SCC0284 / SCC5966
Sistemas de Recomendação

Aula 02: Filtragem Colaborativa
Parte 2

(mmanzato@icmc.usp.br)

FC baseada em modelo

- Baseada em um pré-processamento offline ou fase de "aprendizado de modelo"
- Em tempo de execução, apenas o modelo treinado é usado para calcular predições
- Modelos são atualizados / re-treinados periodicamente
- Construção e atualização do modelo podem ser caras computacionalmente

FC baseada em modelo

- Alguns modelos que veremos no curso:
  - Método baseline
  - Fatoração de matrizes via:
    - Singular Value Decomposition
    - Gradiente Descendente
  - FunkSVD
  - SVD++

Método Baseline

- Método simples para predição de avaliações baseado em tendências de cada usuário e item
- Exemplo:
  - Média global: $\mu = 3.7$
  - Filme Titanic, avaliado com 0.5 estrelas acima da média:

    $$b_i = 0.5$$

  - Usuário Joe, que avalia filmes com 0.3 estrelas abaixo da média:

    $$b_u = -0.3$$

$$r_{ui} \approx \mu + b_i + b_u = b_{ui}$$
$$b_{ui} = 3.7 + 0.5 - 0.3 = 3.9$$

Método Baseline

- Estimativas de b_u e b_i:
  - Média global:

$$\mu = \frac{1}{|r_{ui} \in R|} \sum_{r_{ui} \in R} r_{ui}$$

R = conj. de todas as notas

  - Viés de item:

$$b_i = \frac{1}{\lambda_1 + |R(i)|} \sum_{u \in R(i)} r_{ui} - \mu$$

R(i) = conj. de usuários que avaliaram i
λ_1 = constante

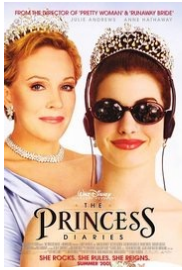
  - Viés de usuário:

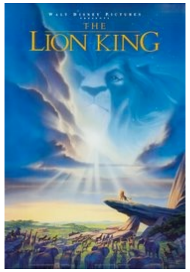
$$b_u = \frac{1}{\lambda_2 + |R(u)|} \sum_{i \in R(u)} r_{ui} - \mu - b_i$$

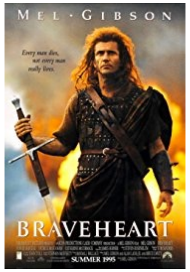
R(u) = conj. de itens que foram avaliados por u
λ_2 = constante

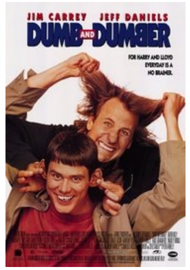
Método Baseline

- (Exemplo) Que nota Dave daria para Ocean's Eleven?









Jessica	5	2	4	3	2	3
Marta	4	3	5	4	3	2
Jose	1	5	3	4	4	5
Dave	1	?	2	3	4	2

Método Baseline

- É possível também estimar b_u e b_i por meio da resolução de um problema de mínimos quadrados:

$$\min_{b^*} \sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i)^2 + \lambda\left(\sum_u b_u^2 + \sum_i b_i^2\right)$$

- Detalhes adiante neste curso.

Fatoração de Matriz (FM)

- Competição Netflix mostrou que métodos de FM podem ser muito úteis na melhoria da acurácia de predições
- Derivam um conjunto de fatores latentes (escondidos) a partir dos padrões de interação
- Caracterizam ambos usuários e itens em termos de um vetor de fatores
  - Fator: aspecto do domínio (interpretável ou não)
- Recomendação é feita quando os fatores do usuário u e do item i são similares

FM

- Idéia de explorar fatores latentes vem da área de Recuperação de Informação (RI)
- Em RI, a fatoração é feita em uma matriz de termos vs. documentos
  - Cada célula representa um peso indicando a importância (ou existência) de um termo para aquele documento
- Em SR, a matriz é de usuários vs. itens
  - Cada célula é uma avaliação / interação

Singular Value Decomposition (SVD)

- Para uma matriz R, $t \times d$, sua decomposição é a fatoração de R em três matrizes tal que:

$$\boxed{R = P\Sigma Q^T}$$

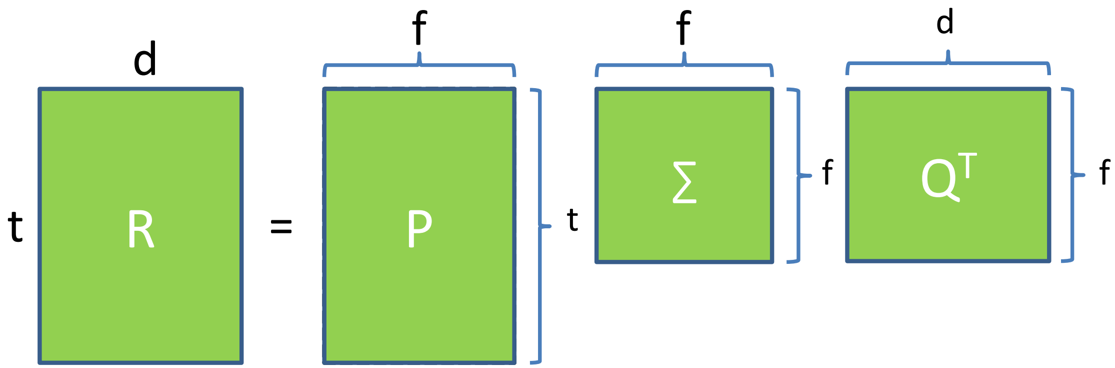
sendo que Σ é uma matriz diagonal cujos elementos σ_i são valores singulares da decomposição, P e Q são ortogonais

- Na forma simples, P tem dimensões $t \times f$, Σ tem dimensões $f \times f$ e Q tem dimensões $d \times f$, onde f é o *rank* (posto) de R

SVD

- Decomposição:

$R = P\Sigma Q^T$



SVD

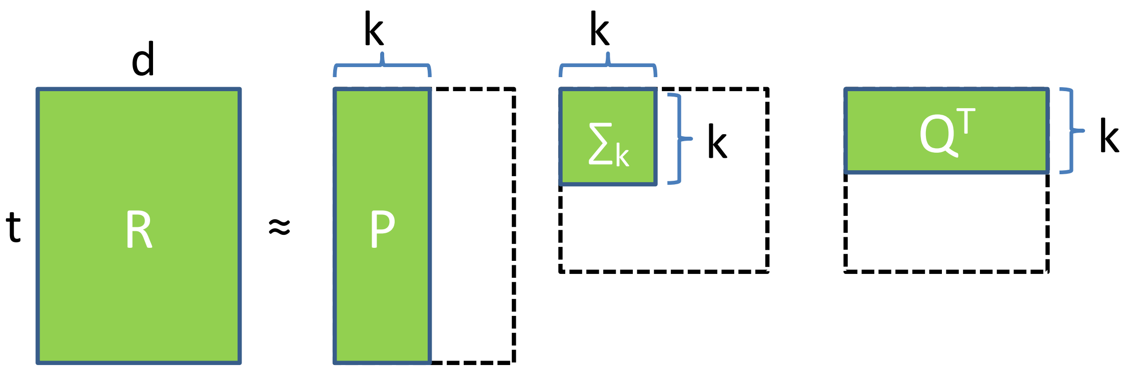
- Redução de dimensionalidade
  - Consiste em considerar apenas os k maiores valores singulares de $\sum$, resultando em uma matriz $\sum_k$ de dimensão k x k

- Vantagens:
  - Redução do espaço vetorial, sendo que os mesmos termos (ou usuários) e documentos (ou itens) podem agora ser representados por vetores de dimensão k
  - Redução de ruído e pequenas protuberâncias por meio da eliminação dos valores singulares menos relevantes

SVD

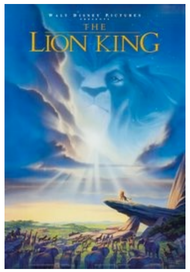
- A computação de SVD na matriz R resulta na seguinte fatoração:

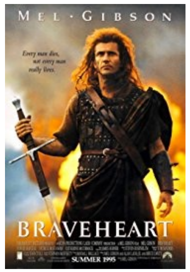
$$R \approx P\Sigma_k Q^T$$

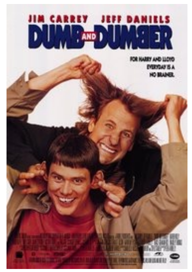


Recomendação baseada em SVD

- Que nota Dave daria para Ocean's Eleven?







Jessica	5	2	4	3	2	3
Marta	4	3	5	4	3	2
Jose	1	5	3	4	4	5
Dave	1	?	2	3	4	2

Recomendação baseada em SVD

- Aplica-se o mesmo princípio na matriz de avaliações…

SVD: $$R_k = P_k \times \Sigma_k \times Q_k^T$$

Predição: $$\hat{r}_{ui} = \sum_{f=1}^{k} p_{uf} \sigma_f q_{if}$$ (notas absolutas)

ou

$$\hat{r}_{ui} = b_{ui} + \sum_{f=1}^{k} p_{uf} \sigma_f q_{if}$$ (notas relativas)

Recomendação baseada em SVD

- Como obter P, Q e Σ?
  - Técnica SVD vem da Álgebra Linear
  - Pacotes disponíveis:
    - R, Matlab, SciPy
    - LINPACK, ARPACK
    - Java matrix libraries

Recomendação baseada em SVD

- Vantagens
  - Elimina ruídos nos dados devido à redução de dimensionalidade
  - Detecta correlações não triviais nos dados
- Desvantagens
  - Avaliações originais não são consideradas (apenas a aproximação)
  - Diferentemente de RI, os “zeros” representam valores desconhecidos
    - O usuário poderia gostar de um item se o conhecesse
    - “Zero” em RI indica que aquele termo não aparece no documento

Gradiente Descendente

- Resolve dois problemas específicos do SVD:
  - Lentidão na decomposição
  - Matriz incompleta
- Considera apenas os valores conhecidos da matriz de interações
- Utiliza uma métrica de erro (e.g. RMSE) para otimizar as matrizes da decomposição
  - Encontrar a melhor aproximação rank-k ao invés de calcular formalmente o SVD usando toda a matriz

Gradiente Descendente

- Simplificação de SVD
  - SVD original:

$$R = P \times \Sigma \times Q^T \qquad \text{ou} \qquad R = B + P \times \Sigma \times Q^T$$

  - Modelo de decomposição:

$$R = B + P \times Q^T$$

  - Predição:

$$\hat{r}_{ui} = b_{ui} + \sum_{f=1}^{k} p_{uf} q_{if} \qquad \text{onde} \quad b_{ui} = \mu + b_u + b_i$$

Gradiente Descendente

- Simplificação de SVD
  - Idéia é minimizar o erro (RMSE)

$$RMSE = \sqrt{\frac{1}{|K|}\sum_{(u,i)\in K}(r_{ui} - \hat{r}_{ui})^2}$$

onde K é o conjunto de pares (u,i) com avaliações conhecidas

  - Ajustar P e Q por meio de gradiente descendente
  - Raiz quadrada e divisão por constante podem ser eliminados
    - Minimizar soma dos erros quadráticos: $\sum_{(u,i)\in K}(r_{ui} - \hat{r}_{ui})^2$

Gradiente Descendente

- O que queremos:

$$\min_{b_*, p_*, q_*} \sum_{(u,i)\in K} \left(r_{ui} - \mu - b_u - b_i - \sum_{f=1}^{k} p_{uf} q_{if} \right)^2$$

- Resolver por diferentes métodos de otimização
  - <u>Gradiente descendente</u>
  - Alternating least squares
  - Etc.

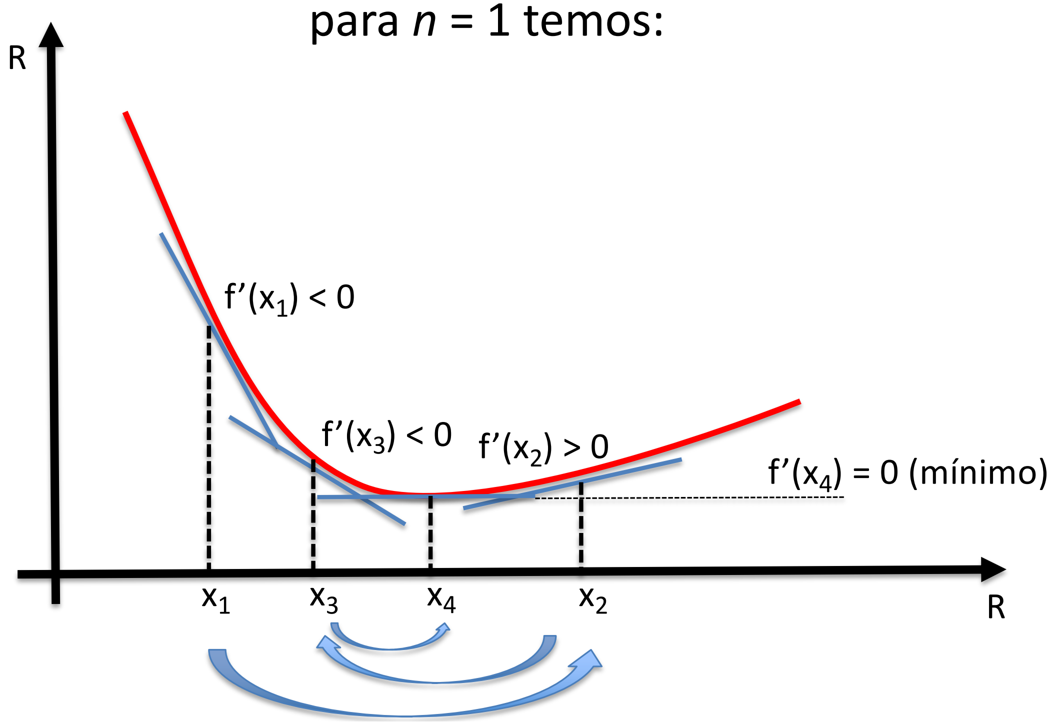
Gradiente Descendente

- Variante: Gradiente Descendente Estocástico (SGD)
  - Chutar um conjunto de valores iniciais para os parâmetros
  - Calcular o erro comparando os dados reais de K com a predição do modelo
  - Usar a derivada do erro para ajustar os valores das matrizes

Gradiente Descendente

suponha $f : R^n \rightarrow R$

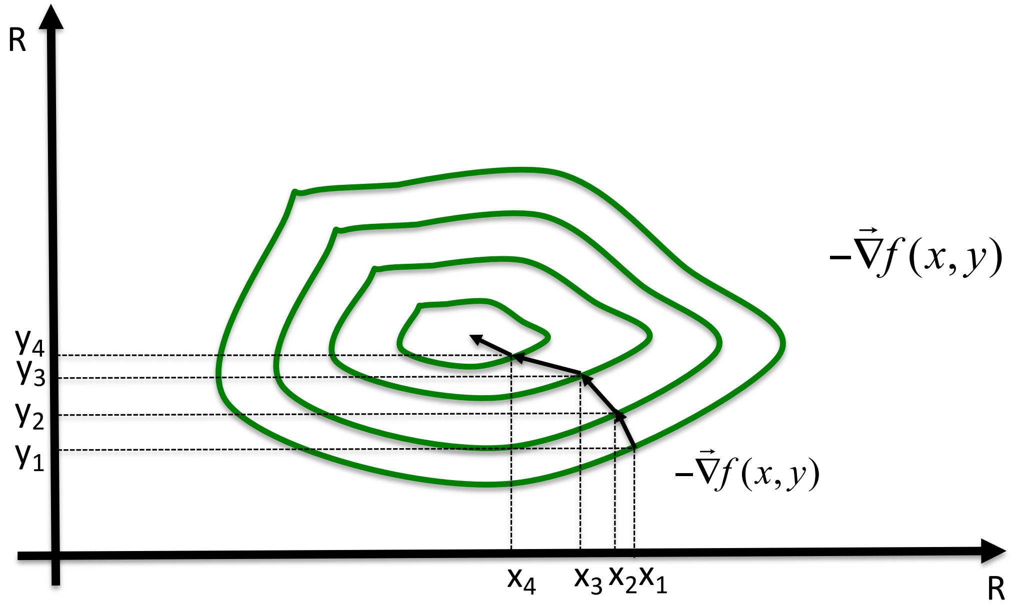
para n = 1 temos:



Gradiente Descendente

suponha $f : R^n \rightarrow R$

para n = 2 temos:



$$-\vec{\nabla} f(x,y) = -\left(\frac{\partial}{\partial x} f, \frac{\partial}{\partial y} f\right)(x,y)$$

Gradiente Descendente

$$\min_{b_*, p_*, q_*} \sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i - \sum_{f=1}^{k} p_{uf} q_{if})^2$$

$$\varepsilon_{ui} = r_{ui} - \hat{r}_{ui} = r_{ui} - \mu - b_u - b_i - \sum_{f=1}^{k} p_{uf} q_{if}$$

$$\left\{ \begin{aligned}
&\frac{\partial}{\partial b_u} \varepsilon_{ui}^2 = 2\varepsilon_{ui} \frac{\partial}{\partial b_u} \varepsilon_{ui} = 2\varepsilon_{ui} \frac{\partial}{\partial b_u} (r_{ui} - \mu - b_u - b_i - \sum_{f'=1}^{k} p_{uf'} q_{if'}) = -2\varepsilon_{ui} \\
&\frac{\partial}{\partial b_i} \varepsilon_{ui}^2 = 2\varepsilon_{ui} \frac{\partial}{\partial b_i} \varepsilon_{ui} = 2\varepsilon_{ui} \frac{\partial}{\partial b_i} (r_{ui} - \mu - b_u - b_i - \sum_{f'=1}^{k} p_{uf'} q_{if'}) = -2\varepsilon_{ui} \\
&\frac{\partial}{\partial p_{uf}} \varepsilon_{ui}^2 = 2\varepsilon_{ui} \frac{\partial}{\partial p_{uf}} \varepsilon_{ui} = 2\varepsilon_{ui} \frac{\partial}{\partial p_{uf}} (r_{ui} - b_{ui} - \sum_{f'=1}^{k} p_{uf'} q_{if'}) = -2\varepsilon_{ui} q_{if} \\
&\frac{\partial}{\partial q_{if}} \varepsilon_{ui}^2 = 2\varepsilon_{ui} \frac{\partial}{\partial q_{if}} \varepsilon_{ui} = 2\varepsilon_{ui} \frac{\partial}{\partial q_{if}} (r_{ui} - b_{ui} - \sum_{f'=1}^{k} p_{uf'} q_{if'}) = -2\varepsilon_{ui} p_{uf}
\end{aligned} \right.$$

Gradiente Descendente

- Ajuste dos parâmetros:

$$\Theta_j = \Theta_{j-1} - \gamma \vec{\nabla}(\Theta_{j-1})$$

- Valores no próximo passo (Θ_j) dependem de:
  - Valores do passo anterior (Θ_{j-1})
  - Taxa de aprendizado (γ)
  - Gradiente do erro

Gradiente Descendente

- Ajuste dos parâmetros:

$$\varepsilon_{ui} = r_{ui} - \hat{r}_{ui}$$
$$b_u = b_u + \gamma\varepsilon_{ui}$$
$$b_i = b_i + \gamma\varepsilon_{ui}$$
$$q_{if} = q_{if} + \gamma(\varepsilon_{ui} p_{uf})$$
$$p_{uf} = p_{uf} + \gamma(\varepsilon_{ui} q_{if})$$

  – Parâmetros:
    - γ : taxa de aprendizado (quão rápido converge)

Gradiente Descendente

- Ajuste dos parâmetros:

$$
\begin{aligned}
\varepsilon_{ui} &= r_{ui} - \hat{r}_{ui} \\
b_u &= b_u + \gamma\varepsilon_{ui} - \lambda b_u \\
b_i &= b_i + \gamma\varepsilon_{ui} - \lambda b_i \\
q_{if} &= q_{if} + \gamma(\varepsilon_{ui} p_{uf} - \lambda q_{if}) \\
p_{uf} &= p_{uf} + \gamma(\varepsilon_{ui} q_{if} - \lambda p_{uf})
\end{aligned}
$$

  – Parâmetros:
    - γ : taxa de aprendizado (quão rápido converge)
    - λ : regularização (viés contra modelos extremos)

Gradiente Descendente

- FunkSVD

inicializar vetores b_u e b_i

inicializar matrizes P e Q

para f = 1 até k faça

    repita

        para (u,i) $\in$ K

            calcular predição para r_{ui}

            atualizar b_u, b_i, p_{uf}, q_{if}

    até convergir

Gradiente Descendente

- SVD otimizado

inicializar vetores b_u e b_i com zero
inicializar matrizes P e Q com distribuição normal
para l = 1 até max_iter faça
    para $(u,i) \in K$
        calcular predição para r_{ui}
        atualizar b_u, b_i
        para f = 1 até k faça
            atualizar p_{uf}, q_{if}

Gradiente Descendente

- SVD++
  - Combina feedback explícito e implícito em um único modelo

$$\hat{r}_{ui} = \underbrace{\mu + b_u + b_i}_{\text{viés do usuário e item}} + \underbrace{q_i^T}_{\text{fatores do item}}\left(\underbrace{p_u}_{\text{fatores do usuário}} + \underbrace{|N(u)|^{-\frac{1}{2}}}_{\text{normalização}} \underbrace{\sum_{j \in N(u)} y_j}_{\text{feedback implícito}}\right)$$

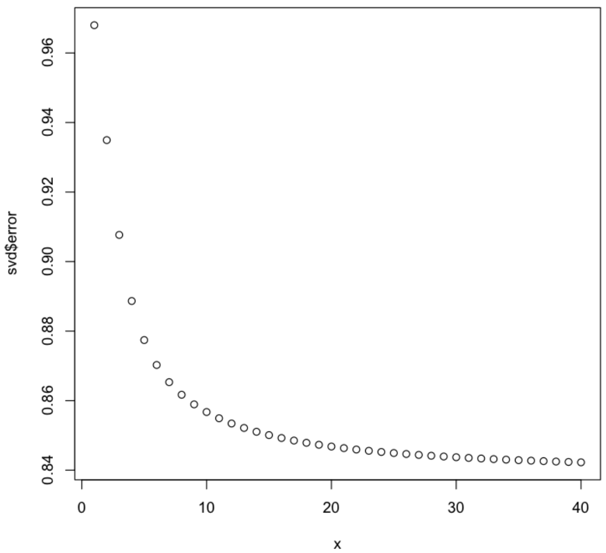
Gradiente Descendente

- SVD++
  - Treinamento do modelo segue o mesmo esquema
  - Parâmetros: $\Theta=\{b_u, b_i, p_u, q_i, y_j\}$

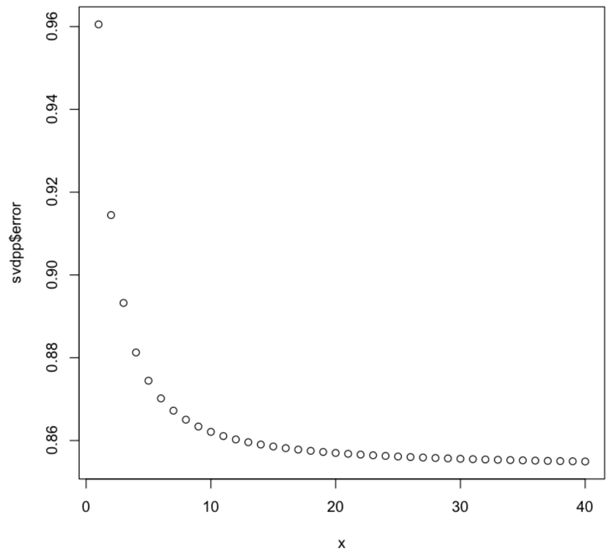
$$
\begin{cases}
b_u = b_u + \gamma(\varepsilon_{ui} - \lambda b_u) \\
b_i = b_i + \gamma(\varepsilon_{ui} - \lambda b_i) \\
q_i = q_i + \gamma(\varepsilon_{ui}(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j) - \lambda q_i) \\
p_u = p_u + \gamma(\varepsilon_{ui} q_i - \lambda p_u) \\
\forall j \in N(u): \\
\qquad y_j = y_j + \gamma \ (\varepsilon_{ui} |N(u)|^{-\frac{1}{2}} q_i - \lambda y_j)
\end{cases}
$$

Gradiente Descendente

- Comparação entre os modelos SVD e SVD++



SVD
no. fatores = 4
no. iterações = 40
RMSE = 1.0303



SVD++
no. fatores = 4
no. iterações = 40
RMSE = 1.0187

Filtragem colaborativa

- Baseada em vizinhança
  - Boa para detectar relacionamentos fortes entre itens próximos entre si (visão local)
- Baseada em fatores latentes (FM)
  - Boa para capturar relações não aparentes na base de dados (visão global)

Filtragem colaborativa

- Vantagens:
  - Técnica bem estudada e entendida
  - Funciona bem em vários domínios
  - Não precisa de conhecimento especializado
- Desvantagens:
  - Requer colaboração da comunidade
  - Esparsidade dos dados
  - Sem integração com outras fontes de conhecimento
  - Na baseada em modelos, é difícil explicar as recomendações

Referências

- [Sarwar et al. 2000a] Application of dimensionality reduction in recommender systems – a case study, Proceedings of the ACM WebKDD Workshop (Boston), 2000

- [Koren et al. 2009] *Matrix factorization techniques for recommender systems*, Computer 42 (2009), no. 8, 30–37

- https://github.com/JacintoCC/FunkSVD/blob/master/FunkSVD.R