3
Sistemas de Recomendação e Filtragem Colaborativa

3.1.
Recomendação Colaborativa

Nos Sistemas de Recomendação Colaborativos, ou seja, baseados em Filtragem Colaborativa, os usuários indicam, através de avaliações, o quanto eles gostam de determinados itens. Analisando as avaliações conhecidas, os sistemas prevêem qual será a avaliação de um usuário para um item ainda não avaliado.

O interesse pelo assunto é elevado não apenas por ele constituir uma área de pesquisa rica em desafios, mas também devido à abundância de aplicações práticas responsáveis por auxiliar os usuários a enfrentar a sobrecarga de informação, fornecendo recomendações personalizadas de conteúdo e serviços. Exemplos incluem a recomendação de livros, CDs e outros produtos na Amazon.com (Linden et al., 2003), de filmes na MovieLens (Miller et al., 2003), e notícias na VERSIFI Technologies (anteriormente AdaptiveInfo.com) (Billsus et al., 2002). Além disto, alguns fornecedores de *software* têm incorporado funcionalidades de recomendação em seus servidores de comércio (Peddy & Armentrout, 2003).

Na recomendação de anúncios, uma tarefa muito importante é predizer o CTR de um par consulta-anúncio. CTR (*Click-Through Rate*) é uma métrica de relevância e equivale à razão entre os cliques e as impressões de um anúncio associado a uma consulta. Sua fórmula é apresentada na eq. 7.

$$CTR = \frac{num\_cliques}{num\_impressões} \tag{7}$$

Desta forma, é possível traçar um paralelo no qual uma consulta representa um usuário, já que seus interesses ou necessidades de informação estão expressos por ela; um anúncio representa um item, pois, de fato, oferece um produto ou serviço; e, finalmente, o CTR representa uma avaliação de um usuário para um item, já que indica o quanto o anúncio é relevante para a consulta. Portanto, podemos especializar os Sistemas de Recomendação para resolver o problema de predição dos CTRs desconhecidos com base em CTRs conhecidos.

A recomendação de filmes funciona de forma análoga. Os usuários representam os clientes, os filmes são os itens e a avaliação do cliente para um dado filme, representa a nota. Assim como na recomendação de anúncios, a idéia é correlacionar os filmes de forma a extrair atributos “escondidos”, como o gênero, por exemplo.

Figura 15: Grupo de filmes correlatos.

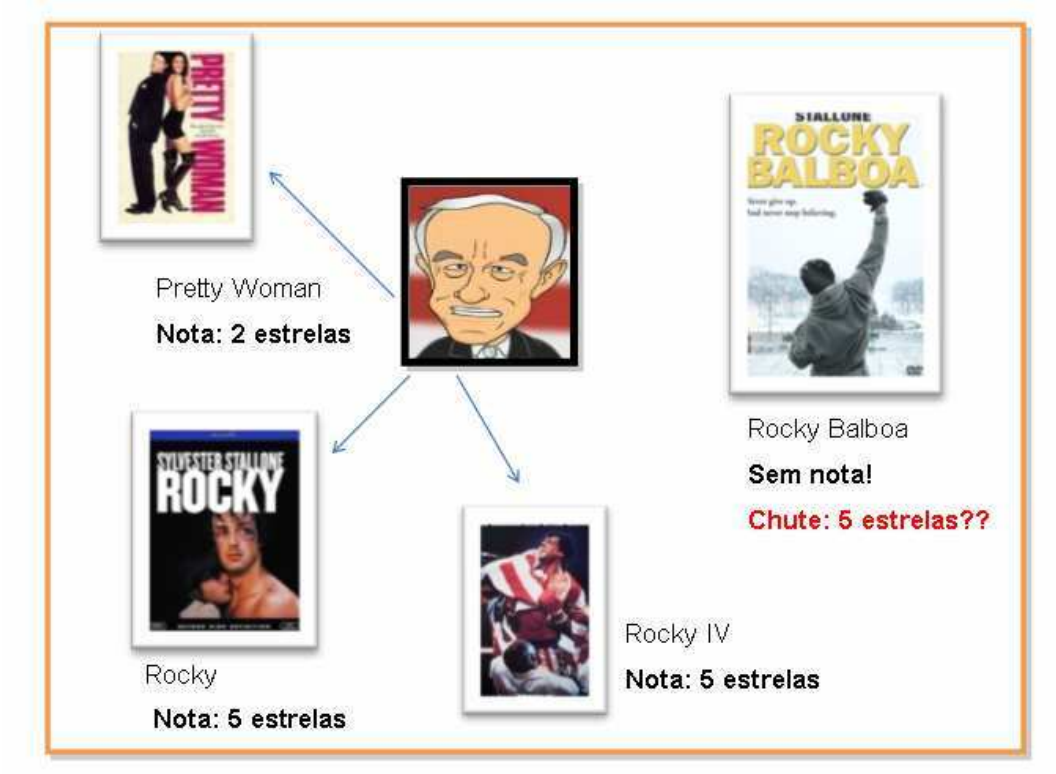

Figura 16: Exemplificação da recomendação de filmes.

3.2. Fatoração de Matrizes

O modelo colaborativo abordado por este trabalho é o de Fatoração de Matrizes (FM) (Takacs et al., 2007). O FM o problema de predição transformando e decompondo matrizes.

Existem dois parâmetros recebidos pelo algoritmo, detalhados por Cavalcante & Milidiú (2008): as épocas de treinamento e os atributos latentes a serem aprendidos. No nosso trabalho, o algoritmo foi alterado para também receber uma distribuição de pesos e um fator de suavização. Uma das restrições desta versão “ponderada” do FM é garantir que seu resultado será exatamente como o da versão original, ou seja,com o peso de todos os exemplos igual a 1 e o fator de suavização igual a 0. Depois de treinado, o FM retorna um valor contínuo como resultado da sua predição.

Não entraremos em detalhes da implementação do FM, pois nossa intenção é melhorar sua performance sem conhecer a estrutura interna. Resumindo: vamos tratá-lo como um “oráculo”, uma “caixa preta”.

3.3. Adaptação do AdaBoost

Um dos inconvenientes de utilizar o AdaBoost aplicado ao FM é o fato do primeiro lidar com problemas discretos e o segundo com problemas contínuos. Problemas discretos são também conhecidos como problemas de classificação e problemas contínuos como problemas de regressão.

Na regressão, ao invés de calcularmos um erro, utilizamos uma função de perda, a qual determina um valor associado ao erro de previsão para cada exemplo de treinamento. O tipo de fórmula (linear, quadrática ou exponencial) utilizada para definira função de perda pode varia de acordo com a natureza do problema em questão.

Para resolver problemas de regressão, Freund & Schapire (1997) propuseram o AdaBoost.R, que os reduz a problemas de classificação. Segundo Solomatine & Shrestha (2004), embora experimentos mostrem a eficácia do AdaBoost.R, existe uma desvantagem em utilizá-lo: o algoritmo expande cada exemplo de regressão em muitos exemplos de classificação.

Para não necessitar reduzir o problema de regressão a um problema de classificação, Drucker (1997) desenvolveu o AdaBoost.R2, o qual é uma modificação ad hoc do AdaBoost.R. Seus experimentos indicam resultados promissores. A Figura 17 apresenta o AdaBoost.R2.

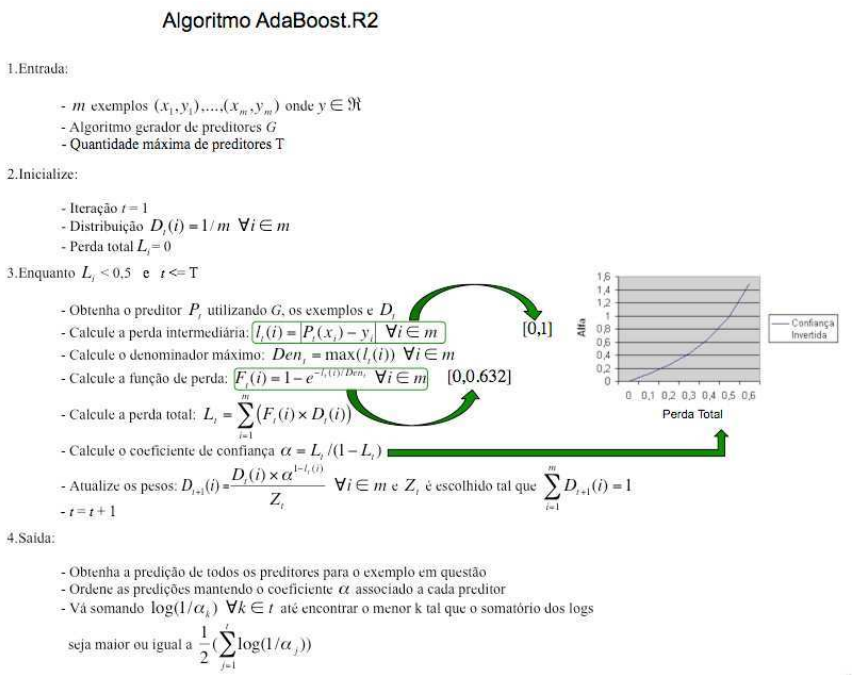

Figura 17: Algoritmo AdaBoost original para problemas regressivos.

Após a realização de alguns testes na nossa instância de recomendação de anúncios, verificamos que o AdaBoost.R2 não melhora o FM. Uma razão para isto pode ser o fato da necessidade de utilizarmos uma distribuição que não é de probabilidade, pois o FM funciona mal quando os pesos são números muito pequenos. Nosso fator de normalização é calculado de forma que, na medida que o algoritmo progride, o somatório dos pesos seja sempre igual ao número de exemplos. Inicialmente, todo exemplo começa com peso 1. Note que a tendência do AdaBoost.R2 é diminuir os pesos dos exemplos com menor perda e manter os pesos dos exemplos com maior perda.

Mediante aos resultados não positivos do AdaBoost.R2, alteramos o modo como é calculada a perda de cada exemplo, suavizando seu valor pelo inverso do do peso elevado a k (Figura 18). Denominamos tal algoritmo de **AdaBoost.RS** (Regressão Suavizada).

A idéia é a seguinte: quanto mais difícil o exemplo é para ser aprendido, menos importância é dada para a sua perda. Este conceito é análogo ao desvio padrão aplicado em alguns exames de vestibulares. Se uma questão é bastante difícil e muitos erram, então o valor do seu erro é reduzido. Em contrapartida, se a questão é fácil e muitos acertam, então o valor do seu erro é ampliado.

Note que tal heurística é controversa à filosofia do *Boosting*. Contudo, apesar de estarmos suavizando essa perda de forma contrária ao *Boosting*, nós reponderamos, a posteriori, a função de perda pelo peso do exemplo. É como se relaxássemos um pouco antes de tratarmos os exemplos "errados" de forma mais severa. De todas as funções de perda experimentadas, a exponencial foi a que apresentou os melhores resultados.

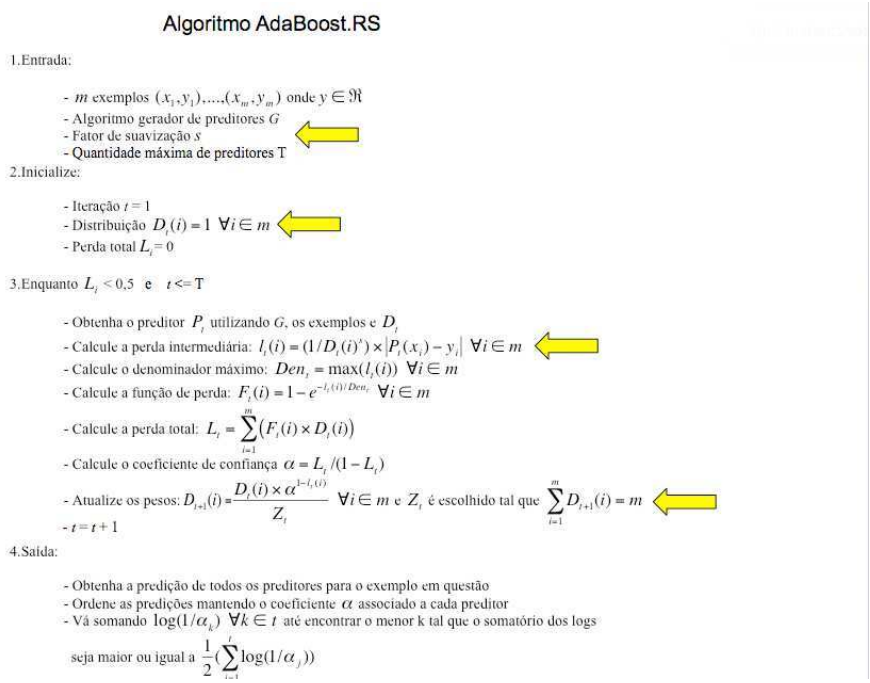

Figura 18: Algoritmo AdaBoost adaptado para a recomendação.