Sistemas de Recomendação

Aplicando Sistemas de Recomendação em Situações Práticas

Renata Ghisloti Duarte de Souza (rgduarte@br.ibm.com)
Software Engineer
Linux Technology Center - IBM

28/Mai/2014

A área de Sistemas de Recomendação vem ganhando bastante importância dentro de Inteligencia Artificial, motivada pelo E-commerce. Diversos algoritmos foram propostos nos ultimos anos, inclusive o de filtragem-colaborativa SlopeOne. Neste artigo visa-se apresentar o tema Sistemas de Recomendação com uma abordagem prática, visando aplicaçoes web.

Introdução

Sistemas de Recomendação podem ser vistos ao realizar buscas em sites de pesquisa da internet, em compras online, ou até mesmo ao visualizamos nossos emails. São o mecanismo por trás da propaganda personalizada que recebemos na web, com indicações de sites para visitarmos ou produtos para compramos.
Com o advento do consumo em dispositivos móveis e a propagação o e-commerce, sistemas de recomendação tornaram-se um tema extremamente atrativo. Através de algoritmos simples e facilmente integráveis a aplicações web, eles agregam valor ao negócio online, promovendo itens de consumo direcionados a um público alvo.

Por trás da singela propaganda, estes sistemas utilizam abstrações matemáticas de dados. Neste artigo, veremos que eles consistem basicamente em algoritmos de filtragem e inferência de dados, que recomendam produtos de acordo com os interesses dos usuários.

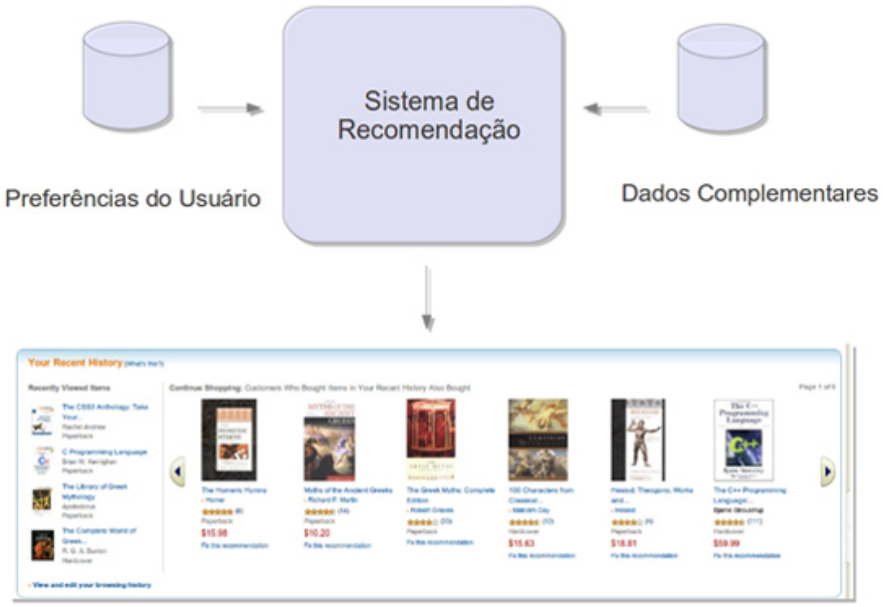
Em Quais Aplicações Podem ser Utilizados?

É importante avaliar para quais aplicações esses sistemas são viáveis.
Primeiramente, estas devem obrigatoriamente basear-se em itens sendo expostos ou oferecidos a usuários. Em outro caso, o algoritmo perde seu sentido.

Outro ponto importante é que esse mecanismo só é aplicável quando há grande volume de dados envolvidos. Isso é necessário para garantir que a metodologia seja eficiente, já que, são feitas abstrações matemáticas e quanto mais dados, mais apurada a função de abstração, e portanto, mais correto o resultado.

Apresentando Sistemas de Recomendação

Os sistemas de recomendação são uma sub-area de aprendizagem de máquina (machine learning) e tem por objetivo sugerir itens a um usuário, com base em seu histórico de preferências. Podem ser recomendados itens diversos como livros, investimentos ou viagens. É amplamente utilizado como uma estratégia de marketing, já que ao recomendar produtos que estejam alinhados ao interesse do usuário, é mais provável que ele venha adquirir tal produto.
É possível fazer recomendações comparando as preferências de um usuário com um grupo de outros usuários. Também é possível fazer recomendações procurando itens com características similares aos que o usuário já demonstrou interesse no passado. As preferências do usuário podem ser colhidas implicitamente ou explicitamente. Na forma implícita, informações são obtidas através de opções de compras passadas, histórico de sites visitados, links clicados, cookies do browser ou até mesmo localidade geográfica. Há também a forma explícita de averiguar preferências, utilizando feedbacks efetivos, como por exemplo notas dadas a um determinado item.



Visualização ampliada

Tipos de Sistemas de Recomendação

Podem ser classificados em basicamente três categorias, a partir de como a recomendação é feita:

Baseado em Conteúdo
Um sistema de recomendação baseado em **conteúdo** recomenda ao usuário produtos que sejam semelhantes ao que ele preferiu no passado. A recomendação é feita a partir de tags "descritoras" de itens. Itens com características próximas destas tags são recomendados.Em um cenário de recomendação de filmes, por

exemplo, um usuário que, assiste e gosta do filme "Matrix" teria recomendações do gênero ação e ficção científica.

Vantagens deste tipo de sistema é que são simples para dados textuais e não necessitam de muitas informações sobre um usuário para sugerir itens. Todavia, além de serem difíceis de aplicar em contextos multimídia, podem oferecer recomendações repetitivas, recomendando sempre assuntos que o usuário já conhece.

Filtragem Colaborativa
Filtragem **colaborativa** consiste na recomendação de itens que pessoas com gosto semelhante preferiram no passado. Analisa-se a vizinhança do usuário a partir da regra: "Se um usuário gostou de A e de B, um outro usuário que gostou de A também pode gostar de B". Esse tipo de recomendação apresenta resultados positivos na prática [LINDEN, Greg. SMITH, Brent. YORK Jeremy. Amazon.com Recommendations Item-to-Item Collaborative Filtering], e evita o problema de recomendações repetitivas. Uma desvantagem é que requer grande número de informações sobre o usuário e sua vizinhança para funcionar precisamente.

Sistemas Híbridos
Por fim, um **sistema híbrido** consiste em combinar as duas abordagens mencionadas, tentando fortificá-las e superar suas desvantagens.
Dentro da categoria Filtragem Colaborativa, pode-se ainda dividir os sistemas em mais duas categorias: Item-Based e User-Based. Destaca-se nesse meio o algoritmo SlopeOne, do tipo Item-Based, uma abordagem simples e eficiente

Sistema de Recomendação SlopeOne

O Slope One é um método de Recomendação fácil de implementar, com teoria simples e que apresenta bons resultados práticos sendo altamente escalável [LEMIRE, Daniel. MACLACHLAN, Anna. Slope One Predictors for Online Rating-Based Collaborative Filtering, In SIAM Data Mining (SDM'05), Newport Beach, California, April 21-23, 2005].
Apresentado em um artigo de Daniel Lemire em 2005, suas predições são calculadas a partir da comparação entre avaliações de usuários a certos itens.
O algoritmo opera supondo que um usuário tenha dado notas não binárias a itens. Essas notas são colocadas em uma matriz de UsuáriosxItens, de tal forma que cada linha corresponda às notas de um usuário j a N itens. Se um usuário j não tiver dado notas a um item i, o elemento $x_{i,j}$ fica igual a 0.
A figura representa a Matriz com um conjunto de notas.

Items

Users					
	3	0	3	2.5	4
	1.5	0	4	0	5
	0	1	1.5	1	0
	4	3	0	1.5	4.5
	2	2.5	0	2	4
	5	0	4.5	0	0
	1	2	1	0	3.5
	0	5	0	1	4

Observando a matriz de notas vemos que uma linha j da matriz representa as notas dadas por um usuário j a todos os itens no espaço definido. Uma coluna i Representa as notas recebidas pelo item i pelos diferentes usuários existentes.

Item i

User j					
	3	0	3	2.5	4
	1.5	0	4	0	5
	0	1	1.5	1	0
	4	3	0	1.5	4.5
	2	2.5	0	2	4
	5	0	4.5	0	0
	1	2	1	0	3.5
	0	5	0	1	4

A partir dessa matriz, podemos obter relações entre os dados. É possível gerar uma interpolação matemática e predizer qual seria a nota dada por um usuário j ao item i que ele ainda não avaliou.

A maioria dos métodos de Filtragem Colaborativa também utiliza a matriz de notas para calcular predições. Comumente, são calculas as similaridades entre as linhas ou colunas usando funções como Pearson ou Cosine Similarity. Dizemos que o método é User-Based quando são comparadas as linhas da matriz e Item-Based para colunas.

Diferentemente de outras abordagens colaborativas, o Slope One cria uma relação linear entre os dados. Dai vem o nome: slope é o multiplicador de x na fórmula f(x) = ax + b, e o slope para esse algoritmo equivale a 1.

Supondo que temos, um usuário A que deu nota 2 para um item i, e nota 4 a um item j e supondo ainda que temos um usuário B que deu nota 3 para o item i, através do SlopeOne, calcularía-se a predição da nota que o usuário B daria ao item j da seguinte forma:

(Nota de A a item i - Nota de A a item j) = (Nota de B a item i – Predição de B em j)

Logo:

$$(2 - 4) = (3 - ?)$$
$$? = 2 + 3$$
$$? = 5$$

De acordo com a predição do algoritmo, o usuário B daria uma nota 5 ao item j.

Para análise de mais dados, obtería-se a média das diferenças entra as notas dos usuários. A fórmula geral para cálculo das predições segue descrita abaixo:

$$P(A,i) = \frac{(R(A,j) + Diff(i,j)) + (R(A,k) + Diff(i,k)) + \ldots + (R(A,z) + Diff(i,z))}{N}$$

Onde Diff (i,j) é a média das diferenças de avaliações entre itens i e j para os outros usuários, R (A, j) é quanto o usuário A deu de nota ao item j, e supondo que tenhamos N itens e que os itens variem de i a z .

Na Prática: Implementando SlopeOne

Uma característica interessante deste método é a redução da computação online do algoritmo, já que a maior parte de seu processamento pode ser feita offline (fase de pré-processamento). Como apresentado abaixo, é criada uma matriz de diferença entre as avaliações dadas por cada usuário. Em seguida, utiliza-se essa matriz pré-computada e calculam-se as predições.

Segue o pseudo-código do algoritmo:

```
# Fase de preprocessamento na qual é calculada a diferença
# entre todos os item-item valores.
1. para cara item i
2.  para cada item j
3.   para cada usuário u que expressa preferência por i e j
4.   adicione a diferença entre as preferências i e j `a matriz de diferenças
(formando uma média das diferenças entre i e j)

# Cálculo da predição
1. para cada item i pelo qual o usuário u não expressa preferência
2.  para cada item j pelo qual o usuário u expressa preferência
3.   ache diferença média entre j e i
4.adicione essa diferença à preferência de u para j
6. retorne os top itens, ranqueados pelo valor dessas preferências médias
```

Segue uma implementação do algoritmo em JAVA, que recomenda filmes a usuários. Foram utilizados dados do MovieLens (http://www.grouplens.org/node/12), constituído de 1 milhão de notas que 6000 usuários deram para 4000 filmes . A implementação completa encontra-se em anexo.

Como sugerido pelo artigo de Lemire, o código foi divido em dois arquivos principais SlopeOne.java e Predict.java. O primeiro contém o pré-processamento e cálculo da matriz de diferenças. O outro arquivo faz a predição a partir da matriz calculada.

O seguinte trecho de código computa a matriz de diferenças entres avaliações dos itens:

```
1.public  void buildDiffMatrix() {
2.
3.    mRatings = new float[maxItemsId+1][maxItemsId+1];
4.    mFreq = new int[maxItemsId+1][maxItemsId+1];
5.
6.    /* Itera por todos os usuários, e então, por todos os itens para
  calcular as diferenças */
7.    for(int cUser : usersMatrix.keySet()){
8.        for(int i: usersMatrix.get(cUser).keySet()){
9.            for(int j : usersMatrix.get(cUser).keySet() ){
10.                mRatings[i][j] = mRatings[i][j]  +
11.                 (usersMatrix.get(cUser).get(i).floatValue() -
                   (usersMatrix.get(cUser).get(j).floatValue()));
12.                mFreq[i][j] = mFreq[i][j] + 1;
13.            }
14.        }
15.    }
16.
17.    /*  Calcula as Médias (diff/freqs) */
18.    for(int i = 1; i<= maxItemsId; i++){
19.        for(int j = i; j <= maxItemsId; j++){
20.            if(mFreq[i][j] > 0){
21.                mRatings[i][j] = mRatings[i][j] / mFreq[i][j];
22.            }
23.        }
24.    }
25.}
```

Nesta função, o cálculo real inicia-se a partir da linha 7. No laço da linha 7, itera-se por todos os usuários (presentes em userMatrix) e então por todos os itens, e em mRatings[i][j] é colocada a soma de notas dadas por todos os usuários que forneceram notas aos itens i e j. Em mFreq[i][j] é colocada a quantidade de usuários que deram notas a ambos os itens.

No laço da linha 18, a matriz mRatings é percorrida dividindo os valores de mRatings pelos de mFreq, obtendo assim, uma média das diferenças entre avaliações de itens.
Ao término desse laço, obtemos a matriz de diferenças completa.

A etapa de predição é descrita no trecho de código a seguir, no qual calculam-se as predições de notas a partir da matriz de diferenças.

```
1.public Predict(){
2.
3.   getUser(targetUser);
4.   float totalFreq[] =  new float [maxItem+1];
5.
6.   for (int j=1; j <= maxItem; j++) {
7.      predictions.put(j,0.0f);
8.   }
9.
10.   for (int j : user.keySet()) {
11.      for (int k = 1; k <= maxItem; k++) {
12.         if( j != k) {
13.            /* Só para itens que o usuário ainda não avaliou */
14.            if(!user.containsKey(k)){
15.               float newVal = 0;
16.               if(k < j) {
17.                  newVal = mFreq[j][k] * (mDiff[j][k] +
  user.get(j).floatValue());
18.               }
19.               else {
20.                  newVal = mFreq[j][k] * (-1 * mDiff[j][k] +
  user.get(j).floatValue());
21.               }
22.               totalFreq[k] = totalFreq[k] + mFreq[j][k];
23.               predictions.put(k, predictions.get(k).floatValue() +
  newVal);
24.            }
25.         }
26.      }
27.   }
28.  /* Calcula a média */
29.   for (int j : predictions.keySet()) {
30.      predictions.put(j, predictions.get(j).floatValue()/(totalFreq[j] ));
31.   }
32.
33.   /* Preenche o vetor de predições com as avaliações já conhecidas */
34.   for (int j : user.keySet()) {
35.      predictions.put(j, user.get(j));
36.   }
37.
38.   /* Imprime predições */
39.   System.out.println("\n" + "#### Predictions #### ");
40.   for (int j : predictions.keySet()) {
41.       System.out.println( j + " " + predictions.get(j).floatValue());
42.   }
43.}
```

O laço da linha 10 itera pelas preferências de um usuário (usuário alvo da predição, obtido por getUser) e para cada item k que o usuário ainda não avaliou e para cada item j que o usuário já avaliou, é calculada a nota que o usuário daria para k em relação a j (newVal = mFreq[j][k] * (mDiff[j][k] + user.get(j).floatValue())). Em seguida, na linha 23 é armazenada a soma de todas as notas computadas entre dois itens (predictions.put(k, predictions.get(k).floatValue() + newVal);).

Posteriormente, no laço da linha 29, são calculadas as predições médias finais, dividindo a soma de todas as predições para um item j pela quantidade de differenças usadas na soma.

Em termos de eficiência, a parte com maior custo de processamento é o calculo da matriz de diferenças, que pode ser realizado offline. O tempo de execução do algoritmo pode chegar a mn^2 passos considerando m usuários e n itens. O espaço de armazenamento requerido pode chegar a $n(n-1)/2$.

Segundo o artigo de Lemire o Erro Médio Absoluto (Mean Absolute Error) do algoritmo segue a margem de outras abordagens colaborativas. Seu MAE é 0.188, o da abordagem de Pearson 0.190 e o método que usa a Cosine Similarity tem MAE de é 0.198 . Ainda segundo Lemire, o SlopeOne apresenta acurácia comparável a métodos mais complexos de predição.

Conclusão

Sistemas de Recomendação são um tema atrativo no contexto atual. O método de Filtragem Colaborativa se destaca por seus resultados, e nesse meio, ressalta-se o algoritmo SlopeOne. Este método apresenta acurácia equiparável a abordagens complexas mesmo sendo de fácil implementação.

Referências

LEMIRE, Daniel. MACLACHLAN, Anna. Slope One Predictors for Online Rating-Based Collaborative Filtering, In SIAM Data Mining (SDM'05), Newport Beach, California, April 21-23, 2005.

ADOMAVICIUS, G. TUZHILIN, A. Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. 2005.

CANDILLIER, Laurent. MEYER, Frank. JACK, Kris. FESSANT, Françoise. State-of-the-Art Recommender Systems.

LINDEN, Greg. SMITH, Brent. YORK Jeremy. Amazon.com Recommendations Item-to-Item Collaborative Filtering.

_____. movie lens - movie recommendations. Disponível em: http://www.movielens.org

Download

Nome	Tamanho	Método de download
SlopeOne.zip	5.63MB	HTTP

Sobre o autor

Renata Ghisloti Duarte de Souza

Renata Ghisloti é formada em Ciência da Computação pela UNICAMP. Trabalha na IBM como Software Engineer desde o inicio de 2011, onde atualmente trabalha como Team Leader do Bugzilla no Linux Technology Center.