Explorando Decomposição por Valor Singular em Filtragem Colaborativa

João Bosco A. Pereira Filho[1], Renato Fileto[1], Pedro Barbetta[1], Guilherme Bittencourt[2]

[1]Programa de Pós-Graduação em Ciência da Computação (PPGCC/INE/CTC/UFSC)
[2]Departamento de Automação e Sistemas (DAS/CTC/UFSC)
Universidade Federal de Santa Catarina (UFSC), Florianópolis-SC, 88040-900, Brasil
bosco@chaordicsystems.com, {fileto,barbetta}@inf.ufsc.br, gb@das.ufsc.br

Abstract. This paper proposes a collaborative filtering algorithm, based on Singular Value Decomposition (SVD) that models the profiles of a large set of users, in order to make personalized recommendations for them. Experiments performed with this algorithm and a comparative analysis with a KNN algorithm shows that the proposed algorithm has better efficacy, despite being slower, for the database used as a case study.

Keywords: recommender systems, collaborative filtering, SVD.

1 Introdução

A facilidade de criação e publicação de conteúdos na Internet e a crescente quantidade de informação disponível fazem com que cada novo item colocado na Internet dispute o recurso mais escasso da sociedade digital: a atenção do usuário. Em resposta aos desafios da sobrecarga de informação e à necessidade de personalização na Internet, tem-se procurado desenvolver sistemas nos quais os usuários possam identificar rapidamente os conteúdos que mais lhes interessam. A dificuldade é ainda maior quando as preferências do usuário são fatores fundamentais para se conhecer a relevância de uma determinada informação. Para saber quais são os itens mais interessantes, proveitosos, valiosos ou divertidos para uma determinada pessoa, é fundamental entender seu gosto.

Sistemas de recomendação têm o objetivo de facilitar o acesso de usuários às informações que mais lhes interessam. Tais sistemas são bastante comuns em *sites* de comércio eletrônico, como o sistema de recomendações utilizado pela empresa Amazon.com, que é responsável por 35% das vendas de produtos da empresa [9].

Os sistemas de recomendação podem ser subdivididos em duas categorias: baseados em conteúdo e baseados em filtragem colaborativa [1]. Os sistemas baseados em conteúdo utilizam as informações sobre os itens disponíveis para fazer uma recomendação [21]. Por exemplo, um sistema de recomendação de artigos científicos que pode usar as palavras-chave para fazer as recomendações [15].

Os sistemas baseados em filtragem colaborativa, por outro lado, fazem recomendações a um usuário com base nas ações de outros usuários. Um algoritmo típico de filtragem colaborativa pode, por exemplo, avaliar que Maria tem preferências parecidas com as do João e, então, recomendar para ele os itens que a

Maria gosta. A combinação de comportamento, preferências ou idéias de um grupo de pessoas para a criação de novos conhecimentos é chamada inteligência coletiva [14].

Dentre as técnicas utilizadas em filtragem colaborativa, uma das mais populares é a baseada em Decomposição por Valor Singular (*Singular Value Decomposition*, SVD) [18]. SVD é uma técnica algébrica de fatoração de matrizes. É utilizada em filtragem colaborativa para descobrir características latentes dos usuários e dos itens sendo avaliados. Desta maneira, pode-se inferir as preferências de cada indivíduo e então fazer recomendações. Existem diversas implementações de SVD disponíveis em produtos comerciais, como o *Matlab*, por exemplo. Entretanto, a aplicação direta de algoritmos de SVD nos problemas de filtragem colaborativa não garante boa eficácia, sendo necessários diversos ajustes para desenvolver bons algoritmos [8].

Este artigo apresenta uma proposta de algoritmo de filtragem colaborativa baseado em SVD. Também apresenta os resultados dos experimentos realizados com o algoritmo proposto e uma análise comparativa com outros algoritmos, na qual pode se observar que o algoritmo proposto possui maior eficácia, apesar de necessitar de mais tempo de processamento. O trabalho é avaliado no contexto do concurso *Netflix Prize* [2], que fornece uma base de dados bastante completa para experimentações, além de uma metodologia de avaliação de eficácia dos algoritmos.

A seção 2 deste artigo descreve mais detalhadamente a recomendação por filtragem colaborativa e explica formalmente o método de avaliação de recomendações utilizado no *Netflix Prize*. A seção 3 apresenta um algoritmo KNN, comumente utilizado em filtragem colaborativa. A seção 4 apresenta diversos detalhes do método SVD e sua relação com filtragem colaborativa. A seção 5 apresenta o algoritmo SVD proposto. A seção 6 apresenta os experimentos realizados. Finalmente, a seção 7 contém as conclusões e indicações de trabalhos futuros.

2 Filtragem Colaborativa

Sistemas de Filtragem Colaborativa utilizam a colaboração de seus usuários para prover recomendações personalizadas. Tais sistemas podem utilizar as notas que os usuários atribuem a itens, as quais representam o grau de afinidade dos usuários com aqueles itens. O objetivo é descobrir itens ainda não avaliados por usuários e que provavelmente irão agradá-los. Ou seja, o problema de recomendação se reduz ao problema de predizer as notas que usuários atribuiriam a itens não avaliados. Notas altas atribuídas a itens significam que estes devem ser recomendados para o usuário em questão.

A Tabela 1 apresenta um exemplo de matriz de dados de avaliações de filmes. Neste exemplo, o usuário *João* atribuiu nota 5 ao filme *O Poderoso Chefão*, nota 1 a *Miss Simpatia*, não avaliou *Duro de Matar* e atribuiu nota 4 a *Matrix*. O objetivo de sistemas de filtragem colaborativa é completar esta matriz predizendo as notas desconhecidas. Note-se que esta pode ser uma tarefa bastante difícil, já que em um sistema real esta matriz pode ter dimensão muito grande e ser extremamente esparsa, pois são poucos os usuários que se dispõe a avaliar itens.

A base de dados disponibilizada pelo *Netflix Prize* [2], usada para avaliar algoritmos neste trabalho, tem forma similar à da Tabela 1. Ela contém 100.480.507

avaliações de 480.189 usuários, referentes a 17.770 filmes, gerando uma matriz de 8,5 bilhões de células, sendo esta 98% esparsa. A *NetFlix* disponibiliza dois arquivos: um de prova (*probe.txt*) e outro de classificação (*qualifying.txt*). O primeiro contém 1.408.395 notas para pares usuário-filme e deve ser utilizado para treinar o algoritmo. O segundo arquivo, *qualifying.txt*, possui 2.817.131 pares usuário-filme para os quais somente a *Netflix* conhece as notas atribuídas. Deve-se então tentar predizer estas notas e enviar o resultado à *Netflix*, que calcula a precisão obtida.

Tabela 1: Matriz de avaliações de filmes

	O Poderoso Chefão	Miss Simpatia	Duro de Matar	Matrix
João	5	1	?	4
Lais	?	4	1	1
Lígia	?	5	?	1
Fátima	2	4	1	1

A avaliação da precisão dos algoritmos concorrentes ao *Netflix Prize* é realizada através da raiz quadrada do erro quadrático médio (*Root Mean Square Error*, RMSE). Seja $\hat{y}_{um}$ a nota predita do usuário u para o filme m, y_{um} a nota verdadeira e n o número de predições realizadas o RMSE é calculado por:

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

Quanto menor o RMSE obtido por um algoritmo, melhor a qualidade do mesmo. O sistema proprietário *Cinematch* da *Netflix* obteve um RMSE de 0,9514 [10].

3 KNN

O algoritmo *K Nearest Neighbor* (KNN) é um método de análise de vizinhança que utiliza uma técnica de aprendizado supervisionado não-paramétrico. Ele é utilizado para fazer predições (classificações ou regressões) nas áreas de *data mining*, reconhecimento de padrões, processamento digital de imagens, etc.

KNN foi uma das primeiras técnicas usadas em filtragem colaborativa [4]. A idéia geral é descobrir usuários parecidos com um determinado usuário e, então, utilizar as notas destes "vizinhos" para predizer a nota que o usuário em questão daria para um determinado item. Tal algoritmo se baseia na semelhança entre usuários para fazer as predições e, portanto, é classificado como *user-based*. Ele pode ser utilizado para analisar a semelhança entre itens, resultando em um algoritmo *item-based*, que normalmente apresenta melhor eficiência e eficácia [13].

O algoritmo KNN utilizado na comparação com o algoritmo proposto neste artigo é *item-based*. Ele utiliza o coeficiente de correlação de Pearson [19] para medir a similaridade entre itens e um tratamento de intervalos de confiança para controlar a precisão de cada estimativa dos coeficientes de correlação. Tal algoritmo atingiu um

RMSE no *Netflix Prize* de 0,9253, o que representa uma melhoria de 2,85% sob o algoritmo de referência do concurso [3].

4 SVD

Decomposição por valor singular (SVD) é uma técnica algébrica de fatoração de matrizes frequentemente usada para descobrir características latentes (i.e. escondidas) nos dados [22]. SVD analisa qualquer matriz em busca de correlações e agrupa os dados correlacionados, causando uma redução dimensional na matriz. Dada uma matriz $\boldsymbol{T}$ de dimensão *nxm* , SVD a transforma no produto de três matrizes:

$$\boldsymbol{T}_{nxm} = \boldsymbol{S}_{nxn} \sum_{nxm} \boldsymbol{V}^{T}_{mxm} \quad (\mathbf{2})$$

onde $\boldsymbol{S}$ é a matriz ortogonal cujas colunas são os vetores singulares da esquerda, $\boldsymbol{V}^{T}$ é a matriz ortogonal cujas colunas são os vetores singulares da direita, e $\sum$ é a matriz diagonal de valores singulares positivos e decrescentes. Um exemplo:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \sqrt{0.2} & 0 & 0 & 0 & \sqrt{0.8} \\ 0 & 0 & 0 & 1 & 0 \\ -\sqrt{0.8} & 0 & 0 & 0 & \sqrt{0.2} \end{bmatrix}$$

Ao observar a fatoração da matriz acima, a redução dimensional não fica clara, pois uma matriz 4x5 foi obtida pelo produto de três matrizes: uma 4x4 à esquerda, uma 4x5 no centro e uma 5x5 à direita. Entretanto, pode-se optar pelo uso de SVD truncado [16], que representa apenas algumas dimensões, reduzindo o tamanho das matrizes de decomposição. Para 2 dimensões no exemplo anterior, tem-se:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \end{bmatrix} \approx \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 4 & 0 \\ 0 & 3 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \end{bmatrix}$$

Neste caso, fica evidente a redução dimensional nas matrizes de decomposição, já que as matrizes obtidas são $\boldsymbol{S}_{4x2}$, $\sum_{2x2}$ e $\boldsymbol{V}^{T}_{2x5}$. Observa-se também que a fatoração gera uma aproximação da matriz inicial e, portanto, existe um erro associado a este processo. É uma propriedade da SVD que este erro é o mínimo erro quadrático de uma aproximação *rank-d* da matriz original [7].

Uma maneira mais simples e comumente utilizada de fatoração SVD utiliza apenas duas matrizes de decomposição para aproximar a matriz inicial. Neste caso, pode-se, por exemplo, multiplicar a matriz diagonal pela matriz da direita. No caso do exemplo anterior, obtém-se a seguinte fatoração:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \end{bmatrix} \approx \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \end{bmatrix}$$

4.1 SVD e Filtragem Colaborativa

SVD é uma das formas de fatoração de matrizes usadas em filtragem colaborativa [17]. Por exemplo, ao aplicar SVD a uma matriz de avaliações de filmes por usuários pode-se descobrir algumas características que fazem os usuários avaliar positiva ou negativamente certos tipos de filmes. Estas características podem ser gêneros de filmes, participação de atores famosos, quantidade de prêmios recebidos, ou mesmo características mais complexas que não se pode entender, mas que estatisticamente fazem sentido. Está fora do escopo deste trabalho tentar descobrir quais são exatamente estas características, sendo necessário apenas entender a representatividade de cada uma delas com relação ao conjunto de dados analisados. Aplicando SVD a uma matriz de avaliações $\boldsymbol{T}$, tem-se:

$$\boldsymbol{T}_{nxm} \approx \boldsymbol{U}_{nxd}\boldsymbol{M}^{T}_{mxd} = \boldsymbol{Q}_{nxm} \quad (\mathbf{3})$$

onde $\boldsymbol{U}$ é a matriz formada por vetores de características de usuários, $\boldsymbol{M}$ é a matriz formada por vetores de características de filmes e $\boldsymbol{Q}$ é uma aproximação de $\boldsymbol{T}$. Cada vetor-linha $\boldsymbol{U}_i \in \boldsymbol{U}$ $(i < n)$ representa a relação do usuário i com as características descobertas e cada vetor-linha $\boldsymbol{M}_j \in \boldsymbol{M}$ $(j < m)$ representa a relação do filme j com essas características.

Por exemplo, ao executar o SVD sobre uma matriz de avaliação descobriu-se três características importantes: C1, C2 e C3 (significando "filmes de ação", "filmes românticos" e "filmes lançamentos", por exemplo). Supõe-se ainda que os vetores do usuário "João", do usuário "Ligia" e do filme "Duro de Matar" são, respectivamente:

$$\boldsymbol{U}_J = [2 \;\; 0{,}4 \;\; 1{,}4] \quad \boldsymbol{U}_L = [0{,}2 \;\; 2 \;\; 1{,}8] \quad \boldsymbol{M}_D = [2 \;\; 0{,}4 \;\; 0{,}6]$$

Ou seja, o usuário "João" prefere filmes de ação (2), não gosta muito de filmes românticos (0,4) e tem preferência por filmes de lançamento (1,4). Já o usuário "Lígia" não gosta de filmes de ação (0,2), prefere filmes românticos (2) e filmes lançamento (1,8). Ainda segundo este exemplo, o filme "Duro de Matar" é um filme de ação (2), não é muito romântico (0,4) e também já é um pouco antigo (0,6). Uma maneira de obter a predição das notas que "João" e "Lígia" dariam ao filme seria através do produto escalar de seus vetores com o vetor do filme em questão:

$$\boldsymbol{U}_J \,.\, \boldsymbol{M}_D^T = 4{,}58 \quad \boldsymbol{U}_L \,.\, \boldsymbol{M}_D^T = 2{,}28$$

Como a nota predita para “João” (4,58) é maior do que a nota predita para ”Lígia” (2,28), provavelmente o primeiro usuário gostaria mais do filme que o segundo.

5 Um algoritmo de filtragem colaborativa baseado em SVD

Existem muitas formas de se fazer fatoração de matrizes [6]. Optou-se neste trabalho por fazer a fatoração através do cálculo do gradiente-descendente de uma função de erro, técnica muito utilizada em treinamento de redes neurais *feed forward* [12]. A função de erro utilizada é definida pela seguinte equação [8]:

$$\boldsymbol{E} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\boldsymbol{I}_{ij}(\boldsymbol{T}_{ij} - p(\boldsymbol{U}_i, \boldsymbol{M}_j))^2 \tag{4}$$

onde:

- $\boldsymbol{T}_{nxm}$: matriz de avaliações conhecidas, contendo as avaliações de n usuários para m filmes, podendo esta matriz ser bastante esparsa;
- $\boldsymbol{I}_{nxm}$: matriz binária de indicação de presença, na qual $\boldsymbol{I}_{ij} = 1$ se o usuário i avaliou o filme j e $\boldsymbol{I}_{ij} = 0$ se o usuário i não avaliou o filme j;
- $p(\boldsymbol{U}_i, \boldsymbol{M}_j)$: função de predição da nota do usuário i para o filme j, definido:

$$p(\boldsymbol{U}_i, \boldsymbol{M}_j) = \begin{cases} 1, & se\ \boldsymbol{U}_i\boldsymbol{M}_j < 1 \\ \boldsymbol{U}_i\boldsymbol{M}_j, & se\ 1 < \boldsymbol{U}_i\boldsymbol{M}_j < 5 \\ 5, & se\ \boldsymbol{U}_i\boldsymbol{M}_j > 5 \end{cases} \tag{5}$$

- $\boldsymbol{U}_{nxd}$: Matriz de vetores de d características de usuários, com $d < n$;
- $\boldsymbol{M}_{mxd}$: Matriz de vetores de d características de filmes, com $d < m$;

Calculando-se os gradientes da função de erro através da derivada parcial de $\boldsymbol{E}$ com relação a $\boldsymbol{U}$ e $\boldsymbol{M}$, obtém-se:

Para i = 1 ...n:

$$\nabla \boldsymbol{U} = \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{U}_i} = \sum_{j=1}^{m}\boldsymbol{I}_{ij}((\boldsymbol{T}_{ij} - p(\boldsymbol{U}_i, \boldsymbol{M}_j))\boldsymbol{M}_j$$

Para j = 1 ...m:

$$\nabla \boldsymbol{M} = \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{M}_j} = \sum_{j=1}^{n}\boldsymbol{I}_{ij}\left((\boldsymbol{T}_{ij} - p(\boldsymbol{U}_i, \boldsymbol{M}_j)\right)\boldsymbol{U}_i \tag{6}$$

Ao implementar algoritmos baseados em um modelo é necessário avaliar a capacidade de generalização do mesmo. Se o modelo gerado se adequar demais aos dados de treinamento, fica caracterizado *overfitting* e a capacidade de generalização do algoritmo pode ficar prejudicada. Ou seja, o algoritmo possui ótima eficácia com os dados de treinamento, mas pode possuir péssima eficácia com novos dados.

Uma forma de se evitar *overfitting* é pelo uso de coeficientes de regularização [20]. Tais coeficientes podem ser usados para introduzir um ruído na função a ser

analisada, impossibilitando, então, que o modelo se adéqüe demais aos dados de treinamento.

O algoritmo aqui proposto utiliza um coeficiente de regularização k, que introduz um ruído na função de erro que se deseja minimizar com o gradiente-descendente:

$$\boldsymbol{E} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\boldsymbol{I}_{ij}(\boldsymbol{T}_{ij} - p(\boldsymbol{U}_i, \boldsymbol{M}_j))^2 + \frac{k}{2}\sum_{i=1}^{n}\|\boldsymbol{U}_i\|^2 + \frac{k}{2}\sum_{j=1}^{m}\|\boldsymbol{M}_j\|^2 \quad (7)$$

Desta forma, os novos gradientes calculados são:

$$\nabla\boldsymbol{U} = \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{U}_i} = \sum_{j=1}^{m}\boldsymbol{I}_{ij}((\boldsymbol{T}_{ij} - p(\boldsymbol{U}_i, \boldsymbol{M}_j))\boldsymbol{M}_j - k\boldsymbol{U}_i$$

$$\nabla\boldsymbol{M} = \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{M}_j} = \sum_{j=1}^{n}\boldsymbol{I}_{ij}\left((\boldsymbol{T}_{ij} - p\left(\boldsymbol{U}_i, \boldsymbol{M}_j\right)\right)\boldsymbol{U}_i - k\boldsymbol{M}_j \quad (8)$$

Um método simples de treinamento é avaliar todos os dados, calcular os gradientes do erro e atualizar as matrizes de decomposição $\boldsymbol{U}$ e $\boldsymbol{M}$ de maneira iterativa até se obter bons valores de $\boldsymbol{U}$ e $\boldsymbol{M}$. Entretanto, este método pode ocasionar tempo computacional muito elevado quando utilizado com grandes bases de dados.

Pode-se, então, atualizar as matrizes $\boldsymbol{U}$ e $\boldsymbol{M}$ sem avaliar todos os dados de treinamento. O algoritmo aqui proposto utiliza uma forma incremental de se calcular SVD, na qual cada característica é treinada separadamente e a atualização das matrizes de decomposição é feita após a análise de cada avaliação do conjunto de treinamento. Ou seja, para cada nota atribuída por um usuário a um filme, calcula-se o gradiente e atualizam-se as matrizes $\boldsymbol{U}$ e $\boldsymbol{M}$. Em pseudo-código:

```
SVD
INPUT: quantidade de características (d), taxa de
aprendizado (μ), numero de épocas (e)
OUTPUT: matrizes de características U e M treinadas
ALGORITMO:
Inicializa as matrizes U e M
Para cada característica:
 Para cada época (ou enquanto RMSE estiver diminuindo):
  Para cada avaliação Tij conhecida:
   Computar ∇Ui
   Computar ∇Mj
   Ui ← Ui − μ∇Ui
   Mj ← Mj − μ∇Mj
```

O método acima computa os gradientes durante certa quantidade de épocas ou enquanto o RMSE continuar diminuindo. Ou seja, ele pára assim que verificar que a eficácia do mesmo está piorando. Esta técnica é chamada de *early stopping* e é realizada para evitar *overfitting*. Entretanto, o *early stopping* pode também achar ótimos locais e prejudicar a eficácia do método [11].

No pseudo-código apresentado anteriormente foi mencionada a inicialização das matrizes $\boldsymbol{U}$ e $\boldsymbol{M}$, porém não explicitou-se como esta inicialização ocorre. Entretanto, a

forma de se inicializar estas matrizes pode influenciar na velocidade de convergência do gradiente-descendente e, portanto, é importante analisar algumas alternativas.

Pode-se inicializar as matrizes $\boldsymbol{U}$ e $\boldsymbol{M}$ de maneira trivial, escolhendo uma constante qualquer, como por exemplo, 0,1 [5].

Outra forma de inicializar as matrizes é fazê-la de forma que o produto escalar das mesmas resulte na média de todas as avaliações. Ou seja, supondo-se que a média de notas dos usuários para os itens do sistema seja $\overline{\boldsymbol{y}}$, tem-se:

$$\boldsymbol{U}.\boldsymbol{M} = \begin{bmatrix} \overline{y} & \cdots & \overline{y} \\ \vdots & \ddots & \vdots \\ \overline{y} & \cdots & \overline{y} \end{bmatrix} \quad (\mathbf{9})$$

Sendo $\boldsymbol{d}$ o numero de dimensões, $\boldsymbol{U_{ij}}$ e $\boldsymbol{M_{ij}}$ os valores iniciais das matrizes, para se inicializar as matrizes de forma que a predição final do algoritmo seja igual à média de avaliações do sistema tem-se:

$$\boldsymbol{U}_{ij} = \boldsymbol{M}_{ij} = \sqrt{\frac{\overline{y}}{d}} \quad (\mathbf{10})$$

Entretanto, inicializar todos os valores das matrizes de maneira igual pode prejudicar a convergência. Portanto, é sugerido adicionar um pequeno ruído $n(r)$ à equação **(9)**, sendo este ruído uma variável aleatória de distribuição uniforme $[-r, r]$, com $\boldsymbol{r}$ pequeno [8]. Assim:

$$\boldsymbol{U}_{ij} = \boldsymbol{M}_{ij} = \sqrt{\frac{\overline{y}}{d}} + n(r) \quad (\mathbf{11})$$

Certamente existem alguns filmes que são de melhor qualidade e que possuem uma nota média maior do que outros. Entretanto, a abordagem anterior trata cada filme de maneira igualitária, o que também reduz a velocidade de convergência.

O algoritmo proposto neste artigo tenta diminuir este problema ao inicializar os valores das matrizes $\boldsymbol{U}$ e $\boldsymbol{M}$ de forma que os valores de predição levem em consideração o filme em questão. Desta forma utiliza-se a média de cada filme ao invés da média geral de notas:

$$\boldsymbol{U}_{ij} = \boldsymbol{M}_{ij} = \sqrt{\frac{\overline{y}_{j}}{d}} + n(r) \quad (\mathbf{12})$$

6 Experimentos

Os experimentos foram realizados com o algoritmo KNN descrito em [3] e com o algoritmo de SVD apresentado neste artigo, sendo o primeiro escrito em C e o segundo em C++. Ambos os algoritmos implementados são *single-threads*, não possuindo nenhum tipo de paralelização, o que poderia diminuir os tempos de processamento dos mesmos.

Os experimentos foram realizados no contexto do *Netflix Prize* e a medida de precisão (RMSE) foi calculada pela própria empresa *Netflix*. Tanto o KNN quanto o SVD foram treinados com os dados puros do concurso, não sendo realizado nenhum tipo de pré-processamento ou normalização.

Foi utilizado um computador com processador Intel Core 2 Duo de 2,80 GHz e 4GB de memória RAM com Linux Ubuntu 32 bits instalado.

Os parâmetros utilizados no SVD foram: a quantidade de dimensões (d) a serem representadas e a quantidade de épocas de treinamento (e). Já no KNN, optou-se pela utilização do número de vizinhos (k) a serem analisados. A Tabela 2 apresenta os resultados dos experimentos, ordenados crescentemente pelo RMSE.

Tabela 2: Resultados dos experimentos realizados

Algoritmo	Parâmetros de entrada	Tempo de processamento	Qualifying RMSE	Melhoria sob a referência (%)
SVD	d=256, e=120	18 horas	0,9171	3,60%
KNN	k=23	50 minutos	0,9253	2,85%
KNN	k=25	50 minutos	0,9258	2,80%
KNN	k=19	50 minutos	0,9273	2,64%
SVD	d=64, e=120	48 minutos	0,9264	2,62%
SVD	d= 10, e=12	15 minutos	0,9665	-1,58%

Os experimentos mostram que o algoritmo baseado em SVD produz resultados com menor RMSE do que o KNN para muitas dimensões. Entretanto, SVD com $\boldsymbol{d}$ grande necessita de muito tempo de processamento, sendo necessárias otimizações de desempenho do algoritmo para melhorar a relação custo-benefício, já que o tempo de processamento do KNN é praticamente constante devido à natureza do algoritmo.

7 Conclusões

Este artigo apresentou um algoritmo SVD para filtragem colaborativa que utiliza diversas abordagens presentes na literatura e algumas novas propostas, como a forma de inicialização das matrizes, por exemplo. Este algoritmo foi implementado e validado usando o *benchmark* da competição *Netflix Prize*, atingindo uma melhoria de 3,60% sobre o algoritmo referência do concurso.

Entre as possibilidades de trabalhos futuros estão (i) a paralelização do algoritmo proposto, para diminuir o tempo de processamento, que atualmente é bastante elevado; (ii) a normalização dos dados, visando maior eficácia e (iii) executar o algoritmo KNN sobre os resíduos do algoritmo SVD proposto, visando identificar padrões não identificados no primeiro algoritmo para também aumentar sua eficácia.

Agradecimentos: Este trabalho foi parcialmente suportado pelo CNPq sob o processo número 48139212007-6.

Referências

1. Balabanović, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Communications of the ACM vol. 40, 66--72 (1997)
2. Bennett, J., Lanning, S.: The Netflix Prize. KDD Cup and Workshop, ACM. San Jose, California (2007)
3. Bernartt, J.: Um sistema de recomendação baseado em filtragem colaborativa. Dissertação de mestrado. Universidade Federal de Santa Catarina, Florianópolis (2008)
4. Bradley, K., Rafter, R., Smyth, B.: Case-Based User Profiling for Content Personalisation. In: Brusilovsky, P., Stock, O., Strapparava, C. (Eds.) Adaptive Hypermedia and Adaptive Web-Based Systems, LNCS, vol. 1892, pp. 62--72. Springer, Heidelberg (2000)
5. Funk, S.: Netflix Update: Try This at Home, http://sifter.org/~simon/journal/20061211.html (Acesso em Junho de 2008)
6. Golub, G. H., Van Loan, C. F.: Matrix Computations. The Johns Hopkins University Press, Baltimore (1989)
7. Gorrell, G., Webb, B.: Generalized Hebbian Algorithm for Incremental Latent Semantic Analysis. European Conference on Speech Communication and Technology. Lisboa (2005)
8. Ma, C.: A Guide to Singular Value Decomposition for Collaborative Filtering. Technical Report, Department of Computer Science, National Taiwan University, Taipei, Taiwan (2008)
9. Marshal, M.: Aggregate Knowledge raises $5M from Kleiner on a roll, http://venturebeat.com/2006/12/10/aggregate-knowledge-raises-5m-from-kleiner-on-a-roll/ (Acesso Fevereiro de 2009)
10. Netflix Inc.: The Netflix Prize, http://www.netflixprize.com/ (Acesso em Outubro de 2008)
11. Prechelt, L.: Early Stopping - But When? In: Orr, G. B., Muller, K. R. (Eds) Neural Networks: Tricks of the Trade, LNCS, vol. 1524, pp. 55--69. Springer, Heidelberg (1998)
12. Rumelhart, D. E., Hinton, G. E., Williams, R. J.: Learning representations by back-propagating errors. In: Polk, T. A., Seifert, C. M. (Eds) Cognitive Modelling., pp. 213--222. The MIT Press (1986)
13. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. International World Wide Web Conference, Hong Kong (2001)
14. Segaran, T.: Programming Collective Intelligence: Building Smart Web 2.0 Applications. O'Reilly Media. Sebastopol, (2007)
15. Torres, R. D.: Combining Collaborative and Content-based Filtering to Recommend Research Papers. Universidade Federal do Rio Grande do Sul, Porto Alegre (2004)
16. Xu, P.: Truncated SVD methods for discrete linear ill-posed problems. Geophysical Journal International vol. 135, pp. 505--514, Royal Astronomical Society (1998)
17. Zhang, S., Wang, W., Ford, J., Makedon, F., Pearlman, J.: Using Singular Value Decomposition aproximation for Collaborative Filtering. In: 7th IEEE International Conference on E-Commerce Technology, pp. 257--264. IEEE Press (2005)
18. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Application of Dimensionality Reduction in Recommender System - A Case Study. ACM WebKDD 2000 Web Mining for E-Commerce. Boston (2000)
19. Rodgers, J.L., Nicewander, W.A.: Thirteen ways to look at the correlation coefficient. American Statistician vol 42, 59--64 (1988)
20. Bishop. C. M.: Training with Noise is Equivalent to Tikhonov Regularization. Neural Computation vol. 7, pp. 108--116 (1995)
21. Pazzani, M. J., Billsus D.: Content-Based Recommendation Systems. In: Brusilovsky, P., Kobsa A., Nejdl, W. (Eds) The adaptative Web. LNCS, vol. 4321, pp. 325-341. Springer, Heidelberg (2007)