Programming Assignment #1

Collaborative Movie Recommendation

Luiza Campos
luiza.chagas@dcc.ufmg.br
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais, Brasil

1 INTRODUÇÃO

O objetivo deste trabalho é implementar um recomendador de filmes colaborativo usando uma abordagem baseada em memória ou em modelo. Conforme discutido em aula [3], opções de implementação afetam a qualidade das recomendações colaborativas: normalização de dados, cálculo de similaridade, seleção de vizinhança, agregação de classificação e redução de dimensionalidade. As recomendações deviam ser enviadas ao Kaggle.

2 MÉTODO

O método de recomendação utilizado neste trabalho foi o SVD (*Singular Value Decomposition*) que foi proposto por [1]. Assim, levando em consideração as aulas da disciplina e a ideia inicial de Funk temos que dado uma matriz R de *ratings* cujas linhas são os usuários e as colunas os itens:

$$R = P\Sigma Q^T$$

Onde Σ é a matriz de fatores que tem os valores singulares na diagonal. Na abordagem básica, a matriz de *ratings* R $m \times n$ é aproximadamente fatorada em uma matriz P $m \times k$ e uma matriz Q $n \times k$:

$$R \approx PQ^T$$

Dessa aproximação extraímos uma fórmula para cada r_{ui} em R:

$$r_{ui} \approx \hat{r}_{ui} = p_u \cdot q_i^T = \sum_{f=1}^{k} p_{uf} \cdot q_{if}^T \tag{1}$$

E podemos definir o erro como:

$$e_{ui} = r_{ui} - \hat{r}_{ui} \tag{2}$$

Conforme descrito em [4], minimizar a soma do erro quadrático minimiza o RMSE. Tirando as derivadas parciais em relação a p_{uf} e q_{if} obtemos regras de atualização para os parâmetros:

$$p_{uf} += -\alpha \frac{\partial}{\partial p_{uf}} e_{ui}^2 = 2 * \alpha(e_{ui} q_{if})$$
$$q_{if} += -\alpha \frac{\partial}{\partial q_{if}} e_{ui}^2 = 2 * \alpha(e_{ui} p_{uf})$$

Tal que α é a taxa de aprendizado, ou *learning rate*. No caso deste trabalho, contudo, foram também usadas regularização e bias. Como explicado em [2], o cálculo de $\hat{r}$ é o mesmo. Contudo, na regularização temos:

$$p_{uf} += \alpha(e_{ui} q_{if} - \lambda p_{uf})$$
$$q_{if} += \alpha(e_{ui} p_{uf} - \lambda q_{if})$$

Grande parte da variação observada na classificação de valores é devido a efeitos associados aos usuários ou itens, que são conhecidos como vieses ou interceptações e são independentes de qualquer interação. A equação do viés, ou bias é:

$$b_{ui} = \mu + b_i + b_u \tag{3}$$

Assim, b_{ui} é o bias envolvido na avaliação r_{ui}. Na equação, μ denota a avaliação média dos usuários, b_i é o viés do item e b_u é o viés do usuário. Com bias, a equação que calcula $\hat{r}_{ui}$ é:

$$\hat{r}_{ui} = \mu + b_i + b_u + p_u \cdot q_i^T \tag{4}$$

Finalmente, as regras de otimização serão dadas por:

$$b_u += \alpha(e_{ui} - \lambda b_u) \tag{5}$$
$$b_i += \alpha(e_{ui} - \lambda b_i) \tag{6}$$
$$p_{uf} += \alpha(e_{ui} q_{if} - \lambda p_{uf}) \tag{7}$$
$$q_{if} += \alpha(e_{ui} p_{uf} - \lambda q_{if}) \tag{8}$$

O trabalho usa uma abordagem de gradiente descendente estocástico no treinamento.

3 IMPLEMENTAÇÃO

O trabalho foi implementado em Python 3, utilizando as bibliotecas padrão e as bibliotecas permitidas em `requirements.txt`.

Um grande desafio da abordagem foi o limite de tempo. Com o tempo limitado em, ao máximo, 5 minutos, o programa tinha pouco tempo de treino. Assim, era necessário que tudo que não fosse relacionado ao treino do recomendador fosse o mais rápido possível.

3.1 Velocidade

Uma medida importante foi diminuir ao máximo o acesso ao dataframe. Isso inclui carregá-lo apenas parcialmente (sem usar a coluna tempo) e especificar os tipos de dados de cada coluna. Além disso, para evitar acessos seguidos a este crio listas e dicionários, estruturas nativas do Python.

O tempo limitado também trouxe o problema da *learning rate*. Uma taxa muito baixa não permitia um treinamento mínimo, mas uma taxa mais alta poderia levar a mínimos locais.

3.2 Classes

Foram criadas duas classes. A classe `setup` tem as informações sobre a configuração de treino: número de fatores, épocas, *learning rate* e taxa de regularização. A classe `Dados` guarda informações importantes: dicionário de usuários, dicionário de itens, lista de

tuplas do tipo (usuário, item, *rating*), μ, as matrizes P e Q além dos vetores de viés b_u e b_i.

3.3 Inicializações

Algumas matrizes e vetores devem ser inicializados. As matrizes P e Q tinham duas opções de inicialização, apenas $\frac{1}{k}$ e valores aleatórios seguindo a distribuição normal no intervalo $[0.0, \frac{1.0}{k})$.

Os valores do bias, numa primeira abordagem eram calculados com base nos dados disponíveis para inicialização. Pela pouca quantidade de tempo disponível para convergir a abordagem não deu certo. A abordagem final inicializa b_u e b_i como 0.

3.4 Algoritmos

O algoritmo utilizado é uma modificação do SVD utilizando viés e regularização. A complexidade de computação do SVD é $O(m^2n + n^3)$. No caso do método utilizado, a complexidade vai depender em boa parte do fator k. Cada época roda por $O(mnk)$, se assumirmos que $m \geq n$, então $n \geq k$, logo no pior caso cada época roda por $O(mn^2)$.

4 RESULTADOS

Todos os resultados exibidos a seguir foram obtidos em uma máquina `Intel® Core™ i7-6500U CPU @ 2.50GHz × 4` com 7.7 GB de memória em uma distribuição `Linux Ubuntu 18.04.5 LTS`. Todos os testes executaram em menos de 5 minutos, medidos utilizando-se o comando `time python3 main.py ratings.csv targets.csv`.

Na tabela 1 "run" indica o número da tentativa, e indica o número de épocas, k indica o número de fatores, α o valor da taxa de aprendizado, λ o valor da taxa de regularização, "rand" indica como as matrizes P e Q foram inicializadas, "RMSE" indica o resultado obtido no Kaggle e "tempo" indica o tempo necessário para executar a instância. É importante salientar que, em instâncias onde o tempo foi inferior a 4 minutos e 20 segundos, o tempo não foi registrado. Nas instâncias com tempo mais próximo de 5 ele será exibido.

Nos resultados é interessante destacar os modelos 5 e 6. Eles tem formatações iguais, mas um RMSE de aproximadamente 0.002 de diferença. Isso se dá pela inicialização aleatória das matrizes P e Q. Espera-se que em um modelo bem treinado essa inicialização deixe de ter peso, mas, como o modelo é treinado de forma rápida, a inicialização tem peso e afeta os resultados.

O início seguindo a distribuição normal aparenta ter melhores resultados. Em comparações diretas ([2, 11],[4,5,7] e[3,8]), os valores utilizando normal foram quase sempre superiores (mesmo que por uma margem de erro pequena). O tempo limitado para treino trouxe outro problema: valores elevados de α e λ quase sempre levavam o problema a cair em um vale (ou mínimo local). O ideal seria treinar por mais tempo com taxas inferiores, de modo a evitar esse mínimo.

Até o final da escrita deste trabalho, o melhor resultado alcançando foi 1.59034 utilizando a abordagem com distribuição normal (3ª run). Se desconsiderarmos estas (`rand = False`), temos a 17ª run com, RMSE de 1.59674.

Table 1: Resultados Obtidos

run	e	k	α	λ	rand	RMSE	tempo
1	25	10	0.01	0.1	True	1.62539	
2	30	10	0.01	0.1	True	1.63592	
3	10	20	0.01	0.1	True	1.59034	
4	20	20	0.01	0.1	True	1.61128	
5	20	20	0.008	0.1	True	1.60420	
6	20	20	0.008	0.1	True	1.60209	
7	20	20	0.008	0.001	True	1.62088	
8	20	20	0.008	0.1	False	1.60149	
9	20	20	0.01	0.1	False	1.60659	4m55.962s
10	20	15	0.008	0.1	False	1.60184	
11	25	15	0.008	0.1	False	1.60577	4m48.950s
12	10	20	0.01	0.1	False	1.59750	
13	15	20	0.01	0.1	False	1.60145	
14	13	20	0.01	0.1	False	1.59948	
15	15	20	0.009	0.1	False	1.59947	
16	13	20	0.009	0.1	False	1.59796	
17	10	20	0.009	0.1	False	1.59674	
18	15	15	0.009	0.1	False	1.59986	
19	13	15	0.009	0.1	False	1.59843	
20	25	15	0.009	0.1	False	1.60843	4m32.445s
21	25	15	0.005	0.1	False	1.59776	4m38.949s

5 CONCLUSÃO

No final do trabalho era esperado um recomendador com bom RMSE. Essa tarefa foi concluída. Acredito que o valor atual de erro poderia ser melhorado com mais tempo e épocas.

REFERENCES

[1] Simon Funk. 2006. Netflix Update: Try This at Home. https://sifter.org/~simon/journal/20061211.html

[2] Stephen Gower. 2014. Netflix Prize and SVD. http://buzzard.ups.edu/courses/2014spring/420projects/math420-UPS-spring-2014-gower-netflix-SVD.pdf

[3] Rodrygo L. T. Santos. 2021. Matrix Factorization. https://www.youtube.com/watch?v=4YiJP__6f9Q

[4] Robin Witte. 2019. Recommender system: Using singular value decomposition as a matrix factorization approach. https://robinwitte.com/wp-content/uploads/2019/10/RecommenderSystem.pdf