

Sistemas de Recomendação

Prof. Danilo Silva

EEL7514/EEL7513 - Tópico Avançado em Processamento de Sinais

EEL410250 - Aprendizado de Máquina

EEL / CTC / UFSC

Introdução

Sistemas de Recomendação

- ▶ Objetivo: a partir de dados sobre um usuário (avaliações, compras passadas, etc), recomendar itens que este usuário tem mais chance de se interessar
 - ▶ Também conhecido como “filtragem” (de itens para um dado usuário)
- ▶ Exemplos: Netflix, Spotify, Amazon, e-commerce em geral
- ▶ Tipos de abordagem:
 - ▶ Filtragem baseada em conteúdo (*content-based filtering*): requer uma descrição do conteúdo dos itens (atributos)
 - ▶ Filtragem colaborativa (*collaborative filtering*): baseia-se exclusivamente nas avaliações de outros usuários
 - ▶ Híbrida

Filtragem Baseada em Conteúdo

Filtragem Baseada em Conteúdo

- ▶ Suponha n itens e f atributos
- ▶ Cada item i é descrito por um vetor de atributos $\mathbf{q}_i \in \mathbb{R}^f$
- ▶ Matriz de atributos dos itens:

$$\mathbf{Q} = \begin{bmatrix} \text{---} \mathbf{q}_1^T \text{---} \\ \vdots \\ \text{---} \mathbf{q}_n^T \text{---} \end{bmatrix} \in \mathbb{R}^{n \times f}$$

- ▶ Seja r_{ui} a avaliação (*rating*) dada ao item i pelo usuário u
- ▶ Seja $\mathcal{I}_u \subseteq \{1, \dots, n\}$ o conjunto de itens avaliados pelo usuário u
- ▶ **Problema:** para um usuário u , dadas \mathbf{Q} e as avaliações conhecidas $r_{ui}, i \in \mathcal{I}(u)$, prever as avaliações desconhecidas $r_{ui}, i \notin \mathcal{I}(u)$

Exemplo

Filme	q_1 (romance)	q_2 (ação)	Alice
Star Wars	0.1	0.93	0
Matrix	0.05	0.99	0
X-Men	0	0.95	?
Titanic	0.99	0.05	4
Casablanca	0.92	0	?

Exemplo

Filme	q_1 (romance)	q_2 (ação)	Alice	Bruno	Carol	Davi
Star Wars	0.1	0.93	0	5	5	0
Matrix	0.05	0.99	0	5	?	?
X-Men	0	0.95	?	?	4	0
Titanic	0.99	0.05	4	0	0	5
Casablanca	0.92	0	?	0	0	5

Exemplo: Regressão Linear

- ▶ A avaliação do item i pelo usuário u é modelada como

$$\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i$$

(possivelmente incluindo um atributo constante $q_{i0} = 1$)

- ▶ Conjunto de treinamento (para o usuário u): $\mathcal{D}_u = \{(\mathbf{q}_i, r_{ui}), i \in \mathcal{I}_u\}$
- ▶ Função custo:

$$J = \sum_{i \in \mathcal{I}_u} (\mathbf{p}_u^T \mathbf{q}_i - r_{ui})^2$$

- ▶ Solução:

$$\mathbf{p}_u = (\mathbf{Q}_u^T \mathbf{Q}_u)^{-1} \mathbf{Q}_u^T \mathbf{r}_u, \quad \text{onde} \quad \begin{cases} \mathbf{Q}_u = \mathbf{Q}[\mathcal{I}_u, :] \\ \mathbf{r}_u = (r_{ui}, i \in \mathcal{I}_u)^T \end{cases}$$

Limitações / Extensões

Extensões:

- ▶ Todo ou parte do vetor \mathbf{p}_u pode ser obtida diretamente a partir de dados de perfil ou de comportamento do usuário
 - ▶ Especialmente útil quando o usuário não efetuou nenhuma avaliação
- ▶ Outros modelos mais sofisticados (não-lineares)

Limitações:

- ▶ Determinação de \mathbf{Q}
 - ▶ Exige conhecimento específico do domínio para encontrar bons atributos
 - ▶ Os atributos que seriam mais preditivos podem não estar presentes nos dados disponíveis

Filtragem Colaborativa

Filtragem Colaborativa

- ▶ Suponha m usuários e n itens
- ▶ Seja r_{ui} a avaliação (*rating*) dada ao item i pelo usuário u
- ▶ Seja $\mathcal{R} = \{(u, i) : \text{usuário } u \text{ avaliou o item } i\}$ o conjunto que indica as avaliações conhecidas
- ▶ **Problema:** dadas as avaliações conhecidas $r_{ui}, (u, i) \in \mathcal{R}$, prever as avaliações desconhecidas $r_{ui}, (u, i) \notin \mathcal{R}$

Exemplo

Filme	Alice	Bruno	Carol	Davi
Star Wars	0	5	5	0
Matrix	0	5	?	?
X-Men	?	?	4	0
Titanic	4	0	0	5
Casablanca	?	0	0	5

Filtragem Colaborativa: Tipos de abordagem

- ▶ **Baseada em memória:** guarda toda a matriz de avaliações e realiza previsões baseadas em similaridade de avaliações
 - ▶ Considera como vizinhos usuários que possuem preferências semelhantes
 - ▶ Recomenda a um usuário itens bem avaliados pelos seus vizinhos
- ▶ **Baseada em modelo:** ajusta aos dados um modelo que possui variáveis ocultas (fatores latentes)
 - ▶ Assume que itens e usuários podem ser representados por vetores em um espaço de dimensão f pequena
 - ▶ Recomenda a um usuário os itens mais próximos a ele neste espaço latente

Modelo de Fatoração Matricial

- ▶ Seja f o número de fatores latentes
- ▶ A avaliação do item i pelo usuário u é modelada como

$$\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i, \quad \text{onde } \mathbf{p}_u, \mathbf{q}_i \in \mathbb{R}^f$$

- ▶ Corresponde a encontrar uma fatoração **aproximada** de posto f para a **matriz com entradas faltantes**

$$\mathbf{R} = [r_{ui}] \approx \mathbf{P}\mathbf{Q}^T$$

- ▶ Conjunto de treinamento: $\mathcal{D} = \{r_{ui}, (u, i) \in \mathcal{R}\}$
- ▶ Função custo:

$$J = \sum_{(u,i) \in \mathcal{R}} (\mathbf{p}_u^T \mathbf{q}_i - r_{ui})^2$$

- ▶ **Não é convexa** pois ambos \mathbf{p}_u e \mathbf{q}_i são variáveis de otimização

Mínimos Quadrados Alternados

Alternating Least Squares

- ▶ Se \mathbf{Q} está fixa, sabemos a solução ótima para \mathbf{P}
- ▶ Se \mathbf{P} está fixa, sabemos a solução ótima para \mathbf{Q} (por simetria)
- ▶ Algoritmo:
 - ▶ Inicialize \mathbf{P} e \mathbf{Q} aleatoriamente
 - ▶ Otimize \mathbf{P} (com \mathbf{Q} fixo) e depois \mathbf{Q} (com \mathbf{P} fixo)
 - ▶ Repita o item anterior até a convergência
- ▶ A cada iteração o custo nunca pode aumentar, portanto deve convergir (para um ótimo local)

Método do Gradiente Estocástico

- ▶ Otimizar a função custo **conjuntamente** em todas as variáveis

$$J = \sum_{(u,i) \in \mathcal{R}} (\mathbf{p}_u^T \mathbf{q}_i - r_{ui})^2$$

Aprimorando o Método

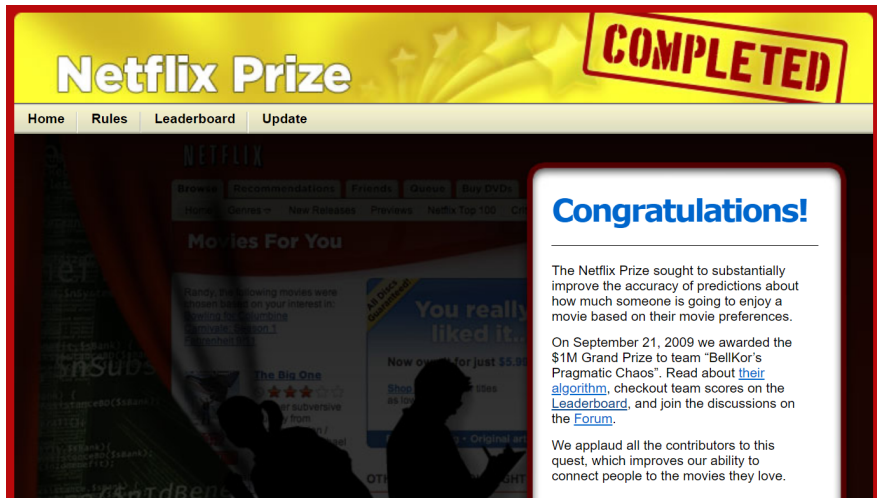
- ▶ Adicionar *biases* explicitamente:

$$\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{p}_u^T \mathbf{q}_i$$

- ▶ Adicionar regularização L2 em todos os pesos (exceto μ):

$$J = \sum_{(u,i) \in \mathcal{R}} (\mathbf{p}_u^T \mathbf{q}_i - r_{ui})^2 + \lambda(\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2 + b_u^2 + b_i^2)$$

Exemplo: Competição Netflix (2006-2009)



The image shows a screenshot of the Netflix Prize website. At the top, there's a yellow banner with the text "Netflix Prize" and a large red stamp that says "COMPLETED". Below the banner, there's a navigation bar with links: Home, Rules, Leaderboard, and Update. The main content area shows a "Movies For You" section with a recommendation for "The Big One" and a "Congratulations!" message. The message states: "The Netflix Prize sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences. On September 21, 2009 we awarded the \$1M Grand Prize to team 'BellKor's Pragmatic Chaos'. Read about [their algorithm](#), checkout team scores on the [Leaderboard](#), and join the discussions on the [Forum](#). We applaud all the contributors to this quest, which improves our ability to connect people to the movies they love."

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
------	-----------	-----------------	---------------	------------------

Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
---	--	--	--	--

1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
---	---	--------	-------	---------------------

Aplicação: Encontrando itens semelhantes

- ▶ O modelo efetivamente aprende uma representação em \mathbb{R}^f para os itens (e para os usuários)
 - ▶ Extração de atributos automática
- ▶ De posse desta representação, podemos resolver outros problemas como:
 - ▶ Encontrar itens semelhantes a um dado item:

$$\min_{j \neq i} \|\mathbf{q}_j - \mathbf{q}_i\|$$

- ▶ Encontrar grupos de itens semelhantes (clustering)

Implementação usando Redes Neurais

- ▶ Considere o conjunto de dados $\mathcal{D} = \{((u, i), r_{ui}), (u, i) \in \mathcal{R}\}$, onde u e i são variáveis categóricas:
 - ▶ $u \in \{1, \dots, m\}$ indica o usuário
 - ▶ $i \in \{1, \dots, n\}$ indica o item
- ▶ Em outras palavras, temos amostras das variáveis (\mathbf{x}, y) , onde
 - ▶ $\mathbf{x} = (u, i)$ é o vetor de atributos
 - ▶ $y = r_{ui}$ é o valor-alvo
- ▶ Desejamos construir um modelo de rede neural que realiza a predição $\hat{y} = r_{ui}$ para $(u, i) \notin \mathcal{D}$

Interpretação via Redes Neurais

- ▶ O problema é que essas variáveis de entrada são categóricas. Para usar um modelo de regressão, podemos antes transformá-las em variáveis binárias usando *one-hot encoding*:

$$\begin{aligned}\mathbf{x}_1 &= (x_{11}, \dots, x_{1m})^T, & x_{1j} &= 1[j = u] \\ \mathbf{x}_2 &= (x_{21}, \dots, x_{2n})^T, & x_{2j} &= 1[j = i]\end{aligned}$$

- ▶ Dessa forma, temos como vetor de entrada $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \{0, 1\}^{m+n}$
- ▶ Sejam

$$\mathbf{P} = \begin{bmatrix} \text{---} \mathbf{p}_1^T \text{---} \\ \vdots \\ \text{---} \mathbf{p}_m^T \text{---} \end{bmatrix} \quad \text{e} \quad \mathbf{Q} = \begin{bmatrix} \text{---} \mathbf{q}_1^T \text{---} \\ \vdots \\ \text{---} \mathbf{q}_n^T \text{---} \end{bmatrix}$$

- ▶ É fácil ver que:

$$\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i = (\mathbf{x}_1^T \mathbf{P})(\mathbf{Q}^T \mathbf{x}_2)$$

Interpretação via Redes Neurais

- ▶ Este modelo pode ser implementado por uma rede neural com uma arquitetura particular:

- ▶ A primeira camada calcula $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2) \in \mathbb{R}^{2f}$, onde

$$\mathbf{z}_1 = \mathbf{P}^T \mathbf{x}_1 \quad (= \mathbf{p}_u)$$

$$\mathbf{z}_2 = \mathbf{Q}^T \mathbf{x}_2 \quad (= \mathbf{q}_i)$$

- ▶ A segunda camada (camada de saída) calcula $\hat{r}_{ui} = \mathbf{z}_1^T \mathbf{z}_2$
- ▶ Na literatura, a operação que mapeia uma variável categórica em um vetor em \mathbb{R}^f , como no mapeamento

$$u \in \{1, \dots, m\} \quad \mapsto \quad \mathbf{z}_1 \in \mathbb{R}^f$$

$$i \in \{1, \dots, n\} \quad \mapsto \quad \mathbf{z}_2 \in \mathbb{R}^f$$

é chamada de *embedding*

- ▶ Esta operação está eficientemente implementada em bibliotecas como Keras, evitando a necessidade de utilizar *one-hot encoding*