Desafio de Pesquisa 1 - Sistemas de Recomendação

VINÍCIUS JULIÃO RAMOS**, Universidade Federal de Minas Gerais, Brasil

Sistemas de Recomendação podem ser desenvolvidos utilizando diversas técnicas. Cada uma delas possui um abordagem diferente e o desempenho correspondente varia de acordo com o tipo do problema. O trabalho prático que será apresentado, testou diferentes algorítimos e testou-os, validando o desempenho e os resultados. Além disso, também experimentou-se diferentes combinações de hiperparâmetros para um dos algoritmos que possuía melhor performance e que fora escolhido como solução final. Ademais, como o trabalho possui uma parte competitiva, a avaliação de desempenho levou em consideração a submissão das saídas dos experimentos na plataforma *Kaggle*, responsável por julgar e classificar os participantes.

CCS Concepts: • **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

Additional Key Words and Phrases: datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Vinícius Julião Ramos. 2018. Desafio de Pesquisa 1 - Sistemas de Recomendação. *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUÇÃO

Sistemas de recomendação são utilizados em larga escala pela principalmente pelas indústrias audiovisual e *e-commerce*. A aplicação dessa tecnologia permite oferecer aos usuários recomendações personalizadas sobre filmes, itens para compra ou músicas, por exemplo. Porém, o objetivo é endereçar itens aos usuários com um bom nível de acertividade, ou seja, os usuários realmente consumirem o item indicado pelos sistemas. Existem diferentes técnicas empregadas para esse fim e aquelas que foram utilizadas neste trabalho prático são categorizadas como **Recomendações Baseadas em Personalização**.

Além da implementação prática dos algoritmos, este trabalho prático propôs um desafio entre os alunos. Então, para obter uma boa colocação, testou-se duas abordagens para sistemas de recomendação. A primeira tentativa foi baseada em similaridade, utilizando a correlação de *Pearson*. Já a segunda tentativa, que resultou na solução final deste trabalho se baseou em fatoração de matrizes. Entretanto, dada a característica de esparsidade dos dados, nenhumas das soluções utilizou-se de matrizes. A justificativa para a não utilização de matrizes, é dada a complexidade de implementação de matrizes esparsas. Uma vez que as bibliotecas permitidas para este trabalho não forneciam suporte a matrizes esparsas, então seria necessária tal implementação.

2 MODELOS IMPLEMENTADOS

Como solicitado pela descrição do trabalho, o algoritmo que seria implementado deveria ser baseado em recomendações personalizadas. Entretanto havia uma barreira inicial na seleção de um algoritmo para implementação desse trabalho: a distribuição dos dados. Ao tentar criar uma matriz $usuario \times item$ para a entrada fornecida para o trabalho, a estrutura ocuparia 168 Giga Bytes de memória. Porém, em uma simples análise de pequenas amostras da entrada, observou-que que trata-se de dados esparsos. Logo, o ideal seria a utilização de uma estrutura de dados adequada para representar dados esparsos com baixo consumo de memória. Entretanto, como as bibliotecas fornecidas não possuem tal suporte, escolheu-se modelos os quais a utilização de matrizes não fosse um requisito.

2.1 Filtragem Colaborativa por Correlação de Pearson

Em linhas gerais, há duas maneiras para a implementação de recomendadores baseados em similaridade: `(i)` baseada em item e `(ii)` baseada em usuários. Estudos como [Sarwar et al. 2001] mostram que as recomendações baseadas em item, no geral, possuem melhor performance e retornam melhores recomendações. Por isso, nesta primeira tentativa de implementação, optou-se pela filtragem colaborativa baseada em itens. Entretanto, essa abordagem não se mostrou tão eficaz, dada a restrição de desempenho imposta pelo trabalho, no qual todo o *pipeline* do trabalho deveria executar ema aproximadamente cinco minutos, enquanto que a solução demorava de cinco a seis vezes mais do que o aceitável.

Apesar do mau desempenho observado, a técnica de substituição da matriz de notas pela lista de adjacências, serviu de base para a solução baseada em gradientes descendentes que foi implementada posteriormente. Pala ilustrar tal substituição, basta imaginar que a relação entre **usuário**, **item** e **nota** é um grafo bidirecional bipartido, no qual uma partição é dada pelos itens e a outra são os usuários. Nessa cenário, as notas formam os pesos das arestas entre os nós. Assim, foi possível resolver o problema que relacionava a distribuição dos dados e a memória utilizada na computação das recomendações. A Figura 1 ilustra a representação em forma de grafo para a tripla $usuro \times item \times nota$.

O comportamento assintótico do algoritmo é uma forte justificativa em que se explica o mau desempenho desta primeira implementação. Isso deve ao fato de que, dada a existência de m usuários e n itens, é necessário computar a similaridade para todos os pares de itens da base dados (ou pares de usuários). Como nesta implementação optou-se pela similaridade de itens, o comportamento assintótico da execução desse algoritmo é

$$O(n^2 m)$$

O calculo da similaridade do cosseno entre dois itens tem custo assintótico linear no numero de usuários, ou seja, $O(m)$. Entretanto, a quantidade de pares possíveis de itens é quadrática, o que nos

*Autor deste trabalho prático

Author's address: Vinícius Julião Ramos, viniciusjuliao@dcc.ufmg.br, Universidade Federal de Minas Gerais, Avenida Presidente Antônio Carlos, 6627, Pampulha, Belo Horizonte, Minas Gerais, Brasil.

0730-0301/2018/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

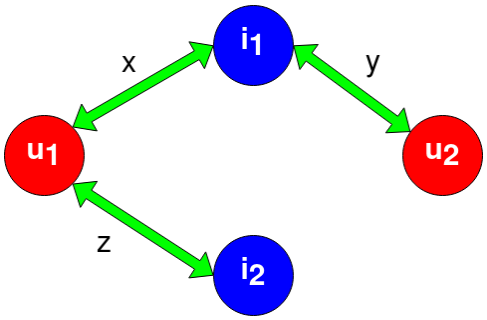

Fig. 1. Grafo da relação entre $usurio \times item \times nota$.

resulta no custo assintótico expresso acima. Vale ainda lembrar que a *Correlação de Pearson* introduz uma etapa de normalização dos dados que possui custo $O(mn)$, mas que não afeta o custo assintótico final, uma vez que $O(nm) + O(n^2m) = O(n^2m)$.

2.2 Fatoração de Matrizes por Gradiente Estocástico Descendente

1: **SGD**$(S, k = 100, \alpha = 0.005, \lambda = 0.02, l = 20)$
2: init P, Q
3: **for** $t = 1 \ldots l$
4: **for each** $r_{ui} \in S$
5: $e_{ui} = r_{ui} - \langle p_u, q_i \rangle$
6: $p_u^+ = p_u + \alpha(e_{ui}q_i - \lambda p_u)$
7: $q_i^+ = q_i + \alpha(e_{ui}p_u - \lambda q_i)$
8: $(p_u, q_i) = (p_u^+, q_i^+)$
9: **return** P, Q

Fig. 2. Algoritmo do calculo do gradiente descendente estocástico.

As técnicas mais populares e difundidas de fatoração de matrizes, estaão relacionadas à obtenção da matriz de valores singulares [Koren et al. 2009]. No geral, essas técnicas exigem a construção de uma matriz completa, na qual se preenchem com detemrinados valores auxiliares aqueles campos não fornecidos. Entretanto, a fatoração proposta por *Simon Funk* [1] não requer tal construção. Então, como tempo e memória são fatores limitantes para este trabalho, decidiu-se por utilizar um algoritmo capaz de gerar bons resultados, sem a necessidade de armazenar os dados das tripla $usuario \times item \times nota$ em uma matriz. Portanto, escolheu-se implementar o algoritmo de fatoração de matrizes utilizando gradiente descendente. Nele, é possível armazenar a entrada em uma estrutura linear, por exemplo uma lista de triplas, e mesmo que o resultado seja dado por duas matrizes P e Q, elas possuem dimensões $m \times k$ e $k \times n$ respectivamente. No final das contas, o custo extra de memória necessário seria $O(kn + km)$, porém como k trata-se de uma constante parametrizável (ou hiperparâmetro), o custo de memória extra, relacionado ao tamanho da entrada seria de:

$$O(n + m)$$

.

Para analisar custo da execução do algoritmo mostrado na Figura [?], vamos consideremos o parâmetro l que representa o número de épocas, k número de fatores, r como o tamanho da lista de triplas, e mantenhamos m e n como a quantidade de usuários e itens respectivamente. O custo de execução do laço mais interno (da linha 5 à linha 8) é $O(k)$. Entretanto, tal laço executará r vezes. Ademais, o `foor-loop` definido na linha 4 será executado uma vez para cada época, o que nos dá um custo assintótico total de:

$$O(klr)$$

.

Vale lembrar que o algoritmo mostrado na Figura 2, sofreu uma pequena modificação, para ajustar problemas de inicialização. Um dos problema de inicialização é caracterizado pela definição de um melhor valor para as matrizes P e Q, que foram atribuías o valor unitário em todas posições. Além disso, a fim de criar um novo ajuste de convergência, o valor de e_{ui} foi multiplicado por 2, antes da execução das linhas 6 e 7.

Table 1. Frequency of Special Characters

k	α	λ	l		tempo $mm:ss$	RMSE
4	0.0001	0.1	22	→	2:45	1.26905
4	0.0001	0.1	25	→	3:09	1.22008
4	0.0001	0.1	30	→	4:07	1.21723
4	0.0001	0.1	33	→	4:06	1.21571
4	0.0001	0.1	38	→	4:16	1.21343
4	0.0001	0.1	40	→	4:39	1.21260
4	0.0005	0.08	39	→	4:43	1.19421
4	0.0005	0.09	39	→	4:43	1.19414

3 RESULTADOS OBTIDOS

A solução final, descrita na Seção 2.2 foi submetida a testes *offline* antes da submissão na plataforma Kaggle. Para isso, a entrada de treinamento foi dividia num fator 80×20 no qual a maior parte dos dados foi utilizada para treinar o modelo, e os 20% restantes davam suporte aos testes que se baseavam no cálculo da Raiz do Erro Quadrático Médio. Esses testes foram feitos de maneira obter boas combinações de hiperparâmetros que executassem dentro dos limites de tempo impostos pelo trabalho. Sendo assim, as submissões com melhor performance e os respectivos resultados dados pelo *Kaggle* estao presentes na Tabela 1.

REFERENCES

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (aug 2009), 30–37. https://doi.org/10.1109/MC.2009.263

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (Hong Kong, Hong Kong) *(WWW '01)*. ACM, New York, NY, USA, 285–295. https://doi.org/10.1145/371920.372071

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

[1] https://sifter.org/ simon/journal/20061211.html – Acessado em 04 de Out. de 2022 às 09:21