Agrupamento de dados baseado em predições de
modelos de regressão: desenvolvimentos e
aplicações em sistemas de recomendação

André Luiz Vazine Pereira

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

André Luiz Vizine Pereira

Agrupamento de dados baseado em predições de modelos de regressão: desenvolvimentos e aplicações em sistemas de recomendação

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Eduardo Raul Hruschka

USP – São Carlos
Junho de 2016

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

P436a Pereira, André Luiz Vazine
Agrupamento de dados baseado em predições de
modelos de regressão: Desenvolvimentos e aplicações
em Sistemas de Recomendação / André Luiz Vazine
Pereira; orientador Eduardo Raul Hruschka. -- São
Carlos, 2016.
106 p.

Tese (Doutorado - Programa de Pós-Graduação em
Ciências de Computação e Matemática Computacional) --
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2016.

1. Sistemas de Recomendação. 2. Cold-start. 3.
Biclusterização. 4. Modelos de Predição. 5. Máquinas
de Aprendizado Extremo. I. Hruschka, Eduardo Raul,
orient. II. Título.

André Luiz Vizine Pereira

**Data clustering based on prediction regression models:
developments and applications in recommender systems**

Doctoral dissertation submitted to the Instituto de
Ciências Matemáticas e de Computação – ICMC-USP,
in partial fulfillment of the requirements for the degree
of the Doctorate Program in Computer Science and
Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Eduardo Raul Hruschka

**USP – São Carlos
June 2016**

À minha família.

AGRADECIMENTOS

Ao meu orientador professor Eduardo Raul Hruschka, muito obrigado pela excepcional orientação, conhecimento, dedicação, ajuda e paciência. Realmente muito obrigado pela oportunidade de trabalharmos juntos, e pelo conhecimento que adquiri ao longo desses anos.

À Andrea, minha esposa, por ter sido paciente e compreensiva ao longo desses anos e por estar sempre a meu lado, seja nos bons momentos ou nos momentos difíceis. Aos meus filhos, Amanda e Mário pelo apoio e pelas palavras de incentivo nas horas de cansaço.

Ao professor Fernando José Von Zuben, pelas valiosas contribuições durante a pesquisa sobre Máquinas de Aprendizado Extremo.

Ao professor Roberto Pinto Souto, pelas contribuições na pesquisa sobre Processamento Paralelo.

Aos professores do ICMC pela competência e dedicação.

A todos os colegas de estudo que conheci no ICMC ao longo desta jornada. Especialmente a Nádia Felix e Luís Paulo Faina pelas longas horas de estudo em grupo.

Aos amigos da Fatec Rubens Lara (alunos, professores e funcionários) que de alguma forma vivenciaram comigo o desenvolvimento deste trabalho. Especialmente ao professor Cirne Trindade pela ajuda nos momentos de discussão sobre assuntos relacionados a esta tese, e aos irmãos Alberto Saraiva e Nilton Saraiva pelo companheirismo e por estarem sempre dispostos a ajudar.

À Intel Software, por ter disponibilizado a plataforma de Computação de Alto Desempenho para que os experimentos fossem realizados. Um agradecimento especial para Antonio Rivera, Túlio Marin e Igor Freitas.

“Sonhos não morrem, apenas adormecem na alma da gente.”
(Chico Xavier)

RESUMO

Sistemas de Recomendação (SR) vêm se apresentando como poderosas ferramentas para portais *web* tais como sítios de comércio eletrônico. Para fazer suas recomendações, os SR se utilizam de fontes de dados variadas, as quais capturam as características dos usuários, dos itens e suas transações, bem como de modelos de predição. Dada a grande quantidade de dados envolvidos, é improvável que todas as recomendações possam ser bem representadas por um único modelo global de predição. Um outro importante aspecto a ser observado é o problema conhecido por *cold-start*, que apesar dos avanços na área de SR, é ainda uma questão relevante que merece uma maior atenção. O problema está relacionado com a falta de informação prévia sobre novos usuários ou novos itens do sistema. Esta tese apresenta uma abordagem híbrida de recomendação capaz de lidar com situações extremas de *cold-start*. A abordagem foi desenvolvida com base no algoritmo SCOAL (*Simultaneous Co-Clustering and Learning*). Na sua versão original, baseada em múltiplos modelos lineares de predição, o algoritmo SCOAL mostrou-se eficiente e versátil, podendo ser utilizado numa ampla gama de problemas de classificação e/ou regressão. Para melhorar o algoritmo SCOAL no sentido de deixá-lo mais versátil por meio do uso de modelos não lineares, esta tese apresenta uma variante do algoritmo SCOAL que utiliza modelos de predição baseados em Máquinas de Aprendizado Extremo. Além da capacidade de predição, um outro fator que deve ser levado em consideração no desenvolvimento de SR é a escalabilidade do sistema. Neste sentido, foi desenvolvida uma versão paralela do algoritmo SCOAL baseada em OpenMP, que minimiza o tempo envolvido no cálculo dos modelos de predição. Experimentos computacionais controlados, por meio de bases de dados amplamente usadas na prática, comprovam que todos os desenvolvimentos propostos tornam o SCOAL ainda mais atraente para aplicações práticas variadas.

Palavras-chave: Sistemas de Recomendação, *Cold-start*, Biclusterização, Modelos de Predição, Máquinas de Aprendizado Extremo.

ABSTRACT

Recommender Systems (RS) are powerful and popular tools for e-commerce. To build its recommendations, RS make use of multiple data sources, capture the characteristics of items, users and their transactions, and take advantage of prediction models. Given the large amount of data involved in the predictions made by RS, it is unlikely that all predictions can be well represented by a single global model. Another important aspect to note is the problem known as cold-start that, despite that recent advances in the RS area, it is still a relevant issue that deserves further attention. The problem arises due to the lack of prior information about new users and new items. This thesis presents a hybrid recommendation approach that addresses the (pure) cold start problem, where no collaborative information (ratings) is available for new users. The approach is based on an existing algorithm, named SCOAL (Simultaneous Co-Clustering and Learning). In its original version, based on multiple linear prediction models, the SCOAL algorithm has shown to be efficient and versatile. In addition, it can be used in a wide range of problems of classification and / or regression. The SCOAL algorithm showed impressive results with the use of linear prediction models, but there is still room for improvements with nonlinear models. From this perspective, this thesis presents a variant of the SCOAL based on Extreme Learning Machines. Besides improving the accuracy, another important issue related to the development of RS is system scalability. In this sense, a parallel version of the SCOAL, based on OpenMP, was developed, aimed at minimizing the computational cost involved as prediction models are learned. Experiments using real-world datasets has shown that all proposed developments make SCOAL algorithm even more attractive for a variety of practical applications.

Key-words: Recommender Systems, Cold-start, Biclustering, Prediction models, Extreme Learning Machines.

LISTA DE ILUSTRAÇÕES

Figura 1 – Comparação das avaliações feitas pelos <i>Usuários 1, 2 e 5</i>	29
Figura 2 – Dados utilizados pelo SCOAL para a construção dos modelos de predição — adaptado de Deodhar e Ghosh (2007).	41
Figura 3 – (a) Solução encontrada pelo SCOAL depois de rearranjar as linhas e colunas da Tabela 1. Linhas e colunas são realocadas para outro bicluster sempre que tal operação possibilite uma melhoria na média de desempenho dos modelos de predição. As diferentes cores representam os biclusters induzidos. (b) Mapeamento de entrada-saída produzido pelo modelo de predição $\{r, c\}$. Os índices de linha e coluna são específicos para cada bicluster.	42
Figura 4 – Visão geral da abordagem proposta para lidar com o problema <i>cold-start</i> . . .	48
Figura 5 – Valores de NMAE para diferentes números de avaliações disponíveis — situações de <i>cold-start</i> parcial.	54
Figura 6 – Valores de NMAE para situações extremas de <i>cold-start</i>	55
Figura 7 – Acurácia da classificação obtida considerando que cada rótulo de <i>cluster</i> encontrado pelo SCOAL é um rótulo de classe para o problema <i>cold-start</i> . . .	56
Figura 8 – NMAE obtido na base Movielens para MK-KNN, NB, J48, Logistic, WP e DC. . .	57
Figura 9 – Arquitetura de uma rede SHLFNN.	60
Figura 10 – Saída do algoritmo SCOAL-ELM com $U = V = 2$	63
Figura 11 – Valores do Erro Médio Absoluto Normalizado (NMAE).	66
Figura 12 – Valores de precisão.	67
Figura 13 – Base Netflix	84
Figura 14 – Base Jester	84
Figura 15 – Base Movielens	85
Figura 16 – Base Música	89
Figura 17 – Tempos computacionais obtidos pelo SCOAL-LR com o <i>offload</i> automático da biblioteca MKL habilitado e desabilitado.	91

LISTA DE ALGORITMOS

Algoritmo 1 – Algoritmo SCOAL (adaptado de (DEODHAR; GHOSH, 2010)).	43
Algoritmo 2 – Algoritmo ELM (adaptado de (HUANG; ZHU; SIEW, 2006)).	61

LISTA DE CÓDIGOS-FONTE

Código-fonte 1 – Exemplo de funções MKL utilizadas na multiplicação de matrizes . . .	77
Código-fonte 2 – Exemplo de código em Armadillo C++ para a multiplicação de matrizes	78
Código-fonte 3 – Construção dos modelos de regressão	79
Código-fonte 4 – Atualização dos <i>clusters</i> de linha	80
Código-fonte 5 – Atualização dos <i>clusters</i> de coluna	81

LISTA DE TABELAS

Tabela 1 – Exemplo de matriz utilizada em um SR.	27
Tabela 2 – Similaridades entre usuários.	28
Tabela 3 – Predições utilizando fatoração de matrizes.	34
Tabela 4 – Base de dados usada para construir um modelo de classificação para situações extremas de <i>cold-start</i> . Os rótulos dos <i>clusters</i> obtidos do SCOAL tornam-se rótulos de classe, os quais são usados para construir um modelo de classificação capaz de encontrar os melhores modelos de predição para os novos usuários.	50
Tabela 5 – Tempo de CPU em segundos decorrido para cada método.	57
Tabela 6 – : Valores NMAE: média (desvio padrão). O texto em negrito refere-se a melhores resultados.	58
Tabela 7 – Valores de U e V adotados para gerar os biclusters.	64
Tabela 8 – Melhores valores obtidos para λ usando a metodologia baseada em Huang <i>et al.</i> (2012)	64
Tabela 9 – Resumo dos resultados obtidos para todas as medidas investigadas. Melhores resultados são destacados em negrito.	68
Tabela 10 – Número de vitórias para cada algoritmo considerando as medidas da Tabela 9.	68
Tabela 11 – Tempo computacional (em minutos): Macbook com 8 GB de memória e 4 núcleos Intel Core i7 2GHz.	69
Tabela 12 – Valores de U e V adotados para gerar os biclusters.	83
Tabela 13 – Tempos de execução em minutos.	86
Tabela 14 – <i>Speedup</i>	86
Tabela 15 – Eficiência Paralela.	87
Tabela 16 – Percentuais de utilização da CPU — SCOAL-LR.	88
Tabela 17 – Percentuais de utilização da CPU — base de dados Música (SCOAL-LR).	88
Tabela 18 – Medidas de desempenho — base de dados Música (SCOAL-LR).	89

LISTA DE ABREVIATURAS E SIGLAS

ACML	...	<i>AMD Core Math Library</i>
API		<i>Application Programming Interface</i>
BCC		<i>Bayesian co-clustering</i>
CELMCF	.	<i>Extreme Learning Machine Combining Matrix Factorization for Collaborative Filtering</i>
CLARE	..	<i>Cross-Level Association Rules</i>
CME		<i>Cluster with Minimum Error</i>
CUDA	...	<i>Compute Unified Device Architecture</i>
DC		<i>Dynamic Classification</i>
ELM		<i>Extreme Learning Machines</i>
ELM		<i>Extreme Learning Machines</i>
ELMR	...	<i>ELM based Recommendation</i>
EM		<i>Expectation Maximization</i>
FC		<i>Filtragem Colaborativa</i>
GFS		<i>Google File System</i>
GPU		<i>Graphics Processing Unit</i>
HDFS		<i>Hadoop Distributed File System</i>
HPC		<i>High Performance Computing</i>
ILP		<i>Instruction Level Parallelism</i>
LAPACK	.	<i>Linear Algebra Package</i>
LDA		<i>Latent Dirichlet Allocation</i>
MAE		<i>Mean Absolute Error</i>
MF-KNN	.	<i>Matrix Factorization K-Nearest Neighbors</i>
MIC		<i>Many Integrated Core</i>
MKL		<i>Math Kernel Library</i>
MLP		<i>Multilayer Perceptron</i>
MLP		<i>Multilayer Perceptron</i>
MPI		<i>Message Passing Interface</i>
MSE		<i>Mean Squared Error</i>
NMAE	...	<i>Normalized Mean Absolute Error</i>
NMF		<i>Non-negative Matrix Factorization</i>

NUMA ... *Non-uniform memory access*
 OS-ELM . *Online Sequential Learning for ELM*
 OWA *Ordered Weighted Averaging*
 PCI *Peripheral Component Interconnect*
 PIP *Proximity-Impact-Popularity*
 pLSA *probabilistic latent semantic analysis*
 RBC *Recomendação Baseada em Conteúdo*
 RC *Recomendação Colaborativa*
 RCF *Real-Time Collaborative Filtering*
 RH *Recomendação Híbrida*
 RMSE *Root Mean Squared Error*
 SCOAL .. *Simultaneous Co-clustering and Learning*
 SHLFNNs *Single Hidden Layer Feedforward Neural Networks*
 SIMD *Single Instruction, Multiple Data*
 SMP *Symmetric Multi-Processing*
 SPMD *Single Program Multiple Data*
 SR *Sistema de Recomendação*
 SVD *Singular Value Decomposition*
 UMA *Uniform Memory Access*
 WNMTF . *Weighted Nonnegative Matrix Tri-Factorization*
 WOT *Web Of Trust*
 WP *Weighted Prediction*

LISTA DE SÍMBOLOS

$\mathbf{Y}$ — Matriz de dados

y_{ij} — Elemento da matriz $\mathbf{Y}$

I — Quantidade de linhas da matriz $\mathbf{Y}$

J — Quantidade de colunas da matriz $\mathbf{Y}$

$\mathbf{U}$ — Matriz de características de usuários

$\mathbf{V}$ — Matriz de características de itens

$\mathbf{u}_i$ — i -ésimo vetor de características dos usuários

$\mathbf{v}_i$ — i -ésimo vetor de características dos itens

$\hat{y}_{ij}$ — Predição calculada para elemento y_{ij}

e_{ij} — Erro da predição para elemento y_{ij}

U — Quantidade de *clusters* de linha

V — Quantidade de *clusters* de colunas

$\mathbb{C}_u$ — Conjunto de *clusters* de linha

$\mathbb{C}_v$ — Conjunto de *clusters* de colunas

$\mathbf{x}_{ij}$ — Vetor que concatena os atributos dos usuários e de itens

$\hat{\boldsymbol{\beta}}_{rc}$ — Vetor de coeficientes de regressão do bicluster $\{r, c\}$

$\mathbf{X}_{rc}$ — Matriz que concatena os atributos de usuários e itens

$\mathbf{y}_{rc}$ — Vetor de respostas do bicluster $\{r, c\}$

$\mathbf{I}$ — Matriz Identidade

λ — Parâmetro que controla a quantidade de regularização

$\mathbf{C}_i^u$ — i -ésimo *cluster* de linha

$\mathbf{C}_j^v$ — j -ésimo *cluster* de coluna

r^* — Índice do *cluster* de linha que minimiza o MSE

$\mathbf{u}_{new}$ — Vetor de características do novo usuário

$\mathbf{x}_{new}$ — Vetor que concatena o vetor de características do novo usuário e vetor de características

do item

$\hat{y}_{new,n}$ — Predição calculada para o n-ésimo item do novo usuário

$P(Class \mid \mathbf{u}_{new})$ — Probabilidade condicional da classe creditada para o novo usuário

$\mathbf{w}$ — Matriz de pesos da camada intermediária de uma SHLFNN

$\boldsymbol{\beta}$ — Matriz de pesos da camada de saída de uma SHLFNN

$\mathbf{H}$ — Matriz de saída de uma ELM

$\mathbf{H}^\dagger$ — Matriz inversa generalizada de Moore-Penrose

$\hat{\boldsymbol{\beta}}$ — Vetor de pesos calculados da ELM

$\mathbb{Q}$ — Conjunto de treinamento

$f(\cdot)$ — Função de ativação

$\mathcal{N}$ — Distribuição normal

$O(\cdot)$ — Custo computacional

$\mathbf{X}$ — Matriz de dados de treinamento

SUMÁRIO

1	INTRODUÇÃO	20
1.1	Motivações e Hipóteses	22
1.2	Organização da Tese	24
2	SISTEMAS DE RECOMENDAÇÃO	26
2.1	Considerações Iniciais	26
2.2	Recomendação Colaborativa	26
2.2.1	<i>Recomendação Baseada no Vizinho mais Próximo</i>	27
2.2.2	<i>Recomendação Baseada em Item</i>	29
2.3	Recomendação Baseada em Conteúdo	30
2.4	Recomendação Híbrida	30
2.5	Recomendações <i>on-line</i>	31
2.5.1	<i>Fatoração de Matrizes</i>	32
2.5.2	<i>Biclusterização</i>	34
2.5.2.1	<i>Biclusterização e SR</i>	34
2.6	Medidas de avaliação	35
2.7	Considerações Finais	38
3	AGRUPAMENTO DE DADOS E APRENDIZADO DE FUNÇÕES DE REGRESSÃO	39
3.1	Considerações Iniciais	39
3.2	Abordagens baseadas em dividir e conquistar	39
3.3	Algoritmo SCOAL	40
3.4	Predição utilizando modelos lineares	42
3.5	Predição utilizando modelos não lineares	43
3.6	Considerações Finais	44
4	SCOAL E O PROBLEMA <i>COLD-START</i>	45
4.1	Considerações Iniciais	45
4.2	Trabalhos relacionados	45
4.2.1	<i>Fatores latentes</i>	45
4.2.2	<i>Redes sociais</i>	46
4.2.3	<i>Novas medidas de similaridade</i>	46
4.2.4	<i>Regras de associação</i>	47

4.3	Abordagem proposta	47
4.3.1	<i>Cluster com erro mínimo</i>	49
4.3.2	<i>Situações extremas de cold-start</i>	49
4.3.3	<i>Predição Ponderada</i>	50
4.3.4	<i>Classificação Dinâmica</i>	50
4.4	Avaliação Experimental	51
4.4.1	<i>Bases de dados</i>	51
4.4.1.1	<i>Netflix</i>	51
4.4.1.2	<i>Jester</i>	52
4.4.1.3	<i>Movielens 100k</i>	52
4.4.2	<i>Parametrização dos experimentos</i>	52
4.4.3	<i>Resultados obtidos com cold-start parcial</i>	53
4.4.4	<i>Resultados obtidos em situações extremas de cold-start</i>	55
4.4.4.1	<i>Comparações com o método MF-KNN</i>	56
4.5	Análise sobre os resultados obtidos	57
4.6	Considerações Finais	58
5	AGRUPAMENTO DE DADOS E APRENDIZADO DE FUNÇÕES DE REGRESSÃO BASEADO EM MÁQUINAS DE APRENDIZADO EXTREMO	59
5.1	Considerações Iniciais	59
5.2	Máquinas de Aprendizado Extremo	59
5.3	Sistemas de Recomendação e ELMs	62
5.4	Algoritmo SCOAL-ELM	62
5.4.1	<i>Avaliação Experimental</i>	63
5.4.1.1	<i>Parametrização dos Experimentos</i>	64
5.4.1.2	<i>Resultados Obtidos</i>	65
5.5	Considerações Finais	69
6	AGRUPAMENTO DE DADOS E APRENDIZADO DE FUNÇÕES DE REGRESSÃO EM PARALELO UTILIZANDO OPENMP	70
6.1	Considerações Iniciais	70
6.2	Tecnologias para programação paralela	71
6.2.1	<i>Arquitetura multi-core</i>	72
6.2.2	<i>Many Integrated Core</i>	72
6.2.3	<i>OpenMP</i>	73
6.2.4	<i>OpenACC</i>	74
6.2.5	<i>OpenCL</i>	74
6.2.6	<i>GPUs e CUDA</i>	75
6.2.7	<i>MPI</i>	75

6.2.8	<i>MapReduce</i>	76
6.3	<i>Algoritmo Proposto</i>	76
6.3.1	<i>Construir um modelo de regressão para cada bicluster</i>	79
6.3.2	<i>Atualização dos clusters de linha</i>	80
6.3.3	<i>Atualização dos clusters de colunas</i>	81
6.4	<i>Avaliação Experimental</i>	82
6.4.1	<i>Parametrização</i>	82
6.4.2	<i>Bases de Dados</i>	82
6.4.3	<i>Resultados Obtidos</i>	83
6.4.3.1	<i>MKL e Xeon Phi</i>	89
6.5	<i>Considerações Finais</i>	91
7	CONCLUSÕES E TRABALHOS FUTUROS	93
7.1	<i>Principais Contribuições</i>	93
7.2	<i>Trabalhos futuros</i>	95
7.2.1	<i>Ensemble de regressores</i>	95
7.2.2	<i>Aumentar a eficiência do paralelismo</i>	95
7.2.3	<i>Instâncias multicluster</i>	96
	REFERÊNCIAS	97

INTRODUÇÃO

Grande parte dos usuários acostumados a comprar produtos pela Internet já teve contato com um Sistema de Recomendação (SR)¹. Os principais sítios de comércio eletrônico já utilizam algum tipo de SR no intuito de auxiliar seus clientes no processo de compra de maneira mais eficiente. Um SR pode ser definido como um conjunto de ferramentas de software e técnicas que fornecem sugestões de itens de interesse para um usuário (RICCI *et al.*, 2011). Assim sendo, um sistema de software que determina quais itens devem ser mostrados para um cliente é um SR (BURKE, 2007; RESNICK; VARIAN, 1997; JANNACH *et al.*, 2010). Usualmente, os sítios de comércio eletrônico disponibilizam para seus clientes uma lista dos “produtos mais vendidos”, o que de certa forma pode ser considerado um tipo de SR impessoal para atrair consumidores. Muito embora tais sistemas de fato atraiam um número razoável de clientes, existe ainda uma grande parcela que não se sente atraída a adquirir um produto devido tão somente à sua popularidade. Atualmente, ao se fazer uma busca por um determinado produto de interesse, é comum em algum lugar do sítio aparecer um aviso intitulado “Pessoas que compraram este item também compraram:...” seguido por uma relação de itens que supostamente serão de interesse do usuário. Tais sistemas utilizam um técnica conhecida por recomendação colaborativa. Se o usuário já é um cliente habitual, o sistema gera uma outra lista, contendo esta recomendações de itens personalizados, na qual cada cliente enxerga uma lista diferente de acordo com as suas preferências. A recomendação personalizada, no entanto, requer que o SR conheça um pouco de cada cliente, sendo necessário manter um perfil do usuário ou modelo do usuário que, por exemplo, armazena suas preferências (JANNACH *et al.*, 2010). O interesse por parte das empresas no desenvolvimento de novos SR pode ser visto como uma via de mão dupla onde tanto os clientes saem beneficiados - recebendo informações úteis sobre itens de interesse - como também os negócios da empresa - possibilitando um aumento nas vendas e a fidelização do cliente (SOTOMAYOR; CARTHY; DUNNION, 2007). Apesar de ser uma área

¹ Por simplicidade, a sigla SR fará referência tanto a Sistema de Recomendação quanto a Sistemas de Recomendação, sendo que o significado em cada parte do texto será facilmente identificável a partir do contexto.

de pesquisa relativamente nova na qual sistemas como Tapestry (GOLDBERG *et al.*, 1992), GroupLens (RESNICK *et al.*, 1994), Ringo (SHARDANAND; MAES, 1995) e Bellcore Video Recommender (HILL *et al.*, 1995) surgiram no início dos anos 90, nos últimos anos o interesse em SR aumentou drasticamente (RICCI *et al.*, 2011), conforme indicam os seguintes fatos:

- SR ocupam um papel importante em empresas de Internet altamente conceituadas, tais como: Amazon.com, YouTube, Netflix, Last.fm, Tripadvisor e Internet Movie Database (IMDb). Em particular, a empresa Netflix, em outubro de 2006, disponibilizou uma base de dados contendo 100 milhões de avaliações de filmes feitas por usuários anônimos e desafiou pesquisadores da área de mineração de dados e aprendizado de máquina a desenvolverem sistemas que pudessem superar o seu SR conhecido por Cinematch (BENNETT; LANNING, 2007). Para tal, foi criada uma competição anual com o objetivo de desenvolver um SR que superasse em 10% a precisão do Cinematch. A equipe que conseguisse o intento seria agraciada com um prêmio de um milhão de dólares. Em setembro de 2009, a equipe BellKor's Pragmatic Chaos ganhou o prêmio;
- O aumento de conferências e workshops relacionados à área de SR, tais como: o ACM Recommender Systems (RecSys) criado em 2007, além das sessões especiais dedicadas à área que são frequentemente inseridas nas conferências mais tradicionais na área de bases de dados, sistemas de informação e sistemas adaptativos. Entre essas conferências, destaca-se a ACM (SIGIR) Grupo de Interesse Especial em Recuperação de Informação-, Modelagem de Usuário, Adaptação e Personalização (UMAP) e o Grupo da ACM de Interesse Especial sobre Gestão de Dados (SIGMOD);
- Várias edições especiais sobre SR foram publicadas em periódicos, tais como: AI Communications (2008); IEEE Intelligent Systems (2007); International Journal of Electronic Commerce (2006); International Journal of Computer Science and Applications (2006); ACM Transactions on Computer-Human Interaction (2005) e ACM Transactions on Information Systems (2004).

Para que os SR sejam capazes de fazer recomendações personalizadas para seus usuários, eles necessitam de informações específicas de cada um. Tais informações podem ser obtidas de forma explícita, por exemplo no momento em que um novo usuário se cadastra no sistema, informando quais são suas preferências, ou de maneira implícita, monitorando o comportamento do usuário no sistema. Além de informações demográficas de seus usuários (*e.g.*, gênero, idade, profissão, endereço) atualmente fontes de informação externas vêm sendo utilizadas no intuito de melhorar a qualidade da recomendação do sistema. Um típico exemplo é a Internet, onde grande parte dos usuários tece comentários de forma espontânea sobre seus gostos e preferências nas redes sociais (*e.g.*, *posts* em *blogs*, *tweets*, *followers*, *followees*).

A seleção de quais fontes de dados serão utilizadas é determinada pela técnica de recomendação empregada no sistema. Algumas técnicas se baseiam no hábito de compra do

usuário e de opiniões de uma comunidade de usuários para fazer as predições de quais itens o usuário do sistema provavelmente ficaria interessado - sem levar em conta nenhuma informação do usuário em si e nem tampouco dos itens propriamente ditos. Outras técnicas são muito mais dependentes de conhecimento prévio, utilizando por exemplo um perfil de usuário e/ou características específicas dos itens para que o sistema possa fazer suas recomendações. Tais técnicas são classificadas como :

- **Recomendação colaborativa:** a recomendação de um item para um determinado usuário é feita com base em opiniões que outros usuários associam ao item. Mantendo um histórico das avaliações feitas pelos usuários, é possível inferir usuários com gostos semelhantes. Para fazer a recomendação de um novo item, o sistema identifica usuários cujos históricos de avaliações sejam semelhantes, e que já tenham avaliado o item em questão. Caso a avaliação seja positiva, o item então será recomendado.
- **Recomendação baseada em conteúdo:** nessa abordagem as informações existentes no perfil do usuário ou as características dos itens previamente comprados pelo usuário são utilizadas, procurando estabelecer uma correlação com novos itens ainda desconhecidos dos usuários para que sejam recomendados.
- **Recomendação híbrida:** As recomendações híbridas combinam as informações obtidas com a recomendação colaborativa, recomendação baseada em conteúdo e outras fontes de informação (*e.g.*, demográficas e redes sociais), procurando extrair o que cada uma oferece.

O desenvolvimento de novos algoritmos e técnicas de recomendação, tem possibilitado a implementação de SR para diferentes domínios. Numa recente pesquisa sobre o estado da arte dos SR (LU *et al.*, 2015), foram selecionados e analisados 104 artigos sobre desenvolvimento de SR, classificando-os em oito diferentes áreas de aplicação: *e-government*, *e-business*, *e-commerce/e-shopping*, *e-library*, *e-learning*, *e-tourism*, *e-resource services*, and *e-group activities*. Os critérios de seleção estabelecidos foram: o fator de impacto dos periódicos e conferências, número de citações, capítulos em livros e a data de publicação, sendo esta estipulada nos últimos 24 meses o que, consequentemente, comprova ser uma importante frente de pesquisa ativa.

1.1 Motivações e Hipóteses

Como será abordado em maiores detalhes no Capítulo 2, uma taxonomia amplamente utilizada divide as técnicas de recomendação baseadas em filtragem colaborativa em duas categorias (BOBADILLA *et al.*, 2013):

- **Técnicas baseadas em memória:** se utilizam de todo histórico de avaliações dos usuários para fazer suas recomendações, as quais são calculadas utilizando medidas de similaridades

para obter a distância entre dois usuários, ou dois itens, baseadas em cada uma das suas avaliações.

- **Técnicas baseadas em modelos:** baseiam-se num modelo compacto inferido a partir dos dados, o qual é utilizado pelo sistema para fazer a recomendação de novos itens. Dentre as técnicas mais utilizadas no aprendizado do modelo destacam-se: classificadores Bayesianos, redes neurais artificiais, computação evolutiva, fatoração de matrizes e agrupamento de dados.

Grande parte dos domínios de aplicações dos SR necessitam que as recomendações sejam geradas em tempo real. A grande quantidade de dados envolvidos em problemas de recomendação inviabiliza a utilização das técnicas baseadas em memória em tais domínios devido ao seu alto custo computacional. Em contrapartida, é improvável que todas as recomendações possam ser bem representadas por um único modelo inferido dessa enorme massa de dados. Geralmente, é melhor fazer uso de múltiplos modelos que representem as recomendações de um subconjunto de usuários para um subconjunto de itens. Por tais razões, esta tese tem como objeto de estudo a utilização de um algoritmo capaz de combinar, de forma inteligente, recomendação colaborativa e informações demográficas, gerando múltiplos modelos de aprendizado.

Um dos principais problemas que afetam a maioria dos SR é conhecido por problema *cold-start*. O *cold-start* ocorre quando não é possível fazer recomendações confiáveis devido à falta de um histórico de avaliações sobre o novo item ou um novo usuário incorporado no sistema. Levando isso em consideração, a primeira hipótese desta tese é definida como:

- **Hipótese 1:** É possível desenvolver uma variante do algoritmo SCOAL para lidar apropriadamente com o problema da ausência de informações de avaliações para novos usuários/itens (problema *cold-start*).

Tal variante deve fornecer uma acurácia preditiva que estimule seu uso na prática. Neste contexto, espera-se obter resultados melhores, ou ao menos competitivos, com aqueles obtidos por metodologias já consolidadas.

Apesar da versão original do SCOAL não lidar com o problema do *cold-start*, é necessário reconhecer que tal algoritmo mostrou ser eficiente e versátil nos experimentos realizados por [Deodhar e Ghosh \(2010\)](#), podendo ser utilizado na resolução de uma gama de problemas de classificação e regressão. Entretanto, a maioria dos experimentos foi feita utilizando modelos lineares apenas. Nos experimentos realizados com o SCOAL utilizando modelos não lineares, baseados em redes neurais, os resultados obtidos foram inferiores aos da versão utilizando modelos de regressão linear. Baseada neste fato, a segunda hipótese desta tese é definida como:

- **Hipótese 2:** Pode-se melhorar o algoritmo SCOAL no tocante a modelos não lineares, deixando-o mais versátil e competitivo em termos de capacidade preditiva, comparativa-

mente a outros algoritmos que representam o estado-da-arte no assunto (*e.g.* biclusterização e fatoração matricial), para que possa ser utilizado como o componente principal de SR variados (*e.g.* para filmes, músicas e redes sociais).

Além da capacidade de predição, um outro fator que atualmente deve ser levado em consideração no desenvolvimento de SR tem a ver com a escalabilidade do sistema. SR têm sido empregados em centenas de diferentes sítios de *e-commerce* onde as recomendações devem ser feitas *on-line*. Centenas ou até mesmo milhares de avaliações são feitas por segundo, sem contar com a chegada de novos usuários e novos itens que são catalogados no sistema a todo instante, produzindo um volume enorme de dados. Levando em consideração este cenário, formula-se a terceira hipótese desta tese:

- **Hipótese 3:** É possível desenvolver uma versão paralelizada do algoritmo SCOAL que seja suficientemente eficiente, do ponto de vista computacional, a ponto de poder ser incorporada como um componente de SR para problemas reais, os quais usualmente envolvem grandes bases de dados.

1.2 Organização da Tese

O restante desta tese está organizado da seguinte forma:

Capítulo 2 - Sistemas de Recomendação: Apresenta os conceitos básicos dos Sistemas de Recomendação (SR) e as principais técnicas de recomendação utilizadas, bem como os aspectos positivos e negativos de cada uma. O capítulo também descreve os métodos utilizados para avaliar a acurácia dos SR.

Capítulo 3 - Agrupamento de Dados e Aprendizado de Funções de Regressão: O funcionamento do algoritmo SCOAL, a partir do qual são desenvolvidas as contribuições dessa tese, é apresentado de forma detalhada neste capítulo.

Capítulo 4 - SCOAL e o problema *cold-start* : Apresenta as funcionalidades que foram desenvolvidas e implementadas numa variante do algoritmo SCOAL que é capaz de lidar com o problema *cold-start*. A abordagem proposta foi avaliada em três bases de dados frequentemente utilizadas na literatura (BOBADILLA *et al.*, 2013) apresentando um resultado superior quando comparado com outros algoritmos que representam o estado-da-arte. O capítulo também apresenta um levantamento bibliográfico realizado sobre diversos trabalhos sobre este tema. As contribuições deste capítulo foram publicadas em Pereira e Hruschka (2015).

Capítulo 5 - Agrupamento de Dados e Aprendizado de Funções de Regressão baseado em Máquinas de Aprendizado Extremo: O algoritmo SCOAL obtém excelentes resultados com modelos de predição lineares, mas ainda há espaço para melhorias com modelos de predição não lineares (DEODHAR; GHOSH, 2010). Neste capítulo é apresentado o algoritmo SCOAL-ELM, uma versão do algoritmo SCOAL que utiliza modelos de predição não lineares

baseado em Máquinas de Aprendizado Extremo (*Extreme Learning Machines* (ELM)) (HUANG; ZHU; SIEW, 2004). Os resultados obtidos pelo SCOAL-ELM demonstram que o algoritmo pode ser útil em diversos cenários, sendo competitivo com o já consolidado algoritmo SCOAL baseado em modelos lineares.

Capítulo 6 - Agrupamento de Dados e Aprendizado de Funções de Regressão em Paralelo utilizando OpenMP: Este capítulo apresenta uma versão paralelizada do algoritmo SCOAL cuja implementação utiliza OpenMP, um modelo de programação portátil e escalável que proporciona aos desenvolvedores uma interface simples e flexível para o desenvolvimento de aplicações paralelas.

Capítulo 7 - Conclusões e Trabalhos Futuros: Este capítulo sumariza as principais conclusões originadas desta tese, além de apresentar possíveis linhas de pesquisa para trabalhos futuros.

SISTEMAS DE RECOMENDAÇÃO

2.1 Considerações Iniciais

Um Sistema de Recomendação (SR) processa diversas fontes de dados a fim de construir suas recomendações. Os dados em questão definem características dos itens, informações sobre o usuário do sistema e até mesmo transações realizadas pelo usuário. A seleção de quais fontes de dados serão utilizadas é determinada pela técnica de recomendação empregada no sistema. Algumas técnicas se baseiam no hábito de compra do usuário e de opiniões de uma comunidade de usuários para fazer as predições de quais itens o usuário do sistema provavelmente ficaria interessado - sem levar em conta nenhuma informação do usuário em si nem tampouco dos itens propriamente ditos. Outras técnicas são muito mais dependentes de conhecimento prévio, utilizando por exemplo um modelo de usuário e/ou características específicas dos itens para que o sistema possa fazer suas recomendações. Seja qual for o caso, os dados utilizados por um SR referem-se a três tipos de objetos: usuários, itens e transações (JANNACH *et al.*, 2010). Um exemplo para ilustrar um cenário de aplicação de um SR é representado pela Tabela 1. O exemplo apresenta uma matriz $\mathbf{Y}_{I \times J}$ na qual cada elemento y_{ij} armazena um determinado valor (*e.g.*, uma avaliação) atribuída pelo usuário i ao item j (itens podem ser filmes, músicas, produtos, etc.). Tendo-se em mente o cenário de aplicação fictício representado pela Tabela 1, as próximas seções apresentam as técnicas mais utilizadas por sistemas de recomendação. Vale notar que essa matriz tem por objetivo apenas ilustrar o funcionamento de algumas técnicas, não sendo representativa das aplicações práticas de um SR, nas quais as dimensões são muito maiores e há uma quantidade significativa de valores ausentes.

2.2 Recomendação Colaborativa

Sistemas baseados em Recomendação Colaborativa (RC) — ou Filtragem Colaborativa (FC) (ADOMAVICIUS; TUZHILIN, 2005) — recomendam itens (documentos, produtos, filmes,

Tabela 1 – Exemplo de matriz utilizada em um SR.

	item 1	item 2	item 3	item 4	item 5
Usuário 1	5	3	4	4	?
Usuário 2	3	1	2	3	3
Usuário 3	4	3	4	3	4
Usuário 4	3	3	1	5	4
Usuário 5	1	5	5	2	1

etc.) baseados em avaliações que outros usuários fizeram sobre a eles. Os usuários compartilham informações e impressões sobre determinado item, atribuindo uma pontuação que poderá servir como referência a um outro usuário que ainda não tenha adquirido o item em questão. Esses tipos de sistemas têm sido explorados por vários anos, e suas vantagens, seus desempenhos e suas limitações são atualmente bem conhecidas (ADOMAVICIUS; TUZHILIN, 2005; BURKE, 2007; BOBADILLA *et al.*, 2013). Vale mencionar que várias técnicas baseadas em RC têm sido propostas e avaliadas nos últimos anos, obtendo resultados satisfatórios em diversas aplicações comerciais (MEO; QUATTRONE; URSINO, 2008; ZHENG; LI, 2011; KWON; HONG, 2011; O'CONNOR *et al.*, 2001; LU *et al.*, 2015). Os dados de entrada utilizados em abordagens baseadas em RC são estruturados na forma de uma matriz de avaliação *usuário-item*, conforme ilustrado na Tabela 1 e, após o processamento, a predição do SR é feita na forma de um valor numérico (geralmente em valores ordinais) indicando o grau de satisfação do usuário com relação a um determinado item. Adicionalmente, uma lista com os n itens melhor ranqueados e que ainda não foram adquiridos pelo usuário são apresentados. A RC se baseia nas seguintes premissas: (a) “Se usuários tiveram gostos similares no passado, então eles terão gostos similares no futuro” e (b) “As preferências do usuário permanecem estáveis e consistentes no decorrer do tempo” (JANNACH *et al.*, 2010). As técnicas utilizadas na RC são frequentemente classificadas em dois tipos:

- **Baseadas em modelo:** Fazem uso de uma coleção de avaliações feitas por usuários para aprender um modelo, que é então utilizado para fazer as predições. Algumas abordagens que utilizam esta técnica são abordadas em maiores detalhes na Seção 2.5.
- **Baseadas em memória:** realizam suas predições tomando por base uma coleção inteira de itens previamente avaliados pelos usuários. Duas abordagens distintas são baseadas em memória, sendo uma conhecida por recomendação baseada no usuário vizinho mais próximo (*user-based nearest neighbor recommendation*) e a outra conhecida por recomendação baseada em item (*item-based recommendation*).

2.2.1 Recomendação Baseada no Vizinho mais Próximo

Dados um conjunto de avaliações e a identificação do usuário ativo como entrada de dados, um SR baseado no vizinho mais próximo tem por objetivo identificar outros usuários

(vizinhos mais próximos) que possuam preferências similares àsquelas do usuário ativo. Dessa forma, para todo item j que o usuário ativo ainda não tenha avaliado, uma predição é feita com base nas avaliações feitas para i pelos usuários vizinhos mais próximos. Analisando o exemplo da Tabela 1, pode-se notar que o *Usuário 1* ainda não explicitou sua opinião a respeito do *Item 5* — visto que o valor de cada avaliação segue uma escala entre 1 e 5, e itens sem avaliação são representados por “?”. Tomando por base este exemplo, a tarefa do sistema seria prever o grau de satisfação do *Usuário 1* pelo *Item 5*. Caso o valor da predição seja próximo a 5, este item poderia ser adicionado a lista de itens recomendados para o *Usuário 1*. Para tanto, a recomendação baseada no vizinho mais próximo segue basicamente dois passos:

- Passo1: Encontrar, na coleção de dados, quais usuários possuem avaliações de itens similares ao *Usuário 1*.
- Passo2: De posse do subconjunto de usuários obtido no Passo 1, calcular a predição para o *Item 5* ponderando as avaliações dos usuários vizinhos mais próximos.

Para a execução do Passo 1 (determinar um conjunto de usuários similares) uma medida de similaridade comumente utilizada é a bem conhecida correlação de Pearson (KONSTAN *et al.*, 1997). A correlação de Pearson mede a correlação linear entre dois vetores de avaliação. O intervalo de valores varia entre -1 (forte correlação negativa) à $+1$ (forte correlação positiva). Uma outra medida utilizada é a correlação restrita de Pearson — (*Constrained Pearson*)(SHARDANAND; MAES, 1995) — uma versão levemente modificada da correlação de Pearson que permite que somente pares de avaliações do mesmo lado (positivo ou negativo) contribuam para a correlação. Por exemplo, tomando-se por base os dados da Tabela 1, as similaridades do *Usuário 1* relativamente aos demais usuários (*Usuário 2*, ..., *Usuário 5*) são representadas na Tabela 2.

Tabela 2 – Similaridades entre usuários.

	correlação de Pearson	correlação restrita de Pearson
<i>Usuários 1 e 2</i>	0,85	0,18
<i>Usuários 1 e 3</i>	0,70	0,86
<i>Usuários 1 e 4</i>	0	0
<i>Usuários 1 e 5</i>	-0,79	-0,33

Observa-se que pela correlação de Pearson os *Usuários 2* e *3* são bastante semelhantes ao usuário em questão. Já para a correlação restrita de Pearson, o *Usuário 3* apresenta maior similaridade. Cabe ressaltar que a Correlação de Pearson considera o fato de que os usuários diferem com relação a como eles interpretam a escala de avaliação. Alguns usuários tendem a dar apenas avaliações elevadas, enquanto outros usuários nunca irão dar nota 5 para qualquer item. Ou seja, embora os valores absolutos das avaliações do *Usuário 1* e *Usuário 2* sejam muito

diferentes, existe uma correlação linear entre suas avaliações e, portanto, a similaridade entre os usuários é detectada. Tal fato pode ser constatado na representação visual da Figura 1, que ilustra tanto a semelhança entre o *Usuário 1* e *Usuário 2* como as diferenças nas avaliações entre o *Usuário 1* e o *Usuário 5*.

Figura 1 – Comparação das avaliações feitas pelos *Usuários 1, 2 e 5*.

2.2.2 Recomendação Baseada em Item

Apesar da abordagem baseada no vizinho mais próximo ser aplicada com sucesso em muitas aplicações comerciais, em domínios com elevado tráfego de transações, como em grandes sítios de comércio eletrônico onde a quantidade de itens de catálogo são da ordem de milhões e o número de usuários é também elevado, o processo de procurar um número de vizinhos similares eleva o custo computacional de tal maneira que inviabiliza realizar os cálculos necessários para se fazer previsões *on-line*. Um famoso SR baseado em item é o da Amazon.com, que em 2003 já dispunha de 29 milhões de usuários cadastrados e milhões de itens de catálogo (LINDEN; SMITH; YORK, 2003). A ideia principal de um SR baseado em item é fazer previsões calculando similaridades entre itens ao invés de calcular similaridades entre usuários, conforme realizado por um SR baseado em vizinhos mais próximos. Para calcular a similaridade entre itens, uma medida bastante utilizada é a do cosseno, para a qual dois itens são tratados como vetores m -dimensionais, e a similaridade entre os dois vetores pode ser medida calculando-se o cosseno do ângulo entre eles. A medida do cosseno é amplamente utilizada na área de Filtragem de Informação e Mineração de Textos (THORNLEY, 2012; REZENDE, 2003).

Novamente usando como base o exemplo da Tabela 1, para fazer a previsão do valor do *Item 5* para o *Usuário 1* o primeiro passo seria procurar itens similares ao *Item 5* utilizando a medida do cosseno. Note que, nesse caso, as similaridades são definidas como uma função das colunas da matriz. Após a escolha dos itens similares, basta verificar como o *Usuário 1* avaliou estes itens e então o valor em questão seria estimado, por exemplo, usando o valor médio das avaliações feitas pelo *Usuário 1* para os itens similares.

2.3 Recomendação Baseada em Conteúdo

Sistemas que utilizam a Recomendação Baseada em Conteúdo (RBC) analisam um conjunto de documentos e/ou uma descrição de itens previamente avaliados pelo usuário para construir um modelo (ou perfil) de usuário — baseado nas características dos itens avaliados. As informações do perfil do usuário também podem ser obtidas pelo próprio usuário de forma a criar uma representação estruturada de seus interesses, que será então utilizada pelo sistema para futuras recomendações de novos itens. O processo de recomendação consiste basicamente em fazer o *casamento* entre os atributos do perfil do usuário com os atributos dos itens. Como resultado, obtém-se um critério de relevância (geralmente variando entre 0 e 1) que representa o grau de interesse do usuário pelo item. Um perfil de usuário que reproduz fielmente seus interesses é de grande vantagem para a eficácia na recomendação de novos itens. Uma técnica muito comum neste tipo de abordagem é a indexação de frequência de termos (*term frequency indexing*) (CAZELLA; NUNES; REATEGUI, 2010). Neste tipo de indexação, informações dos itens e do perfil dos usuários são descritas por vetores com uma dimensão para cada ocorrência de uma determinada palavra. Cada posição do vetor é a frequência com que a respectiva palavra ocorre em um item ou no perfil do usuário. Sendo assim, os vetores dos itens que estão próximos aos vetores do perfil do usuário são considerados aqueles mais relevantes. Após a construção dos vetores de características do item e do perfil do usuário, o critério de relevância pode ser obtido através da medida de similaridade do cosseno. A RBC é mais indicada para a recomendação de itens textuais, para os quais o conteúdo é geralmente descrito com *palavras-chave*.

2.4 Recomendação Híbrida

As técnicas apresentadas nas Seções 2.2 e 2.3 naturalmente possuem vantagens e desvantagens dependendo do domínio de aplicação. Por exemplo, a RC necessita de uma base significativa de avaliações para que as predições sejam feitas com sucesso. Neste contexto, apesar de existirem milhares, ou até mesmo milhões, de itens disponíveis, (parte dos) usuários somente fazem a avaliação de um item após adquiri-lo, desta forma tornando a matriz de avaliação (Tabela 1) muito esparsa. Um outro problema inerente da RC é o *cold-start*, que é caracterizado no momento em que um novo item é adicionado ao catálogo de itens e o sistema não tem como recomendá-lo, visto que as recomendações são feitas com base nas avaliações prévias dos usuários. Na RBC, o fato do SR somente recomendar itens altamente relacionados com o perfil do usuário limita o SR a fazer recomendações de itens que de algum modo fazem parte da experiência do usuário, dessa forma impedindo que o usuário seja exposto a novos itens potencialmente interessantes para o seu perfil. Este problema é conhecido por super-especialização (ADOMAVICIUS; TUZHILIN, 2005). O *cold-start* também afeta a RBC, pois a acurácia do sistema depende de um número considerável de avaliações feitas pelo usuário para que seu perfil esteja bem estruturado. Neste sentido, a tendência é que um usuário novo não seria capaz de

receber recomendações precisas. A Recomendação Híbrida (RH) procura combinar os pontos fortes da RC e da RBC, visando criar SR que possam melhor atender às necessidades do usuário. As diferentes formas de fazer a combinação dessas duas técnicas podem ser categorizadas de acordo com as seguintes abordagens (ADOMAVICIUS; TUZHILIN, 2005):

1. Implementar a recomendação colaborativa e a baseada em conteúdo separadamente, posteriormente combinando suas predições;
2. Incorporar algumas características da abordagem colaborativa na abordagem baseada em conteúdo;
3. Incorporar algumas características da abordagem baseada em conteúdo na abordagem colaborativa;
4. Construir um modelo unificado, incorporando as características de ambas as abordagens e de tal forma que o desempenho do SR possa ser otimizado.

Na análise feita por Burke (2007) sobre SR híbridos, é proposta uma taxonomia de diferentes classes de algoritmos de recomendação. O autor apresenta o *EntreeC*, um SR Híbrido baseado em RC e RBC para fazer a recomendação de restaurantes. Vários outros trabalhos na literatura investigam abordagens híbridas— *e.g.*, (BASU *et al.*, 1998; CONDLIFF *et al.*, 1999; SOBOROFF; NICHOLAS, 1999; POPESCU; PENNOCK; LAWRENCE, 2001; SCHEIN *et al.*, 2002; PORCEL *et al.*, 2012; SHINDE; KULKARNI, 2012).

2.5 Recomendações *on-line*

Além dos problemas relacionados com matrizes esparsas e *cold-start*, um outro fator muito importante é a questão da escalabilidade do sistema. Em um sítio de comércio eletrônico, por exemplo, as recomendações devem ser feitas de forma *on-line*. Em tais domínios, envolvendo milhares ou milhões de itens e usuários, técnicas abordadas anteriormente (*e.g.*, Subseções 2.2.1 e 2.2.2) comprometem a escalabilidade do sistema, devido ao alto custo computacional envolvido no cálculo da recomendação. Por outro lado, nas abordagens baseadas em modelos, os dados são primeiramente utilizados *off-line* para o aprendizado do modelo. Após o aprendizado do modelo, as predições do sistema são feitas em tempo real, com base no modelo gerado. As abordagens baseadas em modelo permitem boa escalabilidade em domínios de aplicação nos quais existam milhões de itens e usuários (*e.g.*, na Amazon.com).

Esta seção apresenta duas técnicas que têm sido recentemente aplicadas com sucesso em sistemas de recomendação. A subseção 2.5.1 apresenta um modelo conhecido por Fatoração de Matrizes, cujas predições se baseiam na extração de fatores latentes de usuários e de itens, caracterizando-os por meio de vetores inferidos a partir da base de dados. Uma outra técnica amplamente utilizada na área de mineração de dados, conhecida por agrupamento de dados

(*clustering*), tem sido utilizada também em sistemas de recomendação. (UNGAR; FOSTER, 1998; PHAM *et al.*, 2011; RONGFEI; MAOZHONG; CHAO, 2010; KIM; PARK; YANG, 2008). A subseção 2.5.2 introduz o conceito de biclusterização — *biclustering* ou *coclustering* (MADEIRA; OLIVEIRA, 2004; CHO, 2008), que se refere a uma classe distinta de algoritmos de agrupamento que executam simultaneamente o agrupamento de linhas e colunas. Algoritmos de biclusterização têm sido utilizados em vários domínios (MADEIRA; OLIVEIRA, 2004; FRANÇA, 2010; MIMAROGLU; UEHARA, 2007; SYMEONIDIS *et al.*, 2008). No contexto de sistemas de recomendação, tais algoritmos são empregados para simultaneamente agrupar usuários e itens que possuam características similares.

2.5.1 Fatoração de Matrizes

Abordagens baseadas em fatoração de matrizes não negativas — *Non-negative Matrix Factorization* (NMF) (LEE; SEUNG, 1999)—tornaram-se populares nos últimos anos por combinarem boa escalabilidade com acurácia preditiva. Além disso, elas oferecem muita flexibilidade para modelar várias aplicações reais (KOREN; BELL; VOLINSKY, 2009). A fatoração de matrizes está intimamente relacionada com a decomposição em valores singulares — *Singular Value Decomposition* (SVD)(GOLUB; LOAN, 2012) — que é uma técnica tradicionalmente empregada na identificação de fatores semânticos na área de recuperação de informação (FURNAS *et al.*, 1988). Modelos de predição baseados em fatoração de matrizes não negativas mapeiam usuários e itens em vetores k -dimensionais (onde k representa a quantidade de fatores latentes) de tal forma que as interações usuário-item são modeladas com base no produto interno desses vetores. A decomposição da matriz $\mathbf{Y}_{I \times J}$ (Tabela 1) em duas outras matrizes, $\mathbf{U}_{I \times k}$ (matriz de características de usuários) e $\mathbf{V}_{J \times k}$, (matriz de características de itens) é realizada de acordo com a equação 2.1.

$$\mathbf{Y} \approx \mathbf{U} \times \mathbf{V}^T \quad (2.1)$$

Após o aprendizado dos fatores latentes (um vetor k -dimensional para cada usuário e um vetor k -dimensional para cada item), a predição da avaliação de itens desconhecidos para um usuário é obtida por meio do produto escalar do vetor de características do usuário pelo vetor de características do item (e.g.):

$$\hat{y}_{ij} = \mathbf{v}_j^T \mathbf{u}_i = \sum_{l=1}^k v_{lj}^T u_{il} \quad (2.2)$$

onde $\mathbf{u}_i = [u_{i1}, u_{i2}, \dots, u_{ik}]^T \in \mathbb{R}^k$ para $i = 1, \dots, I$ são os I vetores linha da matriz $\mathbf{U}$ e $\mathbf{v}_j = [v_{j1}, v_{j2}, \dots, v_{jk}] \in \mathbb{R}^k$ para $j = 1, \dots, J$ são os J vetores linha da matriz $\mathbf{V}$.

A etapa de aprendizado deste modelo é caracterizada pela determinação dos vetores de características dos usuários e itens. Uma maneira de encontrar esses vetores é inicializar os

elementos das duas matrizes $\mathbf{U}$ e $\mathbf{V}$ com valores aleatórios e, por meio da equação 2.2 para os valores conhecidos da matriz $\mathbf{Y}$, minimizar a diferença obtida via algoritmos de otimização. Normalmente, algoritmos de otimização iterativa são aplicados para tal finalidade. Mais especificamente, a diferença entre o valor predito pelo sistema e o valor real é chamado de erro de predição, e pode ser calculado conforme:

$$e_{ij}^2 = (y_{ij} - \hat{y}_{ij})^2 = (y_{ij} - \sum_{l=1}^k v_{lj}^T u_{il})^2 \quad (2.3)$$

Em seguida, ocorre a atualização dos vetores de características dos usuários e itens com o objetivo de minimizar o erro de predição:

$$\mathbf{u}_i \leftarrow \mathbf{u}_i + \alpha(e_{ij}\mathbf{v}_j - \lambda\mathbf{u}_i) \quad (2.4)$$

$$\mathbf{v}_j \leftarrow \mathbf{v}_j + \alpha(e_{ij}\mathbf{u}_i - \lambda\mathbf{v}_j) \quad (2.5)$$

Em (2.4) e (2.5) α é uma constante que define a taxa de aprendizagem e λ é utilizado para controlar as magnitudes dos vetores de características de usuários e itens (regularização) de tal modo a evitar o super-ajuste do modelo à base de treinamento.

Um algoritmo amplamente utilizado para a minimização do erro é o gradiente descendente estocástico, cujo comportamento pode ser resumido na execução dos seguintes passos:

1. Percorrer cada uma das avaliações da base de treinamento e fazer a predição para cada avaliação conhecida (2.2);
2. Calcular o erro da predição (2.3);
3. Fazer a atualização dos vetores de usuários e itens (usando 2.4 e 2.5);
4. Executar os passos 1, 2 e 3 até que a condição de parada seja alcançada (geralmente utiliza-se um valor que define o número de iterações do laço).

Finalizada a fase de treinamento, as predições para os valores desconhecidos podem ser feitas utilizando a 2.2. A Tabela 3 ilustra os valores encontrados pela fatoração de matrizes com os dados da Tabela 1. O algoritmo foi executado com os seguintes parâmetros: $k = 4$ (número de fatores latentes), $\alpha = 0.002$, $\lambda = 0.2$ e a condição de parada estabelecida em 5.000 iterações.

Comparando as Tabelas 1 e 3, percebe-se que as predições do modelo são próximas aos valores conhecidos. Observa-se também que a predição feita para o item não avaliado (*Usuário 1* para o *Item 5*) é de um valor alto (5,03) o que faria com que o sistema recomendasse o *Item 5* para o *Usuário 1*.

Tabela 3 – Predições utilizando fatoração de matrizes.

	Item 1	Item 2	Item 3	Item 4	Item 5
Usuário 1	4,95	2,99	3,99	4,00	5,03
Usuário 2	2,99	1,02	1,97	2,96	2,99
Usuário 3	3,99	2,99	3,98	3,01	3,96
Usuário 4	3,00	2,97	1,02	4,96	3,98
Usuário 5	1,01	4,96	4,96	1,99	1,01

2.5.2 Biclusterização

O termo biclusterização (*biclustering*) foi usado pela primeira vez no trabalho de [Cheng e Church \(2000\)](#), que apresentaram um algoritmo de agrupamento simultâneo de linhas e colunas de uma matriz de expressões gênicas na qual as linhas correspondem a genes e as colunas são amostras (padrões de expressão ou condições). Um bicluster é constituído por um subconjunto de genes e um subconjunto de padrões de expressão que possuem alta similaridade entre si. A técnica de biclusterização pode ser vista como um modelo local de agrupamento, pois considera apenas parte dos atributos dos objetos para definir os grupos. Já as técnicas clássicas de agrupamento podem ser definidas como pertencentes a um modelo global, pois levam em conta todos os padrões no momento de induzir os grupos ([MADEIRA; OLIVEIRA, 2004](#); [FRANÇA, 2010](#)). Por utilizar modelos locais para agrupamento, o espaço de busca empregado por algoritmos de biclusterização tende a ser maior do que aquele explorado por técnicas de agrupamento tradicionais. Segundo [França \(2010\)](#), situações em que o uso de biclusterização é pertinente incluem:

1. Existência de padrões que são encontrados apenas quando alguns atributos são utilizados;
2. Possibilidade de um objeto participar de mais de um grupo, dependendo dos atributos utilizados;
3. Existência de apenas parte dos objetos contendo informações relevantes, dado um conjunto dos atributos.

2.5.2.1 Biclusterização e SR

No contexto de SR, um bicluster é um subconjunto de usuários que exibem comportamentos similares sobre um subconjunto de itens e vice-versa. Apesar do foco de grande parte dos trabalhos sobre biclusterização estar na análise de dados de expressão gênica ([AGRAWAL et al., 1998](#); [TANG et al., 2001](#); [MITRA; BANKA; PAL, 2006](#); [DIVINA; AGUILAR-RUIZ, 2007](#); [MAULIK et al., 2008](#)), recentemente algumas aplicações foram desenvolvidas no contexto de recomendação colaborativa ([CASTRO et al., 2007a](#); [SYMEONIDIS et al., 2008](#); [CASTRO et al., 2007b](#); [GEORGE; MERUGU, 2005](#)).

Segundo [França \(2010\)](#), um dos motivos do número reduzido de publicações sobre biclusterização em sistemas de recomendação deve-se à necessidade de se criar mecanismos para lidar com os dados faltantes das bases de dados, pois geralmente as matrizes a serem processadas são altamente esparsas e grande parte das abordagens de biclusterização necessita da matriz completa.

[Castro et al. \(2007a\)](#) apresentaram uma proposta para o problema causado pelos valores ausentes: inicialmente, todos os dados faltantes da matriz são preenchidos com zeros e após a fase de biclusterização, os dados faltantes são inseridos através de uma média ponderada entre os valores contidos no bicluster. No contexto de aplicações, [Symeonidis et al. \(2008\)](#) propuseram uma abordagem para recomendação de filmes baseada na busca por biclusters com valores constantes. O algoritmo de biclusterização utilizado, Bimax ([PRELIĆ et al., 2006](#)), percorre a matriz de dados procurando por biclusters nos quais todos os elementos são iguais a 1. Os parâmetros do algoritmo Bimax são o número mínimo de usuários e o número mínimo de filmes por bicluster. A base de dados conhecida como MovieLens, que contém avaliações que variam entre 1 e 5, foi utilizada. Nesta base, filmes sem avaliações possuem valor igual a zero. Pelo fato do algoritmo Bimax trabalhar com valores binários, a base de dados foi transformada para a forma binária, na qual as avaliações menores do que 3 foram convertidas para zero e os demais casos receberam o valor 1. Após a discretização dos dados, e passado o estágio de biclusterização, um método de k-vizinhos mais próximos foi utilizado para encontrar os biclusters mais próximos a cada usuário. Estes biclusters foram, então, utilizados para gerar as recomendações. Os resultados obtidos foram confrontados com abordagens tradicionais utilizadas em recomendação colaborativa (seção 2.1) e demonstraram um aumento de 30% de precisão e de 10% em revocação. [Castro et al. \(2007b\)](#) desenvolveram um algoritmo em que uma meta-heurística imuno-inspirada foi utilizada para encontrar biclusters. O processo de formação de clusters conta com um rede imunológica artificial denominada BIC-aiNet. O processo de biclusterização tem como objetivos principais maximizar a área do bicluster (aumentar a quantidade de elementos) sem comprometer a correlação entre eles. Após os biclusters terem sido encontrados, o algoritmo calcula a predição por meio de otimização quadrática, obtendo então uma solução ótima de predição nas condições de correlação dos elementos de cada bicluster ([FRANÇA, 2010](#)). Uma abordagem de recomendação colaborativa proposta por [George e Merugu \(2005\)](#), utiliza uma instância do algoritmo *Bregman co-clustering* ([BANERJEE et al., 2005](#)) que, com base na matriz de avaliações (e.g. Tabela 1), realiza de forma simultânea o agrupamento de usuários e filmes, inferindo uma série de medidas estatísticas relacionadas a cada bicluster que são utilizadas para fazer a recomendação de novos filmes.

2.6 Medidas de avaliação

As técnicas de recomendação apresentadas neste capítulo têm sido empregadas em vários SR que atuam em diferentes domínios. Portanto, é necessário estabelecer medidas de

avaliação para encontrar qual técnica obtém melhor desempenho em uma determinada área ou, até mesmo, para a avaliação comparativa de novas técnicas, algoritmos e sistemas. Os SR têm sido tradicionalmente avaliados em experimentos *offline* que estimam o erro de predição das recomendações utilizando bases de dados públicas e/ou privadas. Empresas e grupos de pesquisa têm colaborado neste sentido, disponibilizando bases de dados representativas (e.g., Movielens, Netflix, Jester) para que a comunidade científica possa replicar experimentos, validar e melhorar suas aplicações (BOBADILLA *et al.*, 2013). Em técnicas de recomendação baseadas em modelo, essas bases de dados são separadas em dois conjuntos de dados distintos — o conjunto de treinamento, destinado ao aprendizado do modelo, e o conjunto de testes utilizado para a avaliação do modelo. Tais medidas podem ser utilizadas na avaliação de SR sobre três aspectos:

- **Qualidade da predição:** Medidas que capturam a *acurácia* dos resultados de um SR. A medida mais utilizada neste sentido é a *Mean Absolute Error* (MAE), definida em 2.6, e outras diretamente relacionadas a esta — *Normalized Mean Absolute Error* (NMAE), *Mean Squared Error* (MSE) e *Root Mean Squared Error* (RMSE).

$$MAE = \frac{1}{n} \sum_{i,j} |\hat{y}_{ij} - y_{ij}| \quad (2.6)$$

Em (2.6), n representa o total de avaliações da base de testes, y_{ij} a avaliação feita pelo i -ésimo usuário para o j -ésimo item e $\hat{y}_{ij}$ a predição calculada pelo SR.

Quando existem diferenças na escalas das avaliações dos conjuntos de dados utilizados — por exemplo a base Movielens na qual as avaliações são representadas por números inteiros entre $[1, 5]$ e base Jester que possui avaliações representadas por números inteiros entre $[-10, +10]$ — fica difícil estabelecer bases de comparação utilizando a 2.6. A medida NMAE (2.7) pode ser utilizada nestas situações. Nela, os valores de saída são normalizados entre o intervalo $[0, 1]$:

$$NMAE = \frac{MAE}{y_{max} - y_{min}} \quad (2.7)$$

Em 2.7, y_{max} representa o maior valor possível na escala de avaliação e y_{min} o menor valor. O erro quadrático médio (MSE) é comumente usado como uma medida do erro de predição:

$$MSE = \frac{1}{n} \sum_{i,j} (\hat{y}_{ij} - y_{ij})^2 \quad (2.8)$$

Uma medida intimamente relacionada ao MSE é o RMSE, computado conforme a equação 2.9. Esta medida tornou-se muito utilizada na avaliação de SR após ter sido escolhida

com forma de comparar a acurácia no prêmio Netflix (BENNETT; LANNING, 2007; EKSTRAND; RIEDL; KONSTAN, 2011).

$$RMSE = \sqrt{\frac{1}{n} \sum_{i,j} (\hat{y}_{ij} - y_{ij})^2} \quad (2.9)$$

- **Medidas que avaliam o conjunto de recomendações:** Em vez de analisar avaliações individuais, estas medidas visam a eficácia do SR na capacidade de fazer recomendações úteis para o usuário. A avaliação de um SR com base apenas na qualidade de predição do sistema não serve como um parâmetro para sua utilização na prática. Do ponto de vista do usuário, os SR devem ser capazes de gerar uma lista reduzida de itens que satisfaçam suas preferências. Para avaliar este requisito do sistema, três medidas amplamente empregadas na área de recuperação da informação (BAEZA-YATES; RIBEIRO-NETO *et al.*, 1999) são utilizadas — (1) Precisão (P), que representa a proporção de recomendações feitas pelo sistema que são consideradas "boas" recomendações; (2) Revocação (R), que indica a proporção de "boas" recomendações recuperadas pelo sistema e (3) Medida $F1$ que é interpretada como a média ponderada entre P e R :

$$F1 = 2 \times \frac{P \times R}{P + R} \quad (2.10)$$

- **Avaliar a lista de recomendação gerada:** O que grande parte dos usuários espera de um SR é uma lista de recomendações ordenada com base nos seus interesses, onde itens com valores de predições mais altos estão no topo da lista e os itens com predições cujos valores são baixos encontram-se no final da lista. A métrica *half-life utility* (ou R-Score) proposta por Breese, Heckerman e Kadie (1998) captura a utilidade da lista gerada, calculando a probabilidade de acesso aos itens da lista:

$$R_i = \sum_j \frac{\max(\hat{y}_{ij} - d, 0)}{2^{(k-1)/(\alpha-1)}} \quad (2.11)$$

Em 2.11, $\hat{y}_{ij}$ representa a predição calculada pelo SR para o usuário i em relação ao item j , d é um valor de avaliação *default*, indicando um valor neutro de avaliação. Tomando como exemplo o cenário representado pela Tabela 1, d poderia assumir um valor intermediário igual a 3, visto que o valor máximo de uma avaliação é 5 e o menor é 1. O parâmetro k representa o posto do item j na lista ordenada e α especifica que a probabilidade do usuário i visualizar um item da lista com posto igual a α seja de 0.5.

A fim de medir o desempenho de um SR de acordo com diferentes usuários, a equação 2.11 é normalizada pela máxima utilização alcançada R_i^{max} resultando em um valor no intervalo $[0, 1]$ representando a fração de utilidade potencial alcançada. O cálculo de R_i^{max}

utiliza uma lista ordenada de todos os itens avaliados pelo usuário i . O resultado global é dado por:

$$R = \frac{\sum_i R_i}{\sum_i R_i^{max}} \quad (2.12)$$

2.7 Considerações Finais

Este capítulo apresentou um resumo das técnicas mais utilizadas no desenvolvimento de SR — Recomendação Colaborativa (Seção 2.2), Recomendação Baseada em Conteúdo (Seção 2.3) e Recomendação Híbrida (Seção 2.4). A seção 2.5 apresentou duas técnicas de recomendação baseadas em modelos. A primeira técnica, amplamente utilizada no contexto de SR, é conhecida por fatoração de matrizes, que constrói seu modelo de predição fazendo a decomposição da matriz de dados em duas outras matrizes que constituem as características latentes de usuários e itens. A segunda técnica abordada foi a biclusterização que, devido à sua natureza de busca por padrões que são encontrados em partes dos atributos e por permitir que um objeto pertença a mais de um grupo, se adapta bem ao contexto de recomendação no qual um bicluster pode ser considerado um subconjunto de usuários que exibem comportamentos similares sobre um conjunto de itens e vice-versa. As medidas de avaliação frequentemente utilizadas na avaliação de SR foram apresentadas na Seção 2.6. O próximo capítulo apresenta o algoritmo chamado SCOAL que explora tanto os atributos presentes na matriz de dados quanto os atributos independentes que caracterizam usuários e itens.

AGRUPAMENTO DE DADOS E APRENDIZADO DE FUNÇÕES DE REGRESSÃO

3.1 Considerações Iniciais

A utilização de um único modelo de aprendizado em tarefas de classificação/regressão é uma prática comum. Outras abordagens que vêm sendo utilizadas com sucesso neste cenário são métodos baseados em *ensemble*, no quais múltiplos modelos são treinados para resolver o mesmo problema. Tipicamente, um *ensemble* é constituído de duas etapas: na primeira, um conjunto de modelos base é produzido; as saídas de cada modelo base são então combinadas na segunda etapa para gerar a resposta do sistema (DIETTERICH, 2000). Entretanto, dada a grande quantidade de dados envolvida em SR é improvável que todas as avaliações (tais como aquelas da Tabela 1) possam ser bem representadas por um único modelo. Ao invés disso, pode ser mais factível aprender modelos que representem as avaliações de somente um subconjunto de usuários para um subconjunto de itens utilizando uma abordagem *dividir e conquistar*, particionando o conjunto de dados em segmentos relativamente homogêneos para então desenvolver modelos específicos para cada segmento (KIM; KIM, 2006; TOSATTO *et al.*, 2002; KOTTE; HEINEMANN, 2009; HSIEH *et al.*, 2012).

3.2 Abordagens baseadas em dividir e conquistar

Além de usualmente prover modelos precisos e confiáveis, a utilização da abordagem *dividir e conquistar* possibilita também uma melhor interpretação, uma vez que os modelos gerados tendem a ser mais simples quando comparados a um único modelo — geralmente mais complexo. Normalmente, o particionamento do conjunto de dados é feito *a priori*, seja baseado

no conhecimento do domínio, seja pela utilização de um procedimento de agrupamento de dados antes da geração dos modelos de predição (DEODHAR; GHOSH, 2010). O algoritmo SCOAL, discutido a seguir, se baseia numa abordagem que faz uso de agrupamento de dados e construção de modelos de predição simultâneos.

3.3 Algoritmo SCOAL

O algoritmo SCOAL (*Simultaneous Co-clustering and Learning*), proposto por Deodhar e Ghosh (2010), implementa uma abordagem *dividir e conquistar* iterativa, intercalando as atividades de agrupamento e aprendizado do modelo de predição. Desta forma, objetiva-se melhorar tanto a atribuição de objetos aos grupos quanto o ajuste do modelo de predição. Esta abordagem simultânea produz melhores resultados do que agrupar os dados primeiro e depois construir modelos de classificação e/ou regressão de forma independente. Neste contexto, Deodhar e Ghosh (2007) apresentam uma função objetivo que é garantidamente minimizada em ambas as etapas do processo iterativo, até que se obtenha um mínimo local (em outras palavras, o algoritmo proposto possui garantias de convergência). Tal abordagem pode ser vista como uma generalização de modelos baseados em biclusterização (subseção 2.5.2). É necessário notar, entretanto, que grande parte das técnicas de biclusterização geram biclusters baseando-se tão somente nos valores das células da matriz de dados, além de não admitir valores ausentes, enquanto que o algoritmo SCOAL permite a associação de conjuntos de variáveis independentes para cada dimensão da matriz de dados.

Inicialmente, a base de dados é particionada em biclusters definidos de forma arbitrária, sendo o número de biclusters definido *a priori*. Um modelo de predição é gerado para cada um dos biclusters usando conjuntos de atributos que compreendem as linhas e colunas de uma matriz, $\mathbf{Y}$, (e.g. Tabela 1) onde linhas representam usuários, colunas representam itens e cada elemento y_{ij} caracteriza uma avaliação feita pelo usuário i para o item j . Neste cenário, a similaridade entre usuários (ou itens) é calculada indiretamente pelo desempenho de predição obtido com a aprendizagem dos modelos. A partição corrente é então atualizada, sendo alguns usuários e/ou itens realocados para outros biclusters desde que haja melhoria na média de desempenho dos modelos de predição atualizados. O modelo de predição associado a cada bicluster é atualizado sempre que ocorrer remoção ou inserção de objetos (usuários ou itens) no bicluster. Esta abordagem simultânea de agrupamento e aprendizado de modelos produz melhores resultados do que agrupar os dados primeiro para então construir os modelos de predição (DEODHAR; GHOSH, 2007; DEODHAR; GHOSH, 2010).

Para um melhor entendimento do funcionamento do SCOAL, considere o cenário de aplicação representado pela Tabela 1. Além de considerar os valores presentes em cada elemento y_{ij} da matriz $\mathbf{Y}$ — cujas linhas representam *Usuários* e cujas colunas representam *Itens* — o algoritmo SCOAL utiliza atributos que descrevem tanto as características específicas dos usuários

como dos itens no processo de aprendizado do modelo de predição. A Figura 2 representa um conjunto ilustrativo de dados utilizado pelo SCOAL para a construção dos modelos de predição.

Figura 2 – Dados utilizados pelo SCOAL para a construção dos modelos de predição — adaptado de Deodhar e Ghosh (2007).

		características dos itens				
		atributos do item 1	atributos do item 2	atributos do item 3	atributos do item 4	atributos do item 5
usuários	atributos do usuário 1	5	3	4	4	?
	atributos do usuário 2	3	1	2	3	3
	atributos do usuário 3	4	3	4	3	4
	atributos do usuário 4	3	3	1	5	4
	atributos do usuário 5	1	5	5	2	1

Num contexto de recomendação de filmes, por exemplo, os atributos dos usuários poderiam ser: *nome, idade, sexo, profissão* etc. Já as características referentes aos itens, ou seja, os atributos dos filmes, poderiam ser: *gênero, data de lançamento, diretor, ator principal* etc. Num modelo de regressão linear, tais atributos são denominados variáveis independentes do modelo e os elementos y_{ij} contêm os valores da variável dependente.

O algoritmo SCOAL particiona a matriz de dados em U grupos de usuários e V grupos de filmes. Os valores de U e V devem ser informados *a priori*. A similaridade entre dois objetos passa agora a ser determinada não pelas avaliações feitas, mas sim pelos (resultados dos) seus modelos preditivos. A Figura 3 ilustra um resultado hipotético obtido pelo algoritmo SCOAL sobre os dados da Tabela 1: há quatro biclusters representados por cores diferentes, cada qual com um modelo preditivo associado. Note que essa solução envolve particionar a matriz de dados em dois *clusters* de linhas ($U = 2$) e em dois *clusters* de colunas ($V = 2$). U e V são parâmetros de entrada do SCOAL, cujos valores foram escolhidos de forma arbitrária para este exemplo.

O algoritmo SCOAL recebe um conjunto de U clusters de linhas (usuários), $\mathbb{C}_u = \{C_1^u, C_2^u, \dots, C_U^u\}$, onde $C_r^u \in \mathbb{C}^u, r = 1, \dots, U$, e um conjunto de V clusters de colunas (itens), $\mathbb{C}_v = \{C_1^v, C_2^v, \dots, C_V^v\}$, onde $C_c^v \in \mathbb{C}^v, c = 1, \dots, V$. Os atributos do i -ésimo usuário são representados pelo vetor $\mathbf{u}_i$, e os atributos do j -ésimo item são representados pelo vetor $\mathbf{v}_j$.

Figura 3 – (a) Solução encontrada pelo SCOAL depois de rearranjar as linhas e colunas da Tabela 1. Linhas e colunas são realocadas para outro bicluster sempre que tal operação possibilite uma melhoria na média de desempenho dos modelos de predição. As diferentes cores representam os biclusters induzidos. (b) Mapeamento de entrada-saída produzido pelo modelo de predição $\{r, c\}$. Os índices de linha e coluna são específicos para cada bicluster.

3.4 Predição utilizando modelos lineares

Presumindo um modelo de regressão linear, o valor armazenado em cada célula da matriz de dados $\mathbf{Y}$ (Tabela 1) pode ser escrito com uma combinação linear dos atributos do usuário i e dos atributos do item j :

$$y_{ij} = \boldsymbol{\beta}_{rc}^T \mathbf{x}_{ij} + \mathcal{N}(0, \sigma^2) \quad (3.1)$$

Em que $\mathbf{x}_{ij}$ é o vetor que concatena os atributos de usuários e de itens, $\mathbf{x}_{ij} = [1, \mathbf{u}_i, \mathbf{v}_j]$, r é o índice do cluster de linha e c é o índice do cluster de colunas do bicluster que contém o usuário i e o item j . A partir dos dados observados associados com o bicluster $\{r, c\}$, pode-se então obter uma estimativa $\hat{\boldsymbol{\beta}}_{rc}$ e depois fazer predições para qualquer par *usuário-item* (i, j) que seja membro do bicluster $\{r, c\}$:

$$\hat{y}_{ij} = \hat{\boldsymbol{\beta}}_{rc}^T \mathbf{x}_{ij}, \quad (3.2)$$

onde $\hat{\boldsymbol{\beta}}_{rc}$ concatena os parâmetros do modelo e tem a mesma dimensão de $\mathbf{x}_{ij}$.

A função objetivo a ser minimizada é :

$$\sum_{ij} w_{ij} (y_{ij} - \hat{y}_{ij})^2 \quad (3.3)$$

na qual $\hat{y}_{ij} = \hat{\boldsymbol{\beta}}_{rc}^T \mathbf{x}_{ij}$, $\hat{\boldsymbol{\beta}}_{rc}$ é o conjunto de parâmetros para o bicluster $\{r, c\}$, $i \in C_r^u, j \in C_c^v$;
e

$$w_{ij} = \begin{cases} 1 & \text{se } y_{ij} \neq ? \\ 0 & \text{se } y_{ij} = ? \end{cases}$$

Para aprender $\hat{\boldsymbol{\beta}}_{rc}$, $r = 1, \dots, U$ e $c = 1, \dots, V$, pelo método “ridge” (HOERL; KENNARD, 1970) temos:

$$\hat{\boldsymbol{\beta}}_{rc} = (\mathbf{X}_{rc}^T \mathbf{X}_{rc} + \lambda \mathbf{I})^{-1} \mathbf{X}_{rc}^T \mathbf{y}_{rc} \quad (3.4)$$

Em que $\mathbf{X}_{rc}$ é uma matriz que concatena os atributos de usuários e itens, $\mathbf{y}_{rc}$ é o vetor de respostas, $\mathbf{I}$ representa a matriz identidade e $\lambda \geq 0$, é um parâmetro que controla a quantidade de regularização. Cada linha de $\mathbf{X}_{rc}$ é dada por $\mathbf{x}_{ij}$, $i \in C_r^u$ e $j \in C_c^v$, tal que $y_{ij} \neq ?$, e cada linha de $\mathbf{y}_{rc}$ é o correspondente y_{ij} . O efeito de aumentar o valor de λ é reduzir a magnitude do coeficiente de regressão $\hat{\boldsymbol{\beta}}_{rc}$. Para determinar λ , geralmente um conjunto de dados de validação é adotado. Os principais passos do algoritmo SCOAL podem ser sumarizados por meio do Algoritmo 1.

Algoritmo 1: Algoritmo SCOAL (adaptado de (DEODHAR; GHOSH, 2010)).

Entrada: $\mathbf{Y}_{I \times J}$, $\mathbf{W}_{I \times J}$, $\mathbf{x}_{ij} = [1, \mathbf{u}_i, \mathbf{v}_j]$, $\forall i \in \{1, \dots, I\}, j \in \{1, \dots, J\}, U, V$.

Saída: Partição de biclusters, (C^u, C^v) , e os respectivos modelos dos biclusters, $\hat{\boldsymbol{\beta}}_{rc}$, $r \in \{1, \dots, U\}$ e $c \in \{1, \dots, V\}$.

1 Inicializar randomicamente os biclusters (C^u, C^v)

2 **repita**

3 Construir um modelo de regressão para cada bicluster;

4 Atualização dos *clusters* de linhas (*usuários*); //associar cada linha ao *cluster* de linhas que minimiza o erro;

5 Atualização dos *clusters* de colunas (*itens*); //associar cada coluna ao *cluster* de colunas que minimiza o erro;

6 **até** critério de convergência ser atingido;

7 **retorna** C^u, C^v , e $\hat{\boldsymbol{\beta}}_{rc}$, $r \in \{1, \dots, U\}$ e $c \in \{1, \dots, V\}$.

3.5 Predição utilizando modelos não lineares

O algoritmo SCOAL não está restrito ao uso de modelos lineares como seus componentes, podendo utilizar modelos preditivos não lineares. Para tal, basta definir apropriadamente como o valor de $\hat{y}_{ij}$ é computado.

Entretanto, experimentos realizados com o SCOAL utilizando modelos não lineares baseados em Redes Neurais Artificiais do tipo Perceptron de Múltiplas Camadas — *Multilayer Perceptron* (MLP) obtiveram um desempenho de predição inferior aos modelos baseados em regressão linear. Segundo [Deodhar e Ghosh \(2010\)](#), a justificativa para tal desempenho é que para determinadas bases de dados, um conjunto de modelos lineares simples é suficiente para capturar a heterogeneidade e estrutura desses dados, além do fato de que um conjunto de modelos não lineares necessita de um número maior de parâmetros, dificultando a sua configuração.

Apesar do algoritmo SCOAL mostrar resultados expressivos utilizando modelos lineares, ainda há espaço para melhorias no tocante a modelos não lineares. Neste sentido, no Capítulo 5 desta tese é apresentada uma instância do algoritmo SCOAL cujos modelos de predição são baseados em Máquinas de Aprendizado Extremo (ELM) (*Extreme Learning Machines*) ([HUANG; ZHU; SIEW, 2004](#)). A adoção de ELMs foi motivada pela sua acurácia, velocidade de treinamento e por demandar pouca intervenção humana na sua parametrização ([HUANG et al., 2015](#)).

3.6 Considerações Finais

Este capítulo apresentou o algoritmo SCOAL, que faz uso de uma abordagem para *dividir e conquistar*. Em particular, o SCOAL intercala atividades de agrupamento e de aprendizado de modelos, e pode ser aplicado a uma ampla variedade de domínios que envolvem problemas complexos. A adoção do SCOAL como um componente de um SR é uma abordagem promissora pois, diferentemente das técnicas tradicionais discutidas nos capítulos anteriores, o algoritmo SCOAL calcula similaridades entre os objetos não apenas utilizando os valores das células da matriz de dados, mas também seus respectivos modelos de predição — construídos com base nas características específicas de usuários e itens.

SCOAL E O PROBLEMA *COLD-START*

4.1 Considerações Iniciais

Um dos principais problemas que afetam grande parte dos SR é conhecido por *cold-start*. O *cold-start* ocorre quando não é possível fazer recomendações devido à falta de um histórico de avaliações sobre um novo item ou um novo usuário incorporado no sistema. Em muitas situações, o problema *cold-start* acarreta a perda de novos usuários que decidem parar de usar o sistema devido à baixa acurácia obtida nas primeiras recomendações (BOBADILLA *et al.*, 2012a). Neste capítulo, é apresentada uma abordagem híbrida de recomendação baseada no algoritmo SCOAL que é capaz de lidar não só com situações parciais de *cold-start* nas quais um número mínimo de avaliações feitas pelo novo usuário é fornecido, mas também em situações extremas de *cold-start* onde não existe nenhum histórico de avaliação do novo usuário do sistema.

4.2 Trabalhos relacionados

Muitas abordagens têm sido propostas para lidar com o problema *cold-start* no contexto de SR. Esta seção apresenta algumas dessas abordagens, separando-as pelo tipo de técnica empregada. Independentemente da técnica utilizada, nota-se que os modelos de predição normalmente fazem uso de informações sobre as características dos itens e dos atributos dos usuários do sistema. Outro fato observado é a necessidade de identificar grupos de usuários e/ou itens semelhantes a fim de melhorar a acurácia da predição.

4.2.1 Fatores latentes

Um modelo probabilístico que combina RC e RBC proposto por Schein *et al.* (2002), utiliza variáveis latentes entre associações de *usuários-filmes* e *filmes-atores* para fazer recomendações para novos filmes. Os parâmetros do modelo são estimados pelo algoritmo *Expectation*

Maximization (EM) (DEMPSTER; LAIRD; RUBIN, 1977). Um SR proposto por Wang e Wang (2014) emprega a análise probabilística de semântica latente — *probabilistic latent semantic analysis* (pLSA) (HOFMANN, 1999) — para modelar o relacionamento entre os atributos de usuários e atributos dos itens e usa o algoritmo EM para ajustar o modelo. Gantner *et al.* (2010) utilizam um método que mapeia ou atributos de usuário ou itens com as características latentes de um modelo de fatoração de matriz. Com tais mapeamentos, pode-se resolver o problema de novos usuários ou novos itens do sistema.

4.2.2 Redes sociais

Em um SR baseado em redes de confiança (VICTOR *et al.*, 2008), os usuários do sistema estão conectados por intermédio de uma “pontuação de confiança” e obtêm recomendações para itens avaliados por pessoas da mesma rede chamada *Web Of Trust* (WOT). Um algoritmo de recomendação baseado em grafos tripartidos (usuários-itens-tags) proposto por Zhang *et al.* (2010) consideram *tags* sociais como uma ponte entre usuários e itens. Sistemas baseados em *tags* permitem que o usuário associe a itens de seu interesse palavras-chave ou termos, estabelecendo, assim, uma classificação informal dos itens por meio das palavras-chave associadas a estes. Dessa forma, as predições são calculadas com base na frequência das *tags* (preferência do usuário) e no relacionamento semântico entre *tags* e itens (informação global). Um outro algoritmo baseado em redes sociais de dados foi proposto por Sahebi e Cohen (2011), no qual a informação de redes sociais em diferentes dimensões (*e.g.*, interações entre usuários, avaliações, nível de popularidade/amizade, etc), é utilizada para detectar comunidades de usuários latentes. Dessa forma, um novo usuário pode ser inserido dentro de uma comunidade que é mais similar ao seu perfil. Então, o sistema é capaz de fazer recomendações baseadas nas avaliações feitas pelos usuários desta comunidade, os quais são os mais próximos de acordo com o perfil de usuário. Lin *et al.* (2013) usam informação extraída dos dados do Twitter para fazer recomendações de aplicativos para dispositivos móveis. A abordagem é baseada em Alocação Latente de Dirichlet — *Latent Dirichlet Allocation* (LDA) (BLEI; NG; JORDAN, 2003) — que é utilizada para gerar grupos latentes. Um novo usuário é mapeado para os grupos latentes encontrados usando as relações transitivas entre eles e os aplicativos.

4.2.3 Novas medidas de similaridade

Uma medida de similaridade chamada *Proximity-Impact-Popularity* (PIP) foi proposta por Ahn (2008) como parte de sua abordagem para lidar com o problema *cold-start*. Tal medida é utilizada para calcular semelhanças entre os usuários para que o sistema possa gerar suas recomendações. Esta abordagem tem mostrado acurácia superior quando comparada a outras medidas amplamente utilizadas em SR baseados em filtragem colaborativa. Bobadilla *et al.* (2012b) propuseram uma nova medida de similaridade que considera não somente a informação numérica contida nas avaliações — tais como as medidas tradicionalmente utilizadas em RC

como correlação de Pearson — mas também usa informação baseada na distribuição e no número de avaliações obtida por cada par de usuários a ser comparado. Os autores argumentam que é mais razoável explicitar uma maior semelhança entre usuários que avaliaram positivamente uma quantidade similar de itens do que entre usuários para os quais a quantidade de itens seja muito diferente. Basiri *et al.* (2010) utilizam toda a informação disponível de cada usuário para criar um operador *Ordered Weighted Averaging* (OWA) (YAGER, 1988) que é usado para fazer recomendações. O operador aplica um conjunto de pesos associados com cada técnica de recomendação (RC ou RBC) e suas possíveis combinações para fazer as predições.

4.2.4 Regras de associação

Regras de associação são usadas por Shaw, Xu e Geva (2010) para expandir os perfis de usuários do sistema com base em padrões e associações de itens, tópicos e categorias, dando assim mais informações para um SR. Um algoritmo que combina regras de associação e agrupamento de dados foi apresentado por Sobhanam e Mariappan (2013). As regras de associação são usadas para criar e expandir o perfil do usuário com o propósito de aumentar o número de avaliações feitas pelo usuário, minimizando assim o problema *cold-start*. A técnica de agrupamento é usada para agrupar itens que, mais tarde, serão utilizados para fazer a predição de novos itens.

Uma outra abordagem baseada em regras de associação, proposta por Leung, Chan e Chung (2008), faz uso de regras de associação multinível para construir um modelo de preferência composto de três camadas, interligando usuários, itens e atributos dos itens, sendo capaz de capturar não só os relacionamentos *usuário-item* como também os relacionamentos *item-item*. O algoritmo *Cross-Level Association Rules* (CLARE) (LEUNG; CHAN; CHUNG, 2008), infere a preferência do usuário para um novo item que ainda não tenha sido avaliado no sistema com base na relação *item-item*. Num contexto de recomendação de filmes, por exemplo, caso o novo filme apresente características similares (*e.g.* gênero, elenco) com outros filmes que foram bem avaliados pelo usuário, o novo filme será então recomendado.

4.3 Abordagem proposta

Apesar do algoritmo SCOAL ser uma abordagem promissora no desenvolvimento de SR, ele não está preparado para lidar com o problema *cold-start*. Esta seção apresenta uma variante do algoritmo SCOAL capaz de fazer recomendações em situações de *cold-start*, seja de um novo usuário ou de um novo item incorporado ao sistema. A Figura 4 representa a abordagem proposta atuando num cenário de *cold-start* de novos usuários mas, de forma análoga, ela pode ser utilizada em situações de *cold-start* de novos itens pois, como visto anteriormente, o algoritmo SCOAL (Capítulo 3) atua no agrupamento da matriz de dados em ambas as dimensões: linhas (*usuários*) e colunas (*itens*). A abordagem proposta é composta de dois módulos principais:

Agrupamento de dados e aprendizado de modelos de forma simultânea: O propósito deste módulo é encontrar grupos de usuários com interesses em comum e simultaneamente construir modelos de predição que são específicos para cada grupo. Após o critério de convergência do SCOAL ser atingido (Algoritmo 1), o SCOAL particiona a matriz $\mathbf{Y}$ em blocos de $\mathbb{C}^u$ grupos de linhas e $\mathbb{C}^v$ grupos de colunas. Considerando o exemplo da Figura 3, depois de rearranjar as linhas e colunas da matriz $\mathbf{Y}$ (Tabela 1), a saída do SCOAL retorna dois *clusters* de linha $\mathbf{C}_1^u = \{\text{usuário 1, usuário 5}\}$, $\mathbf{C}_2^u = \{\text{usuário 2, usuário 3, usuário 4}\}$ e dois *clusters* de coluna $\mathbf{C}_1^v = \{\text{item 4, item 1, item 2}\}$, $\mathbf{C}_2^v = \{\text{item 3, item 5}\}$.

Recomendações cold-start: Este módulo é utilizado para identificar em qual grupo de usuários (gerado pelo SCOAL) o novo usuário do sistema deverá ser alocado para que as predições possam ser feitas com base nos modelos específicos de cada grupo. Este módulo capacita o sistema não somente a lidar com situações de *cold-start* parciais nas quais um número mínimo de avaliações feitas pelo novo usuário é informado, mas também em situações extremas, nas quais não existe nenhuma avaliação prévia do novo usuário. As demais subseções descrevem os componentes deste módulo.

Figura 4 – Visão geral da abordagem proposta para lidar com o problema *cold-start*.

4.3.1 Cluster com erro mínimo

O componente *cluster* com erro mínimo — *Cluster with Minimum Error* (CME) — é similar ao cálculo feito no passo 2 do algoritmo SCOAL (Algoritmo 1). Ele se baseia na premissa de que um número mínimo de avaliações feitas pelo novo usuário seja requerido para que o SR possa construir o seu perfil. O CME utiliza um pequeno número de avaliações informadas inicialmente pelo novo usuário para encontrar o *cluster* de linha r^* , o qual minimiza o MSE para estas avaliações iniciais,

$$r^* = \arg_{r^*} \min \left\{ \sum_{i=1}^N (y_{new,n} - \hat{y}_{new,n})^2 \right\}, \quad (4.1)$$

sendo N o número de avaliações iniciais feitas pelo novo usuário, $n = 1, \dots, N$ índices que representam os itens inicialmente avaliados pelo novo usuário; $\hat{y}_{new,n} = \hat{\beta}_{rc}^T \mathbf{x}_{new,n}$, $\mathbf{x}_{new,n} = [1, \mathbf{u}_{new}, \mathbf{v}_n]$, $\mathbf{u}_{new} \in \mathbf{C}_r^u$; e $\mathbf{v}_n \in \mathbf{C}_c^v$. Depois de identificar r^* , o novo usuário é então inserido neste *cluster* de linha e todas as predições para ele serão feitas baseadas nos modelos de predição deste *cluster*,

$$\hat{y}_{new,n} = \hat{\beta}_{r^*c}^T \mathbf{x}_{new,j}, \mathbf{u}_{new} \in \mathbf{C}_{r^*}^u, \mathbf{v}_j \in \mathbf{C}_c^v. \quad (4.2)$$

4.3.2 Situações extremas de cold-start

A maioria das abordagens propostas para lidar com o problema *cold-start* em SR não são capazes de atuar em situações extremas de *cold-start*, nas quais não existe nenhuma avaliação prévia feita pelo novo usuário do sistema (SCHEIN *et al.*, 2002; VICTOR *et al.*, 2008; ZHANG *et al.*, 2010; SHAW; XU; GEVA, 2010; PARK; CHU, 2009; BOBADILLA *et al.*, 2012b; AHN, 2008). Para atuar nestas situações, o módulo — *Recomendações cold-start* — utiliza os *clusters* de linha induzidos pelo SCOAL para construir um modelo de classificação que é usado para inferir os melhores modelos de predição para o novo usuário do sistema. Para tanto, primeiramente uma base de dados é gerada onde as instâncias são os usuários do sistema descritos pelos seus atributos. Ainda nesta base de dados é atribuído o rótulo do *cluster* de linha ao qual o usuário pertence que, mais tarde, será tratado como rótulo de classe para efeitos de classificação. A Tabela 4 ilustra a representação da base de dados gerada considerando os *clusters* induzidos pelo SCOAL conforme o exemplo da Figura 3, na qual os atributos do usuário são: idade, gênero e profissão.

Uma vez que a base de dados foi gerada (*e.g.* Tabela 4) o objetivo é encontrar r^* , o melhor *cluster* de linha de acordo com a probabilidade condicional da classe creditada para o novo usuário, $P(Class | \mathbf{u}_{new})$. Note que *Class* refere-se ao rótulo do *cluster* de linha que, por sua vez, fornece modelos de regressão para serem utilizados nas predições para o novo usuário. Assim

Tabela 4 – Base de dados usada para construir um modelo de classificação para situações extremas de *cold-start*. Os rótulos dos *clusters* obtidos do SCOAL tornam-se rótulos de classe, os quais são usados para construir um modelo de classificação capaz de encontrar os melhores modelos de predição para os novos usuários.

Usuário	Idade	Gênero	Profissão	Cluster
1	57	masculino	administrador	1
5	36	masculino	administrador	1
2	24	masculino	operador	2
3	53	feminino	professora	2
4	23	feminino	escritora	2

espera-se encontrar:

$$r^* = \arg_{r \max} P(\mathbf{C}_r^u \mid \mathbf{u}_{new}) \quad (4.3)$$

Após encontrar r^* , as recomendações para o novo usuário são feitas como formalizado e em 4.2.

Nos experimentos realizados para ilustrar a abordagem proposta, foram escolhidos três classificadores populares — Naive Bayes (NB), Árvore de Decisão J48 e Regressão Logística (Logistic), os quais estão disponíveis na API Weka — *Application Programming Interface* (API) (HALL *et al.*, 2009). Apesar dos bons resultados obtidos com estes classificadores (como descrito na Seção 4.4), as predições podem ser melhoradas ponderando as saídas dos classificadores conforme abordado nas Subseções 4.3.3 e 4.3.4.

4.3.3 Predição Ponderada

A predição ponderada — *Weighted Prediction* (WP) — é executada em duas etapas. Primeiro, como abordado anteriormente, um classificador estima a distribuição de probabilidade de classe para cada *cluster* de linha. Na segunda etapa, ao invés de escolher um único *cluster* de linha para fazer as predições para o novo usuário, todos eles são utilizados. Para tanto, utilizam-se das predições individuais de cada *cluster* de linha, nas quais os pesos são dados pelas respectivas probabilidades estimadas na primeira etapa *i.e.*:

$$\hat{y}_{new,j} = \sum_{r=1}^U \left[\frac{P(\mathbf{C}_r^u \mid \mathbf{x}_{new}) \cdot \hat{\boldsymbol{\beta}}_{r,c}^T \mathbf{x}_{new,j}}{\sum_{r=1}^U P(\mathbf{C}_r^u \mid \mathbf{x}_{new})} \right] \quad (4.4)$$

4.3.4 Classificação Dinâmica

A abordagem classificação dinâmica — *Dynamic Classification* (DC) — é uma variante da abordagem WP (descrita na Seção 4.3.3). Diferentemente da WP, que executa a primeira etapa somente uma vez, a DC considera um item de cada vez. Em outras palavras, a abordagem DC

constrói um modelo de classificação específico para todo e qualquer item ao qual uma predição tenha que ser calculada. Para isso, o conjunto de dados utilizado para construir o modelo de classificação é montado de forma dinâmica, um item de cada vez, e considerando apenas os usuários que tenham avaliado este item em particular. Assim, pode-se considerar que o conjunto de dados usado para estimar as probabilidades de classe é atualizado dinamicamente, levando-se em conta os diferentes itens a serem avaliados para um novo usuário do sistema. Exceto por este relevante detalhe, as predições são feitas como dado pela equação 4.4. Em princípio, e por razões óbvias, a DC exige classificadores computacionalmente eficientes. Portanto, nesta tese, será ilustrada sua utilização através de um classificador Naive Bayes. Na prática, dependendo dos recursos computacionais disponíveis, classificadores computacionalmente mais custosos podem ser utilizados.

4.4 Avaliação Experimental

A abordagem proposta foi avaliada em dois cenários distintos: (i) situações de *cold-start* parcial, na qual um pequeno número de avaliações por usuários está disponível e (ii) situações extremas de *cold-start*, nas quais não existem avaliações do novo usuário. Três bases de dados tradicionalmente usadas na avaliação de SR foram utilizadas nos experimentos realizados conforme descrito nas demais seções.

4.4.1 Bases de dados

4.4.1.1 Netflix

A base de dados Netflix tem sido amplamente utilizada para avaliar o desempenho de SR. Essa base foi criada em virtude do prêmio Netflix¹ e é composta por mais de 100.000.000 de avaliações feitas por 480.000 usuários. Os dados foram coletados entre outubro de 1998 e dezembro de 2005. As avaliações obedecem a uma escala de 1 até 5. Os experimentos foram feitos utilizando uma amostra randomicamente gerada da base Netflix original composta por 2.000 usuários e 17.770 filmes. Informações sobre os usuários e filmes não estão originariamente disponíveis. Para construir o modelo de predição, características derivadas da matriz $\mathbf{Y}$ foram utilizadas:

- Número de filmes avaliados por cada usuário;
- Valor médio das avaliações feitas por cada usuário;
- Variância das avaliações feitas por cada usuário;
- Número de usuários que avaliaram cada filme;

¹ www.netflixprize.com

- Média geral das avaliações de cada filme;
- Variância geral das avaliações de cada filme.

4.4.1.2 Jester

Esta base de dados foi extraída de um sistema de recomendação de piadas chamado Jester (GOLDBERG *et al.*, 2001). A base reúne avaliações feitas por 24.983 usuários para 100 piadas. Cada usuário avaliou um mínimo de 36 piadas² — cada avaliação varia de -10 a $+10$. Informações sobre os usuários e piadas não estão originariamente disponíveis, mas, como foi feito na base Netflix, informações adicionais foram extraídas da matriz $\mathbf{Y}$, gerando seis atributos — sendo os itens neste contexto piadas ao invés de filmes. Para os experimentos, foram escolhidos randomicamente 10.000 usuários da base de dados original.

4.4.1.3 Movielens 100k

A base Movielens consiste de 100.000 avaliações feitas por 943 usuários sobre 1.682 filmes. O valor de cada avaliação varia de 1 a 5. Cada usuário da base avaliou no mínimo 20 filmes. A base de dados é disponibilizada pelo laboratório de pesquisa GroupLens da Universidade de Minnesota³. Além das avaliações, a base de dados também fornece informações demográficas dos usuários e informações sobre os filmes, dos quais um conjunto de 23 atributos foi extraído e usado para aprender os modelos de predição. Em particular, existem 19 diferentes gêneros de filmes (*e.g.*, drama, ação, aventura, romance, etc), os quais foram representados através de um vetor binário de tamanho 19. Os seis atributos descritos na Subseção 4.4.1.1 foram adicionados à base Movielens original para a realização dos experimentos.

4.4.2 Parametrização dos experimentos

Procurando dar maior ênfase às contribuições propostas neste capítulo (seção 4.3), nenhuma otimização foi feita nos parâmetros do algoritmo SCOAL, pois o objetivo desta tese não é avaliar o desempenho do SCOAL de forma isolada, visto que isto já foi amplamente discutido (DEODHAR; GHOSH, 2007; DEODHAR; GHOSH, 2010). Neste sentido, os valores de U (grupos de usuário) e V (grupos de itens) foram fixados em 4 ($U = V = 4$). O critério de convergência foi definido como a diferença máxima relativa do MSE em duas iterações sucessivas do SCOAL sendo menor ou igual a 10^{-3} . Os experimentos foram realizados utilizando o método de validação cruzada com 10 pastas e os resultados obtidos foram avaliados através da medida NMAE (equação 2.7). Para a abordagem predição ponderada foram utilizados três classificadores (Naive Bayes (NB), Árvore de Decisão J48 e Regressão Logística (Logistic)). Nos experimentos realizados com a abordagem classificação dinâmica, o classificador usado foi o Naive Bayes.

² eigentaste.berkeley.edu/dataset/

³ www.grouplens.org/datasets/movielens/

4.4.3 Resultados obtidos com cold-start parcial

Esta seção apresenta os resultados obtidos em situações de *cold-start* parcial, nas quais um pequeno número de avaliações está disponível para cada novo usuário do sistema. Os experimentos foram realizados variando o número de avaliações disponíveis de 1 até 20. Para cada número de avaliações, uma recomendação foi calculada para cada base de dados e cada medida de similaridade. O método CME (Seção 4.3.1) é avaliado nestes cenários, sendo também comparado com outras abordagens amplamente usadas na literatura — correlação de Pearson, correlação restrita de Pearson (Subseção 2.2.1) e medida de cosseno. A medida de cosseno analisa o ângulo entre os dois vetores de avaliação. Quanto menor o ângulo, maior a similaridade entre os vetores (SARWAR *et al.*, 2001). Embora amplamente utilizadas, essas medidas são vulneráveis ao se usar matrizes de dados esparsas e não são adequadas para o problema *cold-start* (AHN, 2008). O método CME também foi comparado com a medida PIP (AHN, 2008), a qual foi especificamente proposta para atuar em situações parciais de *cold-start* em que um pequeno número de avaliações está disponível. A Figura 5 sumariza os resultados obtidos.

Note que a CME apresenta os melhores resultados para todas as bases de dados avaliadas, e que a medida PIP obtém melhores resultados em relação às demais medidas utilizadas para propósitos de comparação apenas em algumas situações. Especificamente, a medida PIP é melhor do que as outras medidas somente se o número de avaliações é menor do que quatro para as bases Movielens e Netflix, e menor do que 10 avaliações na base Jester.

Figura 5 – Valores de NMAE para diferentes números de avaliações disponíveis — situações de *cold-start* parcial.

4.4.4 Resultados obtidos em situações extremas de cold-start

Em situações extremas de *cold-start*, não existem avaliações feitas por parte do novo usuário do sistema. Para este tipo de problema, medidas que capturam as similaridades entre as avaliações dos usuários não podem ser empregadas e, como consequência, a maioria das abordagens descritas na literatura são inapropriadas. Esta subseção apresenta os resultados obtidos pelas abordagens WP (Seção 4.3.3) e DC (Seção 4.3.4). A Figura 6 sumariza os resultados obtidos. Todas as recomendações foram realizadas para cada base de dados supondo que na base de teste existem apenas novos usuários do sistema — *i.e.*, sem nenhuma avaliação.

Figura 6 – Valores de NMAE para situações extremas de *cold-start*.

Na maioria dos experimentos, a abordagem DC alcançou melhores resultados quando comparada a todas as instâncias de WP. Entretanto, na base de dados Jester, a abordagem WP baseada em J48 obteve melhores resultados do que a abordagem DC. Isto pode ser explicado devido à melhor acurácia do classificador J48 para esta base de dados, como mostrado na Figura 7. Note-se que esta figura ilustra as acurácias de classificação obtidas por cada classificador construído para lidar com a situação de *cold-start* extrema. Levando em conta novamente o exemplo da Tabela 4, na qual os rótulos dos *clusters* obtidos do SCOAL tornam-se rótulos de classe, os quais são então usados para construir um modelo de classificação capaz de encontrar o *cluster* de linha com os melhores modelos de predição para o novo usuário.

Figura 7 – Acurácia da classificação obtida considerando que cada rótulo de *cluster* encontrado pelo SCOAL é um rótulo de classe para o problema *cold-start*.

As abordagens WP e DC também foram comparadas com um método baseado em fatoração de matrizes proposto por [Gantner et al. \(2010\)](#) capaz de lidar com situações extremas de *cold-start*. Tal método é aqui referenciado como *Matrix Factorization K-Nearest Neighbors* (MF-KNN).

4.4.4.1 Comparações com o método MF-KNN

O método MF-KNN ([GANTNER et al., 2010](#)) usa um modelo tradicional de fatoração de matrizes para derivar um conjunto de características latentes da matriz $\mathbf{Y}$ para caracterizar usuários e itens. Nas abordagens baseadas em fatoração de matrizes tradicionais, as características latentes de um usuário ou item podem somente ser configuradas se o usuário/item ocorre na base de treinamento. Num cenário extremo de *cold-start*, este não é o caso. No entanto, é possível fazer uso do modelo de fatoração de matrizes para novos usuários estimando suas características latentes dos atributos dos usuários (*e.g.*, idade, gênero, profissão). O método MF-KNN consiste dos seguintes passos:

1. treinamento de um modelo de fatoração utilizando a matriz $\mathbf{Y}$;
2. aprendizado das funções de mapeamento do espaço de atributos para o espaço de fatores latentes.

Por questão de simplicidade e seguindo a descrição fornecida em [Gantner et al. \(2010\)](#) o algoritmo K-NN foi parametrizado com $k = 1$, e a medida de cosseno foi utilizada como medida de similaridade para encontrar o vizinho mais próximo com base no vetor $\mathbf{u}$ (atributos do usuário). Dessa forma, as predições para um novo usuário são feitas utilizando as características latentes do usuário mais similar na base de treinamento (1-NN).

O algoritmo usado no passo 1 (*Matrix Factorization*) e no passo 2 (*User Attribute KNN*) estão disponíveis na biblioteca *MyMediaLite Recommender System* (GANTNER *et al.*, 2011). Como sugerido em Gantner *et al.* (2011) os parâmetros do algoritmo *Matrix Factorization* foram definidos como $\lambda = 0,015$, $\alpha = 0,01$, iterações=100, desvio padrão=0,1 e número de fatores latentes=32.

As comparações com o método MF-KNN devem ser realizadas em bases de dados que possuam características adicionais (atributos) de usuários/itens, sem que as mesmas sejam derivadas da matriz de avaliações. Entre as bases de dados usadas nos experimentos, a base Movielens é a única que reúne este requisito. Como atributos de usuário foram utilizados: gênero, profissão e idade — segmentada em três faixas (adolescentes, adultos, idosos). Os atributos dos itens são os 19 diferentes gêneros que um filme pode assumir (*e.g.* ação, drama, romance, etc). Esses atributos nominais foram codificados por meio de representação binária. Seguindo o procedimento padrão, os experimentos foram realizados utilizando o método de validação cruzada com 10 pastas. A Figura 8 mostra os resultados obtidos para a medida NMAE, enquanto a Tabela 5 apresenta os tempos médios de execução alcançados numa plataforma Intel Core I7 2GHz Macbook Pro 8Gb.

Figura 8 – NMAE obtido na base Movielens para MK-KNN, NB, J48, Logistic, WP e DC.

Tabela 5 – Tempo de CPU em segundos decorrido para cada método.

NB	WP-NB	J48	WP-J48	Logistic	WP-Logistic	DC	MF-KNN
0,017	0,031	0,014	0,027	0,016	0,028	4,789	0,545

4.5 Análise sobre os resultados obtidos

Como pôde ser notado nos experimentos realizados simulando situações parciais de *cold-start*, o módulo CME apresentou os melhores resultados para todas as bases de dados avaliadas. Além disso, considerando situações extremas de *cold-start*, o módulo DC mostrou os

melhores resultados para a maioria dos experimentos. Um outro aspecto interessante ainda não abordado envolve a comparação dos resultados obtidos pelo método DC com os obtidos pelo método CME. Cabe ressaltar que o método CME faz uso de informações parciais (um número mínimo de avaliações prévias do usuário) ao passo que DC não faz. Dito isto, a Tabela 6 permite que seja feito um comparativo mais detalhado dos resultados obtidos. Esta tabela em particular, mostra que os melhores resultados obtidos são do método CME (*i.e.*, para 20 avaliações). De certa forma é surpreendente observar que, para as bases Movielens e Netflix, o método DC é tão bom quanto CME. Mesmo na base Jester, onde o método CME é melhor que DC, os resultados são muito bons. Com base nos resultados obtidos, pode-se concluir que a abordagem proposta é bastante eficaz em lidar com o problema *cold-start*.

Tabela 6 – : Valores NMAE: média (desvio padrão). O texto em negrito refere-se a melhores resultados.

		Netflix	Jester	Movielens
<i>Cold-start</i> parcial	CME	0,18458(0,002)	0,17435(0,003)	0,18624(0,003)
	Pearson	0,20854(0,009)	0,21921(0,003)	0,20186(0,010)
	Cosine	0,22147(0,008)	0,22152(0,003)	0,22040(0,009)
	Constrained Pearson	0,21368(0,009)	0,22194(0,003)	0,20997(0,010)
	PIP	0,21913(0,008)	0,22328(0,003)	0,21746(0,009)
<i>Cold-start</i> extremo	Naive Bayes (NB)	0,19348(0,001)	0,20643(0,009)	0,19602(0,006)
	NB-WP	0,18953(0,001)	0,20447(0,007)	0,19089(0,007)
	J48	0,19291(0,000)	0,19772(0,005)	0,19651(0,008)
	J48-WP	0,18991(0,001)	0,19553(0,008)	0,19347(0,008)
	Logistic	0,19405 (0,002)	0,20893(0,006)	0,19575(0,007)
	Logistic-WP	0,18988(0,002)	0,20457(0,007)	0,19076(0,007)
	DC	0,18275(0,002)	0,20476(0,006)	0,18647(0,004)

4.6 Considerações Finais

Este capítulo apresentou três módulos capazes de lidar com o problema *cold-start* que foram incorporados a uma instância do algoritmo SCOAL, habilitando-o a enfrentar tal situação. O problema *cold-start* é caracterizado pela falta de informação prévia dos novos usuários e/ou itens, e é frequentemente encontrado em SR, acarretando a perda de novos usuários que decidem parar de usar o sistema devido à sua baixa acurácia nas recomendações. A investigação de ferramentas para lidar com este tipo de problema é de suma importância. Neste sentido, os módulos propostos neste capítulo mostram-se úteis, com base nos resultados obtidos em três bases de dados amplamente usadas na avaliação do desempenho de sistemas de recomendação. Particularmente, o módulo baseado em classificação dinâmica é eficaz e muito promissor, apesar de demandar um alto custo computacional. Os resultados documentados nesse capítulo foram publicados em [Pereira e Hruschka \(2015\)](#).

AGRUPAMENTO DE DADOS E APRENDIZADO DE FUNÇÕES DE REGRESSÃO BASEADO EM MÁQUINAS DE APRENDIZADO EXTREMO

5.1 Considerações Iniciais

O algoritmo SCOAL descrito anteriormente no Capítulo 3 provou ser eficiente e versátil nos experimentos realizados por [Deodhar e Ghosh \(2010\)](#), podendo ser utilizado na resolução de uma ampla gama de problemas de classificação e/ou regressão. Entretanto, a maioria dos experimentos foram feitos utilizando modelos lineares. Nos experimentos realizados com o SCOAL utilizando modelos não lineares, os resultados obtidos pelos autores foram inferiores aos da versão utilizando modelos de regressão linear. No intuito de contribuir com melhorias ao algoritmo SCOAL no sentido de deixá-lo mais versátil por meio do uso de modelos não lineares, este capítulo apresenta uma versão do SCOAL que utiliza modelos de predição baseados em Máquinas de Aprendizado Extremo.

5.2 Máquinas de Aprendizado Extremo

Máquinas de Aprendizado Extremo — *Extreme Learning Machines* (ELM) ([HUANG; ZHU; SIEW, 2004](#)) — são redes neurais compostas por uma camada de entrada, uma camada intermediária e uma camada de saída cuja arquitetura é conhecida como *Single Hidden Layer Feedforward Neural Networks* (SHLFNNs) ([HUANG; CHEN; BABRI, 2000](#)). A Figura 9 apresenta a estrutura de uma SHLFNN, também conhecida na literatura como *Multilayer Perceptron* (MLP).

Figura 9 – Arquitetura de uma rede SHLFNN.

Na Figura 9, $\mathbf{w}$ representa a matriz de pesos da camada intermediária que armazena os pesos das conexões entre as unidades da camada de entrada e os neurônios da camada intermediária. De forma análoga, β é a matriz de pesos da camada de saída, armazenando os pesos das conexões entre os neurônios da camada intermediária e os neurônios da camada de saída.

Todos os parâmetros das SHLFNNs precisam ser ajustados, levando à dependência entre os parâmetros ($\mathbf{w}$, β e *bias* — Figura 9) das diferentes camadas da rede. O treinamento (ajustes dos parâmetros) de uma rede SHLFNN tradicional é feito utilizando variações do algoritmo do gradiente descendente (descrito de forma sucinta na subseção 2.5.1) sendo o algoritmo *Backpropagation* (RUMELHART; HINTON; WILLIAMS, 1988) o mais utilizado. Apesar de amplamente utilizado, o algoritmo *Backpropagation* é computacionalmente custoso, além de estar sujeito a mínimos locais (HUANG; ZHU; SIEW, 2004; HUANG; ZHU; SIEW, 2006; KULAI; VONZUBEN, 2013). As ELMs evitam parte desses problemas, pois os pesos da camada intermediária (e.g. $\mathbf{w}$ — Figura 9) não necessitam ser ajustados. Em vez disso, eles são escolhidos aleatoriamente dentro de um intervalo pré-estabelecido a ser definido pelo usuário, reduzindo assim o custo computacional envolvido no treinamento dos pesos da rede. Quando comparada ao algoritmo *Backpropagation*, as ELMs apresentam um baixo custo computacional na fase de treinamento, além de obter níveis competitivos de desempenho de generalização, razões pelas quais têm atraído o interesse de várias áreas de pesquisa (CAO; LIN; HUANG, 2010; SAVOJARDO; FARISELLI; CASADIO, 2011; YU; CHOI; HUI, 2012; WANG *et al.*,

2013; LEMME *et al.*, 2013; CAMBRIA *et al.*, 2015).

Dado um conjunto de treinamento de N exemplos $(\mathbf{x}_i, y_i)$, o qual pode ser escrito como $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in \mathbb{R}^n$ e $y_i \in \mathbb{R}, i = 1, \dots, N$, uma rede SHLFFNN padrão com m neurônios na camada intermediária e um único neurônio na camada de saída, pode ser matematicamente escrita como (HUANG; ZHU; SIEW, 2006):

$$y_i = \hat{\beta}_0 + \sum_{j=1}^m \hat{\beta}_j f(\mathbf{w}_j^T \mathbf{x}_i + w_{j0}), i = 1, \dots, N \quad (5.1)$$

em que $\mathbf{w}_j = [w_{j1}, w_{j2}, \dots, w_{jn}]^T$ é o vetor de pesos conectando as n unidades de entrada com o j -ésimo neurônio da camada intermediária, $\hat{\boldsymbol{\beta}} = [\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_m]^T$ é o vetor de pesos conectando os m neurônios da camada intermediária com o neurônio da camada de saída, w_{j0} é o *bias* do j -ésimo neurônio da camada intermediária e $\hat{\beta}_0$ é o *bias* do neurônio da camada de saída.

O resultado da ativação de todos os neurônios da camada intermediária para todo o conjunto de treinamento pode ser representado por uma matriz $\mathbf{H}$:

$$\mathbf{H} = \begin{pmatrix} f(\mathbf{w}_1^T \cdot \mathbf{x}_1 + w_{10}) & \dots & f(\mathbf{w}_m^T \cdot \mathbf{x}_1 + w_{m0}) \\ \vdots & \dots & \vdots \\ f(\mathbf{w}_1^T \cdot \mathbf{x}_N + w_{10}) & \dots & f(\mathbf{w}_m^T \cdot \mathbf{x}_N + w_{m0}) \end{pmatrix} \quad (5.2)$$

Adicionando uma coluna de 1's como primeira coluna de $\mathbf{H}$ e aumentando $\hat{\boldsymbol{\beta}}$ para incluir o *bias* do neurônio da camada de saída, a função de transferência de *entrada-saída*, pode ser escrita na forma matricial como:

$$\mathbf{H}\hat{\boldsymbol{\beta}} = \mathbf{y} \quad (5.3)$$

em que $\mathbf{y} = [y_1, y_2, \dots, y_N]^T$. A extensão para $p > 1$ saídas no modelo é simples. O vetor $\hat{\boldsymbol{\beta}}$ torna-se uma matriz de dimensão $(m+1) \times p$ e o vetor $\mathbf{y}$ torna-se uma matriz de dimensão $N \times p$. Uma rede ELM pode ser resumida conforme o Algoritmo 2:

Algoritmo 2: Algoritmo ELM (adaptado de (HUANG; ZHU; SIEW, 2006)).

Entrada: Um conjunto de treinamento $\mathbb{Q} = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \mathbb{R}^n, y_i \in \mathbb{R}, i = 1, \dots, N\}$, uma função de ativação $f(\cdot)$, e o número de neurônios da camada intermediária, m .

Saída: $\hat{\boldsymbol{\beta}}$.

- 1 Aleatoriamente atribuir os pesos de entrada $\mathbf{w}_j$ e *bias* — $w_{j0}, j = 1, \dots, m$.
 - 2 Calcular a matriz de saída da camada intermediária $\mathbf{H}$.
 - 3 Calcular os pesos da camada de saída $\hat{\boldsymbol{\beta}} = \mathbf{H}^\dagger \mathbf{y}$, onde $\mathbf{H}^\dagger$ é a matriz inversa generalizada de Moore-Penrose (HUANG; ZHU; SIEW, 2004).
-

O Algoritmo 2 assume $p = 1$, i.e. um modelo com uma única saída. Para $p > 1$, é necessário substituir $y_i \in \mathbb{R}$ pelo vetor $\mathbf{y}_i = [y_{i1}, y_{i2}, \dots, y_{ip}] \in \mathbb{R}^{1 \times p}$.

5.3 Sistemas de Recomendação e ELMs

A utilização de ELMs no contexto de SR é relativamente recente. [Deng, Zheng e Chen \(2009\)](#) desenvolveram um algoritmo chamado *Real-Time Collaborative Filtering* (RCF), baseado em ELM, que calcula as previsões em tempo real. No contexto de SR, existe um fluxo contínuo de atualizações nos dados (*e.g.* novas avaliações feitas pelos usuários, novos usuários, novos itens, etc). Visando melhorar a acurácia do sistema, o modelo de predição é atualizado de forma incremental utilizando o algoritmo *Online Sequential Learning for ELM* (OS-ELM) ([LIANG et al., 2006](#)) à medida que ocorram atualizações nos dados do sistema. O modelo de regressão não linear chamado *Extreme Learning Machine Combining Matrix Factorization for Collaborative Filtering* (CELMCF) ([SHANG et al., 2013](#)) utiliza uma combinação do modelo *Weighted Nonnegative Matrix Tri-Factorization* (WNMTF) ([GU; ZHOU; DING, 2010](#)) e ELM no intuito de minimizar o problema relacionado com matrizes esparsas no contexto de FC. O algoritmo *ELM based Recommendation* (ELMR) ([BEDI et al., 2014](#)) foi desenvolvido para fornecer recomendações de filmes online (via Twitter). A abordagem é baseada em padrões de comportamento dos usuários e suas interações (baseadas em *tweets* e *retweets*) que expressam as avaliações dos usuários sobre os filmes. Um novo algoritmo de *ranking* regularizado baseado em combinações de diferentes funções de ativações ELM (ELMRank) proposto por [Chen et al. \(2014\)](#) é utilizado na tarefa de recomendação, gerando uma lista ordenada de itens ainda não vistos pelo usuário do sistema com base nas suas preferências.

5.4 Algoritmo SCOAL-ELM

Esta seção apresenta uma extensão do algoritmo SCOAL (Capítulo 3), baseada em ELMs. Esta extensão substitui o modelo linear do algoritmo SCOAL por uma ELM. Isto é feito substituindo a Equação 3.1 pela Equação 5.3. Dessa forma, a linha 3 do Algoritmo 1 “*Construir um modelo de regressão linear para cada bicluster*” é modificada para “*Construir um modelo ELM (Algoritmo 2) para cada bicluster*”. Cada modelo é agora uma ELM, na qual o número de neurônios da camada intermediária é definido *a priori*.

Considerando a Tabela 1 num contexto de recomendação de filmes, depois de rearranjar as linhas e colunas da matriz $\mathbf{Y}$ e considerando $U = 2$ e $V = 2$, a saída do SCOAL retorna dois grupos de usuários $\mathbf{C}_1^u = \{\text{usuário 1, usuário 5}\}$, $\mathbf{C}_2^u = \{\text{usuário 2, usuário 3, usuário 4}\}$, dois grupos de filmes $\mathbf{C}_1^v = \{\text{filme 4, filme 1, filme 2}\}$, $\mathbf{C}_2^v = \{\text{filme 3, filme 5}\}$ e quatro modelos de predição, sendo cada um uma ELM. Cada modelo é responsável por fazer as previsões de um grupo específico de filmes para um grupo específico de usuários, tal como mostrado na Figura 10.

Figura 10 – Saída do algoritmo SCOAL-ELM com $U = V = 2$.

Para melhorar o desempenho de generalização e tornar a solução mais robusta, foi adicionado um termo de regularização como proposto por [Huang, Zhu e Siew \(2006\)](#). Mais precisamente a linha 3 do Algoritmo 2 foi substituída por:

$$\hat{\beta} = (\mathbf{H}^T \mathbf{H} + \lambda \mathbf{I})^{-1} \mathbf{H}^T \mathbf{y} \quad (5.4)$$

Em 5.4, $\mathbf{I}$ representa a matriz identidade e λ é o parâmetro de regularização usado para reduzir a norma dos pesos de saída. Resultados empíricos apresentados por [Bartlett \(1998\)](#) confirmam que a redução na norma dos pesos da camada de saída é mais relevante para a capacidade de generalização do que a otimização do número de neurônios da camada intermediária.

5.4.1 Avaliação Experimental

Esta seção apresenta os resultados obtidos com a abordagem proposta (SCOAL-ELM — seção 5.4) bem como comparações com uma implementação do algoritmo SCOAL na sua forma original baseado em Regressão Linear (SCOAL-LR). As bases de dados Netflix, Jester e Movielens (descritas na seção 4.4.1) foram utilizadas nos experimentos.

5.4.1.1 Parametrização dos Experimentos

A fim de avaliar a abordagem proposta, o algoritmo SCOAL foi parametrizado de forma arbitrária em três diferentes situações, gerando 4 biclusters, 8 biclusters e 16 biclusters. A Tabela 7 mostra o número de grupos de linha (U) e grupos de coluna (V) adotados em cada base de dados. Na prática, estes parâmetros podem ser otimizados de diferentes maneiras, mas tal estudo esta fora do âmbito desta tese. Em particular, o foco deste capítulo é a utilização de ELMs como modelos locais de predição para SR.

Tabela 7 – Valores de U e V adotados para gerar os biclusters.

	Netflix	Jester	Movielens
4 biclusters	$U = 2, V = 2$	$U = 2, V = 2$	$U = 2, V = 2$
8 biclusters	$U = 2, V = 4$	$U = 2, V = 4$	$U = 2, V = 4$
16 biclusters	$U = 4, V = 4$	$U = 8, V = 2$	$U = 4, V = 4$

O critério de convergência foi definido como a diferença máxima relativa do MSE em duas iterações sucessivas do SCOAL sendo menor ou igual a 10^{-3} . Foram utilizados cinco diferentes valores de λ no algoritmo SCOAL-ELM a fim de alcançar uma boa performance de generalização, a saber: $\{2^{-20}, 2^{-10}, 2^{-2}, 2^{-1}, 2^2\}$. Para encontrar o melhor valor de λ no algoritmo SCOAL-LR, foram usados 20 diferentes valores — $\{2^{-10}, 2^{-9}, \dots, 2^{10}\}$. Estes valores foram escolhidos a partir de experimentos previamente realizados, utilizando o intervalo de valores proposto por Huang *et al.* (2012). A Tabela 8 apresenta os resultados obtidos.

Tabela 8 – Melhores valores obtidos para λ usando a metodologia baseada em Huang *et al.* (2012)

Base de dados	SCOAL-ELM	SCOAL-LR
Netflix-4	2^{-10}	2^6
Netflix-8	2^{-10}	2^2
Netflix-16	2^{-10}	2^3
Movielens-4	2^{-2}	2^3
Movielens-8	2^{-2}	2^6
Movielens-16	2^{-2}	2^4
Jester-4	2^{-10}	2^8
Jester-8	2^{-10}	2^8
Jester-16	2^{-10}	2^3

O número de neurônios da camada intermediária do SCOAL-ELM foi configurado para 150 em todos os experimentos. Tal configuração foi baseada em experimentos preliminares executados variando-se a quantidade de neurônios da camada intermediária. Os pesos da camada intermediária foram inicializados aleatoriamente a partir de uma distribuição normal $\mathcal{N}(0, 0, 025)$ para as bases Netflix e Movielens e, para os experimentos realizados com a base de dados Jester,

a partir de uma distribuição normal $\mathcal{N}(0, 0, 1)$. Este intervalo foi escolhido com base em estudos realizados por Kulaif e VonZuben (2013), que mostram que não é aconselhável inicializar os pesos das ELMs em intervalos maiores — essencialmente devido à sua natureza aleatória. Devido às diferentes características das bases de dados, incluindo o número de amostras e o número de atributos, a escala dos atributos dos dados de entrada foi transformada para o intervalo $[-1, +1]$. A função de ativação adotada foi a tangente hiperbólica. Os experimentos foram realizados utilizando o método de validação cruzada com 10 pastas.

5.4.1.2 Resultados Obtidos

A fim de avaliar o algoritmo SCOAL-ELM e compará-lo com o SCOAL-LR, foram utilizadas medidas de qualidade comumente adotadas na avaliação de SR (BOBADILLA *et al.*, 2013). A adoção de medidas variadas permite a avaliação de diferentes aspectos de um sistema de recomendação. Os aspectos avaliados foram:

- **Qualidade da predição:** Este aspecto avalia a acurácia dos resultados de um SR. As medidas utilizadas para avaliar este aspecto foram: NMAE (GOLDBERG *et al.*, 2001), MSE e RMSE (WILLMOTT; MATSUURA, 2005).
- **Conjunto de recomendações:** Ao invés de analisar avaliações de forma individual, este aspecto mede a eficácia do SR na capacidade de fazer recomendações úteis para o usuário. Para avaliar este aspecto do sistema, três medidas foram empregadas: Precisão, Revocação (BUCKLAND; GEY, 1994) e F-Score (medida F1) (SOKOLOVA; JAPKOWICZ; SZPAKOWICZ, 2006).
- **Lista de recomendação gerada:** Este aspecto está relacionado com a capacidade do sistema em gerar uma lista de recomendação ordenada com base nos interesses dos usuários. A métrica R-Score (BREESE; HECKERMAN; KADIE, 1998) captura a utilidade da lista gerada.

Todas as medidas utilizadas nos experimentos realizados foram previamente discutidas na Seção 2.6. A seguir, são abordados os resultados obtidos para cada uma das medidas anteriormente mencionadas, começando a discussão pelas medidas de predição que, embora não sejam tão atraentes como as demais (segundo a visão do usuário do sistema), são provavelmente as mais populares na literatura. A Figura 11 resume os resultados obtidos para a medida NMAE. Na base de dados Netflix, o algoritmo SCOAL-ELM foi melhor do que o SCOAL-LR nos experimentos com 4 e 8 biclusters. Nas bases Jester e Movielens, o algoritmo SCOAL-ELM supera o SCOAL-LR em todos os experimentos realizados.

Figura 11 – Valores do Erro Médio Absoluto Normalizado (NMAE).

Os resultados obtidos para Precisão são mostrados na Figura 12. O algoritmo SCOAL-ELM superou o algoritmo SCOAL-LR na maioria dos experimentos, com exceção da base Movielens com 8 biclusters e Netflix com 16 biclusters.

Figura 12 – Valores de precisão.

A Tabela 9 possibilita que seja feita uma comparação detalhada dos resultados obtidos para todas as medidas investigadas. Pode ser visto que o algoritmo SCOAL-ELM fornece uma melhor predição, tomando por base os valores obtidos para as medidas NMAE, MSE e RMSE.

Além disso, o algoritmo SCOAL-ELM apresenta uma maior Precisão, que é uma propriedade particularmente desejável para SR, onde os erros de predição (*e.g.* NMAE, MSE e RMSE) podem ser tolerados desde que tais erros não conduzam o SR a fazer recomendações ruins. Em outras palavras, o principal objetivo de um SR é fazer boas recomendações para os usuários do sistema. Nesse contexto, erros de predição são, em certa medida, toleráveis contanto que eles não prejudiquem as recomendações.

Tabela 9 – Resumo dos resultados obtidos para todas as medidas investigadas. Melhores resultados são destacados em negrito.

Base	SCOAL	NMAE	Precisão	Revocação	F-score	MSE	RMSE	R-Score
Netflix 4	ELM	0,1752	0,7404	0,7698	0,7547	0,8018	0,8952	0,7138
	LR	0,1755	0,7388	0,7704	0,7541	0,8027	0,8958	0,7224
Netflix 8	ELM	0,1755	0,7385	0,7710	0,7543	0,8077	0,8985	0,7034
	LR	0,1756	0,7376	0,7731	0,7548	0,8090	0,8992	0,7084
Netflix 16	ELM	0,1739	0,7411	0,7743	0,7571	0,7930	0,8903	0,7197
	LR	0,1735	0,7431	0,7728	0,7575	0,7896	0,8884	0,7066
Jester 4	ELM	0,1660	0,7218	0,3087	0,4322	17,5987	4,1946	0,6277
	LR	0,1787	0,7150	0,2488	0,3676	20,0624	4,4789	0,4193
Jester 8	ELM	0,1659	0,7429	0,3053	0,4324	17,6124	4,1962	0,5801
	LR	0,1706	0,7212	0,2896	0,4128	18,5949	4,3116	0,5800
Jester 16	ELM	0,1643	0,7181	0,3360	0,4574	17,4076	4,1717	0,6597
	LR	0,1666	0,6992	0,4629	0,5568	17,9473	4,2358	0,4254
Movielens 4	ELM	0,1917	0,7195	0,7316	0,7254	0,9284	0,9632	0,6880
	LR	0,1920	0,7155	0,7347	0,7249	0,9295	0,9638	0,6944
Movielens 8	ELM	0,1929	0,7141	0,7330	0,7233	0,9429	0,9707	0,6782
	LR	0,1936	0,7156	0,7262	0,7208	0,9443	0,9714	0,6769
Movielens 16	ELM	0,1887	0,7241	0,7387	0,7312	0,9072	0,9523	0,6946
	LR	0,1888	0,7215	0,7404	0,7307	0,9050	0,9511	0,7010

Note que o algoritmo SCOAL-ELM alcança resultados semelhantes aos do algoritmo SCOAL-LR para a Revocação, mas o SCOAL-ELM ainda supera o SCOAL-LR quando a medida F-Score é considerada. Finalmente, os resultados do SCOAL-ELM para o R-Score são ligeiramente melhores do que os obtidos pelo algoritmo SCOAL-LR. A Tabela 10 resume o número de vitórias para cada algoritmo, computada para cada medida apresentada na Tabela 9.

Tabela 10 – Número de vitórias para cada algoritmo considerando as medidas da Tabela 9.

Medida	SCOAL-ELM	SCOAL-LR
NMAE	8	1
Precisão	7	2
Revocação	4	5
F-score	6	3
MSE	7	2
RMSE	7	2
R-Score	5	4

No tocante aos custos computacionais, note que a complexidade do método *Ridge Regression* é $O(n^2 \cdot N + n^3)$ sendo n e N respectivamente o número de variáveis independentes e exemplos, — como definido na Seção 5.2. Visto que geralmente $N \gg n$, o custo computacional

do *Ridge Regression* é $O(n^2 \cdot N)$. Para treinar uma ELM, tem-se um custo computacional adicional de $O(N \cdot n \cdot m)$, sendo m o número de neurônios da camada intermediária. Assim, em termos assintóticos, o treinamento de uma ELM é $O(m^2 \cdot N)$, sendo usualmente $N \gg m > n$. As magnitudes dos termos constantes, negligenciadas pela análise de complexidade de tempo, são mostradas na Tabela 11. Como esperado, as abordagens baseadas em ELM consomem mais tempo computacional do que as baseadas em regressão linear.

Tabela 11 – Tempo computacional (em minutos): Macbook com 8 GB de memória e 4 núcleos Intel Core i7 2GHz.

		Netflix	Jester	Movielens
SCOAL-LR	4 biclusters	4,80	7,08	0,82
	8 biclusters	8,01	9,72	0,83
	16 biclusters	11,89	15,02	0,94
SCOAL-ELM	4 biclusters	37,45	116,19	32,15
	8 biclusters	63,43	128,47	40,51
	16 biclusters	121,84	217,98	69,35

5.5 Considerações Finais

Neste capítulo foi apresentada uma abordagem híbrida de recomendação que combina o algoritmo SCOAL (Capítulo 3) e ELM. Foram realizados experimentos com três bases de dados amplamente utilizadas na avaliação de SR. Os resultados obtidos são promissores para várias medidas que são relevantes em aplicações práticas de recomendação. Em particular a abordagem proposta fornece recomendações muito precisas, sendo assim elegíveis para fazer parte de um rol de ferramentas a serem utilizadas na prática.

As comparações feitas com modelos lineares mostram, como esperado, que não existe um método vencedor para todas as bases de dados. A partir desta perspectiva, um ponto interessante de investigação envolve a combinação de ELMs e modelos lineares em *ensembles*, os quais geralmente produzem melhores predições. No entanto, é provável que questões relacionadas com custos computacionais possam limitar as potenciais vantagens desta abordagem, como aliás, já observado no treinamento das ELMs.

AGRUPAMENTO DE DADOS E APRENDIZADO DE FUNÇÕES DE REGRESSÃO EM PARALELO UTILIZANDO OPENMP

6.1 Considerações Iniciais

Além da capacidade de predição, um outro fator que atualmente deve ser levado em consideração no desenvolvimento de SR está relacionado com a escalabilidade do sistema. SR têm sido empregados em centenas de diferentes sítios de *e-commerce* onde as recomendações devem ser feitas *on-line*. Apesar das abordagens de recomendação baseadas em modelos possibilitarem que as predições sejam feitas em tempo real, um outro aspecto que deve ser considerado diz respeito à dinamicidade envolvida em SR aplicados em *e-commerce*. Centenas ou até mesmo milhares de avaliações são feitas por segundo, sem contar com a chegada de novos usuários e novos itens que são catalogados no sistema a todo instante. Num contexto de recomendação de músicas, por exemplo, surgem novos estilos, bandas e cantores frequentemente. Isto faz com que mesmo as abordagens baseadas em modelos tenham que constantemente atualizar os seus modelos de predição, a fim de manter a acurácia preditiva do sistema.

Como visto anteriormente, um dos pontos fortes do algoritmo SCOAL é justamente a sua capacidade de prover modelos de predição específicos para um subconjunto de usuários e/ou itens. Encontrar formas de paralelizar o treinamento de tais modelos é essencial para que as variantes do algoritmo SCOAL propostas nesta tese possam efetivamente fazer parte de um SR aplicado em domínios *on-line* de larga escala.

Neste capítulo são apresentadas versões paralelizadas dos algoritmos SCOAL-LR e SCOAL-ELM, implementadas utilizando o modelo de programação paralelo OpenMP, um

modelo de programação portátil e escalável que proporciona aos desenvolvedores uma interface flexível para o desenvolvimento de aplicações paralelas. A próxima seção descreve de forma sucinta as arquiteturas de memória e os principais modelos de programação paralela.

6.2 Tecnologias para programação paralela

Na programação paralela, a memória pode ser compartilhada, distribuída ou assumir uma arquitetura híbrida *distribuída-compartilhada* (BARNEY *et al.*, 2010). Na arquitetura de memória compartilhada, todos os processadores acessam toda a memória como um espaço de endereçamento global. Múltiplos processadores podem operar de forma independente, mas compartilham os mesmos recursos de memória. Mudanças em um endereço de memória feitas por um processador são visíveis para todos os outros processadores. Arquiteturas baseadas em memória compartilhada são classificadas como: *Uniform Memory Access* (UMA) e *Non-uniform memory access* (NUMA). A arquitetura UMA também é conhecida por multiprocessamento simétrico — *Symmetric Multi-Processing* (SMP) — pois todos os processadores são idênticos, o que garante que os tempos de acesso a uma dada região de memória obtida por um determinado processador sejam iguais para todos os demais, independentemente da posição de memória a ser acessada, ou do processador que está fazendo o acesso. Já na arquitetura NUMA, os tempos de acesso à memória dependem tanto da posição de memória, como do processador que está fazendo o acesso.

Na arquitetura distribuída, os processadores têm sua própria memória local. Os endereços de memória de um processador não são mapeados para outro processador e, por esta razão, não existe o conceito de espaço de endereço global entre todos os processadores (KANG; LEE; LEE, 2014). Alterações feitas por um processador na sua memória local não afetam as memórias dos outros processadores. Quando existe a necessidade de um processador acessar dados em outro processador, cabe ao programador definir explicitamente como e quando os dados são comunicados. A comunicação entre processadores é feita por intermédio de uma rede de comunicação que interliga os processadores, cuja topologia é um fator determinante na escalabilidade da arquitetura distribuída. A sincronização entre as tarefas também é responsabilidade do programador.

A arquitetura híbrida *distribuída-compartilhada* fornece uma abstração de memória compartilhada numa arquitetura na qual, de fato, as memórias estão distribuídas fisicamente. Ela engloba um conjunto de nós ou *clusters* conectados por uma rede. Os processadores de um mesmo *cluster* compartilham sua memória local, ao passo que a rede de comunicação é utilizada para mover dados de um nó (ou *cluster*) para outro (PROTIC; TOMASEVIC; MILUTINOVIC, 1998). Os maiores e mais rápidos computadores no mundo atualmente empregam ambas as arquiteturas de memória compartilhada e distribuída (BARNEY *et al.*, 2010). Apesar de tal arquitetura apresentar como principal vantagem a sua escalabilidade, um fator importante a ser

levado em consideração é o aumento na complexidade do desenvolvimento de aplicações para tais domínios.

6.2.1 *Arquitetura multi-core*

Na medida em que os processadores *single core* tornaram-se mais rápidos, os problemas relacionados à dissipação de calor e o uso eficiente de paralelismo em nível de instrução — *Instruction Level Parallelism* (ILP) — ficaram mais evidentes. Os principais desenvolvedores como a Intel têm mudado para a arquitetura *multi-core* (RAMANATHAN, 2006). Apesar de processadores *multi-core* possuírem uma taxa de *clock* menor dos que os processadores *single core*, isto é compensado com as inovações incorporadas na sua arquitetura. Estas inovações proporcionam um alto valor agregado para os processadores *multi-core*. O conceito da arquitetura *multi-core* é principalmente centrado na possibilidade da computação paralela, que pode aumentar significativamente a velocidade do computador através da inclusão de dois ou mais núcleos de processamento em um único chip. Processadores *multi-core* podem oferecer benefícios significativos de desempenho, adicionando capacidade de processamento com latência mínima, dada a proximidade dos processadores. A tecnologia *multi-core* é muito eficaz em tarefas e aplicações que demandam um alto grau de processamento. E entretanto, a quantidade de desempenho obtido com a utilização de processadores *multi-core* é fortemente dependente do desenvolvimento de softwares orientados ao processamento paralelo.

6.2.2 *Many Integrated Core*

A arquitetura *Many Integrated Core* (MIC), desenvolvida pela empresa Intel, combina vários *cores* dentro de um único chip. A arquitetura MIC é fortemente direcionada para aplicações que demandem computação de alto desempenho — *High Performance Computing* (HPC). Um dos pontos forte desta arquitetura é o de proporcionar um ambiente de programação de propósito geral semelhante ao ambiente de programação dos processadores da família Xeon. Os coprocessadores Intel Xeon Phi baseados na arquitetura MIC mais atuais possuem mais de 60 *cores* e 16GB de memória GDDR5, além de um sistema operacional Linux capaz de executar programas escritos em linguagens de programação amplamente utilizadas na programação paralela, tais como linguagem C, C++ e Fortran (CHRYSSOS, 2014). Por este motivo, aplicações podem usar os processadores Xeon e os coprocessadores Xeon Phi de forma simultânea, contribuindo com a performance da aplicação.

As aplicações devem fazer uso de um coprocessador em situações nas quais este realmente possa contribuir com o desempenho do nó ou servidor —*e.g.*, situações que envolvam cálculos matemáticos tais como operações com matrizes. Antes de tentar tirar proveito dos recursos do coprocessador Xeon Phi, deve-se, primeiro, maximizar todos os recursos oferecidos pelo processador Xeon. Segundo Jeffers e Reinders (2013), tentar usar um coprocessador Xeon Phi sem ter maximizado o uso do paralelismo no processador Xeon será quase sempre uma

decepção. O coprocessador Xeon Phi está conectado a uma arquitetura Xeon *multi-core* (host), através de um barramento *Peripheral Component Interconnect* (PCI) *Express* permitindo que o usuário acesse o coprocessador com um nó da rede. Isto permite que o usuário esteja conectado diretamente no coprocessador, executando aplicações nativas, além da execução de aplicações heterogêneas, com partes de instruções sendo executadas no host e outras no coprocessador.

6.2.3 OpenMP

O modelo de programação OpenMP reúne um conjunto de diretivas de compilação, rotinas de biblioteca e algumas variáveis de ambiente que são utilizadas para proporcionar um alto nível de paralelismo em programas escritos em C, C++ e Fortran para a paralelização de programas numa arquitetura de memória compartilhada. Com o modelo OpenMP, é possível fazer uma paralelização incremental de um código serial existente, permitindo que se avalie num curto espaço de tempo o ganho obtido com a região paralelizada. Este talvez seja o grande diferencial em relação a outros paradigmas de programação que exigem uma grande modificação do código (CHAPMAN; JOST; PAS, 2008).

Um programa contendo diretivas OpenMP começa sua execução como uma simples *thread* em execução. A execução segue sequencialmente até encontrar a primeira diretiva, declarando o início de uma região paralela — num programa C ou C++, esta diretiva tem a seguinte sintaxe: `#pragma omp parallel`. Quando a *thread* encontra esta diretiva, ela cria um grupo de *threads*, passando a ser a *thread master* deste grupo. Todas as instruções do programa contidas na região paralela são executadas em paralelo por cada *thread* do grupo, incluindo todas as rotinas externas chamadas de dentro da região paralela. Quando uma *thread* encontra o final da região paralela, ela espera até todas as outras *threads* do grupo chegarem. Quando todas as *threads* do grupo completam a execução do código na região paralela, ocorre uma sincronização entre elas e o grupo de *threads* é dissolvido, restando a *thread* inicial que segue na execução do código até que uma nova região paralela seja encontrada. Conforme descrito anteriormente, não basta apenas declarar uma região paralela pois, neste caso, todas as *threads* vão executar as mesmas instruções em paralelo, gerando apenas retrabalho. As diretivas mais utilizadas do modelo OpenMP são as diretivas responsáveis pela distribuição do trabalho entre as *threads* — *single*, *sections* e *for*.

- **#pragma omp single:** Serve para indicar que o bloco de instruções abaixo dessa diretiva deve ser executado apenas por uma única *thread*, ou seja, a primeira *thread* que atingir esse ponto de execução. As outras *threads* desviam dessa região, e ficam esperando no final do bloco *single*, até que a *thread* que está executando as instruções encontre com as outras no final do bloco. A partir daí todas continuam a execução dentro da região paralela.
- **#pragma omp sections:** Utilizado para dividir tarefas entre as *threads* em blocos de código que não possuem iterações. Dentro do bloco da diretiva *sections* é necessário

incluir a diretiva `#pragma omp section`. Cada diretiva `section` inserida será executada por uma única `thread`.

- **# pragma omp for:** É talvez a diretiva mais utilizada dentro de uma região paralela. Faz com que as iterações da estrutura de repetição situada logo abaixo da diretiva sejam executadas em paralelo. As iterações são distribuídas entre as `threads` do grupo criado na região paralela onde o laço se encontra.

Por ser um modelo de programação amplamente utilizado, existe um vasto conteúdo disponibilizado para treinamento *on-line* além de livros específicos sobre o modelo (CHANDRA, 2001; CHAPMAN; JOST; PAS, 2008).

6.2.4 OpenACC

OpenACC é um modelo de programação paralela baseado em diretivas de compilação criado para facilitar o desenvolvimento de aplicações massivamente paralelas. O modelo oferece desempenho e portabilidade para principais plataformas de computação paralela — *e.g.* host-GPU, *multi-core* e *many-core*. As diretivas de compilação inseridas em programas escritos em C, C++ e Fortran, simplificam grande parte da complexidade envolvida na transferência dos dados entre o host e os aceleradores e no mapeamento do paralelismo e escalonamento das `threads`. Por esta razão, OpenACC se apresenta como uma alternativa na migração de aplicações "legadas". Programas científicos com muitas linhas de código podem ser portados para arquiteturas baseadas em aceleradores sem que haja a necessidade da reescrita de todo o código (KIRK; WEN-MEI, 2012).

6.2.5 OpenCL

O modelo OpenCL — *Open Computing Language* — reúne uma série de funcionalidades, incluindo um conjunto de interfaces de programação, uma camada para a abstração de hardware, além de uma linguagem comum de programação, utilizadas para o desenvolvimento de aplicações paralelas que podem ser executadas num conjunto de diferentes fabricantes de hardware. O modelo OpenCL abstrai CPUs, GPUs e outros aceleradores simplesmente como "dispositivos" que contenham um ou mais *cores* de processamento (STONE; GOHARA; SHI, 2010). O desenvolvimento do OpenCL foi motivado pela necessidade de um ambiente unificado para o desenvolvimento de aplicações de alto desempenho capazes de serem executadas de forma conjunta em diferentes arquiteturas de computação paralela atualmente disponíveis. Inicialmente desenvolvido pela empresa Apple, atualmente é mantida e gerenciada pelo grupo Khronos¹, o mesmo grupo que gerencia o padrão OpenGL (KIRK; WEN-MEI, 2012).

¹ <https://www.khronos.org/opencl/>

6.2.6 GPUs e CUDA

A evolução no desenvolvimento dos consoles e jogos alavancou a necessidade de um poderoso hardware computacional dedicado ao processamento de milhares de cálculos matemáticos envolvidos num jogo 3D. Na corrida para atender a este mercado milionário, fabricantes de dispositivos gráficos desenvolveram processadores capazes de executar processamento massivamente paralelo, com altas taxas de transferência e memória de alta performance essenciais para a exibição de gráficos de alta resolução. No entanto, esses dispositivos de *hardware* têm o potencial para serem reaproveitados e usados em outros domínios de aplicação. A computação acelerada por *Graphics Processing Unit* (GPU) é uma arquitetura que reúne uma ou mais unidades GPUs, juntamente com uma CPU para acelerar o processamento de áreas que demandem de computação de alto desempenho (HPC) (NVIDIA, 2007).

Uma GPU é composta por até milhares de núcleos, projetados para executar uma tarefa em múltiplas *threads*. Visando facilitar o desenvolvimento de aplicações para esta arquitetura, a empresa NVIDIA criou um modelo de programação conhecido por *Compute Unified Device Architecture* (CUDA) (NVIDIA, 2008) o qual permite a programação direta ao hardware NVIDIA. O modelo CUDA utiliza uma abordagem híbrida de processamento paralelo na qual partes do código são executadas no host, e outras conhecidas por *kernels* são transferidas e executadas nas GPUs.

6.2.7 MPI

A principal diferença entre programar para um único sistema e um *cluster* é que cada nó do *cluster* tem seu espaço de endereçamento de memória separado. Diferentemente de múltiplas *threads* sendo executadas numa arquitetura de memória compartilhada, a comunicação entre espaços de memória disjuntos (arquitetura distribuída) usualmente requer que o programador defina de forma explícita as chamadas de comunicação. Tais chamadas ocorrem via mensagens utilizando uma especificação de comunicação — *Message Passing Interface* (MPI) — que foi definida através de um fórum aberto de discussão que congrega uma comunidade de fabricantes de hardware para computação paralela, cientistas da computação, e desenvolvedores de sistemas (FORUM, 2015). Ao contrário do que muitos pensam, MPI não é uma linguagem de programação, ela é uma biblioteca cujas operações são expressas como funções ou métodos que podem ser chamados por programas escritos em C, C++ ou Fortran. Programas que utilizam MPI geralmente empregam a abordagem *Single Program Multiple Data* (SPMD) na qual tarefas são divididas e executadas simultaneamente em vários processadores a fim de obter resultados mais rápidos. Múltiplas instâncias (MPI *ranks*) de um mesmo programa são executadas concorrentemente. Cada *rank* computa uma tarefa diferente e utiliza a MPI para fazer a comunicação de dados entre os *ranks* (JEFFERS; REINDERS, 2013).

6.2.8 MapReduce

O modelo de programação MapReduce é projetado para executar programas em grandes *clusters* de forma eficiente, explorando o paralelismo de dados (DEAN; GHEMAWAT, 2008). O paradigma MapReduce tem atraído a atenção da academia e da indústria em razão de ocultar os detalhes da paralelização do código, permitindo que os desenvolvedores se concentrem apenas em estratégias de processamento de dados (LEE *et al.*, 2012). A estrutura de programação utilizada é feita com base na programação funcional, na qual o programador desenvolve a lógica de programação de duas funções:

- **Map:** responsável pelo processamento de cada par (*chave_in*, *valor_in*) dos dados de entrada gerando uma lista intermediária de pares no formato (*chave_inter*, *valor_inter*). Os pares intermediários (*chave_inter*, *valor_inter*) são então agrupados numa estrutura de dados no formato (*chave_inter*, *lista(valor_inter)*).
- **Reduce:** recebe como parâmetros uma chave intermediária e a sua lista dos valores correspondentes (*chave_inter*, *lista(valor_inter)*). Os valores intermediários são então combinados, finalmente resultando em um par de saída (*chave_out*, *valor_out*).

O modelo MapReduce faz uso de um sistema de arquivos otimizado para processamento distribuído de grandes bases de dados em computadores de baixo custo. Na sua implementação original proposta por Dean e Ghemawat (2008) o sistema de arquivos utilizado foi uma sistema proprietário chamado *Google File System* (GFS) (GHEMAWAT; GOBIOFF; LEUNG, 2003). Uma das *frameworks open-source* mais populares desse modelo é Apache Hadoop, cujos módulos incluem uma implementação MapReduce em Java e um sistema de arquivos distribuído — *Hadoop Distributed File System* (HDFS) (LAM, 2010).

6.3 Algoritmo Proposto

Das diferentes tecnologias para paralelização abordadas anteriormente, um fato que deve ser ressaltado é que não existe uma tecnologia "melhor", embora certamente há melhores implementações de algumas tecnologias em detrimento de outras. Esta seção apresenta as versões paralelizadas dos algoritmos SCOAL-LR e SCOAL-ELM utilizando OpenMP numa arquitetura HPC. A adoção de OpenMP tomou por base os seguintes critérios:

- **Facilidade de uso:** OpenMP oferece um conjunto reduzido de diretivas para serem aplicadas em regiões de código cujas instruções podem ser compartilhadas entre os processadores. Desde que a linguagem de programação utilizada seja C, C++ ou Fortran, é possível obter um alto nível de paralelização fazendo intervenções mínimas no código serial.

- **Estrutura do algoritmo:** Como foi descrito no Capítulo 3, o algoritmo SCOAL consiste de um conjunto de iterações sobre três passos principais até que o critério de convergência seja atingido: (i) construir um modelo de regressão para cada *bicluster*, (ii) atualização dos *clusters* de linha, (iii) atualização dos *clusters* de coluna. A independência de algumas operações existentes nesses passos possibilita a paralelização do SCOAL. Entretanto, o número de iterações necessárias para que o critério de convergência seja atingido e consequentemente os modelos de predição sejam gerados, é da ordem de centenas ou milhares de repetições dependendo da complexidade da base de dados. Esse grande número de iterações prejudica de forma significativa a paralelização do algoritmo SCOAL numa arquitetura híbrida, na qual o processamento seria distribuído entre vários computadores interconectados por uma rede. Tal fato pode ser observado numa versão do algoritmo SCOAL baseada em MapReduce proposta por Deodhar, Jones e Ghosh (2010) na qual apesar do *speedup* obtido ser bom, os tempos absolutos de execução ainda são altos devido ao *overhead* e à lentidão apresentados pelo Hadoop em tais cenários (PERERA, 2013).
- **Arquitetura e Portabilidade:** Além de OpenMP ser o modelo de programação paralela mais utilizado em arquiteturas de memória compartilhada, este oferece ainda a portabilidade de código, possibilitando que uma mesma aplicação seja executada tanto num notebook como numa arquitetura HPC. Os programas desenvolvidos para esta tese foram inicialmente executados num notebook e posteriormente executados numa arquitetura HPC sem que fosse necessária nenhuma alteração do código fonte, bastando apenas recompilar os programas. Cabe ressaltar que isto não significa dizer que a aplicação atingirá o máximo de escalabilidade que a arquitetura HPC oferece.

Para agilizar os cálculos matemáticos envolvidos no aprendizado dos modelos de predição, foi utilizada a biblioteca *Math Kernel Library* (MKL) da empresa Intel. A biblioteca MKL foi desenvolvida no sentido de ajudar os programadores a tirarem o máximo proveito da computação de alto desempenho envolvendo as arquiteturas *multi-core* e *many-core*. A biblioteca disponibiliza um conjunto otimizado de funções matemáticas com versões disponíveis para as arquiteturas 32 e 64 bits, podendo ser executada nos principais sistemas operacionais (*e.g.* Windows, Linux e Mac OS) (WANG *et al.*, 2014). As funções matemáticas são executadas em paralelo, explorando todos os recursos de hardware disponíveis, de forma transparente para o desenvolvedor. Apesar de maximizar a paralelização dos cálculos matemáticos de forma transparente para desenvolvedor, a sintaxe utilizada nas funções MKL pode parecer pouco amigável. O código 1 exemplifica a sintaxe envolvida na multiplicação de duas matrizes genéricas.

Código-fonte 1: Exemplo de funções MKL utilizadas na multiplicação de matrizes

```
1 #include "mkl.h"
2 void int main(){
3     double *A, *B, *C;
```

```

4     int m; //número de linhas das matrizes A e C;
5     int n; //número de colunas das matrizes B e C;
6     int k; //número de colunas da matriz A e número de linhas da
      matriz B;
7     double alpha=1.0;
8     double beta=0;
9     //===== Alocação de memória para as matrizes =====
10    A = (double *)mkl_malloc( m*k*sizeof( double ), 64 );
11    B = (double *)mkl_malloc( k*n*sizeof( double ), 64 );
12    C = (double *)mkl_malloc( m*n*sizeof( double ), 64 );
13    //A função cblas_dgemm abaixo,
14    //realiza o seguinte calculo C=alpha*A*B+beta*C;
15    cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, m, n, k,
16               alpha, A, k, B, n, beta, C, n);
17    //CblasRowMajor indica que as matrizes estão ordenadas por linha
18    //CblasNoTrans indica que as matrizes A e B não devem ser
      transpostas
19    //liberação da memória alocada
20    mkl_free(A);
21    mkl_free(B);
22    mkl_free(C);
23 return 0
24 }

```

Visando abstrair a complexidade envolvida na chamada de funções MKL, além de manter uma melhor legibilidade do programa fonte, todos os cálculos envolvendo matrizes foram feitos utilizando a biblioteca Armadillo C++. Trata-se de uma biblioteca de álgebra linear de código aberto cujo objetivo é manter o equilíbrio entre velocidade e facilidade de uso, sendo extremamente útil se a linguagem de programação adotada for C++ ([SANDERSON, 2010](#)). Por ser distribuída sob uma licença que é aplicável tanto em código aberto como em contextos de desenvolvimento de software proprietários, a biblioteca pode ser utilizada seja na prototipagem rápida de experimentos computacionalmente intensivos, ou no desenvolvimento de sistemas comerciais. A biblioteca suporta a representação de números inteiros, reais e complexos, bem como um subconjunto de funções trigonométricas e estatísticas. Várias decomposições matriciais são fornecidas através da integração opcional com a biblioteca *Linear Algebra Package* (LAPACK) ([ANDERSON et al., 1990](#)) ou, com outras bibliotecas de otimização matemática tais como MKL, *AMD Core Math Library* (ACML) ([AMD, 2012](#)) e OPENBLAS ([XIANYI; QIAN; CHOTHIA, 2012](#)). Sua sintaxe é muito semelhante àquelas de outras linguagens amplamente utilizadas em contextos que envolvam cálculos científicos e operações com vetores e matrizes (e.g. Matlab, Octave e R). O código 2 ilustra a sintaxe Armadillo C++ utilizada no mesmo problema de multiplicação de matrizes explorado no Código-fonte 1.

```

1 #include "armadillo.h"
2 void int main(){
3     //declaração das matrizes
4     mat A, B, C;
5     //Atribuição na matriz C do resultado da multiplicação das
      matrizes A e B.
6     C= A * B;
7 return 0;
8 }

```

Além de proporcionar uma maior legibilidade no código, a biblioteca Armadillo C++, quando vinculada a uma biblioteca de otimização matemática como MKL, por exemplo, garante ainda a velocidade no processamento dos cálculos invocando de forma transparente as principais funções responsáveis pela otimização das operações matemáticas. Isto significa dizer que na execução da linha 6 do programa 2 está sendo executada, na realidade, a linha 15 do programa 1.

As próximas subseções descrevem como foram paralelizados os principais passos do algoritmo SCOAL.

6.3.1 Construir um modelo de regressão para cada bicluster

Cada célula y_{ij} da matriz de dados $\mathbf{Y}$ (Tabela 1) que possua uma avaliação ($y_{ij} \neq 0$), está associada a uma posição num vetor, cujo tipo de dados é uma estrutura com os seguintes atributos: o número do *bicluster* ao qual a célula pertence e o vetor $\mathbf{x}_{ij}$ que concatena os atributos dos usuários e itens. Dada a independência dos dados, os modelos de regressão podem ser calculados em paralelo. Para tanto, é definida uma região paralela com uma diretiva OpenMP que faz com que as iterações da estrutura de repetição situada logo abaixo da diretiva sejam executadas em paralelo. As iterações são distribuídas entre as *threads* do grupo criado na região paralela onde o laço se encontra. A quantidade máxima de *threads* criadas na região paralela é limitada pelo valor atribuído à variável de ambiente OMP_NUM_THREADS. No caso dessa variável não ter sido declarada, o número máximo de *threads* criado será igual ao número de *cores* disponíveis. O código 3 ilustra este procedimento.

Código-fonte 3: Construção dos modelos de regressão

```

1 #pragma omp parallel for
2 for(int i=0;i<tot_biclusters;i++){
3     mat X=geraMatriz(i);
4     vetBiclusters[i]=calculaModelo(X);
5 }

```

A função *geraMatriz* é responsável por percorrer todas as células pertencentes ao *bicluster* i , concatenando-as com o seu vetor $\mathbf{x}_{ij}$ associado, gerando uma nova linha na matriz $\mathbf{X}$. Dessa

forma, cada linha da matriz $\mathbf{X}$ armazena um conjunto de atributos que, para o caso do SCOAL-LR, representa o conjunto de variáveis necessárias para o cálculo da regressão linear: as variáveis independentes do modelo ($\mathbf{x}_{ij}$) e a variável dependente do modelo (y_{ij}). A matriz $\mathbf{X}$ é então passada como parâmetro para a função *calculaModelo* que, após calcular o modelo de regressão linear, retorna um vetor com os coeficientes de regressão calculados para o *bicluster* i . Cada posição do vetor *vetBiclusters* armazena um vetor específico de cada modelo calculado.

6.3.2 Atualização dos clusters de linha

Neste passo, efetua-se a somatória dos erros de predição obtidos para todas as células de uma determinada linha, utilizando os modelos de predição pertencentes ao seu *cluster* de linha, e os erros de predição obtidos, utilizando os modelos de predição pertencentes aos demais *clusters* de linha. Caso o erro seja minimizado utilizando os modelos de predição pertencentes a um outro *cluster* de linha, a linha em questão será então removida do seu *cluster* atual e inserida no *cluster* de linha onde o erro de predição é menor.

O código 4 sumariza as operações executadas neste passo. Na linha 4 é declarada uma estrutura de repetição cujo número de iterações é igual a U , o qual representa a quantidade de *clusters* de linha, um dos parâmetros de entrada do algoritmo SCOAL (Algoritmo 1).

Código-fonte 4: Atualização dos clusters de linha

```

1 int    menor;
2 vector<Line>modifica;
3 double erroPredicao[U];
4 for(int i=0;i<U;i++){
5     #pragma omp parallel for private(menor)
6     for(int j=0;j<clusterLinha[i].size();j++){
7         menor=i;
8         for(int k=0;k<U;k++){
9             erroPredicao[k]=calculaPredicao(clusterLinha[i][j],k);
10            if(erroPredicao[k]<erroPredicao[menor]){
11                menor=k;
12            }
13        }
14        if(menor !=i){
15            Line atual;
16            atual.setLin(clusterLinha[i][j]);
17            atual.setMove(menor);
18            atual.setDel(i);
19            modifica.push_back(atual);
20        }
21    } // término da região paralela

```

```

22 }
23 rearranjaLinhas(modifica);

```

Na linha 5, a diretiva *#pragma omp parallel for* cria uma região paralela e distribui as iterações da estrutura de repetição declarada na linha 6 entre o grupo de *threads* criado na região paralela. Dessa forma, o bloco de instruções compreendido entre as linhas 7 e 19 será executado por um grupo de *threads* em paralelo. Na linha 14 existe uma condição que verifica se uma determinada linha deverá ser removida do seu *cluster* de linha atual e transferida para um outro *cluster*. Caso a condição (*menor != i*) retorne verdadeiro, uma instância da classe *Line* é criada (linha 15). A instância de nome *atual* recebe os seguintes parâmetros: o número da linha que será removida (linha 16); o número do *cluster* de linha aonde ela será inserida (linha 17) e o número do *cluster* de linha de onde a linha será removida (linha 18). O objeto é então inserido no vetor *modifica* (linha 19), o qual armazena uma coleção de objetos que possuem todas as atualizações que devem ser feitas nos *clusters* de linha. Terminada a iteração em todos os *U clusters* de linha, a função *rearranjaLinhas* (linha 23) realiza todas as atualizações necessárias.

6.3.3 Atualização dos clusters de colunas

Este passo é análogo ao da atualização dos *clusters* de linha. Entretanto, as iterações que eram feitas na dimensão linha passam agora a ser feitas na dimensão coluna, conforme ilustrado pelo código 5.

Código-fonte 5: Atualização dos clusters de coluna

```

1 int  menor;
2 vector<Column>modifica;
3 double erroPredicao[V];
4 for(int i=0;i<V;i++){
5     #pragma omp parallel for private(menor)
6     for(int j=0;j<clusterColuna[i].size();j++){
7         menor=i;
8         for(int k=0;k<V;k++){
9             erroPredicao[k]=calculaPredicao(clusterColuna[i][j],k);
10            if(erroPredicao[k]<erroPredicao[menor]){
11                menor=k;
12            }
13        }
14        if(menor !=i){
15            Column atual;
16            atual.setCol(clusterColuna[i][j]);
17            atual.setMove(menor);
18            atual.setDel(i);
19            modifica.push_back(atual);
20        }

```

```

21     } // término da região paralela
22 }
23 rearranjaColunas(modifica);

```

Na linha 4 do código 5, o número de iterações da estrutura de repetição é V , que representa o total de *clusters* de coluna. A instância *atual* (linha 15) é da classe *Column*, e o vetor *modifica* (linha 19), armazena uma coleção de objetos que possuem todas as atualizações que devem ser feitas nos *clusters* de coluna. Terminada a iteração em todos os V *clusters* de coluna, a função *rearranjaColunas* (linha 23) efetua todas as atualizações necessárias nos *clusters* de coluna.

6.4 Avaliação Experimental

6.4.1 Parametrização

As versões paralelizadas dos algoritmos SCOAL-LR e SCOAL-ELM foram avaliadas em dois cenários: (i) numa arquitetura *multi-core* composta por 64 *cores* Intel Xeon E5-2699v3 2.3 GHz e (ii) num servidor com arquitetura *multi-core/many-core* composto por 32 *cores* Intel Xeon (E5-2650v2 2.6 GHz) e dois coprocessadores Intel Xeon Phi x100 family com 57 *cores* cada. Os experimentos realizados na arquitetura *multi-core* avaliaram a escalabilidade dos algoritmos na medida em que o número de *threads* é incrementado. Os experimentos no servidor com arquitetura *multi-core/many-core* avaliaram o desempenho da biblioteca MKL em fazer o *offload* automático, recurso que possibilita que chamadas de funções computacionalmente intensivas utilizem os coprocessadores Xeon Phi em conjunto com a arquitetura *multi-core*, de forma automática e transparente. Esta particularidade do MKL permite que os usuários aproveitem os recursos computacionais adicionais fornecidos pelo coprocessador, sem a necessidade de nenhuma modificação no código. O *offload* automático pode ser ativado/desativado por meio de configuração de uma variável de ambiente (MKL_MIC_ENABLE=1/MKL_MIC_ENABLE=0) ou por uma chamada de função no código (*mkl_mic_enable()/mkl_mic_disable()*).

6.4.2 Bases de Dados

Para os experimentos de escalabilidade realizados na arquitetura *multi-core* foram usadas as bases de dados Netflix, Jester e Movielens (descritas na Seção 4.4.1). Nos experimentos com o *offload* automático, foi utilizada uma base de dados de um grande portal de música do Brasil. Além de ser muito conhecido pelas letras de músicas organizadas e catalogadas no site, tal portal possui uma enorme base de dados sobre cantores, estilos musicais, histórico de navegabilidade dos usuários e várias outras informações relacionadas ao mundo da música. A base de dados Música, disponibilizada pela empresa que administra o portal vem sendo testada num sistema de recomendação de músicas. A base é composta por 2.600 usuários e 994 músicas. Cada usuário possui um mínimo de 10 músicas avaliadas — cada avaliação varia de 1 a 5. Além das avaliações,

a base também fornece informações demográficas dos usuários e informações sobre as músicas, das quais um conjunto de 1.365 atributos foi extraído e usado para aprender os modelos de predição. O algoritmos SCOAL-LR e SCOAL-ELM foram parametrizados de forma arbitrária em três diferentes situações, gerando 4 biclusters, 8 biclusters e 16 biclusters. A Tabela 12 mostra o número de grupos de linha (U) e grupos de coluna (V) adotado em cada base de dados.

Tabela 12 – Valores de U e V adotados para gerar os biclusters.

	Netflix	Jester	Movielens	Música
4 biclusters	$U = 2, V = 2$	$U = 2, V = 2$	$U = 2, V = 2$	$U = 2, V = 2$
8 biclusters	$U = 2, V = 4$	$U = 2, V = 4$	$U = 2, V = 4$	$U = 2, V = 4$
16 biclusters	$U = 4, V = 4$	$U = 8, V = 2$	$U = 4, V = 4$	$U = 4, V = 4$

6.4.3 Resultados Obtidos

A fim de avaliar as versões paralelas dos algoritmos SCOAL-LR e SCOAL-ELM, foram utilizadas medidas de desempenho comumente adotadas:

- **Speedup (S):** É a razão entre o tempo para executar o algoritmo serial, utilizando um processador e o tempo executando o algoritmo paralelo, usando N processadores:

$$S = \frac{T_{serial}}{T_N} \quad (6.1)$$

- **Eficiência Paralela (E):** A eficiência paralela é a razão entre o *Speedup* e o número de processadores. Esta métrica indica quanto do paralelismo foi explorado no algoritmo:

$$E = \frac{S}{N} \quad (6.2)$$

As Figuras 13, 14 e 15 mostram o desempenho alcançado pelos algoritmos SCOAL-LR e SCOAL-ELM para as bases Netflix, Jester e Movielens, sendo executados numa arquitetura *multi-core*. Nos experimentos com o SCOAL-LR, pode ser observado que, em aumentando o número *threads*, alcança-se um bom *speedup* até 32 *threads*. Após este número, nos experimentos com as bases Jester e Netflix, praticamente não existe redução de tempo com 64 *threads* exceto para o experimento Jester com 16 biclusters e Netflix com 8 biclusters.

Figura 13 – Base Netflix

Figura 14 – Base Jester

Figura 15 – Base Movielens

As curvas dos gráficos referentes aos experimentos feitos com o SCOAL-ELM demonstram um comportamento semelhante com relação ao *speedup* obtido pelo SCOAL-LR, isto é, os tempos de execução se estabilizam com 32 *threads*. Os experimentos com a base Jester com 8 e 16 biclusters com 64 *threads* ainda apresentam um aumento significativo de *speedup*. Observa-se também que, devido ao custo computacional envolvido no cálculo dos modelos de predição não lineares ser maior do que nos modelos lineares, o *speedup* alcançado pelo SCOAL-ELM é bem superior ao SCOAL-LR. O maior *speedup* alcançado nos experimentos com o SCOAL-LR foi de $7.5\times$ no experimento Netflix com 8 biclusters contra $11.5\times$ no experimento Jester com 16 biclusters obtidos com o SCOAL-ELM. Para um melhor efeito comparativo, as Tabelas 13 e 14 apresentam os tempos de execução e o *speedup* respectivamente.

Tabela 13 – Tempos de execução em minutos.

Base	SCOAL	Número de threads						
		1	2	4	8	16	32	64
Netflix 4	ELM	57,435	42,749	25,894	17,911	12,296	8,615	8,607
	LR	5,277	3,656	2,122	1,874	1,376	0,948	1,097
Netflix 8	ELM	59,833	43,614	35,898	25,899	15,181	9,969	9,324
	LR	10,376	6,757	5,234	3,624	3,091	1,920	1,380
Netflix 16	ELM	84,927	43,474	35,573	31,503	17,834	13,545	11,399
	LR	8,357	5,966	4,575	3,434	1,816	1,312	1,321
Jester 4	ELM	114,143	73,76	57,715	49,640	30,592	32,928	27,148
	LR	2,859	2,806	3,398	2,485	1,447	1,155	1,108
Jester 8	ELM	289,23	205,75	145,07	88,383	68,274	44,092	26,633
	LR	9,496	7,687	5,866	6,2245	3,751	2,556	2,230
Jester 16	ELM	545,295	441,883	269,098	175,64	102,389	91,306	46,757
	LR	22,620	16,760	14,74	14,07	8,952	7,387	4,382
Movielens 4	ELM	32,854	28,942	20,170	14,788	10,663	7,125	7,681
	LR	0,889	0,726	0,552	0,527	0,390	0,291	0,333
Movielens 8	ELM	52,394	46,639	27,525	16,548	12,303	8,893	7,818
	LR	1,376	1,030	0,822	0,652	0,527	0,400	0,390
Movielens 16	ELM	58,221	51,186	35,723	31,82	20,110	11,053	9,765
	LR	1,144	1,056	1,031	0,645	0,428	0,301	0,320

Tabela 14 – *Speedup*.

Base	SCOAL	Número de threads					
		2	4	8	16	32	64
Netflix 4	ELM	1,344	2,218	3,207	4,671	6,667	6,673
	LR	1,443	2,486	2,815	3,834	5,562	4,810
Netflix 8	ELM	1,372	1,667	2,330	3,941	6,001	6,417
	LR	1,536	1,983	2,863	3,356	5,403	7,516
Netflix 16	ELM	1,953	2,387	2,695	4,762	6,269	7,450
	LR	1,400	1,826	2,433	4,600	6,367	6,323
Jester 4	ELM	1,547	1,977	2,299	3,731	3,466	4,204
	LR	1,018	0,841	1,150	1,975	2,473	2,579
Jester 8	ELM	1,405	1,993	3,272	4,236	6,559	10,859
	LR	1,235	1,168	1,525	2,531	3,714	4,256
Jester 16	ELM	1,234	2,026	3,104	5,325	5,972	11,662
	LR	1,349	1,534	1,607	2,526	3,061	5,161
Movielens 4	ELM	1,135	1,628	2,221	3,081	4,611	4,277
	LR	1,223	1,610	1,686	2,277	3,054	2,666
Movielens 8	ELM	1,123	1,903	3,166	4,258	5,891	6,701
	LR	1,336	1,673	2,110	2,610	3,440	3,525
Movielens 16	ELM	1,137	1,629	1,829	2,895	5,267	5,961
	LR	1,083	1,109	1,773	2,671	3,796	3,577

Apesar dos algoritmos SCOAL-LR e SCOAL-ELM alcançarem um bom *speedup*, analisando os gráficos de eficiência e a Tabela 15, fica evidente que os algoritmos apresentam um escalonamento deficiente, ocasionando em uma baixa eficiência paralela. A Tabela 16 apresenta o percentual de tempo de CPU gasto por cada passo paralelizado do algoritmo SCOAL-LR com base numa análise feita com o software Intel Vtune Amplifier². Nos experimentos realizados com as bases de dados Netflix, Jester e Movielens, percebe-se que o tempo de CPU gasto com o cálculo dos modelos de predição (Subseção 6.3.1) é bem menor quando comparado ao tempo gasto com a atualização dos *clusters* de linhas (Subseção 6.3.2) e a atualização dos *clusters* de colunas (Subseção 6.3.3). A análise feita com o VTune também detectou valores discrepantes

² <https://software.intel.com/en-us/intel-vtune-amplifier-xe>

de *Spin time*. Altos valores de *Spin time* em regiões paralelas indicam um desbalanceamento de carga entre as *threads* ou uma concorrência ineficiente entre as mesmas.

Realmente existe um desbalanceamento elevado de carga de trabalho entre as *threads* envolvidas tanto na atualização dos *clusters* de linhas (Código-Fonte 4) como na atualização dos *clusters* de colunas (Código-Fonte 5). No caso da atualização dos *clusters* de colunas, por exemplo, cada *thread* é responsável pelo cálculo da predição total de uma coluna inteira da matriz de dados. Isto implica em percorrer todas as células da respectiva coluna identificando quais delas possuem avaliações para, em seguida, calcular o valor da predição e compara-lo com o valor de predição obtido utilizando os modelos de predição dos outros *clusters* de coluna. Ocorre é que a matriz de dados é altamente esparsa, fazendo com que determinadas linhas e/ou colunas estejam com várias células preenchidas ao passo que a grande maioria esteja sem valores de avaliação.

Tomando por exemplo o contexto de recomendação de filmes, é natural que um filme consagrado pela crítica tenha muito mais avaliações do que os demais. Dessa forma a *thread* responsável pelo cálculo da predição global deste filme (coluna) vai ter muito mais trabalho devido à grande quantidade de avaliações feitas ao filme do que a maior parte das *threads* cujos filmes tiveram pouca audiência e consequentemente pouquíssimas avaliações. O mesmo acontece na dimensão linha onde um usuário pode ser, por exemplo, um crítico de cinema, razão pela qual o deixa com um número muito maior de avaliações em relação ao demais usuários (linhas). Tais situações contribuem para que determinadas *threads* fiquem sobrecarregadas enquanto outras fiquem ociosas e "presas" no final da região paralela, consumindo tempo de CPU de forma desnecessária. Dessa forma, apesar do aumento do número de processadores (N), a eficiência diminui, pois grande parte das *threads* não está atuando de maneira concorrente.

Tabela 15 – Eficiência Paralela.

Base	Modelo	Número de threads					
		2	4	8	16	32	64
Netflix 4	ELM	0,671	0,554	0,400	0,291	0,208	0,104
	LR	0,721	0,621	0,351	0,239	0,173	0,075
Netflix 8	ELM	0,685	0,416	0,288	0,246	0,187	0,100
	LR	0,767	0,495	0,357	0,239	0,173	0,117
Netflix 16	ELM	0,976	0,596	0,336	0,297	0,195	0,116
	LR	0,700	0,456	0,304	0,287	0,198	0,098
Jester 4	ELM	0,773	0,494	0,287	0,093	0,108	0,065
	LR	0,509	0,210	0,143	0,123	0,077	0,040
Jester 8	ELM	0,702	0,498	0,409	0,264	0,204	0,169
	LR	0,617	0,404	0,190	0,158	0,116	0,066
Jester 16	ELM	0,617	0,506	0,388	0,332	0,142	0,182
	LR	0,674	0,383	0,200	0,157	0,095	0,080
Movielens 4	ELM	0,56	0,407	0,277	0,192	0,144	0,066
	LR	0,611	0,402	0,210	0,142	0,095	0,041
Movielens 8	ELM	0,561	0,475	0,395	0,266	0,184	0,104
	LR	0,668	0,418	0,263	0,163	0,107	0,055
Movielens 16	ELM	0,568	0,407	0,228	0,180	0,164	0,093
	LR	0,541	0,277	0,221	0,166	0,118	0,055

Apesar da baixa eficiência obtida nos experimentos realizados com as bases de dados

Tabela 16 – Percentuais de utilização da CPU — SCOAL-LR.

	cálculo dos modelos	cálculo dos <i>clusters</i> de linhas	cálculo dos <i>clusters</i> de colunas
Netflix 4	5,6%	16,4%	38,8%
Netflix 8	6,4%	16,2%	41%
Netflix 16	5,8%	22,5%	44%
Jester 4	9,5%	31,1%	32,9%
Jester 8	6,7%	45,4%	28%
Jester 16	4,7%	60%	21,9%
Movielens 4	26,6%	19,9%	24,5%
Movielens 8	21%	21,6%	37,4%
Movielens 16	16,2%	30,3%	35%

Netflix, Jester e Movielens, cabe ressaltar que o mesmo algoritmo — SCOAL-LR — é capaz de apresentar um desempenho muito bom em situações nas quais as bases de dados apresentam alta dimensionalidade. Em tais bases, o tempo de CPU dedicado aos cálculos dos modelos de predição é muito maior do que ao utilizado na atualização dos *clusters* de linhas e colunas. A Tabela 17 apresenta os tempos de CPU coletados pela análise do Vtune na execução do algoritmo SCOAL-LR processando a base de dados Música em três situações distintas — 4 biclusters, 8 biclusters e 16 biclusters. Em todas as situações, o tempo de CPU destinado ao cálculo dos modelos de predição fica acima dos 90% do tempo total do processamento. Dessa forma, o desbalanceamento de carga que ocorre na atualização dos *clusters* de linhas e de colunas passa despercebido pois o tempo de CPU destinado a estes passos é reduzido. A Figura 16 e a Tabela 18, apresentam o desempenho obtido pela execução do algoritmo SCOAL-LR no processamento da base de dados Música em diferentes situações. No experimento com 4 biclusters, a eficiência ultrapassa a casa dos 80%, com 8 biclusters permanece em 70% e somente com 16 biclusters cai para 31%.

Tabela 17 – Percentuais de utilização da CPU — base de dados Música (SCOAL-LR).

	cálculo dos modelos	cálculo dos <i>clusters</i> de linha	cálculo dos <i>clusters</i> de coluna
4 biclusters	91%	2,6%	2,9%
8 biclusters	91,5%	2,5%	4,4%
16 biclusters	91%	3,8%	3,9%

Figura 16 – Base Música

Tabela 18 – Medidas de desempenho — base de dados Música (SCOAL-LR).

		Número de threads					
		1	2	4	8	16	32
4 biclusters	tempo	88,07	48,413	26,696	25,847	24,703	25,119
	speedup		1,819	3,299	3,407	3,565	3,506
	eficiência		0,910	0,825	0,426	0,223	0,110
8 biclusters	tempo	78,620	41,526	22,202	14,056	13,589	14,224
	speedup		1,893	3,541	5,593	5,786	5,527
	eficiência		0,947	0,885	0,699	0,362	0,173
16 biclusters	tempo	91,656	48,978	29,248	20,528	18,290	14,626
	speedup		1,871	3,134	4,465	5,011	6,267
	eficiência		0,936	0,783	0,558	0,313	0,196

6.4.3.1 MKL e Xeon Phi

As funções MKL podem ser executadas no coprocessador Xeon Phi de três formas — execução nativa, *offload* assistido e *offload* automático. Na forma nativa, o programa fonte contendo funções MKL é compilado com uma diretiva de compilação que gera um binário específico para ser executado no coprocessador Xeon Phi. No *offload* assistido, cabe ao programador definir quais regiões do código serão executadas no Xeon Phi. No *offload* automático, não existe a necessidade de nenhuma alteração no código fonte e nem mesmo qualquer diretiva de compilação. A própria biblioteca MKL decide em tempo de execução a quantidade de carga de trabalho que será executada no(s) coprocessador(es). Dependendo do tamanho do problema e do estado atual do(s) coprocessador(es), a biblioteca pode optar por executar todo o trabalho,

parte dele ou até mesmo não executar nada no(s) coprocessador(es) (JEFFERS; REINDERS, 2013). A função *DGEMM*, utilizada na multiplicação de matrizes, exige que pelo menos uma das dimensões de todas as matrizes envolvidas no cálculo, tenha um tamanho mínimo de 1.280 linhas ou colunas. Por este motivo, os experimentos realizados nesta seção utilizaram somente a base de dados Música dado que as bases Netflix, Jester e Movielens não atendem esta exigência. O objetivo primário deste experimento é a avaliar os benefícios da utilização do coprocessador Xeon Phi com MKL na sua forma automática.

Foram realizados experimentos em dois cenários distintos. O primeiro cenário avaliou a escalabilidade do algoritmo SCOAL-LR em três diferentes situações: 4 biclusters, 8 biclusters e 16 biclusters, com as funções MKL utilizando somente a arquitetura *multi-core* do servidor, ou seja, com o *offload* desabilitado. No segundo cenário, foi habilitado o *offload* automático para avaliar a utilização dos coprocessadores Xeon Phi em conjunto com os processadores do host pelo MKL realizando os mesmos experimentos. A Figura 17 apresenta os resultados obtidos pelo SCOAL-LR nos dois cenários. Pode-se perceber que quando o número de *threads* disponíveis no host é restrito (< 4), obtém-se um bom *speedup* mas, na medida em que o número de *threads* disponíveis no host é incrementado, não se observa nenhum ganho. Possivelmente, isto está relacionado com o tamanho da carga de trabalho que está sendo processada. A transferência de dados do host para o Xeon Phi é um processo custoso e, para problemas pequenos (a matriz de dados possui 1.300 colunas quando o mínimo necessário para o *offload* automático são 1280), qualquer aumento de *speedup* pode ser anulado pelo custo associado com a transferência dos dados. Entretanto, quando se aumenta a quantidade de modelos de predição a serem calculados, nota-se que o desempenho da MKL com *offload* automático se mantém superior até com 16 *threads*.

Figura 17 – Tempos computacionais obtidos pelo SCOAL-LR com o *offload* automático da biblioteca MKL habilitado e desabilitado.

6.5 Considerações Finais

Este capítulo apresentou as versões paralelas dos algoritmos SCOAL-LR e SCOAL-ELM baseadas em OpenMP. Dos diferentes modelos de programação paralela descritos de forma sucinta na Seção 6.2, a adoção por OpenMP foi motivada pela sua facilidade de uso e sua portabilidade. As rotinas de código apresentadas na Seção 6.3 comprovam que é possível alcançar um bom nível de paralelização com intervenções mínimas no código serial. A portabilidade do OpenMP também pôde ser comprovada nos experimentos realizados — inicialmente foram feitos num notebook e, posteriormente, numa arquitetura HPC sem que fosse necessária nenhuma alteração na codificação. Os resultados obtidos nos experimentos realizados na arquitetura *multi-core* comprovam a redução do tempo de execução na medida em que é incrementado o número de *threads*.

Apesar da baixa eficiência obtida com as bases de dados Netlix, Jester e Movielens, num cenário de aplicação real envolvendo bases de dados com alta dimensionalidade a versão paralela do algoritmo SCOAL obteve um desempenho promissor. Um outro aspecto avaliado foi a utilização do *offload* automático da biblioteca MKL numa arquitetura HPC. Apesar do código não estar devidamente escrito para maximizar a utilização dessa arquitetura, os resultados iniciais mostram que, com poucas *threads* disponíveis no host, torna-se vantajosa a utilização

do coprocessador Xeon Phi, sobretudo pelo fato de que não existe a necessidade de nenhuma modificação no código. Um fator preponderante para se obter performance no Xeon Phi com o *offload* automático diz respeito ao tamanho do problema a ser processado, devido ao alto custo associado com a transferência de dados entre o host e o Xeon Phi.

CONCLUSÕES E TRABALHOS FUTUROS

7.1 Principais Contribuições

Nesta tese, variantes do algoritmo SCOAL foram apresentadas para lidar com os desafios associados aos SR. Em linhas específicas, buscou-se confirmar as hipóteses levantadas no Capítulo 1. Para conveniência do leitor, e na Seção 1.1, a Hipótese 1 levantada nesta tese é:

Hipótese 1: É possível desenvolver uma variante do algoritmo SCOAL para lidar apropriadamente com o problema da ausência de informações de avaliações para novos usuários/itens (problema *cold-start*).

Com o intuito de confirmar esta hipótese, foi proposta uma abordagem híbrida de recomendação baseada no algoritmo SCOAL que é capaz de lidar não só com situações parciais de *cold-start*, nas quais um número mínimo de avaliações feitas pelo novo usuário é fornecido, mas também em situações extremas de *cold-start* em que não existe nenhum histórico de avaliação do novo usuário do sistema. Considerando todos os resultados obtidos apresentados no Capítulo 4, a Hipótese 1 desta tese foi confirmada. Os resultados dessas análises foram publicados em [Pereira e Hruschka \(2015\)](#).

Vale destacar que em situações parciais de *cold-start*, a simplicidade conceitual do método CME, aliada à acurácia de predição do algoritmo SCOAL, obteve os melhores resultados ao ser empiricamente comparada com outras abordagens amplamente utilizadas na prática. No contexto de situações extremas de *cold-start* o método DC provou ser bastante eficaz. O alto custo computacional associado ao método DC pode ser extremamente reduzido utilizando uma versão paralela do classificador Naive Bayes presente na biblioteca *Data Analytics Acceleration* (DAAL)¹ recentemente lançada pela empresa Intel (agosto/2015). A biblioteca DAAL contempla uma variedade de algoritmos utilizados na área de *Big Data* que foram otimizados no intuito de extrair o máximo do recurso computacional que uma arquitetura HPC pode oferecer.

¹ <https://software.intel.com/en-us/daal>

Uma outra frente de pesquisa investigada foi norteadada pela Hipótese 2:

Hipótese 2: Pode-se melhorar o algoritmo SCOAL no tocante a modelos não lineares, deixando-o mais versátil e competitivo em termos de capacidade preditiva, comparativamente a outros algoritmos que representam o estado-da-arte no assunto (*e.g.* biclusterização e fatoração matricial), para que possa ser utilizado como o componente principal de SR variados (*e.g.* para filmes, músicas e redes sociais).

Para validar a Hipótese 2, foi proposto o algoritmo SCOAL-ELM, uma instância do algoritmo SCOAL que utiliza modelos de predição não lineares baseados em Máquinas de Aprendizado Extremo. O SCOAL-ELM foi então avaliado via experimentos realizados com três bases de dados amplamente utilizadas na avaliação de SR. Os resultados obtidos são promissores para várias medidas de qualidade avaliadas. Num estudo comparativo feito com o já consagrado algoritmo SCOAL baseado em regressão linear, o algoritmo SCOAL-ELM obteve um desempenho superior em boa parte dos experimentos realizados oferecendo boas evidências de que a hipótese 2 pode ser confirmada.

As principais razões que motivaram a adoção de Máquinas de Aprendizado Extremo como modelos de regressão não lineares foram: (i) alta acurácia (ii) rapidez na fase do treinamento dos modelos e (iii) o fato de demandar pouca intervenção humana na sua parametrização. Os resultados obtidos com o SCOAL-ELM comprovam a acurácia alcançada pelas ELMs. No que diz respeito ao tempo envolvido na fase de treinamento dos modelos, se comparado às tradicionais redes *Single Hidden Layer Feedforward Neural Networks*, o tempo computacional necessário é muito inferior. Entretanto, fazer a parametrização de uma ELM, a fim de se obter uma boa acurácia, não é uma tarefa das mais simples. São necessários vários experimentos para que se encontre o número de neurônios da camada oculta e o valor do parâmetro de regularização usado para reduzir a norma dos pesos de saída. Um outro problema que foi observado durante os experimentos realizados, é que ELMs são muito sensíveis tanto na forma de inicialização dos pesos da rede quanto na escala dos atributos dos dados de entrada.

Finalmente, foi investigada a terceira hipótese:

Hipótese 3: É possível desenvolver uma versão paralelizada do algoritmo SCOAL que seja suficientemente eficiente, do ponto de vista computacional, a ponto de poder ser incorporada como um componente de SR para problemas reais, os quais usualmente envolvem grandes bases de dados.

Para validar essa terceira hipótese, versões paralelas dos algoritmos SCOAL-LR e SCOAL-ELM foram implementadas utilizando o modelo de programação paralelo OpenMP. Foi demonstrado via experimentos empíricos que tanto o SCOAL-LR como o SCOAL-ELM atingem um bom nível de *speedup* com intervenções mínimas no código serial. Os resultados obtidos nos experimentos realizados comprovam a redução do tempo de execução na medida em que é incrementado o número de *threads*.

Um fator importante observado nos experimentos realizados com as versões paralelas do algoritmo SCOAL propostas nesta tese é que, na forma como os algoritmos estão implementados atualmente, para que se obtenha uma boa eficiência paralela a matriz de dados deve apresentar alta dimensionalidade. Caso contrário, a eficiência fica comprometida devido ao escalonamento ineficiente de *threads* detectado nos métodos responsáveis pela atualização dos *clusters* de linhas e colunas.

7.2 Trabalhos futuros

Como trabalhos futuros promissores destacam-se: (i) construir um *ensemble* de regressores (ii) aumentar o nível de paralelismo e (iii) adaptar o algoritmo SCOAL de forma a permitir que uma determinada instância (*e.g.* usuário ou item) possa pertencer em mais de um *cluster*.

7.2.1 *Ensemble de regressores*

As comparações feitas entre o SCOAL-LR e o SCOAL-ELM na Seção 6.4 comprovam que, conforme o esperado, não existe um método vencedor para todas as bases de dados. Diante desta perspectiva, uma frente de investigação está relacionada com o estudo de formas de construir um *ensemble* de regressores combinando ELMs e modelos lineares para o contexto específico de SR baseados no algoritmo SCOAL, levando em conta a diversidade entre os modelos regressores e seus respectivos pesos para realizar as predições.

7.2.2 *Aumentar a eficiência do paralelismo*

Apesar das versões paralelas dos algoritmos SCOAL-LR e SCOAL-ELM atingirem um bom *speedup*, faz-se necessária uma reestruturação desses algoritmos para que se possa maximizar a capacidade de processamento paralelo que uma arquitetura HPC oferece. Neste sentido, uma das principais frentes de investigação é a reescrita das funções que atualizam os *clusters* de linhas e colunas, fazendo com que cada *thread* fique responsável pelo cálculo da predição de uma única célula da matriz de dados ao invés de calcular a predição de todas as células de uma linha e/ou coluna. Isto ocasionará uma melhor divisão da carga de trabalho entre as *threads*, aumentando a eficiência dos algoritmos SCOAL-LR e SCOAL-ELM em cenários de aplicações que envolvam matrizes de dados com dimensões reduzidas.

Um outro aspecto avaliado na Subseção 6.2.2, foi a execução de aplicações no coprocessador Xeon Phi sendo realizadas de forma nativa — sem utilizar os recursos de processamento do host, ou de forma heterogênea — *offload* assistido — no qual partes do código são executadas no host e outras no coprocessador. Nos experimentos descritos na Seção 6.4.3.1 (MKL *offload* automático), coube à biblioteca MKL a responsabilidade em decidir quando, e de que forma, o coprocessador Xeon Phi é utilizado. As rotinas da biblioteca MKL estão escritas de forma vetorizada, o que possibilita a execução de instruções de 512 bits — *Single Instruction, Multiple*

Data (SIMD) — de uma única vez, obtendo o máximo de performance que uma arquitetura HPC pode oferecer. Entretanto, o *offload* automático restringe a utilização do coprocessador Xeon Phi a um número limitado de funções MKL, além de estabelecer um tamanho mínimo das dimensões das matrizes envolvidas no cálculo. Visando maximizar a capacidade de utilização do coprocessador Xeon Phi, um estudo aprofundado sobre técnicas de vetorização de código (*e.g.* estruturas de dados, alinhamento de dados, *prefetching*, diretivas SIMD) e a utilização do *offload* assistido deve ser feito, bem como uma análise sobre qual tamanho a base de dados deve ter para que o tempo envolvido na transferência dos dados entre o host e o coprocessador Xeon Phi possa ser compensado pelo ganho em eficiência computacional.

7.2.3 *Instâncias multicluster*

A abordagem *dividir e conquistar* do algoritmo SCOAL particiona a matriz de dados em U grupos de usuários (linhas) e V grupos de itens (colunas) sendo que cada usuário e item pertence exatamente a um único grupo de usuário e item respectivamente. No contexto de um SR de filmes por exemplo, um filme pertence a um único grupo de filmes, o que pode não condizer com a realidade de determinados filmes que podem estar relacionados a mais de um grupo de filmes. Um estudo aprofundado de como capacitar o algoritmo SCOAL para lidar com tais cenários é promissor. Neste sentido, uma possível frente de investigação seria uma versão do algoritmo SCOAL baseada em modelos de biclusterização Bayesianos que permitem que uma instância (*e.g.* *usuário* e/ou *item*) possa pertencer a mais de um *cluster*.

REFERÊNCIAS

- ADOMAVICIUS, G.; TUZHILIN, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. **Knowledge and Data Engineering, IEEE Transactions on**, IEEE, v. 17, n. 6, p. 734–749, 2005. Citado 4 vezes nas páginas 26, 27, 30 e 31.
- AGRAWAL, R.; GEHRKE, J.; GUNOPULOS, D.; RAGHAVAN, P. Automatic subspace clustering of high dimensional data for data mining applications. **ACM Communications**, ACM, v. 27, n. 2, p. 94–105, 1998. Citado na página 34.
- AHN, H. J. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. **Information Sciences**, v. 178, p. 37–51, 2008. Citado 3 vezes nas páginas 46, 49 e 53.
- AMD, A. Core math library (acml). URL <http://developer.amd.com/acml.jsp>, 2012. Citado na página 78.
- ANDERSON, E.; BAI, Z.; DONGARRA, J.; GREENBAUM, A.; MCKENNEY, A.; CROZ, J. D.; HAMMERLING, S.; DEMMEL, J.; BISCHOF, C.; SORENSEN, D. Lapack: A portable linear algebra library for high-performance computers. **Proceedings of the 1990 ACM/IEEE conference on Supercomputing**, IEEE Computer Society Press, p. 2–11, 1990. Citado na página 78.
- BAEZA-YATES, R.; RIBEIRO-NETO, B. *et al.* Modern information retrieval. **ACM**, ACM press New York, v. 463, 1999. Citado na página 37.
- BANERJEE, A.; MERUGU, S.; DHILLON, I. S.; GHOSH, J. Clustering with bregman divergences. **The Journal of Machine Learning Research**, JMLR. org, v. 6, p. 1705–1749, 2005. Citado na página 35.
- BARNEY, B. *et al.* Introduction to parallel computing. **Lawrence Livermore National Laboratory**, v. 6, n. 13, p. 10, 2010. Citado na página 71.
- BARTLETT, P. L. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. **Information Theory, IEEE Transactions on**, IEEE, v. 44, n. 2, p. 525–536, 1998. Citado na página 63.
- BASIRI, J.; SHAKERY, A.; MOSHIRI, B.; HAYAT, M. Z. Alleviating the cold start problem of recommender systems using a new hybrid approach. **Telecommunications (IST), 2010 5th International Symposium (IEEE)**, p. 962–967, 2010. Citado na página 47.
- BASU, C.; HIRSH, H.; COHEN, W. *et al.* Recommendation as classification: Using social and content-based information in recommendation. **AAAI/IAAI**, p. 714–720, 1998. Citado na página 31.
- BEDI, P.; AGARWAL, S. K.; VASHISTH, P.; SHARMA, S.; AGGARWAL, T. Online tweet recommendation using extreme learning machine. **Proceedings of the 2014 Recommender Systems Challenge**, p. 62, 2014. Citado na página 62.

- BENNETT, J.; LANNING, S. The netflix prize. **Proceedings of KDD cup and workshop**, v. 2007, p. 35, 2007. Citado 2 vezes nas páginas 21 e 37.
- BLEI, D. M.; NG, A. Y.; JORDAN, M. I. Latent dirichlet allocation. **Journal of machine Learning research**, JMLR, v. 3, p. 993–1022, 2003. Citado na página 46.
- BOBADILLA, J.; ORTEGA, F.; HERNANDO, A.; BERNAL, J. A collaborative filtering approach to mitigate the new user cold start problem. **Knowledge-Based Systems**, Elsevier, v. 26, p. 225–238, 2012. Citado na página 45.
- _____. _____. **Knowledge-Based Systems**, v. 26, p. 225–238, 2012. Citado 2 vezes nas páginas 46 e 49.
- BOBADILLA, J.; ORTEGA, F.; HERNANDO, A.; GUTIÉRREZ, A. Recommender systems survey. **Knowledge-Based Systems**, Elsevier, v. 46, p. 109–132, 2013. Citado 5 vezes nas páginas 22, 24, 27, 36 e 65.
- BREESE, J. S.; HECKERMAN, D.; KADIE, C. Empirical analysis of predictive algorithms for collaborative filtering. **Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence**, p. 43–52, 1998. Citado 2 vezes nas páginas 37 e 65.
- BUCKLAND, M.; GEY, F. The relationship between recall and precision. **Journal of the American society for information science**, American Documentation Institute, v. 45, n. 1, p. 12, 1994. Citado na página 65.
- BURKE, R. Hybrid web recommender systems. **The adaptive web**, Springer, p. 377–408, 2007. Citado 3 vezes nas páginas 20, 27 e 31.
- CAMBRIA, E.; GASTALDO, P.; BISIO, F.; ZUNINO, R. An ELM-based model for affective analogical reasoning. **Neurocomputing**, Elsevier, v. 149, p. 443–455, 2015. Citado 2 vezes nas páginas 60 e 61.
- CAO, J.; LIN, Z.; HUANG, G.-b. Composite function wavelet neural networks with extreme learning machine. **Neurocomputing**, Elsevier, v. 73, n. 7, p. 1405–1416, 2010. Citado 2 vezes nas páginas 60 e 61.
- CASTRO, P. A. D.; FRANÇA, F. O. de; FERREIRA, H. M.; ZUBEN, F. J. V. Evaluating the performance of a biclustering algorithm applied to collaborative filtering—a comparative analysis. **Hybrid Intelligent Systems**, p. 65–70, 2007. Citado 2 vezes nas páginas 34 e 35.
- CASTRO, P. A. de; FRANÇA, F. O. de; FERREIRA, H. M.; ZUBEN, F. J. V. Applying biclustering to text mining: an immune-inspired approach. **Artificial Immune Systems**, Springer, p. 83–94, 2007. Citado 2 vezes nas páginas 34 e 35.
- CAZELLA, S. C.; NUNES, M.; REATEGUI, E. B. A ciência da opinião: Estado da arte em sistemas de recomendação. **XXX Congresso da SBC Jornada de Atualização da Informática**, 2010. Citado na página 30.
- CHANDRA, R. Parallel programming in openmp. **Morgan kaufmann**, 2001. Citado na página 74.
- CHAPMAN, B.; JOST, G.; PAS, R. V. D. Using openmp: portable shared memory parallel programming. **MIT press**, v. 10, 2008. Citado 2 vezes nas páginas 73 e 74.

- CHEN, H.; PENG, J.; ZHOU, Y.; LI, L.; PAN, Z. Extreme learning machine for ranking: Generalization analysis and applications. **Neural Networks**, Elsevier, v. 53, p. 119–126, 2014. Citado na página 62.
- CHENG, Y.; CHURCH, G. M. Biclustering of expression data. **Ismb**, v. 8, p. 93–103, 2000. Citado na página 34.
- CHO, H. Co-clustering algorithms: Extensions and applications. 2008. Citado na página 32.
- CHRYSSOS, G. Intel® xeon phi™ coprocessor-the architecture. **Intel Whitepaper**, 2014. Citado na página 72.
- CONDLIFF, M. K.; LEWIS, D. D.; MADIGAN, D.; POSSE, C. Bayesian mixed-effects models for recommender systems. **ACM SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation**, v. 15, n. 5, 1999. Citado na página 31.
- DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. **Communications of the ACM**, ACM, v. 51, n. 1, p. 107–113, 2008. Citado na página 76.
- DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood from incomplete data via the em algorithm. **Journal of the Royal statistical Society**, v. 39, p. 1–38, 1977. Citado na página 46.
- DENG, W.; ZHENG, Q.; CHEN, L. Real-time collaborative filtering using extreme learning machine. **Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on**, v. 1, p. 466–473, 2009. Citado na página 62.
- DEODHAR, M.; GHOSH, J. A framework for simultaneous co-clustering and learning from complex data. **Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining**, p. 250–259, 2007. Citado 4 vezes nas páginas 9, 40, 41 e 52.
- _____. Scoal: A framework for simultaneous co-clustering and learning from complex data. **ACM Transactions on Knowledge Discovery from Data**, v. 4, p. 11, 2010. Citado 8 vezes nas páginas 10, 23, 24, 40, 43, 44, 52 e 59.
- DEODHAR, M.; JONES, C.; GHOSH, J. Parallel simultaneous co-clustering and learning with map-reduce. **Granular Computing (GrC), 2010 IEEE International Conference on**, p. 149–154, 2010. Citado na página 77.
- DIETTERICH, T. G. Ensemble methods in machine learning. **Multiple classifier systems**, Springer, p. 1–15, 2000. Citado na página 39.
- DIVINA, F.; AGUILAR-RUIZ, J. S. A multi-objective approach to discover biclusters in micro-array data. **Proceedings of the 9th annual conference on Genetic and evolutionary computation**, p. 385–392, 2007. Citado na página 34.
- EKSTRAND, M. D.; RIEDL, J. T.; KONSTAN, J. A. Collaborative filtering recommender systems. **Foundations and Trends in Human-Computer Interaction**, Now Publishers Inc., v. 4, n. 2, p. 81–173, 2011. Citado na página 37.
- FORUM, M. P. I. Mpi: a message passing interface standard: Version 3.1; message passing interface forum, june 4, 2015. **University of Tennessee**, 2015. Citado na página 75.

FRANÇA, F. O. de. **Biclusterização na análise de dados incertos**. Tese (Doutorado) — Universidade Estadual de Campinas, 2010. Citado 3 vezes nas páginas 32, 34 e 35.

FURNAS, G. W.; DEERWESTER, S.; DUMAIS, S. T.; LANDAUER, T. K.; HARSHMAN, R. A.; STREETER, L. A.; LOCHBAUM, K. E. Information retrieval using a singular value decomposition model of latent semantic structure. **Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval**, p. 465–480, 1988. Citado na página 32.

GANTNER, Z.; DRUMOND, L.; FREUDENTHALER, C.; RENDLE, S.; SCHMIDT-THIEME, L. Learning attribute-to-feature mappings for cold start recommendations. **Data Mining (ICDM), 2010 IEEE 10th International Conference on**, p. 176–185, 2010. Citado 2 vezes nas páginas 46 e 56.

GANTNER, Z.; RENDLE, S.; FREUDENTHALER, C.; SCHMIDT-THIEME, L. Mymedialite: A free recommender system library. **Proceedings of the fifth ACM conference on Recommender systems**, p. 305–308, 2011. Citado na página 57.

GEORGE, T.; MERUGU, S. A scalable collaborative filtering framework based on co-clustering. **Data Mining, Fifth IEEE International Conference on**, p. 4–pp, 2005. Citado 2 vezes nas páginas 34 e 35.

GHEMAWAT, S.; GOBIOFF, H.; LEUNG, S.-T. The google file system. **ACM SIGOPS operating systems review**, v. 37, n. 5, p. 29–43, 2003. Citado na página 76.

GOLDBERG, D.; NICHOLS, D.; OKI, B. M.; TERRY, D. Using collaborative filtering to weave an information tapestry. **Communications of the ACM**, ACM, v. 35, n. 12, p. 61–70, 1992. Citado na página 21.

GOLDBERG, K.; ROEDER, T.; GUPTA, D.; PERKINS, C. Eigentaste: A constant time collaborative filtering algorithm. **Information Retrieval**, Springer, v. 4, n. 2, p. 133–151, 2001. Citado 2 vezes nas páginas 52 e 65.

GOLUB, G. H.; LOAN, C. F. V. Matrix computations. **JHU Press**, v. 3, 2012. Citado na página 32.

GU, Q.; ZHOU, J.; DING, C. H. Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. **SDM**, p. 199–210, 2010. Citado na página 62.

HALL, M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P.; WITTEN, I. H. The weka data mining software: an update. **ACM SIGKDD explorations newsletter**, ACM, v. 11, n. 1, p. 10–18, 2009. Citado na página 50.

HILL, W.; STEAD, L.; ROSENSTEIN, M.; FURNAS, G. Recommending and evaluating choices in a virtual community of use. **Proceedings of the SIGCHI conference on Human factors in computing systems**, p. 194–201, 1995. Citado na página 21.

HOERL, A. E.; KENNARD, R. W. Ridge regression: Biased estimation for nonorthogonal problems. **Technometrics**, Taylor & Francis Group, v. 12, n. 1, p. 55–67, 1970. Citado na página 43.

- HOFMANN, T. Probabilistic latent semantic indexing. **Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval**, p. 50–57, 1999. Citado na página 46.
- HSIEH, C.-J.; BANERJEE, A.; DHILLON, I. S.; RAVIKUMAR, P. K. A divide-and-conquer method for sparse inverse covariance estimation. **Advances in Neural Information Processing Systems**, p. 2330–2338, 2012. Citado na página 39.
- HUANG, G.; HUANG, G.-B.; SONG, S.; YOU, K. Trends in extreme learning machines: A review. **Neural Networks**, Elsevier, v. 61, p. 32–48, 2015. Citado na página 44.
- HUANG, G.-B.; CHEN, Y.-Q.; BABRI, H. A. Classification ability of single hidden layer feedforward neural networks. **Neural Networks**, IEEE, v. 11, n. 3, p. 799–801, 2000. Citado na página 59.
- HUANG, G.-B.; ZHOU, H.; DING, X.; ZHANG, R. Extreme learning machine for regression and multiclass classification. **Systems, Man, and Cybernetics, Part B: Cybernetics**, IEEE, v. 42, n. 2, p. 513–529, 2012. Citado 2 vezes nas páginas 12 e 64.
- HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme learning machine: a new learning scheme of feedforward neural networks. In: IEEE. **Neural Networks, 2004. Proceedings. 2004.** [S.l.], 2004. v. 2, p. 985–990. Citado 5 vezes nas páginas 25, 44, 59, 60 e 61.
- _____. Extreme learning machine: theory and applications. **Neurocomputing**, Elsevier, v. 70, n. 1, p. 489–501, 2006. Citado 4 vezes nas páginas 10, 60, 61 e 63.
- JANNACH, D.; ZANKER, M.; FELFERNIG, A.; FRIEDRICH, G. Recommender systems: an introduction. **Cambridge University Press**, 2010. Citado 3 vezes nas páginas 20, 26 e 27.
- JEFFERS, J.; REINDERS, J. Intel xeon phi coprocessor high-performance programming. **Newnes**, 2013. Citado 3 vezes nas páginas 72, 75 e 90.
- KANG, S. J.; LEE, S. Y.; LEE, K. M. Performance comparison of openmp, mpi, and mapreduce in practical problems. **Advances in Multimedia**, Hindawi Publishing Corporation, 2014. Citado na página 71.
- KIM, S.-H.; KIM, S.-H. A divide-and-conquer approach in applying em for large recursive models with incomplete categorical data. **Computational Statistics & Data Analysis**, Elsevier, v. 50, n. 3, p. 611–641, 2006. Citado na página 39.
- KIM, T. H.; PARK, S. I.; YANG, S. B. Improving prediction quality in collaborative filtering based on clustering. **Proceedings of the 2008 IEEE WIC ACM International Conference on Web Intelligence and Intelligent Agent Technology Volume 01**, p. 704–710, 2008. Citado na página 32.
- KIRK, D. B.; WEN-MEI, W. H. Programming massively parallel processors: a hands-on approach. **Newnes**, 2012. Citado na página 74.
- KONSTAN, J. A.; MILLER, B. N.; MALTZ, D.; HERLOCKER, J. L.; GORDON, L. R.; RIEDL, J. Grouplens: applying collaborative filtering to usenet news. **Communications of the ACM**, ACM, v. 40, n. 3, p. 77–87, 1997. Citado na página 28.
- KOREN, Y.; BELL, R.; VOLINSKY, C. Matrix factorization techniques for recommender systems. **IEEE Computer**, n. 8, p. 30–37, 2009. Citado na página 32.

- KOTTE, O.; HEINEMANN, M. A divide-and-conquer approach to analyze underdetermined biochemical models. **Bioinformatics**, Oxford Univ Press, v. 25, n. 4, p. 519–525, 2009. Citado na página 39.
- KULAIF, A.; VONZUBEN, F. Improved regularization in extreme learning machines. **Annals of the Brazilian Congress on Computational Intelligence**, 2013. Citado 2 vezes nas páginas 60 e 65.
- KWON, H.-J.; HONG, K.-S. Personalized smart tv program recommender based on collaborative filtering and a novel similarity method. **IEEE Transactions on Consumer Electronics**, IEEE, v. 57, n. 3, p. 1416–1423, 2011. Citado na página 27.
- LAM, C. Hadoop in action. **Manning Publications Co.**, 2010. Citado na página 76.
- LEE, D. D.; SEUNG, H. S. Learning the parts of objects by non-negative matrix factorization. **Nature**, Nature Publishing Group, v. 401, n. 6755, p. 788–791, 1999. Citado na página 32.
- LEE, K.-H.; LEE, Y.-J.; CHOI, H.; CHUNG, Y. D.; MOON, B. Parallel data processing with mapreduce: a survey. **ACM SIGMOD RECORD**, ACM, v. 40, n. 4, p. 11–20, 2012. Citado na página 76.
- LEMME, A.; FREIRE, A.; BARRETO, G.; STEIL, J. Kinesthetic teaching of visuomotor coordination for pointing by the humanoid robot icub. **Neurocomputing**, Elsevier, v. 112, p. 179–188, 2013. Citado 2 vezes nas páginas 60 e 61.
- LEUNG, C. W.-k.; CHAN, S. C.-f.; CHUNG, F.-l. An empirical study of a cross-level association rule mining approach to cold start recommendations. **Knowledge-Based Systems**, Elsevier, v. 21, n. 7, p. 515–529, 2008. Citado na página 47.
- LIANG, N.-Y.; HUANG, G.-B.; SARATCHANDRAN, P.; SUNDARARAJAN, N. A fast and accurate online sequential learning algorithm for feedforward networks. **Neural Networks**, IEEE, v. 17, n. 6, p. 1411–1423, 2006. Citado na página 62.
- LIN, J.; SUGIYAMA, K.; KAN, M.-Y.; CHUA, T.-S. Addressing cold start in app recommendation: latent user models constructed from twitter followers. **Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval**, p. 283–292, 2013. Citado na página 46.
- LINDEN, G.; SMITH, B.; YORK, J. Amazon. com recommendations: Item-to-item collaborative filtering. **IEEE Internet Computing**, IEEE, v. 7, n. 1, p. 76–80, 2003. Citado na página 29.
- LU, J.; WU, D.; MAO, M.; WANG, W.; ZHANG, G. Recommender system application developments: A survey. **Decision Support Systems**, Elsevier, v. 74, p. 12–32, 2015. Citado 2 vezes nas páginas 22 e 27.
- MADEIRA, S. C.; OLIVEIRA, A. L. Biclustering algorithms for biological data analysis: a survey. **IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)**, IEEE Computer Society Press, v. 1, n. 1, p. 24–45, 2004. Citado 2 vezes nas páginas 32 e 34.
- MAULIK, U.; MUKHOPADHYAY, A.; BANDYOPADHYAY, S.; ZHANG, M. Q.; ZHANG, X. Multiobjective fuzzy biclustering in microarray data: method and a new performance measure. **Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence)**. **IEEE Congress on**, p. 1536–1543, 2008. Citado na página 34.

- MEO, P. D.; QUATTRONE, G.; URSINO, D. A decision support system for designing new services tailored to citizen profiles in a complex and distributed e-government scenario. **Data & Knowledge Engineering**, Elsevier, v. 67, n. 1, p. 161–184, 2008. Citado na página 27.
- MIMAROGLU, S.; UEHARA, K. Bit sequences and biclustering of text documents. **ICDMW**, p. 51–56, 2007. Citado na página 32.
- MITRA, S.; BANKA, H.; PAL, S. K. A moe framework for biclustering of microarray data. **18th International Conference on Pattern Recognition, ICPR 2006.**, v. 1, p. 1154–1157, 2006. Citado na página 34.
- NVIDIA, C. Compute unified device architecture programming guide. 2007. Citado na página 75.
- _____. **Programming guide**. 2008. Citado na página 75.
- O’CONNOR, M.; COSLEY, D.; KONSTAN, J. A.; RIEDL, J. Polylens: a recommender system for groups of users. **ECSCW 2001**, p. 199–218, 2001. Citado na página 27.
- PARK, S. T.; CHU, W. Pairwise preference regression for cold-start recommendation. **Proceedings of the third ACM conference on Recommender systems**, p. 21–28, 2009. Citado na página 49.
- PEREIRA, A. L. V.; HRUSCHKA, E. R. Simultaneous co-clustering and learning to address the cold start problem in recommender systems. **Knowledge-Based Systems**, Elsevier, v. 82, p. 11–19, 2015. Citado 3 vezes nas páginas 24, 58 e 93.
- PERERA, S. Hadoop mapreduce cookbook. **Packt Publishing Ltd**, 2013. Citado na página 77.
- PHAM, M. C.; CAO, Y.; KLAMMA, R.; JARKE, M. A clustering approach for collaborative filtering recommendation using social network analysis. **UCS**, v. 17, n. 4, p. 583–604, 2011. Citado na página 32.
- POPESCU, A.; PENNOCK, D. M.; LAWRENCE, S. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In: MORGAN KAUFMANN PUBLISHERS INC. **Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence**. [S.l.], 2001. p. 437–444. Citado na página 31.
- PORCEL, C.; TEJEDA-LORENTE, A.; MARTÍNEZ, M.; HERRERA-VIEDMA, E. A hybrid recommender system for the selective dissemination of research resources in a technology transfer office. **Information Sciences**, Elsevier, v. 184, n. 1, p. 1–19, 2012. Citado na página 31.
- PRELIĆ, A.; BLEULER, S.; ZIMMERMANN, P.; WILLE, A.; BÜHLMANN, P.; GRUISSEM, W.; HENNIG, L.; THIELE, L.; ZITZLER, E. A systematic comparison and evaluation of biclustering methods for gene expression data. **Bioinformatics**, Oxford Univ Press, v. 22, n. 9, p. 1122–1129, 2006. Citado na página 35.
- PROTIC, J.; TOMASEVIC, M.; MILUTINOVIĆ, V. Distributed shared memory: Concepts and systems. **John Wiley & Sons**, v. 21, 1998. Citado na página 71.
- RAMANATHAN, R. Intel® multi-core processors. **Making the Move to Quad-Core and Beyond**, 2006. Citado na página 72.

- RESNICK, P.; IACOVOU, N.; SUCHAK, M.; BERGSTROM, P.; RIEDL, J. Grouplens: an open architecture for collaborative filtering of netnews. **Proceedings of the 1994 ACM conference on Computer supported cooperative work**, p. 175–186, 1994. Citado na página 21.
- RESNICK, P.; VARIAN, H. R. Recommender systems. **Communications of the ACM**, ACM, v. 40, n. 3, p. 56–58, 1997. Citado na página 20.
- REZENDE, S. O. Sistemas inteligentes: fundamentos e aplicações. **Editora Manole Ltda**, 2003. Citado na página 29.
- RICCI, F.; ROKACH, L.; SHAPIRA, B.; KANTOR, P. B. Recommender systems handbook. **Springer**, 2011. Citado 2 vezes nas páginas 20 e 21.
- RONGFEI, J.; MAOZHONG, J.; CHAO, L. A new clustering method for collaborative filtering. **Networking and Information Technology (ICNIT), 2010 International Conference on**, p. 488–492, 2010. Citado na página 32.
- RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Cognitive modeling**, v. 5, p. 3, 1988. Citado na página 60.
- SAHEBI, S.; COHEN, W. Community-based recommendations: a solution to the cold start problem. **Workshop on Recommender Systems and the Social Web, RSWEB**, 2011. Citado na página 46.
- SANDERSON, C. Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments. NICTA, 2010. Citado na página 78.
- SARWAR, B.; KARYPIS, G.; KONSTAN, J.; RIEDL, J. Item-based collaborative filtering recommendation algorithms. **Proceedings of the 10th international conference on World Wide Web**, p. 285–295, 2001. Citado na página 53.
- SAVOJARDO, C.; FARISELLI, P.; CASADIO, R. Improving the detection of transmembrane β -barrel chains with n-to-1 extreme learning machines. **Bioinformatics**, Oxford Univ Press, v. 27, n. 22, p. 3123–3128, 2011. Citado 2 vezes nas páginas 60 e 61.
- SCHEIN, A. I.; POPESCU, A.; UNGAR, L. H.; PENNOCK, D. M. Methods and metrics for cold-start recommendations. **Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval**, p. 253–260, 2002. Citado 3 vezes nas páginas 31, 45 e 49.
- SHANG, T.; HE, Q.; ZHUANG, F.; SHI, Z. Extreme learning machine combining matrix factorization for collaborative filtering. In: IEEE. **Neural Networks (IJCNN), The 2013 International Joint Conference on**. [S.l.], 2013. p. 1–8. Citado na página 62.
- SHARDANAND, U.; MAES, P. Social information filtering: algorithms for automating “word of mouth”. **Proceedings of the SIGCHI conference on Human factors in computing systems**, p. 210–217, 1995. Citado 2 vezes nas páginas 21 e 28.
- SHAW, G.; XU, Y.; GEVA, S. Using association rules to solve the cold-start problem in recommender systems. **Advances in Knowledge Discovery and Data Mining**, Springer, p. 340–347, 2010. Citado 2 vezes nas páginas 47 e 49.

- SHINDE, S. K.; KULKARNI, U. Hybrid personalized recommender system using centering-bunching based clustering algorithm. **Expert Systems with Applications**, Elsevier, v. 39, n. 1, p. 1381–1387, 2012. Citado na página 31.
- SOBHANAM, H.; MARIAPPAN, A. Addressing cold start problem in recommender systems using association rules and clustering technique. **Computer Communication and Informatics (ICCCI), 2013 International Conference on**, p. 1–5, 2013. Citado na página 47.
- SOBOROFF, I.; NICHOLAS, C. Combining content and collaboration in text filtering. In: **Proceedings of the IJCAI**. [S.l.: s.n.], 1999. v. 99, p. 86–91. Citado na página 31.
- SOKOLOVA, M.; JAPKOWICZ, N.; SZPAKOWICZ, S. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In: **AI 2006: Advances in Artificial Intelligence**. [S.l.]: Springer, 2006. p. 1015–1021. Citado na página 65.
- SOTOMAYOR, R.; CARTHY, J.; DUNNION, J. **The design and implementation of an intelligent online recommender system**. Tese (Doutorado) — University College Dublin, 2007. Citado na página 20.
- STONE, J. E.; GOHARA, D.; SHI, G. Opencl: A parallel programming standard for heterogeneous computing systems. **Computing in science & engineering**, Institute of Electrical and Electronics Engineers, Inc., y USA United States, v. 12, n. 1-3, p. 66–73, 2010. Citado na página 74.
- SYMEONIDIS, P.; NANOPOULOS, A.; PAPADOPOULOS, A. N.; MANOLOPOULOS, Y. Nearest-biclusters collaborative filtering based on constant and coherent values. **Information retrieval**, Springer, v. 11, n. 1, p. 51–75, 2008. Citado 3 vezes nas páginas 32, 34 e 35.
- TANG, C.; ZHANG, L.; ZHANG, A.; RAMANATHAN, M. Interrelated two-way clustering: an unsupervised approach for gene expression data analysis. **Bioinformatics and Bioengineering Conference, 2001. Proceedings of the IEEE 2nd International Symposium on**, p. 41–48, 2001. Citado na página 34.
- THORNLEY, C. Advances in information retrieval. 32nd european conference on ir research, ecir 2010 milton keynes, uk, march 28-31 2010 proceedings. **Journal of Documentation**, Emerald Group Publishing Limited, v. 68, n. 2, p. 270–270, 2012. Citado na página 29.
- TOSATTO, S. C.; BINDEWALD, E.; HESSER, J.; MÄNNER, R. A divide and conquer approach to fast loop modeling. **Protein engineering**, Oxford Univ Press, v. 15, n. 4, p. 279–286, 2002. Citado na página 39.
- UNGAR, L. H.; FOSTER, D. P. Clustering methods for collaborative filtering. **AAAI workshop on recommendation systems**, v. 1, p. 114–129, 1998. Citado na página 32.
- VICTOR, P.; COCK, M. D.; CORNELIS, C.; TEREDESAL, A. Getting cold start users connected in a recommender system's trust network. **Computational Intelligence in Decision and Control**, v. 1, p. 877–882, 2008. Citado 2 vezes nas páginas 46 e 49.
- WANG, E.; ZHANG, Q.; SHEN, B.; ZHANG, G.; LU, X.; WU, Q.; WANG, Y. Intel math kernel library. **High-Performance Computing on the Intel® Xeon Phi™**, Springer, p. 167–188, 2014. Citado na página 77.

WANG, G.; WANG, Y. Probabilistic attribute mapping for cold start recommendation. **Foundations and Applications of Intelligent Systems**, Springer, p. 421–431, 2014. Citado na página 46.

WANG, J.-N.; JIN, J.-L.; GENG, Y.; SUN, S.-L.; XU, H.-L.; LU, Y.-H.; SU, Z.-M. An accurate and efficient method to predict the electronic excitation energies of bodipy fluorescent dyes. **Journal of computational chemistry**, Wiley Online Library, v. 34, n. 7, p. 566–575, 2013. Citado 2 vezes nas páginas 60 e 61.

WILLMOTT, C. J.; MATSUURA, K. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. **Climate research**, INTER-RESEARCH NORDBUNTE 23, D-21385 OLDENDORF LUHE, GERMANY, v. 30, n. 1, p. 79, 2005. Citado na página 65.

XIANYI, Z.; QIAN, W.; CHOTHIA, Z. Openblas. URL: <http://xianyi.github.io/OpenBLAS>, 2012. Citado na página 78.

YAGER, R. R. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. **Systems, Man and Cybernetics, IEEE Transactions on**, IEEE, v. 18, n. 1, p. 183–190, 1988. Citado na página 47.

YU, Y.; CHOI, T.-M.; HUI, C.-L. An intelligent quick prediction algorithm with applications in industrial control and loading problems. **IEEE Transactions on Automation Science and Engineering**, IEEE, v. 9, n. 2, p. 276–287, 2012. Citado 2 vezes nas páginas 60 e 61.

ZHANG, Z.-K.; LIU, C.; ZHANG, Y.-C.; ZHOU, T. Solving the cold start problem in recommender systems with social tags. **EPL (Europhysics Letters)**, IOP Publishing, v. 92, n. 2, p. 28002, 2010. Citado 2 vezes nas páginas 46 e 49.

ZHENG, N.; LI, Q. A recommender system based on tag and time information for social tagging systems. **Expert Systems with Applications**, Elsevier, v. 38, n. 4, p. 4575–4587, 2011. Citado na página 27.