SCC0284 – Sistemas de Recomendação

Aula 03: 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: μ = 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

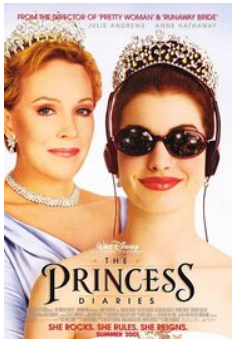
  - Viés de usuário:

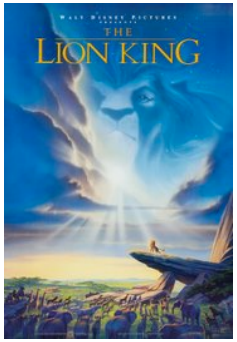
$$b_u = \frac{1}{\lambda_2 + |R(u)|} \sum_{i \in R(u)} r_{ui} - \mu - b_i$$

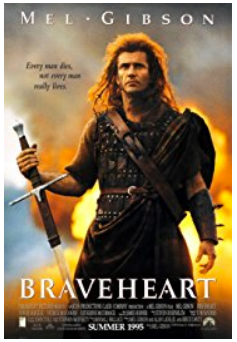
R(u) = conj. de itens que foram avaliados por u
λ_2 = constante

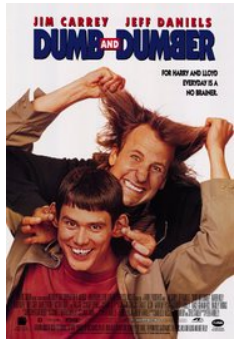
Método Baseline

- (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

- Código em R:

```
setwd("/Users/manzato/Documents/Ensino/USP/2018-1o/SCC0284 - RecSys/Aula 03")
r = read.table("svd.csv", header=TRUE, sep=",", row.names=1)
mean <- mean(as.matrix(r), na.rm = TRUE)
item_bias <- function (col) {
	return(mean(col - mean, na.rm = TRUE))
}
bi <- apply(r, 2, FUN=item_bias)
user_bias <- function (row) {
	return(mean(row - mean - bi, na.rm = TRUE))
}
bu <- apply(r, 1, FUN=user_bias)
pred <- mean + bu[4] + bi[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 x d, sua decomposição é a fatoração de R em três matrizes tal que:

$$R = P\sum Q^T$$

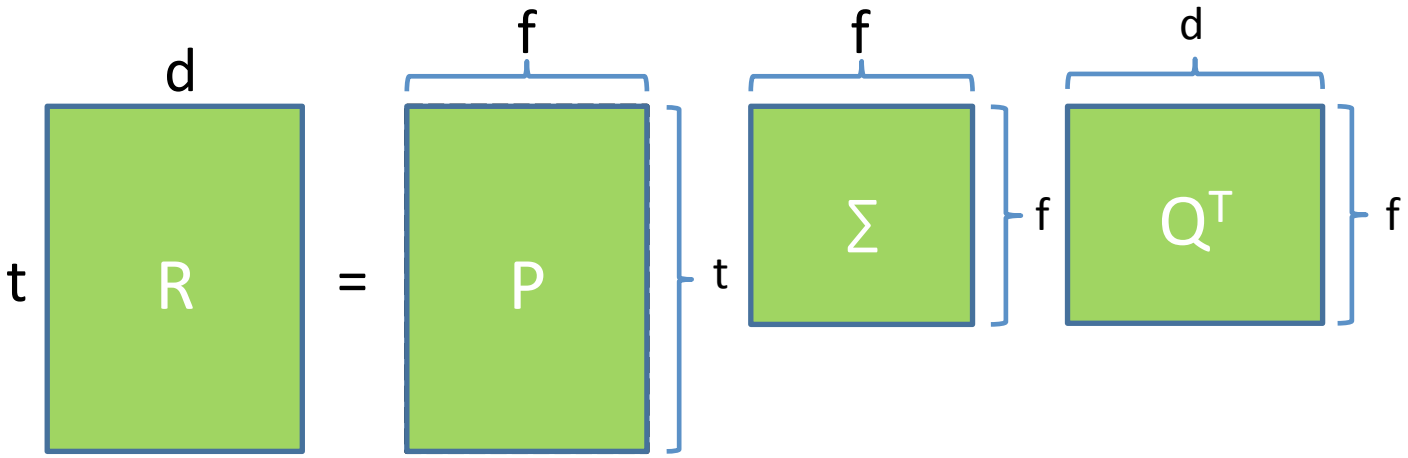
sendo que $\sum$ é uma matriz diagonal cujos elementos σ_i são valores singulares da decomposição, P e Q são ortogonais

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

SVD

- Decomposição:

$$R = P \Sigma Q^T$$



SVD

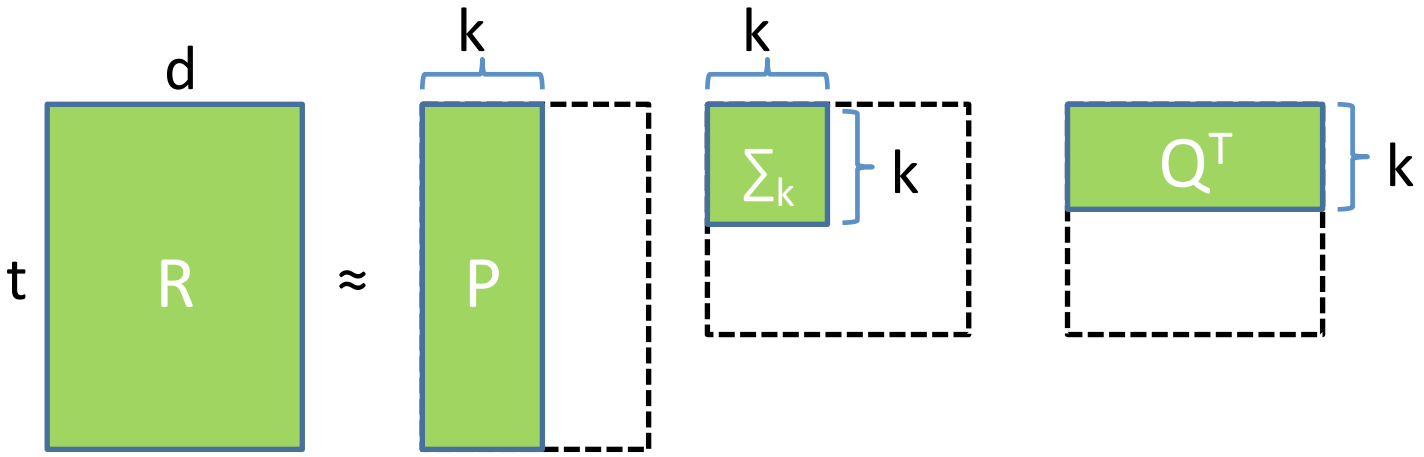
- Redução de dimensionalidade
  - Consiste em considerar apenas os k maiores valores singulares de $\sum$, resultando em uma matriz $\sum_k$ de dimensão k x k

- Vantagens:
  - Redução do espaço vetorial, sendo que os mesmos termos e documentos 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 de termo vs. documento resulta na seguinte fatoração:

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



SVD (exemplo)

R

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

P

	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

Σ

	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

Q^T

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

SVD (exemplo)

- Código em R:

```
r = read.table("tdm.csv", header=TRUE, sep=",", row.names=1)
svd <- svd(r)
rownames(svd$v) <- colnames(r)
rownames(svd$u) <- rownames(r)
svd$u
svd$d
svd$v
```

SVD (exemplo)

P	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

Σ_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

Q^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

Redução de dimensionalidade (k = 2)

SVD (exemplo)

- Código em R:

```
p2 <- svd$u[,1:2]
q2 <- svd$v[,1:2]
s2 <- svd$d[1:2]

library(ggplot2)
df <- data.frame(x = p2[,1], y = p2[,2], label = p2[,0])
ggplot(df, aes(x=x, y=y)) + geom_point(data = df, aes(x=x, y=y))
+ geom_text(data = df, aes(x=x, y=y-0.009, label=row.names(p2)))
```

SVD (exemplo)

R	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

R_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

$$R_2 = P\Sigma_2 Q^T$$

Similaridade entre d_2 e d_3 no espaço original: 0.
Similaridade entre d_2 e d_3 no espaço reduzido:

$$0.52 * 0.28 + 0.36 * 0.16 + 0.72 * 0.36 + 0.12 * 0.20 + -0.39 * -0.08 \approx 0.52$$

→ d_2 tem 'boat' e d_3 tem 'ship'

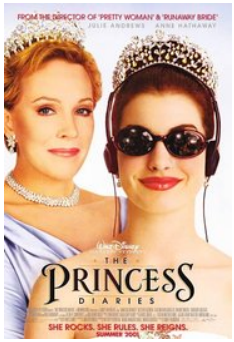
SVD (exemplo)

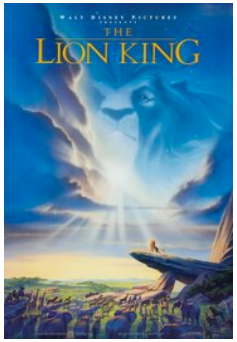
- Código em R:

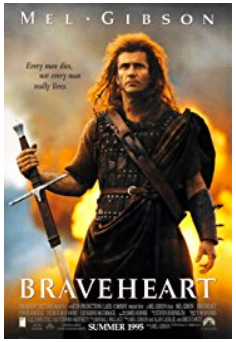
```
r2 <- p2 %*% diag(s2) %*% t(q2)
r2
r
sum(r2[,2] * r2[,3])
sum(r[,2] * r[,3])
```

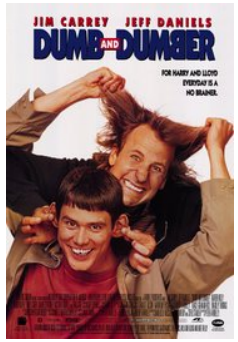
Recomendação baseada em SVD

- 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

```
r = read.table("svd.csv", header=TRUE, sep=",", row.names=1)
svd <- svd(r)
rownames(svd$v) <- colnames(r)
rownames(svd$u) <- rownames(r)
p2 <- svd$u[,1:2]
q2 <- svd$v[,1:2]
s2 <- svd$d[1:2]
x = c(p2[,1], q2[,1])
y = c(p2[,2], q2[,2])
label = c(rownames(p2), rownames(q2))
df <- data.frame(x = x, y = y, label = label)
ggplot(df) + geom_point(data = df, aes(x, y)) + geom_text(data = df, aes(x,
y-0.02), label=label)
pred <- p2[4,] %*% diag(s2) %*% q2[2,]
pred
```

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) conhecidos

  - 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
  - Gradiente descendente
  - Alternating least squares
  - Etc.

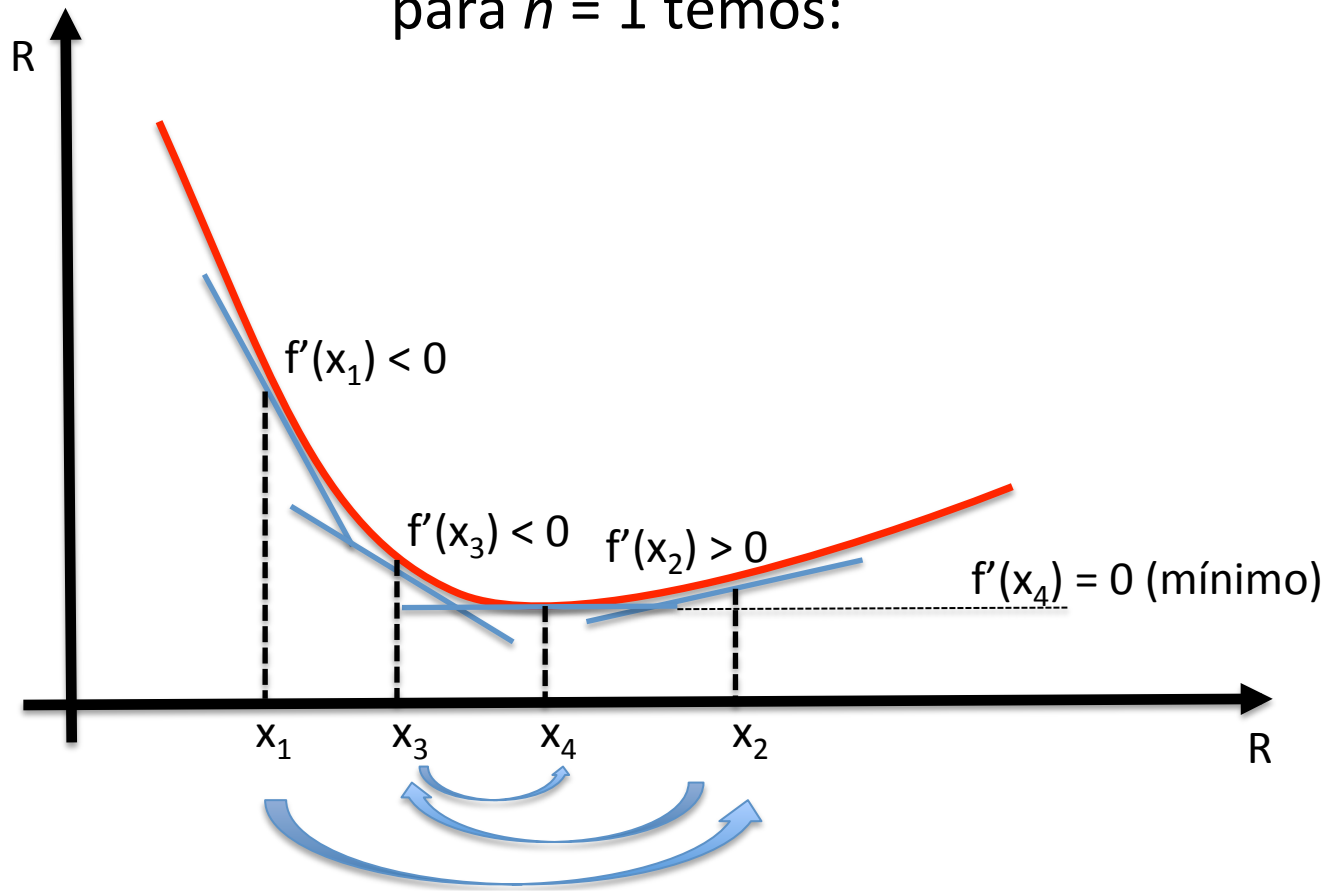
Gradiente Descendente

- 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 $f : R^n \rightarrow R$

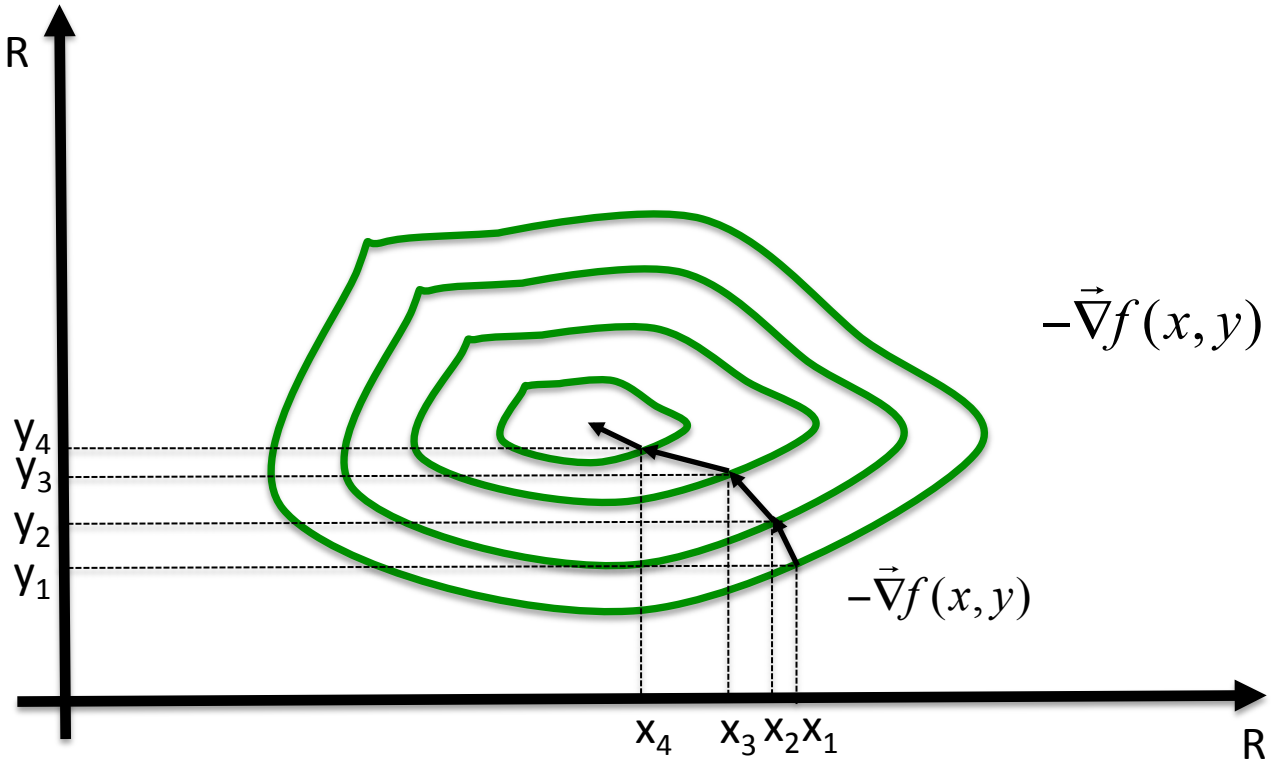
para $n = 1$ temos:



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

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

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

$$\frac{\partial}{\partial b_u} \varepsilon_{ui}^2 = 2\varepsilon_{ui} \frac{\partial}{\partial b_u} \varepsilon_{ui} = 2\varepsilon_{ui} \frac{\partial}{\partial b_u} (r_{ui} - \mu - b_u - b_i - \sum_{f'=1}^{k} p_{uf'} q_{if'}) = -2\varepsilon_{ui}$$

$$\frac{\partial}{\partial b_i} \varepsilon_{ui}^2 = 2\varepsilon_{ui} \frac{\partial}{\partial b_i} \varepsilon_{ui} = 2\varepsilon_{ui} \frac{\partial}{\partial b_i} (r_{ui} - \mu - b_u - b_i - \sum_{f'=1}^{k} p_{uf'} q_{if'}) = -2\varepsilon_{ui}$$

$$\frac{\partial}{\partial p_{uf}} \varepsilon_{ui}^2 = 2\varepsilon_{ui} \frac{\partial}{\partial p_{uf}} \varepsilon_{ui} = 2\varepsilon_{ui} \frac{\partial}{\partial p_{uf}} (r_{ui} - b_{ui} - \sum_{f'=1}^{k} p_{uf'} q_{if'}) = -2\varepsilon_{ui} q_{if}$$

$$\frac{\partial}{\partial q_{if}} \varepsilon_{ui}^2 = 2\varepsilon_{ui} \frac{\partial}{\partial q_{if}} \varepsilon_{ui} = 2\varepsilon_{ui} \frac{\partial}{\partial q_{if}} (r_{ui} - b_{ui} - \sum_{f'=1}^{k} p_{uf'} q_{if'}) = -2\varepsilon_{ui} p_{uf}$$

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:

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

  - 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

```
funkSVD <- function(R, k, lr = 0.05, reg = 0.02, miter = 10) {
	global_mean <- mean(as.matrix(R), na.rm = TRUE)
	bu <- rep(0, nrow(R))
	bi <- rep(0, ncol(R))
	P <- matrix(0.1, nrow = nrow(R), ncol = k)
	Q <- matrix(0.1, nrow = ncol(R), ncol = k)
	K <- which(!is.na(R), arr.ind = TRUE)
	for(f in sample(k)) {
		for(l in 1:miter) {
			for(j in 1:nrow(K)){
				u <- K[j,1]
				i <- K[j,2]
				r_ui <- R[u,i]
				if(!is.na(r_ui)){
					pred <- global_mean + bu[u] + bi[i] + P[u, ] %*% Q[i, ]
					e_ui <- r_ui - pred
					bu[u] <- bu[u] + lr * e_ui
					bi[i] <- bi[i] + lr * e_ui
					temp_uf <- P[u,f]
					P[u,f] <- P[u,f] + lr * (e_ui * Q[i,f] - reg * P[u,f])
					Q[i,f] <- Q[i,f] + lr * (e_ui * temp_uf - reg * Q[i,f])
				}
			}
		}
	}
	return(list(bu = bu, bi = bi, P = P, Q = Q))
}
```

Gradiente Descendente

- Chamada

```
r = read.table("funksvd.csv", header=TRUE, sep=",", row.names=1)
source("funksvd.r")
svd <- funkSVD(r, 4)
global_mean <- mean(as.matrix(r), na.rm = TRUE)

rating <- global_mean + svd$bu[4] + svd$bi[2] + svd$P[4, ] %*% svd$Q[2,]
```

Gradiente Descendente

- SVD otimizado

inicializar vetores b_u e b_i com zero

inicializar matrizes P e Q com distribuição normal (média = ??, desvio padrão = ??)

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}

```
svdopt <- function(R, k, lr = 0.05, reg = 0.002, miter = 10) {
	global_mean <- mean(as.matrix(R[,3]), na.rm = TRUE)
	bu <- rep(0, max(R[,1]))
	bi <- rep(0, max(R[,2]))
	nusers <- max(R[,1])
	nitems <- max(R[,2])
	P <- matrix(rnorm(nusers*k, mean=0, sd=0.1), nusers, k)
	Q <- matrix(rnorm(nitems*k, mean=0, sd=0.1), nitems, k)
	error <- list()
	for(l in 1:miter) {
		sq_error <- 0
		for(j in 1:nrow(R)) {
			u <- R[j,1]
			i <- R[j,2]
			r_ui <- R[j,3]
			pred <- global_mean + bu[u] + bi[i] + P[u, ] %*% Q[i,]
			e_ui <- r_ui - pred
			sq_error <- sq_error + e_ui^2
			bu[u] <- bu[u] + lr * e_ui
			bi[i] <- bi[i] + lr * e_ui
			for(f in 1:k) {
				temp_uf <- P[u,f]
				P[u,f] <- P[u,f] + lr * (e_ui * Q[i,f] - reg * P[u,f])
				Q[i,f] <- Q[i,f] + lr * (e_ui * temp_uf - reg * Q[i,f])
			}
		}
		error <- c(error, sqrt(sq_error/nrow(R)))
	}
	return(list(bu = bu, bi = bi, P = P, Q = Q, error = error))
}
```

Gradiente Descendente

- Código em R
  - Utilizando base MovieLens-100k
  - Chamada:

```
r = read.table("ua.base", header=FALSE, sep="\t")
R <- as.matrix(r[1:3])
source("svdopt.r")
svd <- svdopt(R, 2)

# visualização do RMSE diminuíndo:
x <- seq(1, length(svd$error), 1)
plot(x, svd$error)

# nota do usuário id. 500 para o item id. 100:
global_mean <- mean(as.matrix(R[,3]), na.rm = TRUE)
pred <- global_mean + svd$bu[500] + svd$bi[100] + svd$P[500,] %*% svd$Q[100,]
```

Gradiente Descendente

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

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right)$$

viés do usuário e item

fatores do item

fatores do usuário

norma-lização

feedback implícito

Gradiente Descendente

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

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

```
svdpp <- function(R, N, k, lr = 0.05, reg = 0.002, miter = 10) {
	global_mean <- mean(as.matrix(R[,3]), na.rm = TRUE)
	bu <- rep(0, max(R[,1]))
	bi <- rep(0, max(R[,2]))
	nusers <- max(R[,1])
	nitems <- max(R[,2])
	P <- matrix(rnorm(nusers*k, mean=0, sd=0.1), nusers, k)
	Q <- matrix(rnorm(nitems*k, mean=0, sd=0.1), nitems, k)
	Y <- matrix(rnorm(nitems*k, mean=0, sd=0.1), nitems, k)
	for(l in 1:miter) {
		for(r in 1:nrow(R)) {
			u <- R[r,1]
			i <- R[r,2]
			r_ui <- R[r,3]
			N_u <- N[N[,1] == u,2]
			y_sum <- rep(0, k)
			for(j in N_u) {
				y_sum <- y_sum + Y[j,]
			}
			p_plus_y <- P[u, ] + (1/sqrt(length(N_u))) * y_sum
			pred <- global_mean + bu[u] + bi[i] + p_plus_y %*% Q[i,]
			e_ui <- r_ui - pred
			sq_error <- sq_error + e_ui^2
			bu[u] <- bu[u] + lr * e_ui
			bi[i] <- bi[i] + lr * e_ui
			for(f in 1:k) {
				P[u,f] <- P[u,f] + lr * (e_ui * Q[i,f] - reg * P[u,f])
				temp_if <- Q[i,f]
				Q[i,f] <- Q[i,f] + lr * (e_ui * p_plus_y[f] - reg * Q[i,f])
				for(j in N_u) {
					Y[j,f] <- Y[j,f] + lr * (e_ui * (1/sqrt(length(N_u))) * temp_if - reg * Y[j,f])
				}
			}
		}
	}
	return(list(bu = bu, bi = bi, P = P, Q = Q, Y = Y, error = error))
}
```

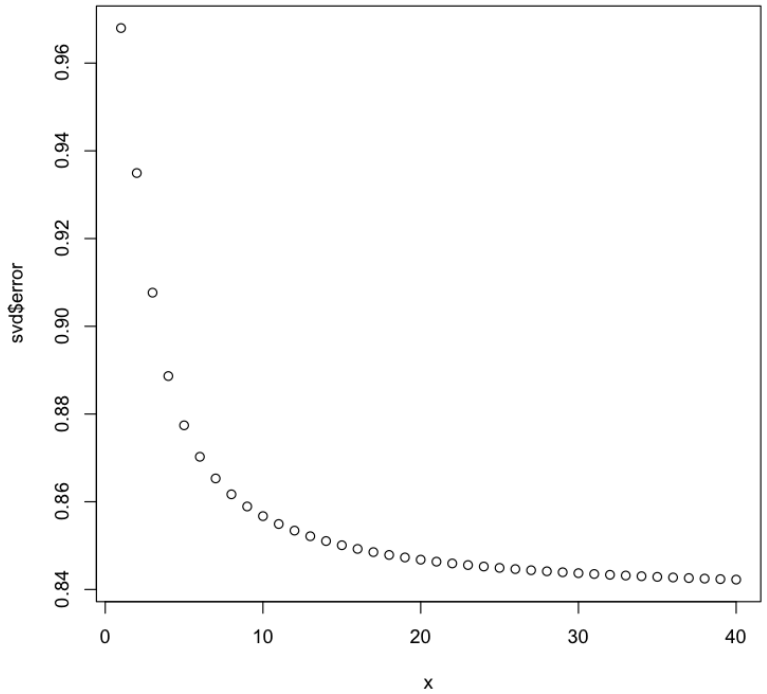
Gradiente Descendente

- Chamada:

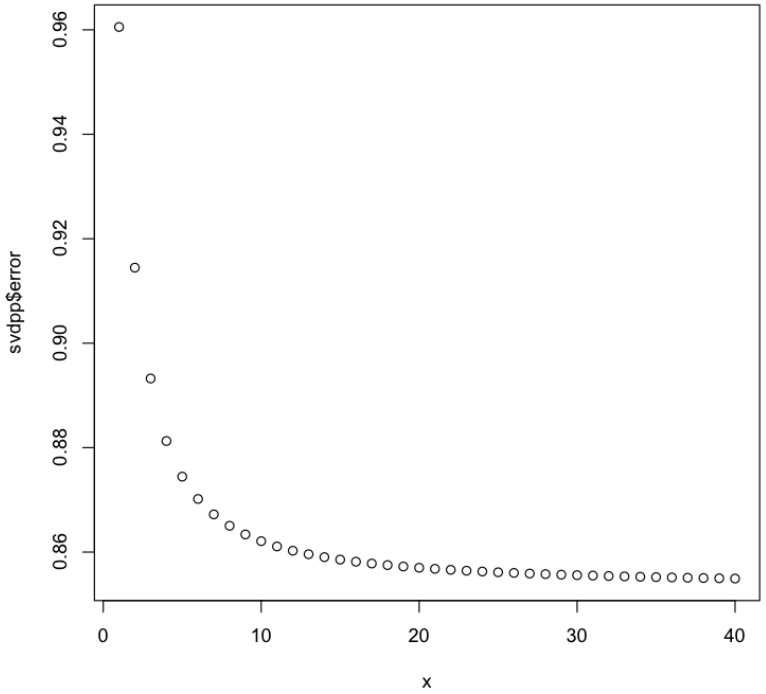
```
r = read.table("ua.base", header=FALSE, sep="\t")
R <- as.matrix(r[1:3])
# feedback implícito:
N <- as.matrix(r[1:2])

source("svdpp.r")
svdpp <- svdpp(R, N, 2)

# nota do usuário id. 500 para o item id. 100:
N_u <- N[N[,1] == 500,2]
y_sum <- rep(0, ncol(svdpp$Y))
for(j in N_u) {
        y_sum <- y_sum + svdpp$Y[j,]
}
p_plus_y <- svdpp$P[500, ] + (1/sqrt(length(N_u))) * y_sum
pred <- global_mean + svdpp$bu[500] + svdpp$bi[100] + p_plus_y %*% svdpp$Q[100,]
```

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