X$, calculamos e armazenamos sua utilidade através da fórmula apresentada na equação 3.13. Escolhemos portanto o item com menor valor de utilidade; inserimos o mesmo em L; recalculamos $\hat{q}$ com base na esquação 3.14; e repetimos o processo a fim de encontrarmos o segundo melhor item e assim sucessivamente até L conter N itens. Uma vez completa, restanos solicitar ao usuário i que avalie os itens.

Algoritmo 2 A Estratégia Livre de Viés

1: $[U, \Sigma, V] \leftarrow SVD(B, k)$
2: $\hat{p} \leftarrow KDE(V, \sigma)$
3: **para cada** usuário $i \in U$ **faça**
4: $\quad L \leftarrow \emptyset$
5: $\quad \hat{q} \leftarrow KDE(K, \sigma)$
6: $\quad$ **enquanto** $|L| < N$ **faça**
7: $\quad\quad$ **para cada** item $j \in X$ **faça**
8: $\quad\quad\quad util(j) \leftarrow \pi\hat{q}(j) - \hat{p}(j)$
9: $\quad\quad$ **fim para**
10: $\quad\quad j' \leftarrow \text{argmin}_j\, util(j)$
11: $\quad\quad L \leftarrow L \cup \{j'\}$
12: $\quad\quad K \leftarrow K \cup \{j'\}$
13: $\quad\quad X \leftarrow X \setminus \{j'\}$
14: $\quad\quad \hat{q} \leftarrow \pi\hat{q} + (1 - \pi)G(x, j', \sigma)$
15: $\quad$ **fim enquanto**
16: $\quad i \xleftarrow{solicita} L$
17: **fim para**

3.5 Escolha de parâmetros

A *Estratégia Livre de Viés* depende de apenas dois parâmetros: o número de dimensões onde usuários e itens serão mapeados (k) e a matriz de covariância que será utilizada pelo KDE (σ), também conhecida como janela de Parzen [52].

Não há na literatura uma metodologia definida para a escolha de tais parâmetros. Portanto, escolhemos empiricamente, através diversos experimentos que seguiram a metodologia descrita no capítulo 4. Em outras palavras, várias versões da *Estratégia Livre de Viés* foram comparadas, cada uma possuindo um valor específico para k e σ. Ao final, optamos pela que apresentou melhor desempenho e acatamos seus parâmetros. Os testes comparando o desempenho desses parâmetros estão descritos na seção 4.6.

É interessante notar que poderíamos ter utilizado valores de σ distintos para as estimativas de $\hat{p}$ e $\hat{q}$. Fizemos a opção de utilizar o mesmo valor por motivos de simplicidade, afinal, usar σ's diferentes tornaria a escolha de parâmetros ainda mais complicada. Como o objetivo deste trabalho é apresentar a *Estratégia Livre de Viés* e provar que ela é uma boa opção para *Elicitação de Preferências para fins de Incentivo*, deixamos as questões referentes ao ajuste fino para trabalhos futuros.

Capítulo 4

Metodologia

Nossos experimentos seguem rigidamente o processo descrito em [13], uma vez que este é o trabalho mais completo sobre o tema em questão. Há, contudo, uma importante diferença que deve ser ressaltada. Os autores em [13] abordam a questão da *Elicitação de Preferências para fins de Arranque Frio*, com isso, não podem assumir que os usuários avaliarão todos os itens solicitados. É possível que eles desconheçam algum dos itens (ou até mesmo todos) solicitados pela estratégia de AA, tanto que o percentual de preferências adquiridas é uma das métricas utilizadas para se medir o desempenho das estratégias.

Em nosso trabalho, pelo contrário, já que estamos abordando o problema de *Elicitação de Preferências para fins de Incentivo*, podemos assumir que os usuários avaliarão todos os itens solicitados pela estratégia. Esta suposição se sustenta pelo fato de que os itens solicitados serão sempre itens que já foram adquiridos pelos usuários, além, claro, da existência de um incentivo individual para quando os usuários contribuem com suas preferências.

4.1 Nomenclatura

A fim de mantermos a relação com [13] ainda mais estreita, optamos por utilizar boa parte da nomenclatura que este usa ao descrever seus experimentos. Assim, o leitor que desejar ler os dois trabalhos não terá problema algum para perceber a correspondência entre ambos e poderá compará-los com extrema facilidade.

Como de costume, os problemas referentes a SR fazem uso de uma matriz esparsa como sendo a estrutura que representa as preferências dos usuários em relação aos itens. Nossa entrada é, portanto, uma matriz R, de n usuários por m itens ($n \times m$), onde cada posição preenchida r_{ij} representa a nota dada pelo usuário i ao item j. Conforme esperado, o número de posições preenchidas da matriz R é uma pequena fração do número total de posições $n \times m$, o que a classifica como esparsa.

R é dividida em duas matrizes disjuntas T e Tr, com as mesmas dimensões,

a primeira contendo 20% das avaliações de R e a segunda com os remanescentes 80%. A matriz T será usada para fins de teste, enquanto que Tr será novamente dividida em K e X, com 5% e 95% de Tr, respectivamente. Todas essas divisões são realizadas de maneira aleatória.

Para utilizarmos uma técnica de recomendação, ou modelo, é necessário que haja uma base de preferências ainda que limitada. Desta forma, K é inicializada com 5% da base original, representando as preferências conhecidas pelo sistema em um momento inicial. X, por sua vez, contém as preferências que serão adquiridas pelo sistema, i.e., os itens que os usuários compraram, porém não avaliaram explicitamente. As preferências contidas em X serão elicitadas através de uma estratégia S e, uma vez elicitadas, serão colocadas em K, incrementando o conhecimento do sistema.

Uma estratégia $S(i, N, K, C_i)$ é uma função que, dado um determinado usuário i, atribui uma pontuação para os itens dentro do conjunto C_i. Tal conjunto representa os itens que o usuário i adquiriu, porém não avaliou. Logo, são itens passíveis de serem solicitados pela estratégia, i.e., $C_i \subset X_i$, onde X_i representa a linha da matriz X correspondente ao usuário i. A pontuação dada por S é fruto de alguma heurística calculada com base nas preferências conhecidas até então pelo sistema (matriz K). O retorno de S é uma lista L contendo os N itens com maior pontuação. Caso o número de itens em C_i seja menor que N, retorna-se todos os itens de C_i.

4.2 Experimentos

Primeiramente, é necessário definir o número de itens (N) que serão elicitados a cada iteração e o número total de iterações (it) a serem executadas. Em nossos experimentos, utilizamos $N = 10$ e $it = 50$. O número de usuários (n) e itens de (m) varia conforme a base de dados que será utilizada.

Com a finalidade de garantirmos certa segurança estatística em nossos resultados, realizamos a *validação cruzada* com 5 partições. Ou seja, dividimos a matriz R em 5 partições disjuntas, todas com o mesmo número de preferências, chamadas de R_y, e repetimos o procedimento principal 5 vezes. A cada rodada, utilizou-se uma das partições, contento 20% de R, como sendo o conjunto teste T e as demais como sendo o conjunto de treinamento Tr.

O procedimento principal, por sua vez, inicia-se quando alocamos aleatoriamente 5% das preferências de Tr para K e o restante para X. Durante it iterações, empregamos uma estratégia S para buscar os melhores N itens, que, quando avaliados pelo usuário i, trarão o maior ganho de acurácia para o modelo. A preferência de i por tais itens é então elicitada, o que, na prática, significa que as avaliações de i referentes aos itens em L são removidas de X e acrescentadas em K. Feito isto

para todos os usuários, treinamos o modelo θ com a matriz K, agora com novas preferências, e o utilizamos para prever as preferências contidas em T.

O Erro Médio Absoluto das previsões, em inglês, *Mean Absolute Error* (MAE), descrito na seção 4.3, é calculado e armazenado com sendo o erro referente à iteração e à rodada. Ao fim de todas as rodadas, calculamos a média e o desvio padrão dos erros a cada iteração para que, desta maneira, resultados muito discrepantes sejam amenizados.

Um esquema completo de nosso experimento pode ser visto no algoritmo 3. Nele, a função $RAND$ é responsável por alocar 5% das preferências de Tr para K; a função Γ é responsável calcular o MAE do modelo θ, treinado com a matrix K, no conjunto de teste T; as funções $MEAN$ e STD são responsáveis por calcular a média e o desvio padrão, respectivamente; e por $ERROR[:, x]$, denotamos toda a coluna x da matriz $ERROR$.

Como estamos interessados em comparar o desempenho de diversas estratégias de AA, o procedimento descrito no algoritmo 3 foi aplicado para um total de 17 estratégias, incluindo a *Estratégia Livre de Viés*. Uma descrição de cada uma delas pode ser encontrada na seção 4.5 e os resultados de nossos experimentos serão apresentados e discutidos no capítulo 5.

Algoritmo 3 Procedimento para se avaliar uma estratégia S

$N \leftarrow 10$ ▷ Número de itens retornados
$it \leftarrow 50$ ▷ Número de iterações
para $y \leftarrow 1$ **até** 5 **faça** ▷ Rodadas da validação cruzada
    $T \leftarrow R_y$
    $Tr \leftarrow R \setminus R_y$
    $K \leftarrow RAND(Tr, 5\%)$
    $X \leftarrow Tr \setminus K$
    para $t \leftarrow 1$ **até** it **faça**
        para $i \leftarrow 1$ **até** n **faça**
            $C_i \leftarrow X_i > 0$ ▷ Itens cujas preferências podem ser elicitadas
            $L \leftarrow S(i, N, K, C_i)$
            $K \leftarrow K + L$
            $X \leftarrow X \setminus L$
        fim para
        $ERROR[y, t] \leftarrow \Gamma(\theta(K), T)$ ▷ Avaliação do desempenho de S
    fim para
fim para
para $x \leftarrow 1$ **até** it **faça** ▷ Estatísticas da validação cruzada
    $media[x] = MEAN(ERROR[:, x])$
    $desvio[x] = STD(ERROR[:, x])$
fim para

4.3 Modelo e Métrica de Avaliação

Conforme visto no capítulo 2, diversos algoritmos foram propostos para a tarefa de prever as preferência dos usuários. Em particular, mencionamos que os modelos baseados na decomposição SVD se sobressaíram aos demais, não apenas por causa dos bons resultados, mas, sobretudo, devido a capacidade do SVD gerar um espaço vetorial ajustável onde usuários e itens podem ser mapeados. Ou seja, tais modelos além de fornecerem as previsões, possibilitam que os projetistas do SR compreendam melhor os usuários e itens conforme suas disposições dentro do espaço gerado.

Para calcular as previsões, optamos pelo modelo baseado em SVD conhecido como *Regularized SVD*, descrito na seção 2.1.5. Conforme aconselhado por [30], utilizamos $\rho = 0,001$, $\lambda = 0,02$, $\varepsilon = 0,1$ e $k = 96$. Na prática, poderíamos ter utilizado qualquer modelo capaz de fazer previsões paras as tuplas (usuário, item) presentes em T, uma vez que não estamos interessados em obter o menor erro possível, e sim avaliar como o decaimento do erro se comporta dependendo da estratégia escolhida.

Em [13], os autores utilizam o modelo descrito em [54]. Ambos são baseados na decomposição SVD, a única diferença é que o modelo de [54] faz uso da média global (μ), do viés do usuário ($bias_i$) e do viés do item ($bias_j$) para o cálculo de previsão. Neste sentido, ele é mais parecido com o *Improved Regularized SVD*, descrito na seção 2.1.5, do que com o *Regularized SVD*. Apesar de apresentar resultados levemente superiores, nossa opção pelo *Regularized SVD* se justifica pela simplicidade de implementação, já que não é necessário atualizar os parâmetros $bias_i$ e $bias_j$.

Todavia, para afirmar que um modelo apresenta desempenho bom ou ruim, é preciso que haja uma métrica capaz de avaliá-lo. Nos problemas relacionados a SR, duas métricas se popularizaram dentro da literatura, são elas o Erro Médio Absoluto, em inglês, *Mean Absolute Error* (MAE); e a Raiz do Erro Médio Quadrático, em inglês, *Root Mean Square Error* (RMSE) [4]. Vale ressaltar que ambas as técnicas visam avaliar o sistema como um todo, tratando todas as previsões como possuindo igual importância. Há casos onde deseja-se avaliar a capacidade do modelo fazer previsões para um grupo específico de usuários ou itens, ou então, deseja-se avaliar a capacidade do modelo prever determinados valores de preferência. Nesses casos não é recomendado fazer uso do MAE nem do RMSE, no entanto, como em nossos experimentos estamos interessados no desempenho global do sistema, tanto MAE quanto RMSE são métricas apropriadas.

Por outro lado, utilizar ambas as técnicas seria redundante, visto que, em essência, elas traduzem a mesma informação, que é a diferença média entre a previsão dada pelo modelo e a verdadeira preferência contida no conjunto de teste. Portanto, optamos por medir nossos resultados apenas com o MAE a fim de poder-

mos realizar comparações mais diretas com [13], que também acatou pela mesma métrica.

Quanto ao cálculo propriamente dito, seja $\hat{r}_{ij}$ a previsão dada pelo modelo para a preferência do usuário i pelo item j, ambos pertencentes ao conjunto de teste T; e seja r_{ij} a verdadeira preferência deste usuário pelo item. Assim, o valor de MAE é dado segundo a equação 4.1, enquanto que o valor de RSME é dado segundo a equação 4.2.

$$MAE = \sum_{(i,j) \in T} |r_{ij} - \hat{r}_{ij}| \tag{4.1}$$

$$RMSE = \sqrt{\sum_{(i,j) \in T} (r_{ij} - \hat{r}_{ij})^2} \tag{4.2}$$

4.4 Base de Dados

A primeira base escolhida foi a *MovieLens 100k*, ou simplesmente *MovieLens*, disponibilizada pelo grupo *GroupLens* [55]. Esta base contém 100 mil preferências, oriundas de 943 usuários e 1682 itens, que, neste caso, representam filmes. Logo, ela contém apenas 6,3% de todas as possíveis preferências, sendo que 6.110 possuem valor igual a 1; 11.370 igual a 2; 27.145 igual a 3; 34.174 igual a 4; e 21.201 igual a 5.

A segunda base escolhida foi a do desafio *Netflix* [6]. Completa, esta base possui possui mais de 100 milhões de preferências, o que torna inviável a execução de nossos experimentos. Realizamos portanto um "corte", coletando as 100 mil primeiras preferências fornecidas. Este corte é oriundo de 1.499 usuários e 2.382 itens, também filmes, sendo que o número de preferências corresponde a apenas 2,8% do total possível. Dessas preferências, 8.149 possuem valor igual 1; 14.253 igual a 2; 34.503 igual 3; 27.076 igual a 4; e 16.019 igual 5.

Há várias bases de recomendação, abordando inúmeros tipos de itens, porém, as bases apresentadas nesta seção são muito bem conhecidas dentro da literatura, havendo diversos trabalhos que fazem uso das mesmas. Além disso, ambas são utilizadas em [13], inclusive, o corte da base *Netflix* foi proposto por este trabalho.

4.5 Estratégias comparadas

Definir quais estratégias serão utilizadas e avaliadas em nossos experimentos é, por si só, uma tarefa árdua, visto não há um consenso claro dentro da literatura sobre quais estratégias servem como referência para o problema abordado. Aliás, o problema em si foi pouco explorado dentro da literatura de SR.

Mesmo quando tratamos do problema análogo, a *Elicitação de Preferências para fins de Arranque Frio*, ainda não existem referências bem consolidadas, com exceção da estratégia aleatória. Todavia, há estratégias que se popularizaram pelo fato de terem sido usadas em trabalhos importantes da área. Decidimos portanto escolher 16 estratégias para comparação com a *Estratégia Livre de Viés*, algumas sendo as mais "populares" da literatura e outras sendo combinações dessas.

4.5.1 *random*

A estratégia *random* é a estratégia mais básica possível, ou seja, *random* pode ser interpretada como sendo a ausência de estratégia. Assim, para uma estratégia S ser considerada minimamente eficaz, ela deve ter desempenho melhor que *random*. Em nosso caso, utilizar a estratégia *random* consiste em selecionar, de forma aleatória, N itens dentre os que estão presentes no conjunto C_i.

4.5.2 *popularity*

Esta estratégia se propõe a ordenar os itens de acordo com o seu valor de popularidade. Para calcular a popularidade de um item, verifica-se quantos usuários já avaliaram este item. Em termos formais, a função que determina o valor de popularidade do item j, $pop(j)$, é dada conforme a equação 4.3. Calculamos então o valor de *pop* para todos os itens em C_i e solicitamos que o usuário avalie os N itens com maior popularidade.

$$pop(j) = \sum_{i \in K, r_{ij} > 0} 1 \tag{4.3}$$

4.5.3 *entropy*

A estratégia *entropy* visa ordenar os itens com base no seu valor de entropia. Proposta pela primeira vez em [9], a estratégia busca associar a entropia de um determinado item ao consenso médio que os usuários possuem sobre este item. Entretanto, neste mesmo trabalho, os autores afirmam que esta relação só pode ser traçada caso o item possua um valor significativo de popularidade. A função $ent(j)$ retorna o valor de entropia para o item j, onde r representa o valor das possíveis avaliações em K, e $p_j(r)$ a probabilidade do item j receber a nota r.

$$ent(j) = -\sum_{r \in K, r > 0} p_j(r) \log p_j(r) \tag{4.4}$$

4.5.4 *entropy0*

As deficiências de se utilizar apenas a entropia pura são extensamente abordadas em [10]. Em linhas gerais, não se pode tirar conclusões sobre a entropia de itens com baixa popularidade. Para contornar tal problema, [10] propõe incorporar a "impopularidade" do item no cálculo da entropia, considerando as avaliações com valor 0. Contudo, como o número de avaliações 0 é muito superior as outras, atribui-se um peso maior para as avaliações positivas. Esta estratégia ficou conhecida como *entropy0* e a função que exprime sua fórmula é dada através da equação 4.5, onde w_r representa o peso da nota r. Em nossos experimentos utilizamos $w_0 = 0,5$ e $w_1 = w_2 = w_3 = w_4 = w_5 = 1$.

$$ent0(j) = \frac{-\sum_{r \in K} w_r p_j(r) \log p_j(r)}{\sum_{r \in K} w_r} \tag{4.5}$$

4.5.5 *log(pop)*ent*

Uma maneira alternativa de se amenizar as deficiências da entropia pura é combinar seu valor com a popularidade do item. Assim, itens com alta entropia, mas que possuem baixa popularidade, seriam penalizados e itens de alta entropia, com também alta popularidade, seriam destacados. Esta é solução proposta por [9], porém os autores verificaram que o valor bruto da popularidade iria prevalecer sobre o valor da entropia, logo, ao invés disso, propuseram o logaritmo da popularidade. A equação 4.6 mostra o cálculo de *log(pop)*ent*.

$$logpopent(j) = \log pop(j) \times ent(j) \tag{4.6}$$

4.5.6 *log(pop)*ent0*

Apesar de nenhum trabalho apresentar esta combinação, optamos por combinar a popularidade com *entropy0* de maneira análoga à combinação realizada em *log(pop)*ent*. Desta forma, criou-se a estratégia *log(pop)*ent0*, cuja fórmula é dada pela equação 4.7.

$$logpopent0(j) = \log pop(j) \times ent0(j) \tag{4.7}$$

4.5.7 *helf*

Outra maneira de se combinar o logaritmo da popularidade e a entropia pura é apresentado em [10]. Nele, ao invés da simples multiplicação, optou-se por calcular a média harmônica das funções $\log pop$ e ent, uma medida muito conhecida na área de RI [24]. A grosso modo, tanto *helf* quanto *log(pop)*ent* são implementações

diferentes de uma mesma ideia, que é tentar amenizar as desvantagens individuais de cada estratégia através da combinação de ambas. A equação 4.8 apresenta a fórmula para o cálculo de *helf*, onde n é o número total de usuários e r_{max} é o maior valor de preferência que um usuário pode dar.

$$helf(j) = \frac{2 \times (\log pop(j)/\log n) \times (ent(j)/\log r_{max})}{(\log pop(j)/\log n) + (ent(j)/\log r_{max})} \tag{4.8}$$

4.5.8 *helf0*

Semelhantemente ao caso de *log(pop)*ent0*, criamos uma estratégia que utiliza a média harmônica para combinar os valores $\log pop$ e $ent0$, a qual chamamos de *helf0*. O seu cálculo se dá conforme a equação 4.9.

$$helf0(j) = \frac{2 \times (\log pop(j)/\log n) \times (ent0(j)/\log r_{max})}{(\log pop(j)/\log n) + (ent0(j)/\log r_{max})} \tag{4.9}$$

4.5.9 *igcn*

A estratégia *igcn*, cujo nome é a sigla para *Information Gain through Clustered Neighbors*, foi proposta em [10] e se baseia no conceito do Ganho de Informação (GI). Este conceito procura mensurar a "quantidade" de informação que uma determinada variável agrega à resolução de um problema, tendo se tornado famoso por causa das *Árvores de Decisão* [56]. Na prática, o *igcn* procura criar uma Árvore de Decisão que classifica os usuários em *clusters*, de acordo com a nota que cada um deu a cada item.

Primeiramente, aplica-se a decomposição SVD na matriz K e, com isso, obtém-se um mapeamento dos usuários em um espaço vetorial de ω dimensões. A partir da disposição dos usuários nesse espaço, utilizamos um algoritmo de clusterização hierárquica para agrupar os usuários em φ *clusters*. Assim calculamos os itens que, quando avaliados, melhor distribuem os usuários em *clusters*, ou seja, que trazem o maior decréscimo na entropia referente a distribuição dos usuários em *clusters*. Tais itens são os que possuem maior GI.

Todavia, calcular o GI dos itens não uma tarefa trivial, tendo em vista que deve-se levar em consideração as avaliações de valor zero. Como essas superabundam em nossas bases, é preciso contrabalançar os efeitos de tais avaliações através de um peso artificial. A equação 4.10 apresenta como se dá o cálculo do *igcn*, onde $|U|$ é a quantidade total de usuários; $|U_j^r|$ é a quantidade de usuários que avaliaram o item j com nota r; w_r é o peso referente a nota r; $H(U)$ é a entropia da distribuição de todos os usuários em *clusters*; e $H(U_j^r)$ é a entropia da distribuição dos usuários que avaliaram o item j com nota r em *clusters*.

Em nossos experimento, utilizamos $\omega = 100$ e $\varphi = 300$. Para facilitar o cálculo de $H(U)$ e $H(U_j^r)$, consideramos apenas a distribuição em *clusters* dos 150 usuários mais próximos do usuário i [10].

$$igcn(j) = H(U) - \sum_{r \in K} \frac{w_r \frac{|U_j^r|}{|U|}}{E(j)} H(U_j^r) \tag{4.10}$$

$$E(j) = \sum_{r \in K} w_r \frac{|U_j^r|}{|U|} \tag{4.11}$$

4.5.10 *variance*

Esta estratégia busca ordenar os itens pela variância das avaliações recebidas. Em essência, ela está intimamente ligada a *entropy*, uma vez que ambas procuram mensurar o consenso dos usuários sobre os itens através de uma medida de dispersão. Inclusive, a variância pura padece do mesmo problema que a entropia pura, i.e., quando o item possui poucas avaliações, tirar conclusões a respeito do valor de sua variância pode ser enganoso e, por esta razão, poucos trabalhos a utilizam. Apesar de [13] afirmar que utilizou *variance* em seus experimentos, este menciona que seus resultados não se destacaram e, por isto, não foram apresentados. Calculamos a variância de um item j através da equação 4.12, onde n é o número de avaliações que o item j recebeu e $\overline{r_j}$ é a média das mesmas.

$$var(j) = \frac{1}{(n-1)} \times \sum_{i \in K, r_{ij} > 0} (r_{ij} - \overline{r_j})^2 \tag{4.12}$$

4.5.11 *sqrt(pop)*var*

Com o intuito de amenizar as desvantagens da variância pura, [46] propôs a combinar a variância com a popularidade, analogamente ao que foi proposto em *log(pop)*ent*. Entretanto, ao invés de utilizar a escala logarítmica para os valores de popularidade, os autores optaram pelo uso da raiz quadrada desses valores. A equação 4.13 apresenta a fórmula para o cálculo de *sqrt(pop)*var*.

$$sqrtpopvar(j) = \sqrt{pop(j)} \times var(j) \tag{4.13}$$

4.5.12 *log(pop)*var*

Apesar de não ter sido encontrado nenhum trabalho fazendo uso desta estratégia, julgamos cabível testar também o logaritmo da popularidade para se amenizar os efeitos da variância, conforme foi empreendido em *log(pop)*ent*. A fórmula para

se calcular esta estratégia, nomeada de *log(pop)*var*, pode ser descrita segundo a equação 4.14.

$$logpopvar(j) = \log pop(j) \times var(j) \tag{4.14}$$

4.5.13 *bin pred*

A estratégia *bin pred* (abreviação para *Binary Prediction*) proposta em [13], procura prever quais itens o usuário irá consumir e, a partir de então, assume que será melhor para o sistema adquirir as avaliações a respeito desses itens. A previsão $\hat{b}_{ij}$ indica se o usuário i irá consumir ou não o item j e é calculada a partir da matriz binária B. Esta matriz, por sua vez, é gerada a partir da matriz K, de tal forma que as avaliações positivas de K possuem valor 1 na matriz B e as avaliações com valor zero são mantidas. O modelo descrito na seção 4.3 é treinado com a matriz B e nos retorna previsões $\hat{b}_{ij}$ para todos os itens. Por fim, a estratégia solicita os N itens em C_i com maior previsão.

4.5.14 *high pred*

Na mesma linha de *bin pred*, *high pred* (abreviação para *Highest Predicted*) realiza o treinamento do modelo com a matriz K e calcula as previsões $\hat{r}_{ij}$ para as notas faltantes. Os N itens em C_i com as maiores previsões são solicitados ao usuário i. Esta estratégia, também proposta em [13], assume que é melhor para o SR adquirir as avaliações altas do que as baixas.

4.5.15 *low pred*

Ao contrário de *high pred*, *low pred* (abreviação para *Lowest Predicted*) assume que é melhor para o SR solicitar as avaliações baixas em detrimento das altas [13]. O funcionamento de *low pred* se assemelha muito ao de *high pred*, o modelo é treinado com a matriz K e as previsões para as notas faltantes $\hat{r}_{ij}$ são calculadas. Feito isso, calcula-se uma nova previsão dada por $\hat{\alpha}_{ij} = r_{max} - \hat{r}_{ij}$, onde r_{max} é o maior valor de preferência que um usuário pode dar. Solicita-se então os N itens de C_i com maior previsão $\hat{\alpha}_{ij}$.

4.5.16 *high-low pred*

Por último, temos a estratégia *high-low pred*, proposta em [13], que nada mais é do que uma tentativa de se combinar *high pred* e *low pred*. Analogamente a essas, utiliza-se o modelo, treinado com a matriz K, a fim de se calcular as previsões $\hat{r}_{ij}$ para as notas faltantes. A partir de então, calcula-se uma nova previsão dada por

$\hat{\beta}_{ij} = |\frac{r_{max}+r_{min}}{2} - \hat{r}_{ij}|$, que indica o quão distante a previsão está do valor médio (r_{max} e r_{min} são os valores máximo e mínimo de preferência, respectivamente). Escolhe-se os N itens pertencentes a C_i com maior valor de $\hat{\beta}_{ij}$.

4.6 Parametrização

Conforme comentado na seção 3.5 a escolha dos parâmetros da *Estratégia Livre de Viés* (o número de dimensões k e a largura da janela σ) foi realizada de maneira empírica, analisando o comportamento da estratégia para diversos valores diferentes. Definido um valor de k, avaliamos qual o valor de σ que resulta no melhor desempenho da estratégia, lembrando que o eixo vertical representa o MAE enquanto que o eixo horizontal representa o número de iterações ($it = 15$).

Apesar das comparações entre as estratégias terem sido realizadas em duas bases diferentes, os testes para escolha de parâmetros foram realizados apenas na base *MovieLens*. Além disso, com a finalidade de simplificar a notação, nos referimos aos valores de σ como sendo escalares. No entanto, σ é uma matriz quadrada $k \times k$ cujas posições da diagonal principal possuem o valor mencionado. Por exemplo, se $k = 2$, quando mencionamos $\sigma = 0,5$ isto significa que $\sigma = \left(\begin{smallmatrix} 0,5 & 0 \\ 0 & 0,5 \end{smallmatrix}\right)$.

A figura 4.1 apresenta o desempenho da *Estratégia Livre de Viés* com diversos valores de σ para $k = 2$. Como é possível observar, há uma pequena vantagem quando utilizamos $\sigma = 0,00005$ nas primeiras 8 iterações. Da mesma forma, na figura 4.2, onde $k = 3$, vemos que $\sigma = 0,00005$ ainda resulta no melhor desempenho, porém a vantagem em relação as outras é bem mais sutil.

Na figura 4.3, onde $k = 5$, novamente temos desempenhos muito próximos, entretanto notamos que, nas primeiras 4 iterações, $\sigma = 10$ é melhor, enquanto que, a partir da 8ª iteração, $\sigma = 0,00005$ é melhor. Já na figura 4.4, onde $k = 10$, $\sigma = 10$ é claramente melhor. Por fim, quando tomamos $k = 50$, na figura 4.5, temos novamente o melhor resultado com $\sigma = 0,00005$.

Dado um valor para k, podemos concluir que as variações de desempenho percebidas, ao se mudar o parâmetro σ, são mínimas. Contudo, a fim de visualizarmos as variações de desempenho resultantes das mudanças em k, apresentamos na figura 4.6 o desempenho da *Estratégia Livre de Viés*, para cada valor de k testado, com seu respectivo melhor valor de σ. Como é possível observar, o melhor resultado ocorre com $k = 2$ e $\sigma = 0,00005$ e foram esses os parâmetros escolhidos para a *Estratégia Livre de Viés* que será comparada com as demais no capitulo 5.

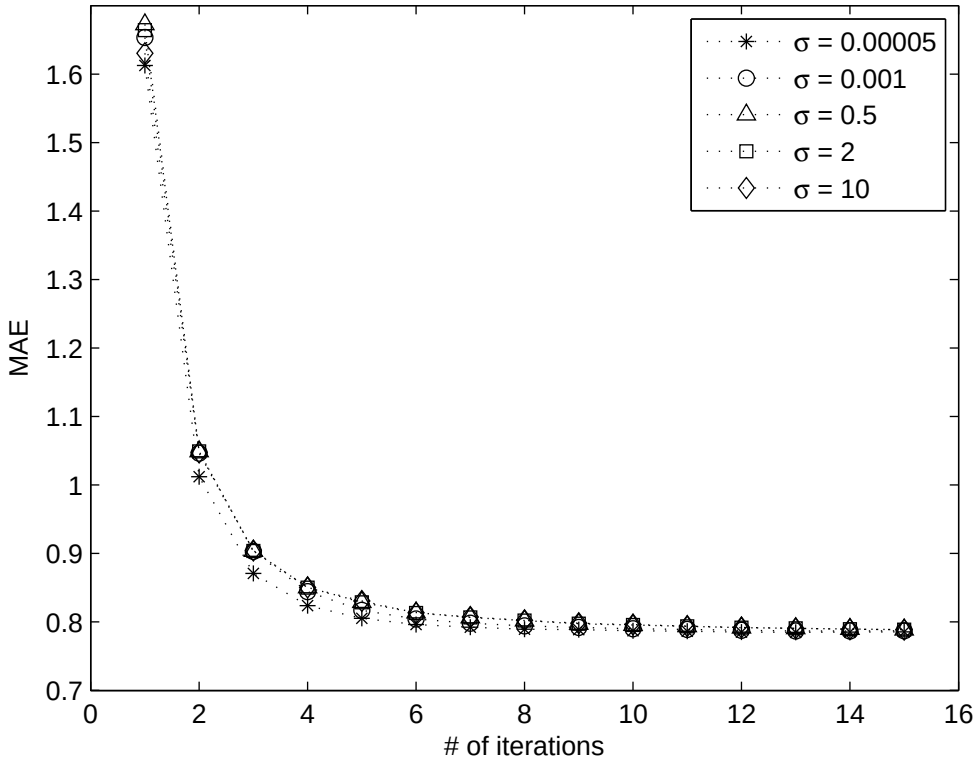

Figura 4.1: Comparação de diferentes σ's utilizando $k = 2$

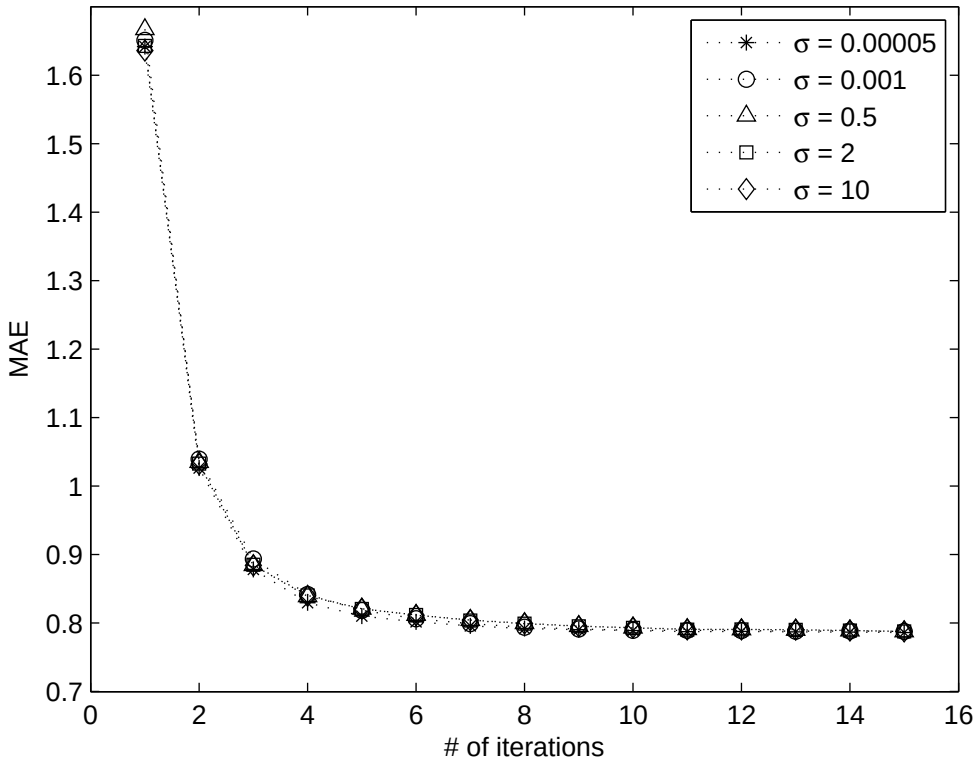

Figura 4.2: Comparação de diferentes σ's utilizando $k = 3$

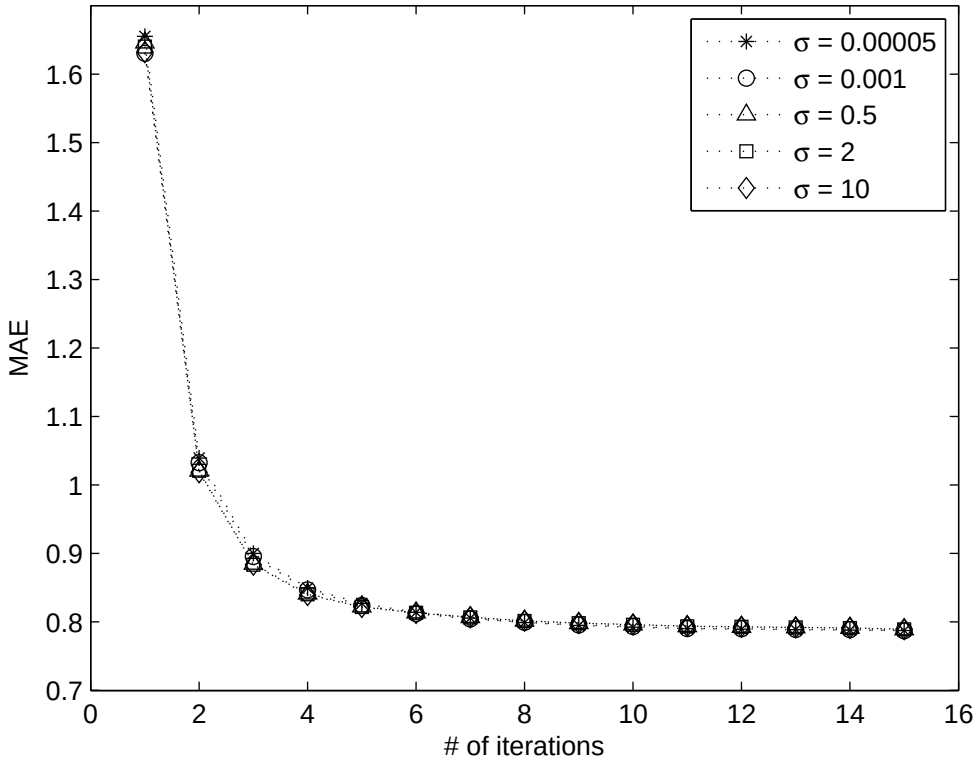

Figura 4.3: Comparação de diferentes σ's utilizando $k = 5$

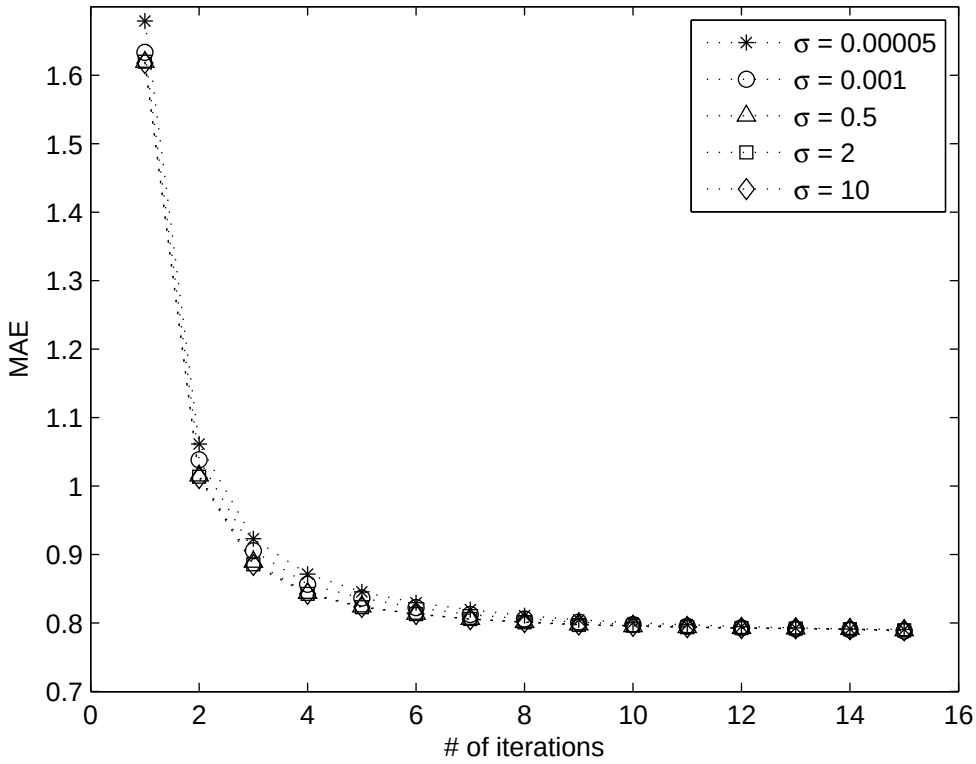

Figura 4.4: Comparação de diferentes σ's utilizando $k = 10$

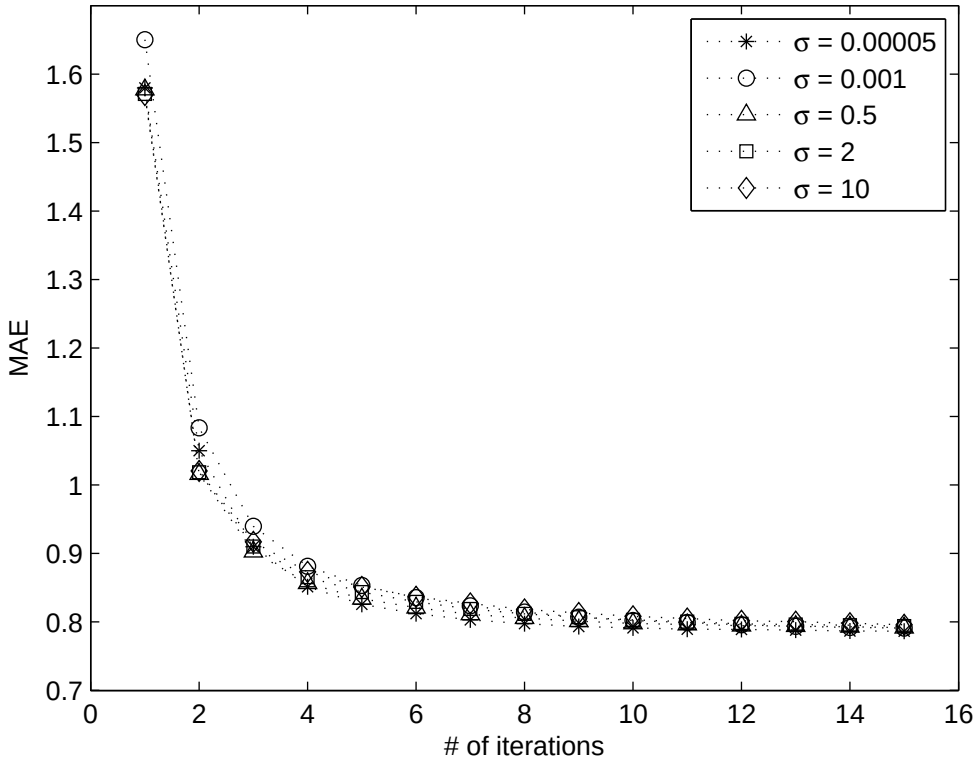

Figura 4.5: Comparação de diferentes σ's utilizando $k = 50$

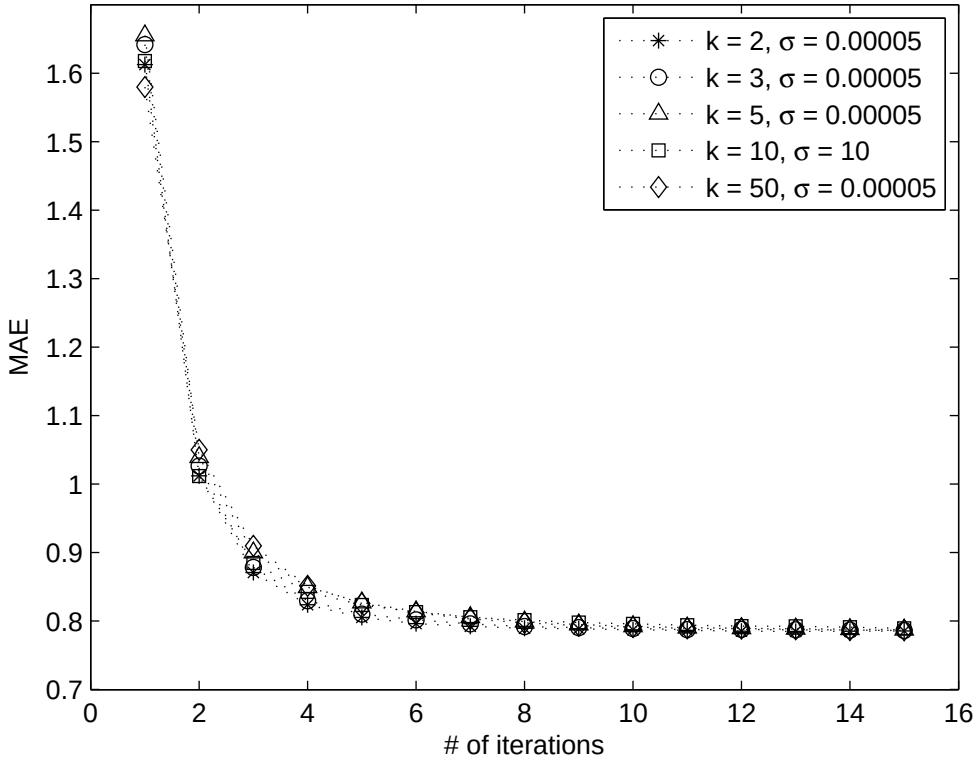

Figura 4.6: Comparação de diferentes k's com seus melhores σ's

Capítulo 5

Resultados

Antes de apresentarmos propriamente os resultados, faz-se necessário esclarecer o leitor sobre a maneira como os mesmos foram organizados. Em nossos experimentos, assim com em [13], um grande número de estratégias, 17 no total, foram comparadas em termos de acurácia global do modelo (MAE). Visualizar os resultados de todas as estratégias ao mesmo tempo, ou seja, em um mesmo gráfico, acaba sendo improdutivo devido a grande quantidade de informação apresentada de uma só vez. Logo, optamos por dividir as estratégia em famílias e subdividir estas famílias em grupos, a fim de podermos analisar com melhor exatidão o desempenho de cada estratégia. Aliás, uma das críticas passíveis de ser feita a [13] é o fato de que, ao mostrar todas as estratégias ao mesmo tempo, os gráficos acabam por ficar confusos, tolhendo o poder de análise.

5.1 Organização

Em linhas gerais, as estratégias foram divididas em 3 famílias: *Entropia*, *Variância* e *Modelo*. Conforme o nome indica, a família *Entropia* é constituída das estratégias que fazem uso da entropia das preferências. Ao todo, 7 estratégias pertencem à esta família, o que ainda é muita informação para ser analisada de maneira detalhada. Decidimos então por subdividir esta família em 4 grupos: *Entropia Pura*; *Logaritmo da Popularidade e Entropia*; *Média Harmônica da Popularidade e Entropia*; e *Ganho de Informação*.

As estratégia foram divididas em grupos de acordo com suas características, assim, ao primeiro grupo pertencem as estratégias *entropy* e *entropy0*; ao segundo grupo pertencem as estratégias *log(pop)*ent* e *log(pop)*ent0*; ao terceiro grupo pertencem as estratégias *helf* e *helf0*; e, por fim, *igcn* constitui sozinha o quarto grupo.

A família *Variância*, por sua vez, é constituída pelas estratégias que se baseiam na variância das preferências, i.e., *variance*, *log(pop)*var* e *sqrt(pop)*var*. Por serem apenas 3 estratégias, não há necessidade de subdividi-las em grupos, posto que a

apresentação das 3 em um mesmo gráfico não gera grandes dificuldades de análise.

Por fim, a família *Modelo* agrega as estratégias que se baseiam no modelo, ou seja, no próprio algoritmo de recomendação. Decidimos por subdividi-las em dois grupos para facilitar a análise dos resultados. Estes foram formados de acordo com o desempenho das mesmas, assim, o primeiro, *Pior que random*, é formado pelas estratégias *bin pred* e *high pred*. Já o segundo, *Melhor que random*, é formado por *low pred* e *high low pred*.

Cada grupo será visualizado em conjunto com as estratégias de referência *random* e *popularity*. A avaliação de cada estratégia pode ser inferida de acordo com seu comportamento em relação a estas.

5.2 Avaliação

Estamos avaliando cada estratégia no que diz respeito ao erro global do SR, i.e., o MAE. Obviamente quanto menor o valor do MAE, melhor é a estratégia. Contudo, estratégias de AA não são avaliadas em relação a um valor pontual de erro, e sim em relação ao decaimento dos erros pontuais, também conhecido como *comportamento* da estratégia. Portanto, quanto mais acentuado, ou hiperbólico, for o decaimento do MAE, melhor é a estratégia.

Há duas estratégias clássicas que servirão de parâmetro para as outras, são elas *random* e *popularity*. A primeira, como veremos, possui um decaimento abrupto e bem acentuado, enquanto que a segunda possui um decaimento estável e pouco acentuado. Existem basicamente 3 áreas de desempenho para uma estratégia, ela pode ser pior que *popularity*, i.e., seu comportamento se encontra acima da curva dada por esta; entre *popularity* e *random*, i.e., seu comportamento se encontra entre essas duas curvas; e melhor que *random*, o que significa que seu comportamento se encontra abaixo desta curva, o melhor dos casos.

Ao fim, todas as estratégias atingem o mesmo valor de erro, visto que irão inserir todas as avaliações de X em K e, portanto, não importa qual estratégia usarmos, sempre atingiremos o limite inferior do nosso modelo. Em outras palavras, todas as possíveis estratégias irão convergir para o mesmo limiar definido apenas pelo modelo empregado. A este fenômeno, damos o nome de *convergência* dos comportamentos.

5.3 Convergência

Em termos práticos, a única diferença que existe entre os experimentos voltados para incentivos e os experimentos voltados para AF é o fato que, no caso de incentivos, o usuário sempre responde com as avaliações solicitadas. Isto não pode ser assumido como verdade no caso do AF, pois é possível que o usuário desconheça

alguns dos itens solicitados, ou até mesmo todos. Quando tratamos de incentivos, é muito mais plausível assumir que o usuário consiguirá avaliar todos os itens solicitados, posto que são itens que ele já comprou e que haverá uma forma de "recompensa" se ele os avaliar.

Com isso, podemos esperar uma convergência mais rápida do MAE para o limiar do modelo. Vemos que isto se verifica em nossos resultados, uma vez que as estratégias parecem convergir a partir da 30ª iteração, em contrate com [13], onde a convergência se dá apenas por volta da 80ª iteração.

5.4 Diferenças em relação a outros trabalhos

Ainda em relação a [13], apesar de nossos experimentos serem semelhantes, esta suposição de que os usuários sempre colaboram avaliando todos os itens soliciados afeta significativamente nossos resultados em comparação com os de [13]. As diferenças não aparecem apenas na questão da convergência, mas no comportamento das estratégias. Por exemplo, [13] faz distinção entre estratégias *monótonas* e *não-monótonas*, enquanto que, em nossos resultados, nenhuma estratégia apresenta comportamento não-monótono.

Assim, embora os experimentos sejam muito parecidos, esta simples diferença torna os resultados difíceis de se comparar diretamente. Claro que o paralelo com [13] foi realizado e, em alguns casos, houve forte correspondência entre os resultados. Porém, é importante o leitor ter em mente que as estratégias podem apresentar um comportamento diferente devido a estrutura do experimento em si, que é semelhante, mas não idêntica.

Outro trabalho que também nos serviu de inspiração e de balizamento foi [10]. Várias das estratégias utilizadas em nossos experimentos foram extraídas de [10] que, além de serem aplicadas diretamente, serviram de base para a criação de estratégias compostas. Todavia, há enormes diferenças entre nosso experimento e o descrito em [10] que não derivam apenas do fato de [10], assim como [13], ser voltado para AF.

Primeiramente, [10] procura elicitar as notas apenas dos usuários que avaliaram 80 itens ou mais, ignorando o resto. Além disso, o processo de elicitação é realizado em apenas uma iteração, buscando-se elicitar o maior número de itens possível de cada usuário ($N \geq 75$). Por último, a avaliação das estratégias é realizada considerando somente as notas fornecidas pelo mesmo usuário na base de teste. Ou seja, [10] compara o desempenho das estratégias com base nos usuários individuais e não no SR por completo.

Todas essas diferenças tornam a comparação de nossos resultados com os de [10] muito difícil. Contudo, há casos onde os comportamentos das estratégias são semelhantes o que indica que, apesar das diferenças entre os experimentos, há uma

forte sintonia entre os trabalhos.

5.5 Entropia

5.5.1 Entropia Pura

As figuras 5.1 e 5.2 apresentam os comportamentos das estratégias *entropy* e *entropy0* nas bases *MovieLens* e *Netflix*, respectivamente. Em linhas gerais, podemos dizer que o comportamento de tais estratégias foi praticamente idêntico ao comportamento de *popularity*, uma vez que as curvas de ambas parecem se sobrepor a esta última. No entanto, nas primeiras iterações, sobretudo na figura 5.1, *entropy* parece ter um desempenho pior que *popularity*, estando ligeiramente acima desta.

Isto pode ser explicado pela própria diferença entre as estratégias. Enquanto *entropy* busca solicitar os itens olhando apenas o valor de entropia dos mesmos, *entropy0* procura levar em consideração também a impopularidade dos itens, adicionando no cálculo da entropia as preferências com valor zero. Pode-se dizer então que *entropy0* leva em consideração a popularidade do item na medida em que considera a impopularidade do mesmo.

A deficiência da entropia pura foi comentada em detalhes em [10]. Segundo este trabalho, calcular a entropia sem levar em consideração a popularidade do item pode ser enganoso, visto que itens que receberam poucas preferências podem possuir valores de entropia muito altos.

Infelizmente [13] não apresenta resultados para estas estratégias. Todavia, devido ao desempenho ruim de ambas, concluímos que a heurística que as norteia resulta na construção de conjuntos de treinamento enviesados. Na prática, ao tentar reduzir a incerteza através da entropia, estamos adicionando no conjunto de treinamento apenas os itens mais controversos (os mais difíceis de se prever) e deixando de lado aqueles onde há maior consenso entre os usuários (os mais fáceis de se prever). A controvérsia dos itens acaba por se refletir na acurácia do modelo, que fica mais duvidoso e menos preciso ao calcular previsões.

5.5.2 Logaritmo da Popularidade e Entropia

Do mesmo modo, as figuras 5.3 e 5.4 correspondem ao emprego de *log(pop)*ent* e *log(pop)*ent0* nas bases *MovieLens* e *Netflix*, respectivamente. Como é possível observar, ambas possuem comportamento muito similar a *popularity*, entretanto havia indícios para suspeitarmos disso, uma vez que conhecemos os resultados do grupo *Entropia Pura*.

Primeiramente, ambas estratégias estão baseadas no valor de popularidade, que representa o número de avaliações que o filme recebeu. Desta forma, tal valor, ainda

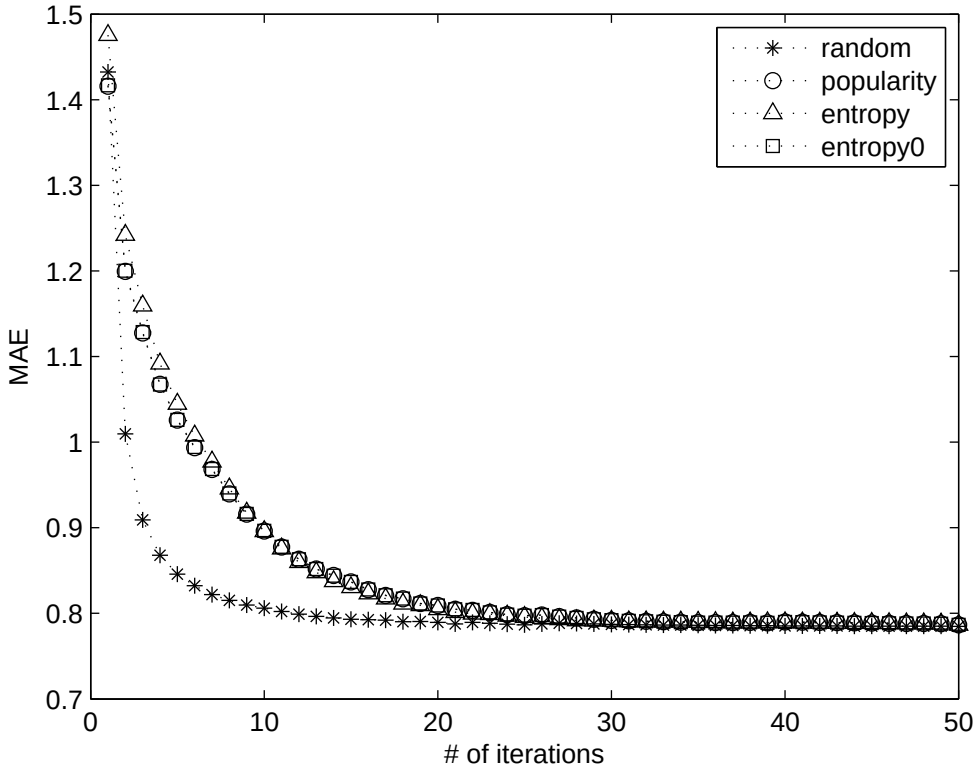

Figura 5.1: Avaliação do grupo *Entropia Pura* na base *MovieLens*

que amenizado por uma função logarítmica, exerce forte influência sobre o comportamento da estratégia. Em segundo lugar, o outro componente que as compõe é o valor da entropia, dado pelas estratégias *entropy* e *entropy0*. É sabido que essas duas estratégias também não obtiveram comportamento muito destoante de *popularity*. Logo, era esperado que tanto *log(pop)*ent* como *log(pop)*ent0* apresentassem comportamentos semelhantes à *popularity*.

Fazendo uma analogia com o mercado de ações, podemos dizer que *log(pop)*ent* está "indexada" por *popularity* e *entropy*, enquanto que *log(pop)*ent0* está "indexada" por *popularity* e *entropy0*. Ou seja, assim como um fundo de investimento que está indexado por um determinado ativo (composto majoritariamente por este ativo) não apresentará rendimentos que destoam significativamente do rendimento deste ativo, da mesma maneira, o comportamento de tais estratégias não será muito distante do comportamento das suas estratégias "índices".

Em [13] são apresentados apenas as estratégias *log(pop)*ent* e *popularity* e pode-se verificar que os comportamentos de ambas seguem bem próximos, o que confirma nossa analogia.

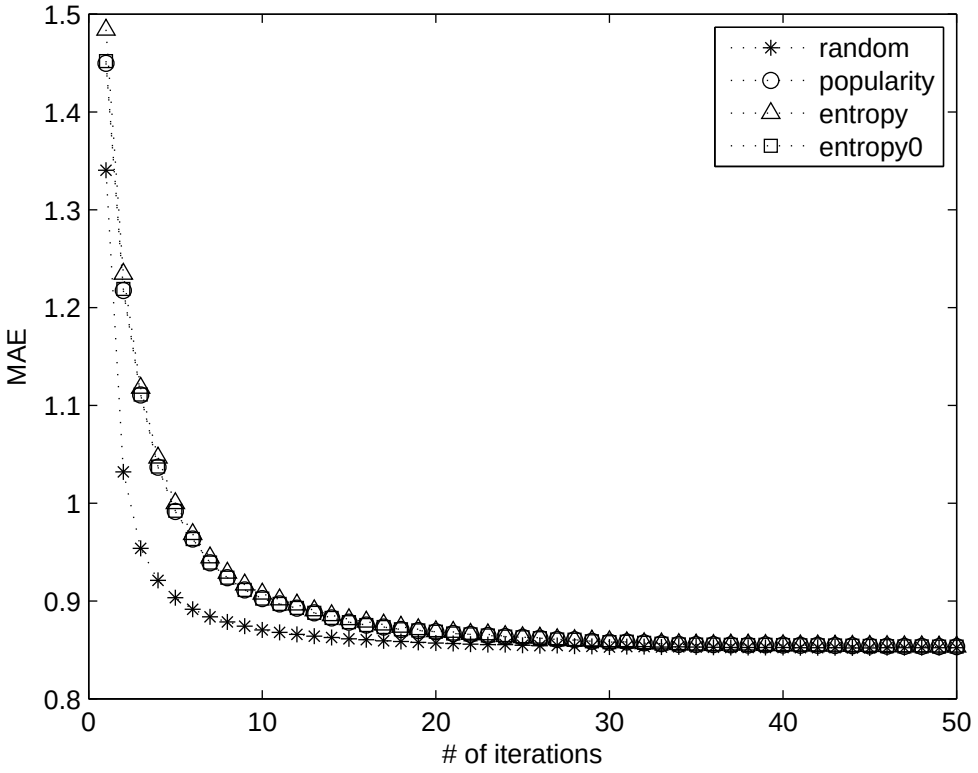

Figura 5.2: Avaliação do grupo *Entropia Pura* na base *Netflix*

5.5.3 Média Harmônica da Popularidade e Entropia

Novamente, pode-se observar a atração que as estratégias índices aplicam sobre suas indexadas. Vemos através das figuras 5.5 e 5.6 que *helf* e *helf0* recaem sobre *popularity*.

Este resultado, juntamente com o anterior, nos informa que tanto combinações simples (e.g., função logarítmica) como combinações complexas (e.g., média harmônica) não suprimem a atração que as estratégias compostas sofrem daquelas que as compõem. Concluímos então que a utilização de estratégias compostas, ou indexadas, não trará ganhos significativos quando as estratégias simples, as que lhes servem como índices, não possuem um bom comportamento elas mesmas.

Apesar de [13] não fazer uso de *helf* nem de *helf0* em seus experimentos, os autores em [10] apresentam os resultados que obtiveram com *helf*. Conforme deduzido aqui, eles seguem bem próximos ao comportamento de *popularity*.

5.5.4 Ganho de Informação

Dentre as estratégias da família *Entropia*, *igcn* parece ter obtido o melhor comportamento, conforme as figuras 5.7 e 5.8 indicam. Entretanto, mesmo sendo a melhor estratégia da família *Entropia*, ela ainda fica muito aquém do desejado, que

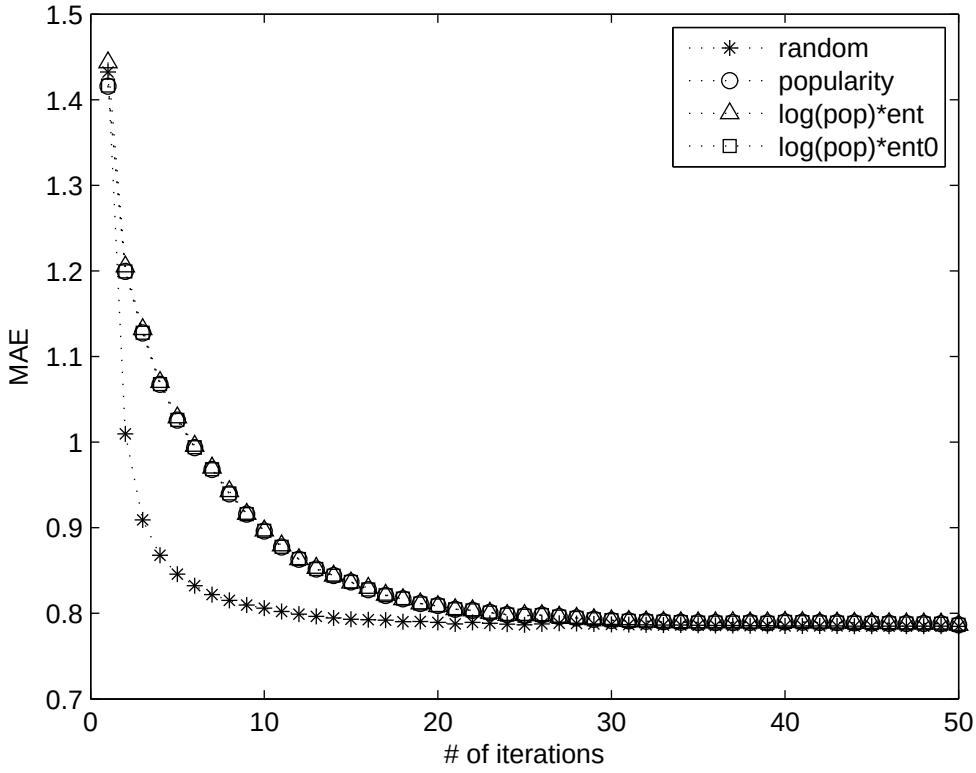

Figura 5.3: Avaliação do grupo *Logaritmo da Popularidade e Entropia* na base *MovieLens*

seria ultrapassar a estratégia *random*.

Apesar de estar inserida nesta família, a *igcn* não se baseia na entropia das preferências, mas sim na entropia dos usuários. Segundo [10], trabalho que propôs a estratégia, os usuários são divididos em grupos, chamados de *clusters*, de acordo com suas disposições no espaço dado pela decomposição SVD. Os itens solicitados são aqueles que, se avaliados, reduzirão a incerteza associada a distribuição dos usuários em *clusters*, i.e., trarão a maior redução na entropia da distribuição dos usuários.

Aqui, como no caso já comentado da entropia pura, há uma heurística subjacente à estratégia que parece introduzir viés no conjunto de treinamento. Buscar os itens com maior potencial para "organizar" os usuários, dentro de um espaço vetorial construído com base nas avaliações, não implica necessariamente em redução do erro do modelo. Afinal, é possível que o estado natural dos dados, incluindo as disposições dos usuários em tal espaço, seja um estado de desordem. Ao solicitar os itens que trarão uma maior organização, estamos induzindo uma ordem que é estranha aos dados e, consequentemente, podemos estar enviesando o conjunto de treinamento.

Somente [10] apresenta resultados para *igcn*, porém, infelizmente, não há comparações com *random*. Todavia, faz-se comparações com *helf*, *entropy0* e *popularity*,

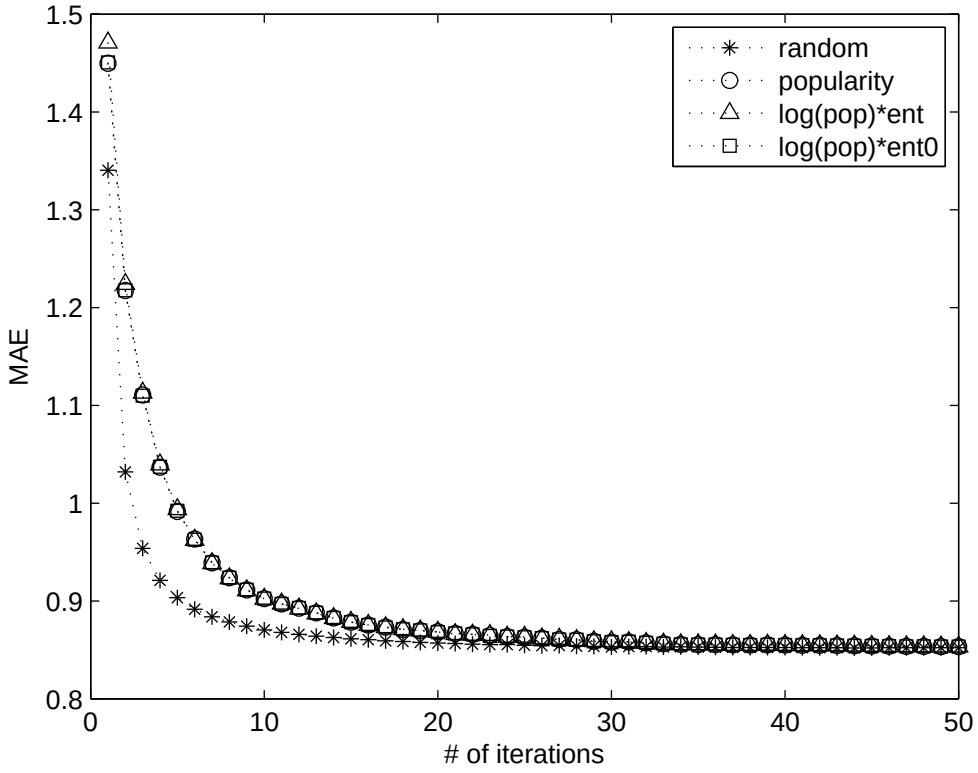

Figura 5.4: Avaliação do grupo *Logaritmo da Popularidade e Entropia* na base *Netflix*

as quais se mostraram inferiores à *igcn*, similar ao que foi verificado em nossos experimentos.

5.6 Variância

Na família *Variância*, como há apenas três estratégias, decidimos por não dividir em grupos e apresentá-las todas de uma só vez. Os resultados podem ser vistos nas figuras 5.9 e 5.10, paras as bases *Movielens* e *Netflix*, respectivamente.

Vemos que a estratégia *variance* apresenta uma ligeira vantagem sobre as demais, sobretudo na figura 5.9, nas primeiras 10 iterações. Contudo, seu comportamento passa logo a seguir o de *popularity*, o que aponta para a presença de viés em K. Novamente, como no caso da entropia, tentar minimizar a incerteza dos itens através da variância, acarreta na construção de um conjunto de treinamento dominado pelos itens mais controversos. Tais itens são justamente os mais difíceis de se prever avaliações e, portanto, o modelo tem grande dificuldade em capturar o padrão que determina as avaliações dadas pelos usuários.

As estratégias compostas utilizando *variance* e *popularity* caem no mesmo inconveniente das estratégias compostas que fazem uso de *entropy* (ou *entropy0*) e

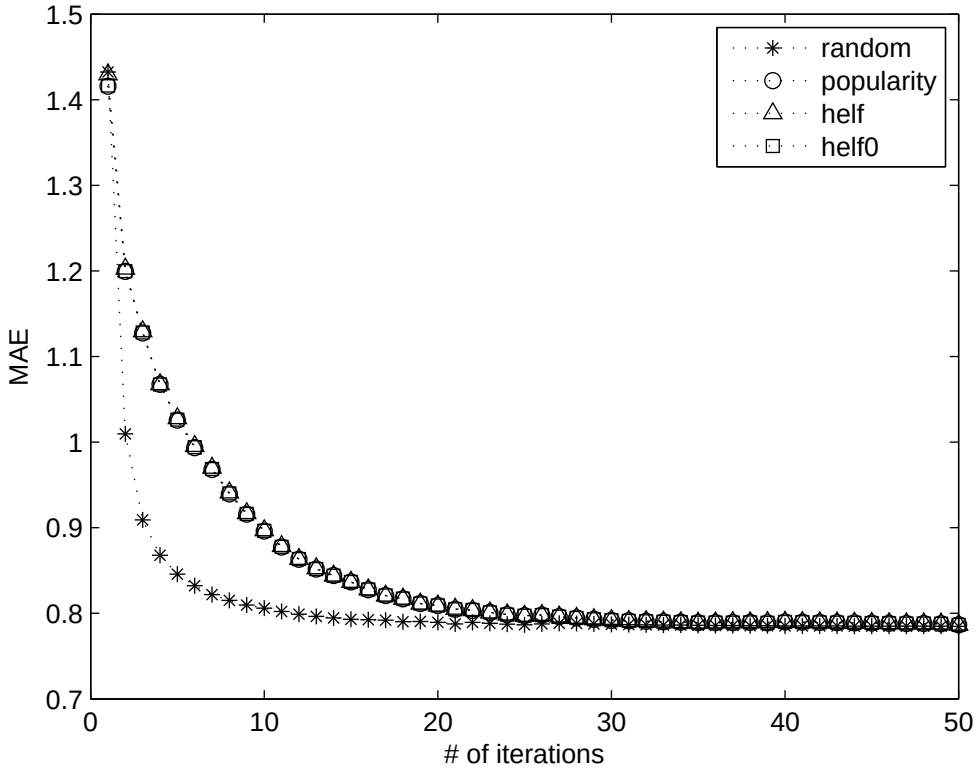

Figura 5.5: Avaliação do grupo *Média Harmônica da Popularidade e Entropia* na base *MovieLens*

popularity, ou seja, acabam por serem atraídas para o comportamento de *popularity*. O que explica a semelhança de comportamento tanto de *log(pop)*var* de *sqrt(pop)*var* com *popularity*.

5.7 Modelo

Dentre as estratégias da família *Modelo*, duas delas obtiveram desempenho pior que a estratégia *random*, enquanto que as outras duas foram melhores. Aliás, deve-se mencionar que foram as únicas de todo nosso experimento que conseguiram tal feitio, com exceção da *Estratégia Livre de Viés*. Cabe à nós então analisarmos o porquê deste comportamento inusitado, tendo sempre em mente a heurística por trás de cada estratégia.

5.7.1 Pior que *random*

Começaremos por analisar as estratégias que obtiveram desempenho pior que *random*, isto é, *high pred* e *bin pred*, exibidas nas figuras 5.11 e 5.12. Observamos que ambas, até a 10ª iteração, apresentam desempenho ligeiramente superior à

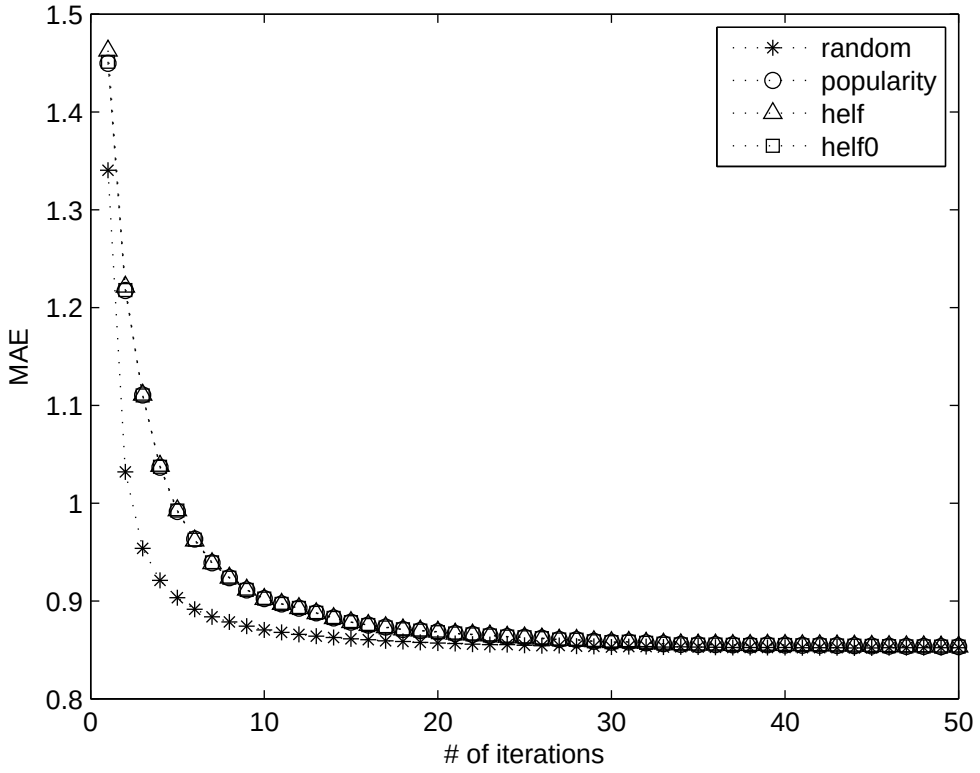

Figura 5.6: Avaliação do grupo *Média Harmônica da Popularidade e Entropia* na base *Netflix*

popularity, convergindo mais tarde para o comportamento desta.

É importante levar em consideração que ambas as bases contêm preferências em forma de números inteiros, variando de 1 a 5. Das 100 mil avaliações presentes na base *MovieLens*, cerca de 6% das notas possuem valor igual a 1; 12% igual a 2; 27% igual a 3; 34% igual a 4; e 21% igual 5. Na base *Netflix*, por sua vez, das 100 mil avaliações, cerca de 8% possuem valor igual a 1; 14% igual a 2; 35% igual a 3; 27% igual a 4; e 16% igual a 5.

Se considerarmos as notas 1 e 2 como avaliações "baixas", e as notas 3, 4 e 5 como avaliações "altas", verificamos que *MovieLens* e *Netflix* possuem 82% e 78% das notas como sendo altas, respectivamente. Esta característica pode elucidar muito a respeito do desempenho que tais estratégias tiveram nessas bases.

A estratégia *high pred* busca solicitar os itens que receberam as melhores previsões do modelo, ou seja, itens cujas previsões apontam para preferências altas. Desta maneira, ela insere no conjunto de treinamento K sempre preferências altas, agravando o desequilíbrio natural que existe entre as preferências e tornando o conjunto ainda mais desbalanceado. Como o modelo é treinado com as preferências em K, este desequilíbrio transfere-se para a acurácia do mesmo, que acaba por ficar mais preciso em prever preferências altas do que baixas.

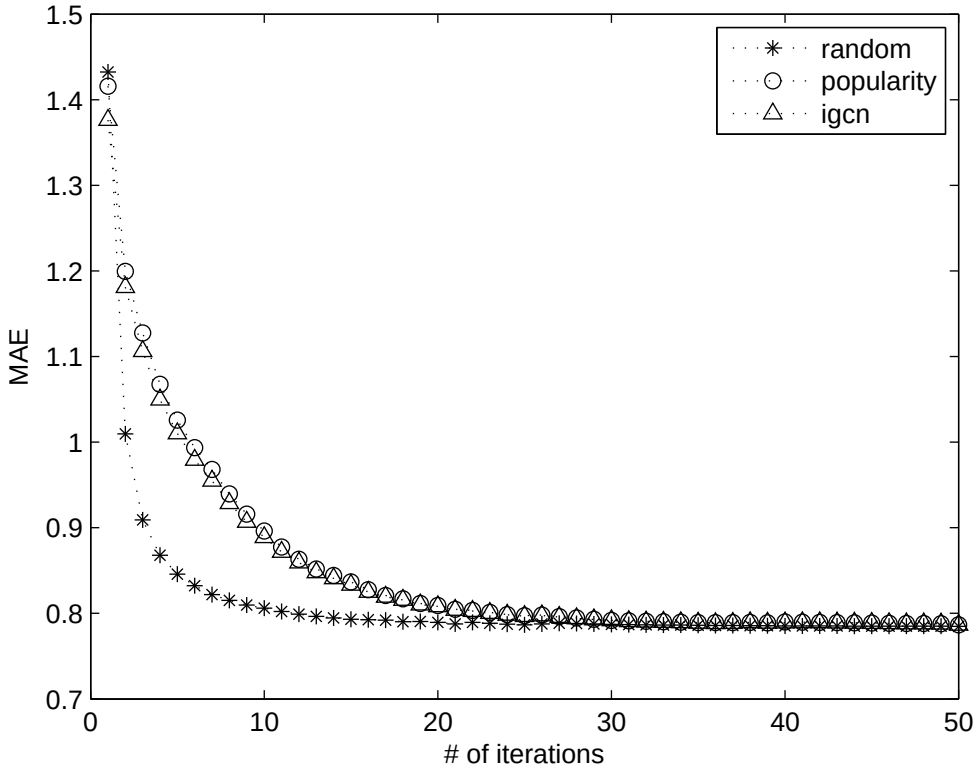

Figura 5.7: Avaliação do grupo *Ganho de Informação* na base *MovieLens*

Por sua vez, *bin pred* procura inserir em K os itens cujas previsões apontam para alguma preferência, seja ela baixa ou alta. Em outras palavras, *bin pred* solicita os itens cujas previsões indicam que os mesmos serão "consumidos". Esta abordagem não faz distinção entre as preferências, reduzindo-as a simples informações binárias. Com isso, acabamos simplesmente mantendo o desequilíbrio natural das preferências, visto que, ao contrário do caso de *high pred*, não temos nenhuma ideia de como o usuário avaliará os itens solicitados.

Ambas as estratégias apresentam desempenho ruim em [13], ou seja, pior que *random*. Contudo, no referido trabalho, *bin pred* é nitidamente melhor que *high pred*, o que não ficou claro em nossos experimentos. Na medida em que solicita apenas os itens de previsão alta, *high pred* reforça o desequilíbrio entre as preferências em K, agravando a disparidade entre as previsões. Como *bin pred* não sabe qual foi o valor efetivo da preferência prevista, o desequilíbrio em K apenas se mantem com os itens solicitados. Por exemplo, mesmo que haja itens com previsão alta, é possível que *bin pred* solicite itens com previsão baixa, o que não seria possível em *high pred*.

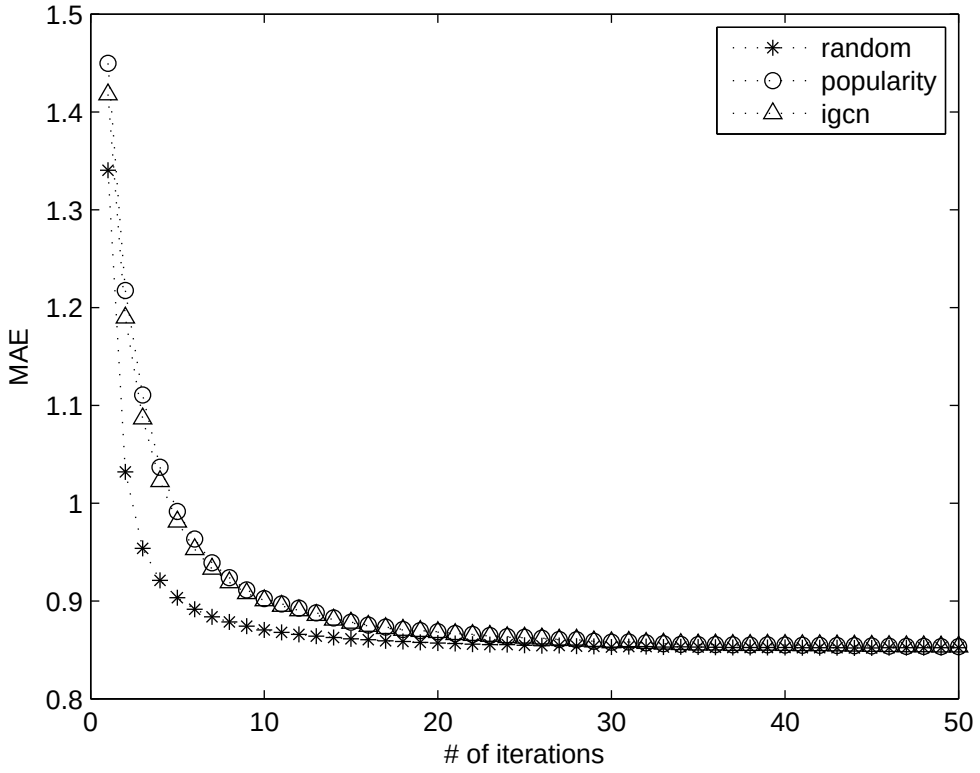

Figura 5.8: Avaliação do grupo *Ganho de Informação* na base *Netflix*

5.7.2 Melhor que *random*

As únicas estratégias que obtiveram desempenho melhor que *random*, com exceção da *Estratégia Livre de Viés*, foram *low pred* e *high-low pred*, conforme pode ser observado nas figuras 5.13 e 5.14.

Ao contrário de *high pred*, *low pred* busca solicitar os itens cujas previsões apontam para preferências baixas. Uma vez que há uma carência natural por tais preferências, ao solicitá-las, esta estratégia introduz em K justamente as preferências faltantes, amenizando o desequilíbrio entre preferências baixas e altas. O conjunto de treinamento acaba ficando mais equilibrado e, consequentemente, a acurácia do modelo fica mais balanceada, capaz de fazer previsões mais precisas para todos os tipos de preferência.

Ou seja, a estratégia *low pred* se beneficia da carência natural de preferências baixas. A heurística que a norteia, solicitar os itens com previsões baixas, mostrou-se eficaz visto que as bases utilizadas em nossos experimentos carecem de tais preferências. Caso as características das bases fossem outras ou se modificassem com o tempo, o desempenho da estratégia poderia ser diferente. Em outras palavras, o simples fato de que a estratégia obteve um bom desempenho não significa que ela não introduz viés no conjunto de treinamento. Pelo contrário, a própria concepção da

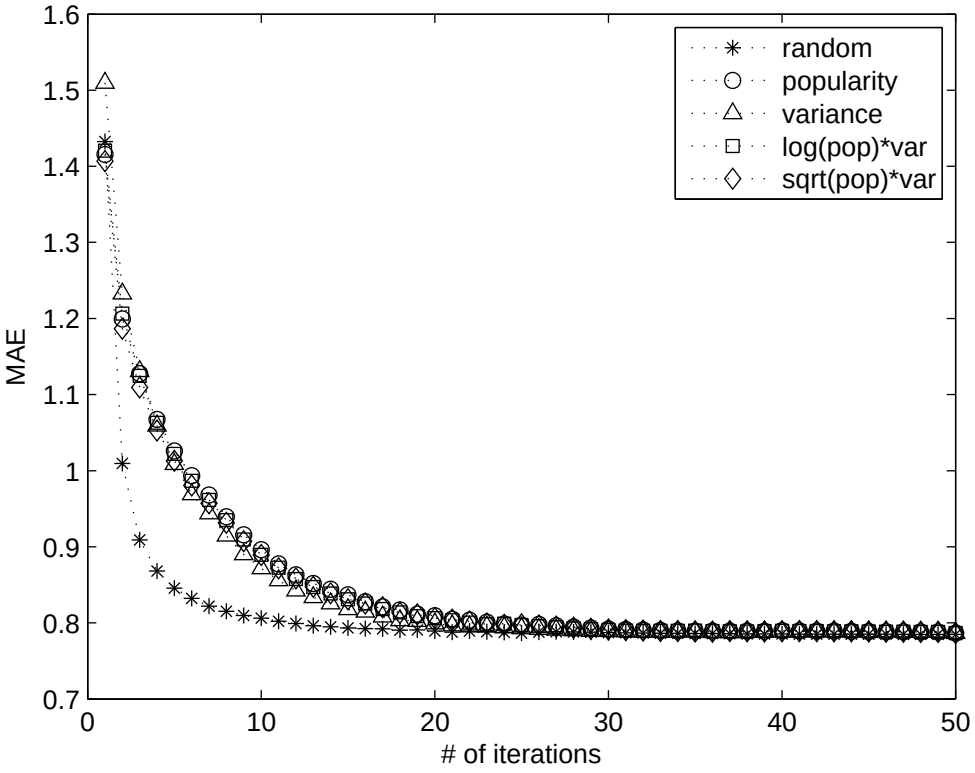

Figura 5.9: Avaliação da família *Variância* na base *MovieLens*

estratégia implica na construção de um conjunto de treinamento enviesado, porém, neste caso, o viés nos foi útil.

Da mesma forma, *high-low pred* assume que é preferível solicitar itens cujas previsões se distanciam do valor médio de preferência. Em ambas as bases, as avaliações podem possuir valores inteiros de 1 a 5, sendo o valor médio 3. Logo, *high-low pred* procura solicitar itens cuja previsão é igual a 1 ou 5 (valores mais distantes do valor médio). Se definirmos as notas 1 e 5 como sendo avaliações "extremas" e as notas 2, 3, e 4 como avaliações "medianas", constatamos que 73% das avaliações em *MovieLens* são medianas, enquanto que, em *Netflix*, este percentual é de 76%.

Ou seja, a distinção entre avaliações baixas e altas esconde um pouco o verdadeiro desequilíbrio entre as preferências. Na verdade, as notas 3 e 4 compõem a grande maioria das avaliações nas duas bases, sendo 61% das avaliações em *MovieLens* e 62% em *Netflix*. Assim como *low pred*, ao solicitar os itens cujas previsões apontam para preferências extremas, *high-low pred* ameniza o desequilíbrio natural que existe nas bases e promove um treinamento mais balanceado do modelo. A heurística subjacente à estratégia acaba sendo, mais uma vez, útil, pois ataca justamente a desproporção entre as preferências.

Em [13], tanto *low pred* quanto *high-low pred* apresentam desempenho pior que *random* nas primeiras iterações, porém conseguem alcançar e superar esta estratégia.

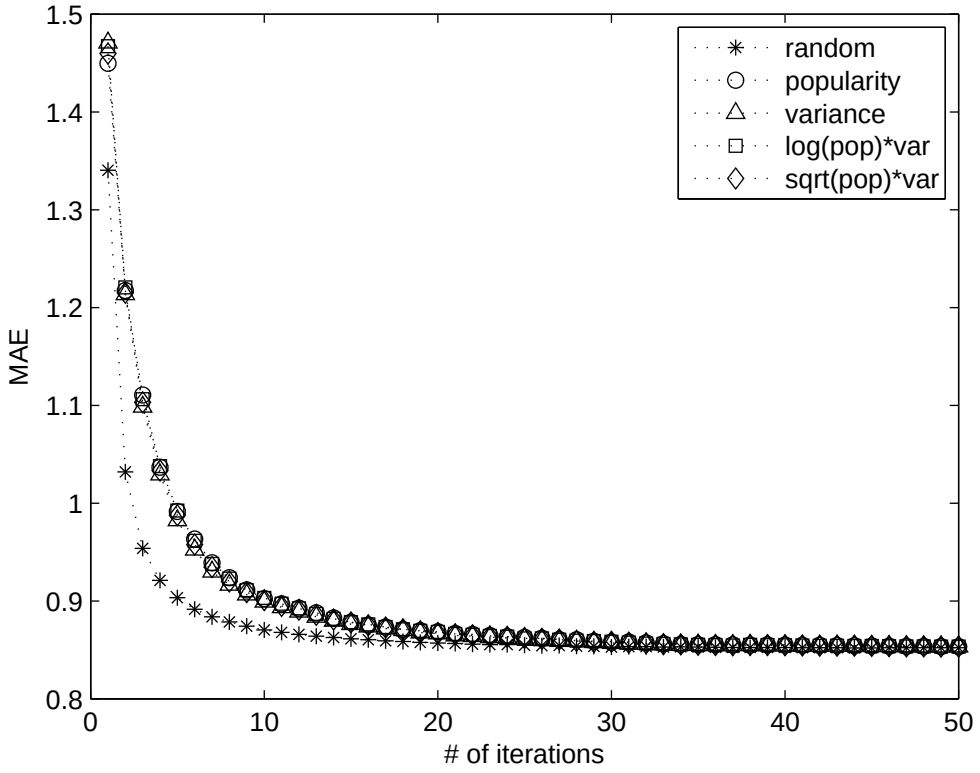

Figura 5.10: Avaliação da família *Variância* na base *Netflix*

No entanto, *high-low pred* mostrou-se melhor que *low pred*, o que não ficou claro em nosso trabalho. Isto significa que a distinção entre preferências "extremas" e "medianas" representa melhor a realidade dos dados do que a distinção entre preferências "baixas" e "altas".

5.8 Estratégia Livre de Viés

Iremos analisar nas próximas seções o comportamento da *Estratégia Livre de Viés*, nomeada como *unbiased* em nossos experimentos, comparando seu comportamento com as duas melhores estratégias até então: *low pred* e *high-low pred*. Além de uma análise global, visualizando todas as iterações, optamos também por realizar uma análise ampliada, com foco nas primeiras 15 primeiras iterações, onde é possível perceber mais nitidamente a superioridade de *unbiased*.

5.8.1 *unbiased vs. low pred*

As figuras 5.15 e 5.16 apresentam uma visão global, ou seja, uma visualização dos comportamentos de *low pred* e *unbiased* durante todas as iterações, nas bases *MovieLens* e *Netflix*, respectivamente. É possível verificar que, nas duas primeiras

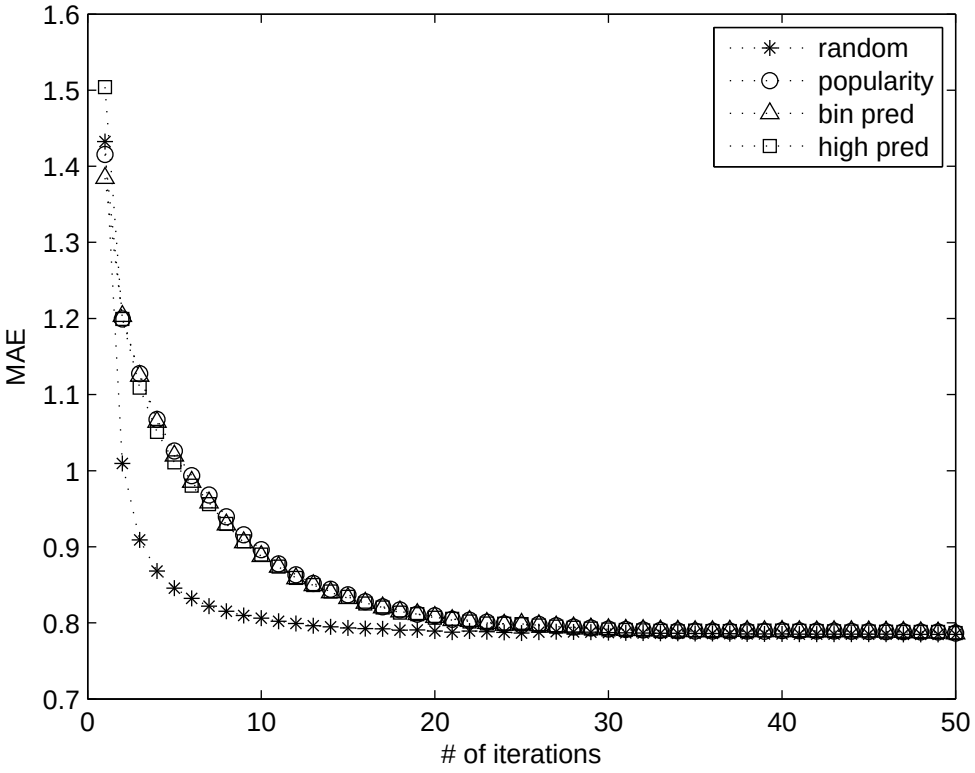

Figura 5.11: Avaliação do grupo *Pior que random* na base *MovieLens*

iterações, *low pred* apresenta certa vantagem, entretanto, a partir da 3ª iteração, *unbiased* parece alcançá-la e acompanhá-la até o final. Infelizmente, como o ganho de *unbiased* é muito sutil, não conseguimos percebê-lo nesta escala.

Assim, decidimos por apresentar uma visão ampliada dos comportamentos de *low pred* e *unbiased*, até a 15ª iteração, de modo que seja possível visualizar com clareza a vantagem desta última. As figuras 5.17 e 5.18 exibem esta visão ampliada nas bases *MovieLens* e *Netflix*, respectivamente.

Vemos que *unbiased* começa pior que *low pred* em ambas as bases, porém, passadas algumas iterações, a primeira alcança a segunda e a supera. É interessante notar que, na base *MovieLens*, *unbiased* alcança *low pred* já na 3ª iteração, enquanto que, em *Netflix*, o empate só se dá por volta da 9ª iteração. Contudo, após o ponto de empate, em ambas as bases, pode-se observar que *unbiased* toma a preeminência sobre a acurácia.

Um detalhe interessante é que esta preeminência de *unbiased* é muito mais explícita na base *MovieLens* do que na base *Netflix*. Além do ponto de empate ocorrer mais cedo, o desvio padrão da acurácia, denotado pelas barras verticais, de *unbiased* está abaixo do de *low pred* em *MovieLens* e eles não se sobrepõem. Isto significa que, mesmo no seu pior momento, o desempenho de *unbiased* ainda é melhor do que o melhor momento de *low pred*.

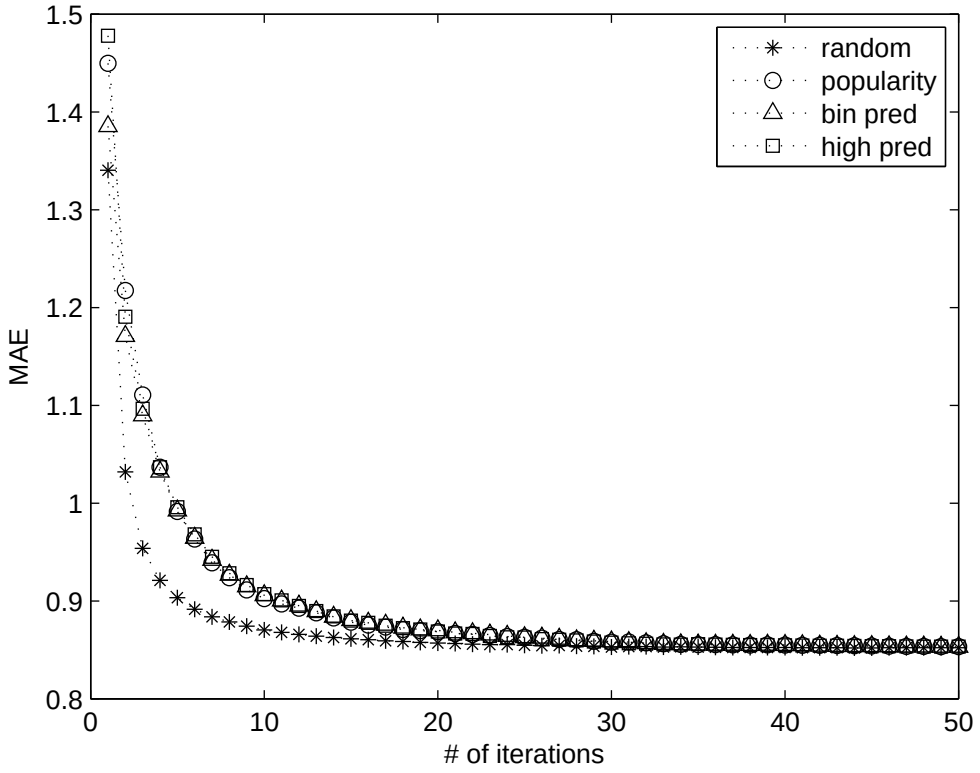

Figura 5.12: Avaliação do grupo *Pior que random* na base *Netflix*

Embora este resultado nos remeta a conclusão de que *unbiased* é efetivamente melhor que *low pred*, é preciso lembrar que nossas conclusões devem se limitar ao dados em questão. Ou seja, apesar do comportamento de *unbiased* ser nitidamente melhor que *low pred*, na base *MovieLens*, os resultados na base *Netflix* já nos levam a uma conclusão diferente. Na média, o comportamento de *unbiased* até foi superior ao de *low pred* em *Netflix*, no entanto, vemos que o desvio padrão das duas estratégias sobrepõem-se nesta base, o que indica que *low pred*, em seus melhores momentos, pode superar *unbiased*.

Esta superioridade de *unbiased* em *MovieLens* pode ser explicada pelo fato de que a escolha dos parâmetros de *unbiased* foi realizada com base apenas em *MovieLens*. É possível que, utilizando parâmetros diferentes, possamos atingir resultados em *Netflix* similares aos de *MovieLens*.

5.8.2 *unbiased vs. high-low pred*

Quando comparamos *unbiased* com *high-low pred* encontramos resultados muito parecidos com a comparação feita com *low pred*. As figuras 5.19 e 5.20 exibem a visão global para as bases *MovieLens* e *Netflix*, respectivamente. Através delas, observa-se que, em ambas as bases, *unbiased* começa com uma notória desvantagem

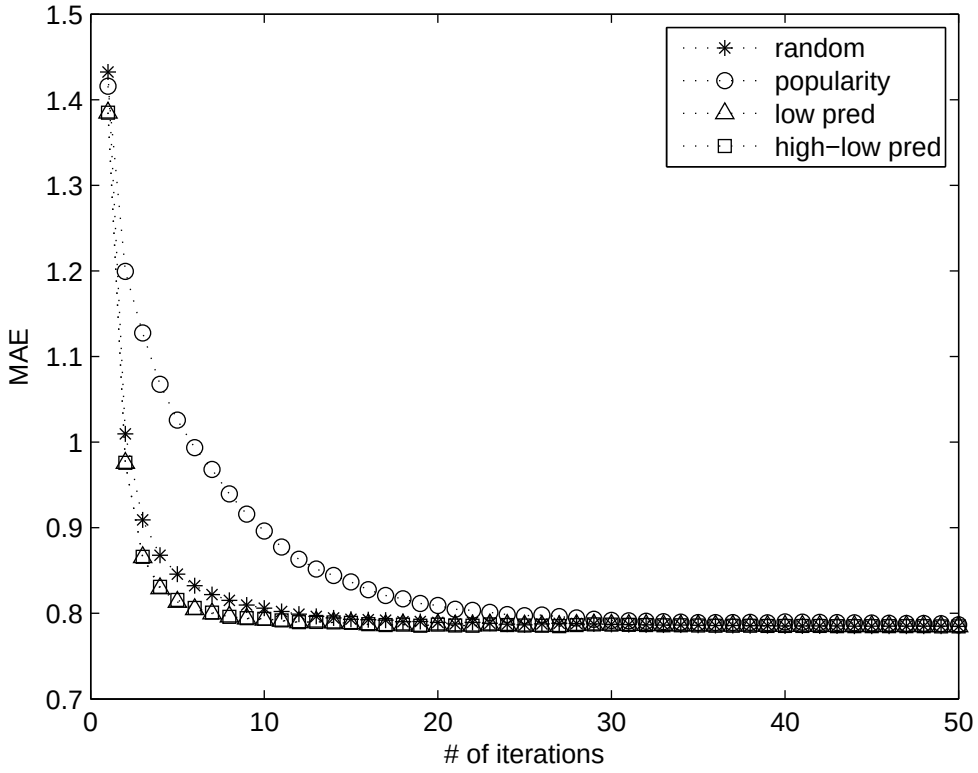

Figura 5.13: Avaliação do grupo *Melhor que random* na base *MovieLens*

frente a *high-low pred*, porém, já na 3ª iteração, ambas parecem confluir para um mesmo valor e seguem assim até o final do experimento.

Na visão ampliada para as bases *MovieLens* e *Netflix*, obtida através das figuras 5.21 e 5.22, respectivamente, vê-se que o ponto de empate de *unbiased* e *high-low pred*, em *MovieLens*, se dá pela 3ª iteração, enquanto que, em *Netflix*, este ponto se dá pela 9ª iteração. Além disso, a relação entre os desvios padrão também se mantem a mesma, já que em *MovieLens* não há sobreposição e em *Netflix* há.

De fato, não deveríamos esperar um resultado diferente para a comparação com *high-low pred*. Tanto *low pred* e *high-low pred* se baseiam em heurísticas que procuram tornar o conjunto de treinamento mais balanceado. *low pred* dá prioridade a itens com previsões 1 e 2 por serem avaliações "baixas", enquanto que *high-low pred* também dá prioridade a itens com previsão 1 e 2 por serem "extremas". Assim, é esperado que haja uma enorme interseção dos itens solicitados por ambas as estratégias e, consequentemente, uma sintonia muito forte de seus comportamentos. Além disso, a questão dos parâmetros também influencia a obtenção do melhor resultado em *MovieLens*.

Verificamos portanto que o desempenho do modelo pode ser impulsionado ou prejudicado dependendo da estratégia empregada. No caso de *low pred* e *high-low pred*, por exemplo, suas heurísticas atendem ao desequilíbrio natural de preferências, for-

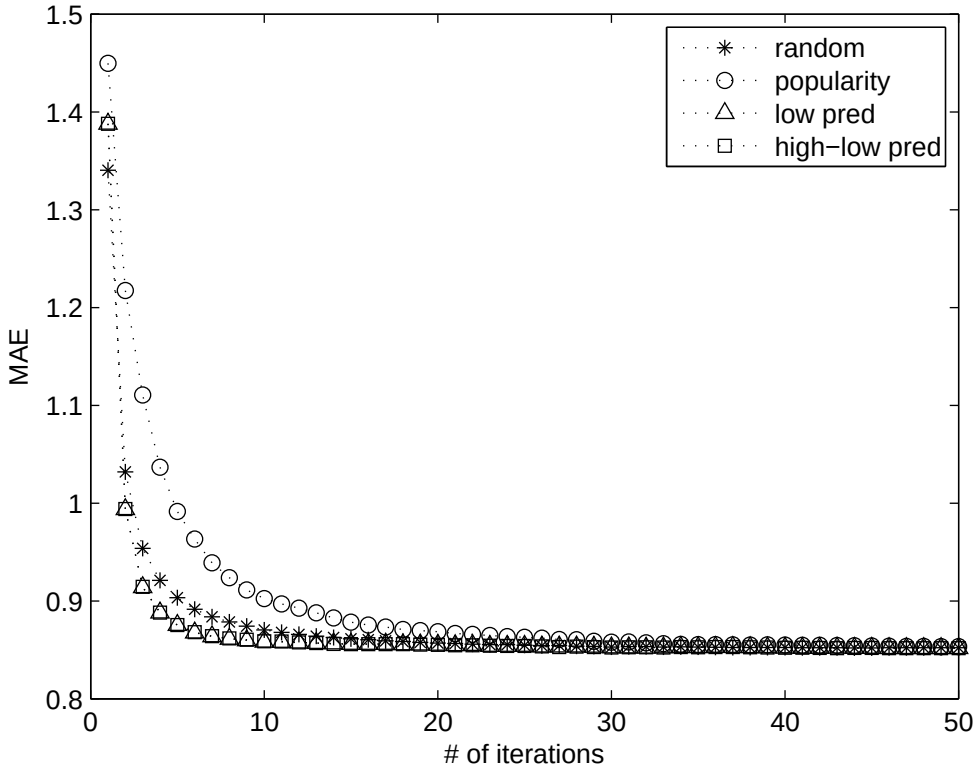

Figura 5.14: Avaliação do grupo *Melhor que random* na base *Netflix*

mando conjuntos de treinamento mais balanceados. No caso das outras estratégias, suas heurísticas agravam ou, no melhor dos casos, mantêm tal desequilíbrio. Em aplicações reais, nem sempre se conhece as características dos dados, sem mencionar que tais características estão em constante mudança conforme a base aumenta. Desta forma, a estratégia "ideal" é aquela que pode ser empregada a qualquer momento, ou seja, independentemente da distribuição dos dados. Assim, sob este aspecto pragmático, a *Estratégia Livre de Viés* é a estratégia "ideal", uma vez que sua heurística pode ser aplicada a qualquer base de dados rendendo bons valores de acurácia do modelo.

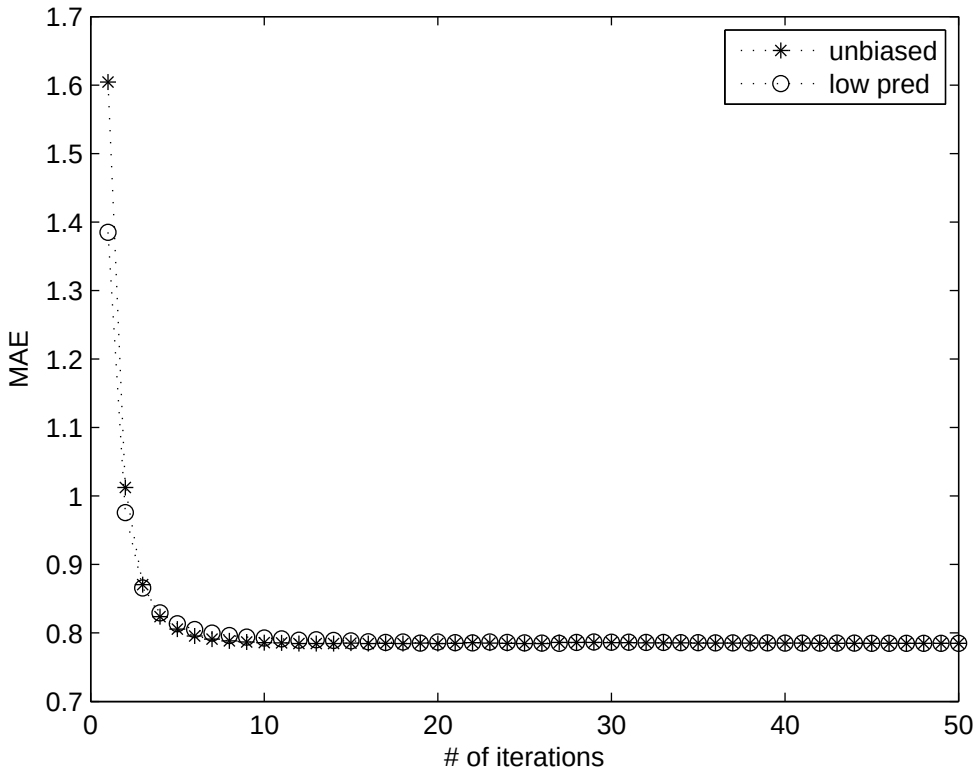

Figura 5.15: Visão global de *unbiased vs. low pred* na base *MovieLens*

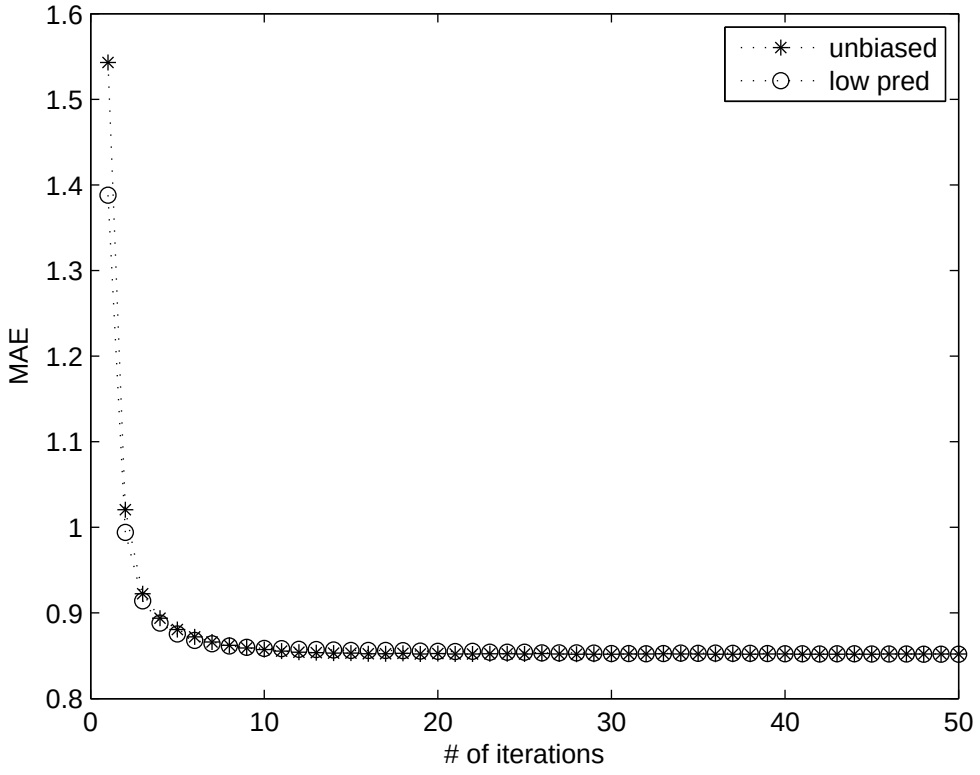

Figura 5.16: Visão global de *unbiased vs. low pred* na base *Netflix*

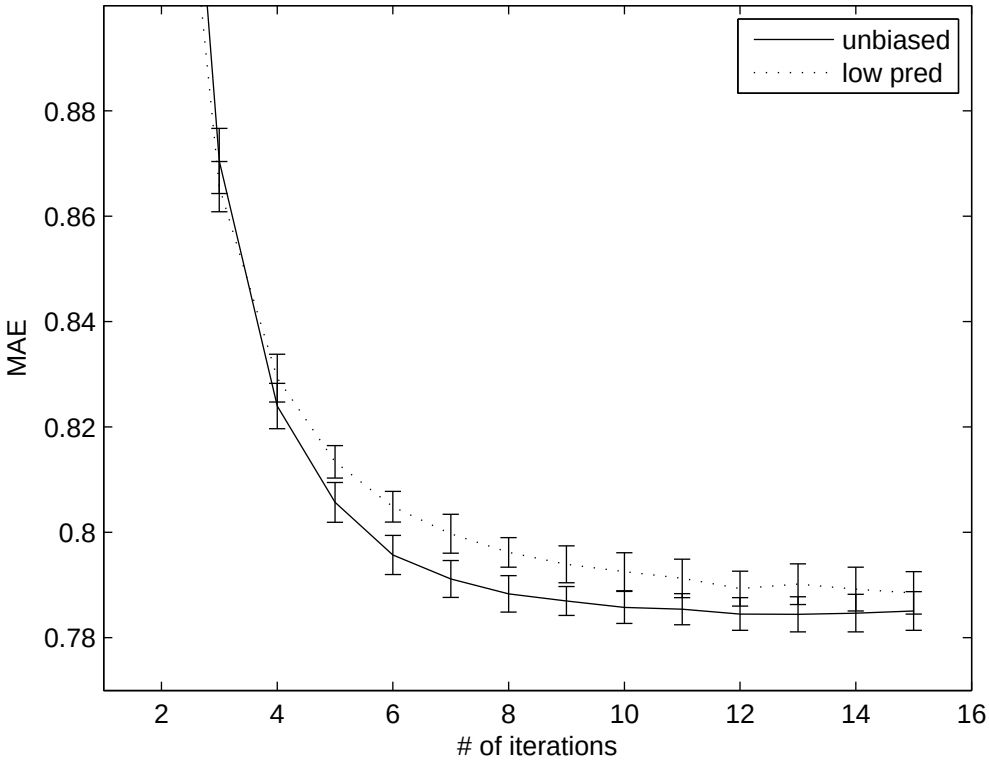

Figura 5.17: Visão ampliada de *unbiased vs. low pred* na base *MovieLens*

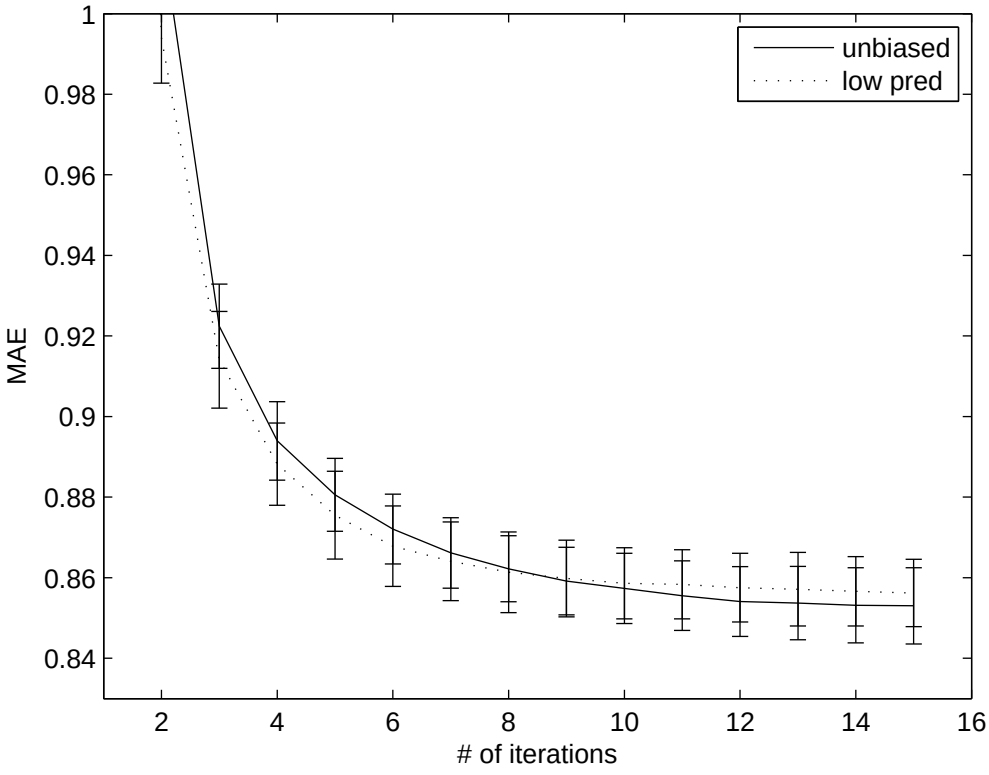

Figura 5.18: Visão ampliada de *unbiased vs. low pred* na base *Netflix*

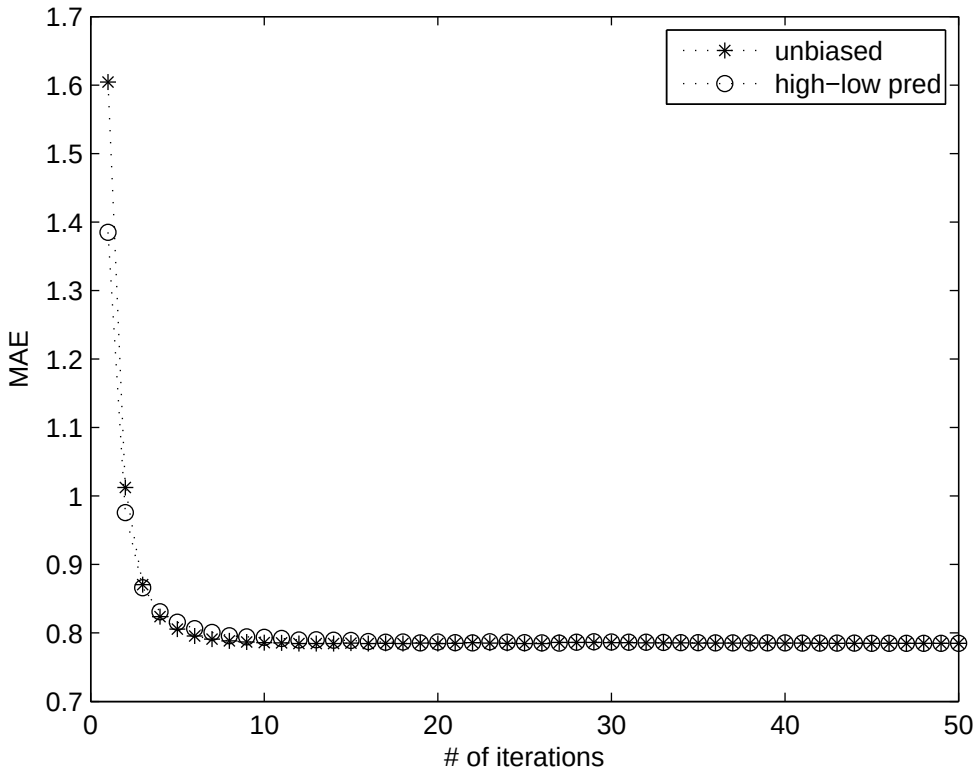

Figura 5.19: Visão global de *unbiased vs. high-low pred* na base *MovieLens*

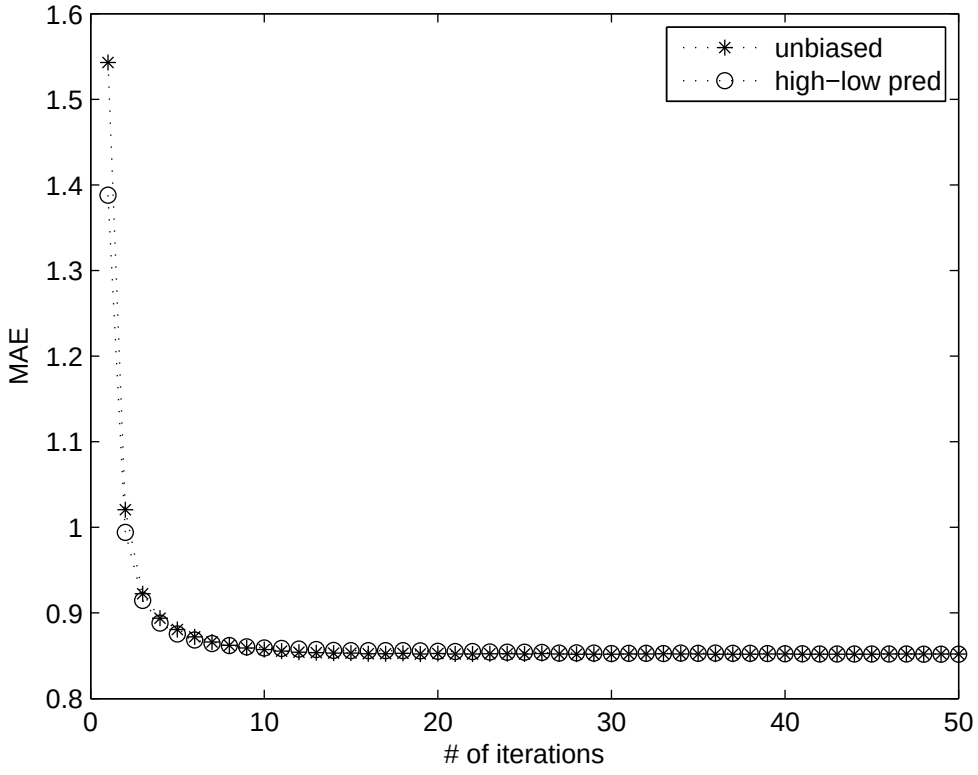

Figura 5.20: Visão global de *unbiased vs. high-low pred* na base *Netflix*

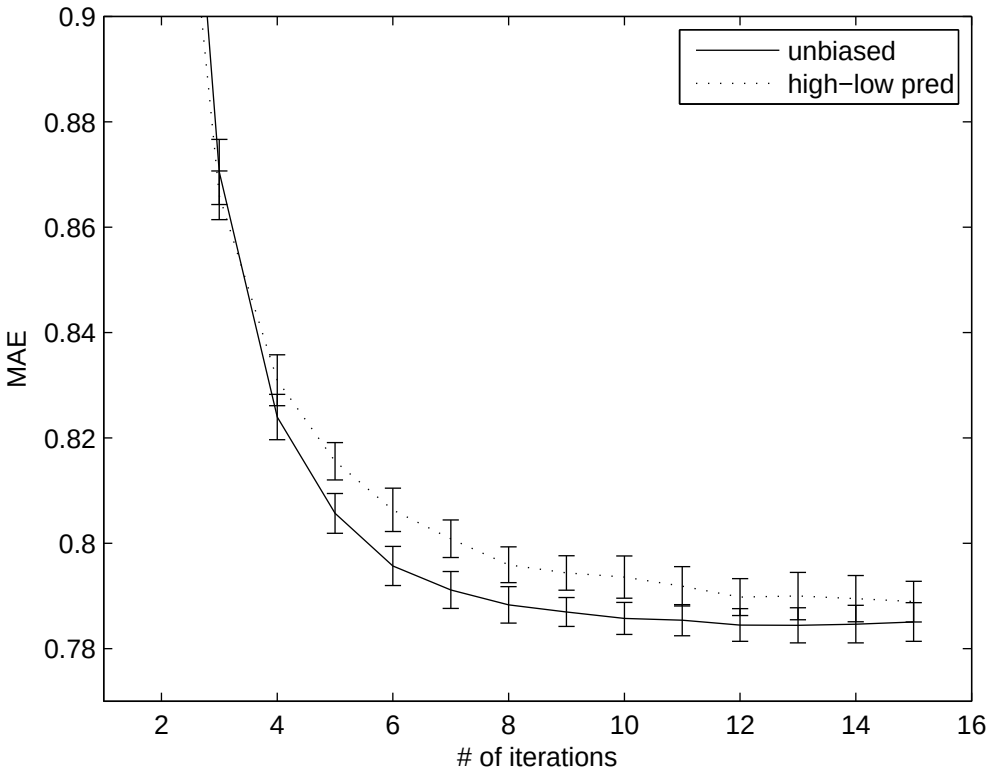

Figura 5.21: Visão ampliada de *unbiased vs. high-low pred* na base *MovieLens*

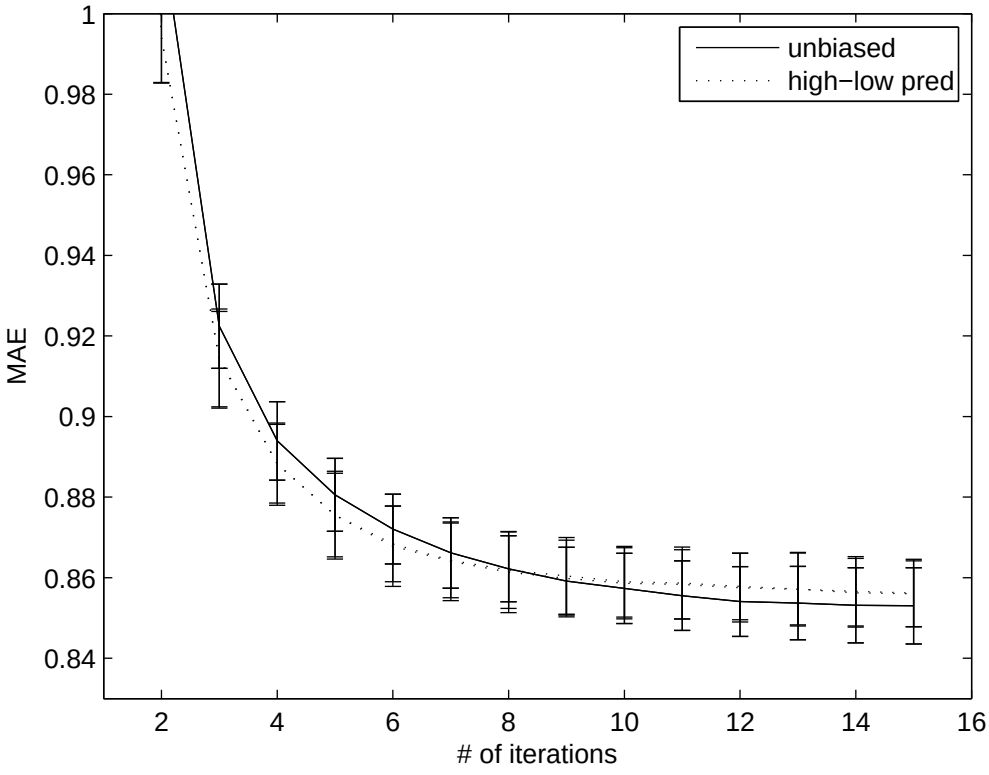

Figura 5.22: Visão ampliada de *unbiased vs. high-low pred* na base *Netflix*

Capítulo 6

Conclusões e Trabalhos Futuros

Neste trabalho, apontamos para a possibilidade de dar incentivos aos usuários de um SR em troca de suas preferências. A *Elicitação de Preferências para fins de Incentivo* foca em usuários com histórico de compras/aquisições, enquanto que a *Elicitação de Preferências para fins de Arranque Frio* se dirige aos nos novos usuários. A fim de escolhermos os melhores itens a serem solicitados, propomos o emprego de estratégias de Aprendizado Ativo, em especial, a *Estratégia Livre de Viés*, que busca formar conjuntos de treinamento de acordo com a distribuição dos dados.

Há três resultados importantes em nossos experimentos que merecem atenção. O primeiro é o bom desempenho obtido por *random* ou, olhando sob outro aspecto, o desempenho ruim obtido pela maioria das estratégias, com exceção de *low pred* e *high-low pred*. As heurísticas que formulam as estratégias ditam a maneira como os itens serão selecionados para inclusão no conjunto de treinamento. Por exemplo, *entropy*, *variance* e suas derivadas assumem que os melhores itens a serem incluídos no conjunto de treinamento são aqueles mais controversos; *igcn* assume que os melhores itens são aqueles que melhor distribuem os usuários em *clusters*; *high pred* assume que o melhor conjunto de treinamento é aquele formado por itens com as maiores avaliações; enquanto que *bin pred* assume que o melhor conjunto é aquele formado por itens com grande chance de serem consumidos.

Em todos esses casos, há uma suposição, ou heurística, que guia a seleção dos itens, gerando assim um conjunto formado por itens que satisfazem esta suposição. Seria correto afirmar então que o emprego de estratégias de AA quase sempre acarreta em conjuntos de treinamento enviesados, o que, à primeira vista, pode parecer indesejado. No entanto, ter um conjunto de treinamento enviesado nem sempre é algo ruim. Nosso segundo resultado digno de ser destacado é o fato de que *low pred* e *high-low pred* obtiveram bons resultados mesmo formando conjuntos de treinamento enviesados. Ou seja, há casos onde um conjunto enviesado pode ser útil.

Em nossos experimentos, como as preferências 3 e 4 excedem as demais, um

conjunto equilibrado promoveu um bom desempenho do modelo, apesar de ser, tecnicamente, enviesado. Todavia, em situações reais muitas vezes não se conhece as características da base de dados, sem mencionar que as mesmas estão em constante mutação conforme a base cresce. Mesmo em um ambiente isolado e controlado como o nosso, foi necessário testar várias estratégias, baseadas em diversas heurísticas diferentes, para encontrarmos aquela que é adequada aos dados (apenas duas heurísticas se mostraram adequadas). Efetuar tal comparação em um sistema real seria muito complexo, sem contar que o resultado seria provisório dado o carácter volátil de um SR. Ou seja, os projetistas de SR necessitam de uma estratégia que possa ser utilizada em qualquer ocasião, sejam quais forem as características da base de dados e, de preferência, independentemente do modelo adotado.

Isto nos leva ao terceiro resultado que é o excelente desempenho obtido pela *Estratégia Livre de Viés*. Tal estratégia assume que o melhor conjunto de treinamento é simplesmente aquele que melhor representa a população, ou seja, aquele onde não há viés. Nossos experimentos mostraram que a *Estratégia Livre de Viés* de fato promove o bom desempenho do modelo, superando as demais. Embora ainda desejamos realizar mais testes, em outras bases, acreditamos ter encontrado uma excelente opção, dentre as diversas estratégias da literatura, que pode ser empregada em qualquer ocasião, sob quaisquer circunstâncias e com qualquer modelo de recomendação. Assim, a *Estratégia Livre Viés* é uma ótima opção para um SR, visto que o risco relacionado à adoção de outra estratégia, cuja heurística pode não condizer com os dados, é alto.

Como trabalhos futuros, além de analisar o comportamento da *Estratégia Livre de Viés* com bases maiores, pretendemos analisá-la sob situações reais. O problema de *Elicitação de Preferências para fins de Incentivo* possui muitas nuances que só podem ser examinadas em um ambiente *online*. Para tal, desejamos utilizar a plataforma *Amazon Mechanical Turk* [19] onde seria possível remunerar usuários em troca de suas avaliações. A partir de então poderíamos tentar responder perguntas do tipo: "Como saber se o usuário está dando sua verdadeira preferência ou se está avaliando aleatoriamente apenas para receber o incentivo?"; "Quanto se deve investir em incentivos para se ter ganhos significativos na acurácia das recomendações?"; "Qual o valor mínimo de incentivo para motivar um usuário a dar sua preferência?"; "Vale a pena um sistema investir em incentivos em troca de avaliações?". Acreditamos que essas questões são, na atualidade, as questões mais cruciais na área de SR.

Quanto a aspectos teóricos, há casos onde um conjunto de treinamento equilibrado pode ser mais benéfico que um conjunto de treinamento sem viés. Por exemplo, em problemas de classificação onde uma das classes é muita rara, é necessário construir um conjunto de treinamento balanceado, senão correremos o risco

do modelo nunca conseguir prever instâncias da classe rara. Neste caso, o conjunto balanceado é enviesado, porém é a maneira mais eficaz de se treinar o modelo. Portanto, pretendemos estudar quando e sob quais circunstâncias vale a pena optar pelo conjunto enviesado em detrimento do mais representativo e vice-versa, procurando estabelecer um limiar teórico com base na proporção das classes.

Referências Bibliográficas

[1] CASTELLS, M. *The information age: economy, society and culture*. Oxford, Blackwell, April 1999. ISBN: 0631215948 9780631215943.

[2] SCHWARTZ, B. *The paradox of choice*. New York, ECCO, January 2005. ISBN: 0060005688 9780060005689.

[3] REHMAN, S., COUGHLAN, J. "Smart agent for automated e-commerce". In: *Sustainable Technologies (WCST), 2011 World Congress on*, pp. 124–128, November 2011.

[4] Ricci, F., Rokach, L., Shapira, B., et al. (Eds.). *Recommender Systems Handbook*. New York, Springer, 2011. ISBN: 978-0-387-85819-7. doi: 10.1007/978-0-387-85820-3. Disponível em: <http://www.springer.com/computer/ai/book/978-0-387-85819-7>. Acesso em: 2014-02-24.

[5] SCHAFER, J. B., KONSTAN, J., RIEDL, J. "Recommender Systems in e-Commerce". In: *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, pp. 158–166, New York, NY, USA, 1999. ACM. ISBN: 1-58113-176-3. doi: 10.1145/336992.337035. Disponível em: <http://doi.acm.org/10.1145/336992.337035>. Acesso em: 2014-02-24.

[6] BENNETT, J., LANNING, S., NETFLIX, N. "The Netflix Prize". In: *In KDD Cup and Workshop in conjunction with KDD*, 2007. doi: 10.1.1.115.6998.

[7] BELL, R. M., KOREN, Y. "Lessons from the Netflix Prize Challenge", *SIGKDD Explor. Newsl.*, v. 9, n. 2, pp. 75–79, December 2007. ISSN: 1931-0145. doi: 10.1145/1345448.1345465. Disponível em: <http://doi.acm.org/10.1145/1345448.1345465>. Acesso em: 2014-02-24.

[8] SU, X., KHOSHGOFTAAR, T. M. "A Survey of Collaborative Filtering Techniques", *Advances in Artificial Intelligence*, v. 2009, January

2009. ISSN: 1687-7470. doi: 10.1155/2009/421425. Disponível em: <http://dx.doi.org/10.1155/2009/421425>. Acesso em: 2014-02-24.

[9] RASHID, A. M., ALBERT, I., COSLEY, D., et al. "Getting to Know You: Learning New User Preferences in Recommender Systems". In: *Proceedings of the 7th International Conference on Intelligent User Interfaces*, IUI '02, pp. 127–134, New York, NY, USA, 2002. ACM. ISBN: 1-58113-459-2. doi: 10.1145/502716.502737. Disponível em: <http://doi.acm.org/10.1145/502716.502737>. Acesso em: 2014-02-25.

[10] RASHID, A. M., KARYPIS, G., RIEDL, J. "Learning Preferences of New Users in Recommender Systems: An Information Theoretic Approach", *SIGKDD Explor. Newsl.*, v. 10, n. 2, pp. 90–100, December 2008. ISSN: 1931-0145. doi: 10.1145/1540276.1540302. Disponível em: <http://doi.acm.org/10.1145/1540276.1540302>. Acesso em: 2014-02-25.

[11] ELAHI, M., REPSYS, V., RICCI, F. "Rating Elicitation Strategies for Collaborative Filtering". In: Huemer, C., Setzer, T. (Eds.), *E-Commerce and Web Technologies*, v. 85, *Lecture Notes in Business Information Processing*, Springer Berlin Heidelberg, pp. 160–171, January 2011. ISBN: 978-3-642-23013-4, 978-3-642-23014-1. doi: 10.1007/978-3-642-23014-1_14. Disponível em: <http://dx.doi.org/10.1007/978-3-642-23014-1_14>. Acesso em: 2014-02-25.

[12] ELAHI, M., RICCI, F., REPSYS, V. "System-wide effectiveness of active learning in collaborative filtering". In: *Proceedings of the International Workshop on Social Web Mining, Co-located with IJCAI, Barcelona, Spain (July 2011)*, July 2011.

[13] ELAHI, M., RICCI, F., RUBENS, N. "Active Learning Strategies for Rating Elicitation in Collaborative Filtering: A System-wide Perspective", *ACM Trans. Intell. Syst. Technol.*, v. 5, n. 1, pp. 1–33, January 2014. ISSN: 2157-6904. doi: 10.1145/2542182.2542195. Disponível em: <http://doi.acm.org/10.1145/2542182.2542195>. Acesso em: 2014-02-25.

[14] CARENINI, G., SMITH, J., POOLE, D. "Towards More Conversational and Collaborative Recommender Systems". In: *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, pp. 12–18, New York, NY, USA, 2003. ACM.

ISBN: 1-58113-586-6. doi: 10.1145/604045.604052. Disponível em: <http://doi.acm.org/10.1145/604045.604052>. Acesso em: 2014-02-26.

[15] LING, K., BEENEN, G., LUDFORD, P., et al. "Using Social Psychology to Motivate Contributions to Online Communities", *Journal of Computer-Mediated Communication*, v. 10, n. 4, 2005. ISSN: 1083-6101. doi: 10.1111/j.1083-6101.2005.tb00273.x. Disponível em: <http://dx.doi.org/10.1111/j.1083-6101.2005.tb00273.x>. Acesso em: 2014-02-26.

[16] RASHID, A. M., LING, K., TASSONE, R. D., et al. "Motivating Participation by Displaying the Value of Contribution". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pp. 955–958, New York, NY, USA, 2006. ACM. ISBN: 1-59593-372-7. doi: 10.1145/1124772.1124915. Disponível em: <http://doi.acm.org/10.1145/1124772.1124915>. Acesso em: 2014-02-26.

[17] HARPER, F., LI, X., CHEN, Y., et al. "An Economic Model of User Rating in an Online Recommender System". In: Ardissono, L., Brna, P., Mitrovic, A. (Eds.), *User Modeling 2005*, v. 3538, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 307–316, January 2005. ISBN: 978-3-540-27885-6. doi: 10.1007/11527886_40. Disponível em: <http://dx.doi.org/10.1007/11527886_40>. Acesso em: 2014-02-26.

[18] AVERY, C., RESNICK, P., ZECKHAUSER, R. "The Market for Evaluations", *The American Economic Review*, v. 89, n. 3, pp. 564–584, 1999. ISSN: 00028282. Disponível em: <http://www.jstor.org/stable/117033>. Acesso em: 2014-02-26.

[19] LEE, J., JANG, M., LEE, D., et al. "Alleviating the Sparsity in Collaborative Filtering using Crowdsourcing". In: *Workshop on Crowdsourcing and Human Computation for Recommender Systems (CrowdRec)*, p. 5, 2013.

[20] LINDEN, G., SMITH, B., YORK, J. "Amazon.com recommendations: Item-to-item collaborative filtering", *Internet Computing, IEEE*, v. 7, pp. 76–80, 2003.

[21] GOLDBERG, D., NICHOLS, D., OKI, B. M., et al. "Using Collaborative Filtering to Weave an Information Tapestry", *Commun. ACM*, v. 35, n. 12, pp. 61–70, December 1992. ISSN: 0001-0782. doi: 10.1145/138859.138867.

Disponível em: <http://doi.acm.org/10.1145/138859.138867>. Acesso em: 2014-06-15.

[22] SHARDANAND, U., MAES, P. "Social Information Filtering: Algorithms for Automating &Ldquo;Word of Mouth&Rdquo;". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pp. 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN: 0-201-84705-1. doi: 10.1145/223904.223931. Disponível em: <http://dx.doi.org/10.1145/223904.223931>. Acesso em: 2014-06-15.

[23] RESNICK, P., IACOVOU, N., SUCHAK, M., et al. "GroupLens: An Open Architecture for Collaborative Filtering of Netnews". In: *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pp. 175–186, New York, NY, USA, 1994. ACM. ISBN: 0-89791-689-1. doi: 10.1145/192844.192905. Disponível em: <http://doi.acm.org/10.1145/192844.192905>. Acesso em: 2014-06-15.

[24] BAEZA-YATES, R. A., RIBEIRO-NETO, B. *Modern Information Retrieval*. Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN: 020139829X.

[25] ADOMAVICIUS, G., TUZHILIN, A. "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.*, v. 17, n. 6, pp. 734–749, June 2005. ISSN: 1041-4347. doi: 10.1109/TKDE.2005.99. Disponível em: <http://dx.doi.org/10.1109/TKDE.2005.99>. Acesso em: 2014-06-15.

[26] DO CARMO, F. B. *Transformando o Problema de Recomendação em Aprendizado de Máquina Supervisionado*. Tese de Mestrado, Federal University of Rio de Janeiro (UFRJ), Brazil, 2013.

[27] STRANG, G. *Introduction to Linear Algebra*. Wellesley, MA, Wellesley-Cambridge Press, 2009. ISBN: 0980232716.

[28] DUMAIS, S. T. "Latent semantic analysis", *Annual Review of Information Science and Technology*, v. 38, n. 1, pp. 188–230, 2004. ISSN: 1550-8382. doi: 10.1002/aris.1440380105. Disponível em: <http://dx.doi.org/10.1002/aris.1440380105>. Acesso em: 2014-06-27.

[29] FUNK, S. "Netflix Update: Try This at Home". December 2006. Disponível em: <http://http://sifter.org/~simon/journal/20061211.html>. Acesso em: 2014-06-30.

[30] PATEREK, A. "Improving regularized singular value decomposition for collaborative filtering". In: *Proc. KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pp. 39–42, 2007.

[31] CLAYPOOL, M., GOKHALE, A., MIRANDA, T., et al. "Combining Content-Based and Collaborative Filters in an Online Newspaper". 1999.

[32] PAZZANI, M. J. "A Framework for Collaborative, Content-Based and Demographic Filtering", *Artif. Intell. Rev.*, v. 13, n. 5-6, pp. 393–408, December 1999. ISSN: 0269-2821. doi: 10.1023/A:1006544522159. Disponível em: <http://dx.doi.org/10.1023/A:1006544522159>. Acesso em: 2014-07-03.

[33] BALABANOVIĆ, M., SHOHAM, Y. "Fab: Content-based, Collaborative Recommendation", *Commun. ACM*, v. 40, n. 3, pp. 66–72, March 1997. ISSN: 0001-0782. doi: 10.1145/245108.245124. Disponível em: <http://doi.acm.org/10.1145/245108.245124>. Acesso em: 2014-07-03.

[34] MELVILLE, P., MOONEY, R. J., NAGARAJAN, R. "Content-boosted Collaborative Filtering for Improved Recommendations". In: *Eighteenth National Conference on Artificial Intelligence*, pp. 187–192, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. ISBN: 0-262-51129-0. Disponível em: <http://dl.acm.org/citation.cfm?id=777092.777124>. Acesso em: 2014-07-03.

[35] NICHOLAS, I. S. C., NICHOLAS, C. K. "Combining Content and Collaboration in Text Filtering". In: *In Proceedings of the IJCAI'99 Workshop on Machine Learning for Information Filtering*, pp. 86–91, 1999.

[36] BERRY, M. W., BROWNE, M. *Understanding Search Engines: Mathematical Modeling and Text Retrieval*. Philadelphia, PA, USA, Society for Industrial and Applied Mathematics, 1999. ISBN: 0-89871-437-0.

[37] ANSARI, A., ESSEGAIER, S., KOHLI, R. "Internet Recommendation Systems", pp. 363–375, 2000.

[38] CONDLIFF, M. K., LEWIS, D. D., MADIGAN, D. "Bayesian Mixed-Effects Models for Recommender Systems". In: *In ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

[39] POPESCUL, R., UNGAR, L. H. "Probabilistic models for unified collaborative and content-based recommendation in sparsedata environments". In: *In UAI '01, 437–444*, 2001.

[40] SETTLES, B. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[41] RUBENS, N., KAPLAN, D., SUGIYAMA, M. "Active Learning in Recommender Systems". In: Kantor, P., Ricci, F., Rokach, L., et al. (Eds.), *Recommender Systems Handbook*, Springer, pp. 735–767, 2011. doi: 10.1007/978-0-387-85820-3_23.

[42] BOUTILIER, C., ZEMEL, R. S., MARLIN, B. "Active Collaborative Filtering". In: *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, UAI'03, pp. 98–106, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc. ISBN: 0-127-05664-5. Disponível em: <http://dl.acm.org/citation.cfm?id=2100584.2100596>. Acesso em: 2014-07-23.

[43] JIN, R., SI, L. "A Bayesian Approach Toward Active Learning for Collaborative Filtering". In: *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pp. 278–285, Arlington, Virginia, United States, 2004. AUAI Press. ISBN: 0-9749039-0-6. Disponível em: <http://dl.acm.org/citation.cfm?id=1036843.1036877>. Acesso em: 2014-07-23.

[44] HARPALE, A. S., YANG, Y. "Personalized Active Learning for Collaborative Filtering". In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pp. 91–98, New York, NY, USA, 2008. ACM. ISBN: 978-1-60558-164-4. doi: 10.1145/1390334.1390352. Disponível em: <http://doi.acm.org/10.1145/1390334.1390352>. Acesso em: 2014-07-23.

[45] SEUNG, H. S., OPPER, M., SOMPOLINSKY, H. "Query by Committee". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pp. 287–294, New York, NY, USA, 1992. ACM. ISBN: 0-89791-497-X. doi: 10.1145/130385.130417. Disponível em:

<http://doi.acm.org/10.1145/130385.130417>. Acesso em: 2014-07-24.

[46] GOLBANDI, N., KOREN, Y., LEMPEL, R. "On Bootstrapping Recommender Systems". In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pp. 1805–1808, New York, NY, USA, 2010. ACM. ISBN: 978-1-4503-0099-5. doi: 10.1145/1871437.1871734. Disponível em: <http://doi.acm.org/10.1145/1871437.1871734>. Acesso em: 2014-04-11.

[47] MELLO, C. E., AUFAURE, M.-A., ZIMBRAO, G. "Active Learning Driven by Rating Impact Analysis". In: *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pp. 341–344, New York, NY, USA, 2010. ACM. ISBN: 978-1-60558-906-0. doi: 10.1145/1864708.1864782. Disponível em: <http://doi.acm.org/10.1145/1864708.1864782>. Acesso em: 2014-07-25.

[48] GOLBANDI, N., KOREN, Y., LEMPEL, R. "Adaptive Bootstrapping of Recommender Systems Using Decision Trees". In: *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pp. 595–604, New York, NY, USA, 2011. ACM. ISBN: 978-1-4503-0493-1. doi: 10.1145/1935826.1935910. Disponível em: <http://doi.acm.org/10.1145/1935826.1935910>. Acesso em: 2014-07-29.

[49] DAVENPORT, T. H. *Big Data at Work: Dispelling the Myths, Uncovering the Opportunities*. Boston Massachusetts, Harvard Business Review Press, February 2014. ISBN: 9781422168165.

[50] DE MELLO, C. E. R. *Active Learning: An Unbiased Approach*. Tese de Doutorado, Federal University of Rio de Janeiro (UFRJ), Brazil, 2013.

[51] DEGROOT, M. H. *Probability and Statistics*. Addison-Wesley, 1986.

[52] DUDA, R. O., HART, P. E., STORK, D. G. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000. ISBN: 0471056693.

[53] PRINCIPE, J. C. *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*. Springer Publishing Company, Incorporated, 2010. ISBN: 1441915699, 9781441915696.

[54] KOREN, Y., BELL, R. M. "Advances in Collaborative Filtering." In: Ricci, F., Rokach, L., Shapira, B., et al. (Eds.), *Recommender Systems Handbook*, Springer, pp. 145–186, 2011. ISBN: 978-0-387-85819-7. Disponível em: <http://dx.doi.org/10.1007/978-0-387-85820-3_5>. Acesso em: 2014-04-07.

[55] HERLOCKER, J. L., KONSTAN, J. A., BORCHERS, A., et al. "An Algorithmic Framework for Performing Collaborative Filtering". In: *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pp. 230–237, New York, NY, USA, 1999. ACM. ISBN: 1-58113-096-1. doi: 10.1145/312624.312682. Disponível em: <http://doi.acm.org/10.1145/312624.312682>. Acesso em: 2014-04-08.

[56] ROKACH, L., MAIMON, O. *Data Mining with Decision Trees: Theory and Applications*. River Edge, NJ, USA, World Scientific Publishing Co., Inc., 2008. ISBN: 9789812771711, 9812771719.