Sistema de Recomendação de Itens com Abordagem Model-Based com Gradiente Descendente Estocástico*

LUIZ HENRIQUE DE MELO SANTOS†, Departamento de Ciência da Computação, UFMG, BR

Com o avanço dos serviços de streaming e o aumento das vendas de produtos e serviços pela internet, a exigência por sistemas que promovam uma recomendação personalizada de itens para seus usuários tem se tornado cada vez maior. Existem várias arquiteturas que possibilitam suas implementações, cada uma com suas especificidades quanto às formas de abordagem dos dados que são utilizados. Neste Trabalho foi elaborado um sistema recomendador de itens com abordagem model-based, que tem como algoritmo principal o Gradiente Descendente Estocástico.

Additional Key Words and Phrases: recommender systems, stochastic gradient descent, model based, items

1 INTRODUÇÃO

O objetivo deste Trabalho Prático é a criação de um Sistema de Recomendação capaz de predizer qual será a avaliação aproximada que um determinado usuário dará para um determinado item. Durante a implementação teve de ser considerado situações de cold-start, ou seja, predizer uma avaliação vinda de um usuário ou de algum item que ainda não está presente na base.

Durante o processo de desenvolvimento foram testados diversos algoritmos, sendo eles o Gradiente Descendente, o Gradiente Descendente Estocástico e a Fatoração Probabilística de Matrizes, sendo que foram implementados os dois últimos. A Fatoração Probabilística apresentou excelentes resultados, entretanto seu tempo de execução inviabilizou o seu desenvolvimento, bem como a complexidade para uma possível otimização. Já o Gradiente Estocástico permitiu a combinação entre eficiência e tempo de execução aceitáveis, sendo ele o optado para o projeto.

2 MODELAGEM

A estrutura de dados escolhida para ser utilizada neste projeto foi a de matriz esparsa, uma vez que facilita a manipulação de valores bem como a recuperação deles para serem utilizados para a definição em novos dados.

O tratamento de dados feito sobre o arquivo de treinamento recebido como entrada consistiu na separação das coluna '*UserId:ItemId*' em duas, com o posterior processo de encoding dos usuários e dos itens presentes no dataset. Afim de propiciar uma boa representação deles foi utilizado um *dict*, que possui como chaves o ID original de identificação de cada um deles. O encoding dos dados permite a criação de um *np.array*, que favorece e simplifica a realização de operações entre matrizes.

Uma estratégia adotada a fim de aumentar a eficiência e reduzir consideravelmente o tempo de treinamento do modelo foi a seleção de uma base de treinamento que possuísse apenas usuários que avaliaram um mínimo de 35 itens. Já para o conjunto de validação é escolhido aleatoriamente 15% destes usuários.

*A implementação completa da última versão deste algoritmo está no GitHub.
†Número de matrícula: 2017014464.

Author's address: Luiz Henrique de Melo Santos, luiz.melo@ufmg.dcc.br, Departamento de Ciência da Computação, UFMG, Belo Horizonte, Minas Gerais, BR.

Nos casos de cold-start foram feitos os seguintes tratamentos:

- caso o item não exista: é utilizado a média das avaliações feitas pelo usuário;
- caso o usuário não exista: é utilizado a média das avaliações do item;
- caso ambos não existam: é utilizado a média de avaliações global da matriz.

3 ALGORITMO

O algoritmo de Gradiente Descendente Estocástico é resultado de uma modificação no Gradiente Descendente original. Neste utilizado é calculado o gradiente utilizando apenas uma pequena parcela aleatória das observações, ao invés de todas elas. Em questões de desempenho, esta modificação pode otimizar o tempo de computação, bem como reduzir a ocorrência de biases durante o treinamento.

Este algoritmo tem como objetivo principal a otimização da função (1), a fim de encontrar os valores que permitam a sua minimização para obtenção do resultado objetivo desejado.

$$w^{new} := w^{old} - n\nabla Q_i(w) \tag{1}$$

Não foi utilizada nenhuma forma de otimização. O fine-tuning consistiu em buscar pelas melhores medidas nos parâmetros de regularização, de número de épocas (que tende a não contribuir muito após a 50º época), e da taxa de apredizado.

4 EXPERIMENTAÇÃO

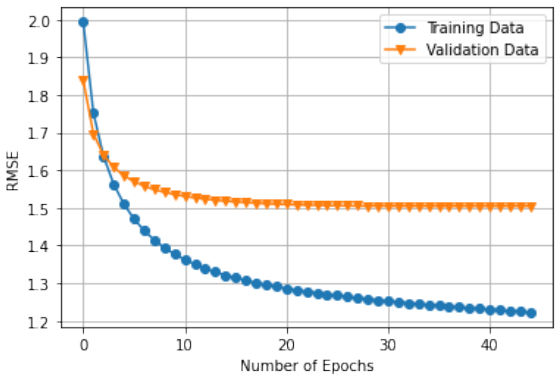
O modelo final utilizado para a geração da submissão final deste Trabalho foi treinado em 25 épocas em máquinas do CRC-DCC-UFMG, com as configurações: Intel Core i7 4ª geração, RAM 16Gb DDR3 166MHz, HD 1TB e sistema operacional Ubuntu 16.04.

Os tempos de execução obtidos com os treinamento oscilaram no intervalo de 4:25min-4:56min, entretanto sem nunca ultrapassar a marca de 5:00min - tempo limite estipulado pela especificação deste Trabalho Prático.

A métrica de avaliação utilizada para a determinação da eficiência do algoritmo foi o RMSE (Root Mean Square Error). O comportamento desta métrica com o decorrer das épocas de treinamento do modelo pode ser observado na Figura 1. Após a 40º época o modelo não apresentou melhoras significativas do desempenho do modelo sobre a base de validação, o que demonstra um possível alcance da convergência do algoritmo neste ponto, logo uma computação além deste ponto pode não ser mais tão vantajoso.

Um fator que dificultou bastante os testes do algoritmo foi a exigência pelo teste do algoritmo em máquinas do CRC-DCC-UFMG, uma vez que o comando '*sudo*', que seria necessário para a instalação do pacote de máquinas virtuais no Python3, não é permitido. Desta forma o código foi testado por meio da submissão às configurações mais recentes das bibliotecas '*Pandas*' e '*Numpy*'.

Fig. 1. RMSE metrics between epochs
(generated in JupyterLab)



5 CONCLUSÃO

Por meio deste Trabalho Prático foi possível a abstração, modelagem e desenvolvimento de um sistema de recomendação de itens. Devido às várias tentativas de implementação deste Trabalho, bem como a inclusão de algum método de otimização, bastante tempo foi gasto nele.

A etapa que conteve a maior parcela do tempo de desenvolvimento foi a criação de uma representação para a matriz esparsa, bem como o encoding dos dados, uma vez que a sua montagem e posterior padronização de uso apresenta uma grande facilidade na ocorrência de erros, além de um trabalhoso degub deles.

Por fim, a melhor submissão feita no Kaggle (com RMSE 1.77) não condiz com o RMSE obtido neste algoritmo final (com RMSE 1.78), uma vez que foram necessárias algumas modificações no código que afetaram a precisão do modelo. Entretanto, como o Kaggle considera a melhor métrica e não a última submissão não foi possível realizar esta modificação.

6 TRABALHO FUTURO

Uma possível atualização deste código pode ser feita por meio da adição de algum algoritmo de otimização, tal como Momentum, Adagrad, RMSprop, Adam e até mesmo o AMSGrad. O que poderia reduzir o tempo de execução e minimizar o erro do algoritmo.