CSE 8803RS: Recommendation Systems

Lecture 3: Matrix Factorization for CF

Hongyuan Zha

School of Computational Science & Engineering
College of Computing
Georgia Institute of Technology

Basic Problem Formulation

Rating based paradigm

- Users: $u, v \in \mathcal{U}$; Items: $i, j \in \mathcal{I}$
- Ratings: r_{ui} indicating degree of preference of user u for item j, higher values $\Rightarrow$ stronger preference
- **Problem.** Ratings are not defined over all $\mathcal{U} \times \mathcal{I}$, need to predict those missing ratings
- Incomplete rating matrix

	Casablanc	God Father	Harry Potter	Lion King
David	5	4	2	?
John	3	2	?	5
Jenny	5	2	5	?

Structure of the Rating Matrix

Assume we have all the ratings we want, can we say something about the structure of the rating matrix?

- Assume an extreme case: all the users rated all the items in *the same way*, i.e., the rows are repetition of one single row vector g^T,

$$A = eg^T, \quad e = [1, \dots, 1]^T$$

  Prediction is also easy

- A is a special case of a rank-one matrix. More generally,

$$A = fg^T, \quad A_{ui} = f_u g_i$$

  Rough interpretation: f_i indicates how much user u likes movies, and g_i how much popular movie i is

Structure of the Rating Matrix

- The rank-one model is coarse, in fact, there are many different genres of movies, say k of them
- Rank-k model

$$A_{ui} = f_{u1}g_{i1} + \cdots + f_{uk}g_{ik}$$

- Rough interpretation:
  - $g_{i\ell}$ relative score for movie i in genre ℓ
  - $f_{u\ell}$ the affinity of user u for genre ℓ
- In matrix format,

$$A = FG^T, \quad F \in R^{M \times k}, G \in R^{N \times k}$$

- A is a rank-k matrix

Netflix Matrix Example

- Ratings: 100M (from 1 to 5)
- Movies: 17K
- Users: 500K
- Potential entries: 8.5B, and 8.4B empty cells
- Let $k = 40$, then 40*(17K+500K)= 21M, 400 times less than 8.5B

Latent Profiles

Latent variable models

- Latent profiles
  - User latent profiles: $F_u = [F_{u1}, \ldots, F_{uk}]$
  - Item latent profiles: $G_i = [G_{i1}, \ldots, G_{ik}]$
- Rating $A_{ui} = F_u G_i^T$, dot-product of the profiles
- Projection viewpoint: users and items projected to k-dimensional Euclidean space R^k
  — Geometry in $R^k \Leftrightarrow$ domain-specific relations
  — Similar users, similar items etc.

But generally, we only have $A \approx FG^T$

Singular Value Decomposition

- Given $A \in R^{M \times N}$, $M \geq N$,

$$A = U \Sigma V^T$$

U and V are orthogonal matrices, $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$,

$$\sigma_1 \geq \cdots \geq \sigma_N$$

- A can be written as a linear combination of rank-one matrices

$$A = \sum_{i=1}^{N} \sigma_i u_i v_i^T$$

SVD: Examples

```
X =
1 2
3 4
5 6
7 8
```

Matlab command

```
[U,S,V] = svd(X)
U =
-0.1525 -0.8226 -0.3945 -0.3800
-0.3499 -0.4214 0.2428 0.8007
-0.5474 -0.0201 0.6979 -0.4614
-0.7448 0.3812 -0.5462 0.0407
```

SVD: Examples

```
S =
14.2691 0
0 0.6268
0 0
0 0

V =
-0.6414 0.7672
-0.7672 -0.6414
```

Best Rank-k Approximation

- Given $A \in R^{M \times N}$, $M \geq N$, let

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

  Then $\mathrm{rank}(A_k) = k$.

- A_k is the best rank-k approximation of A,

$$A_k = \mathrm{argmin}_{\mathrm{rank}(B) \leq k} \|A - B\|$$

- If $\|\cdot\| = \|\cdot\|_F$, the Frobenius norm, then

$$\|A - B\|_F^2 = \sum_{u,i} (A_{ui} - B_{ui})^2$$

Best Rank-k Approximation: Incomplete Data

- Rewrite $B = FG^T$, where $F \in R^{M \times k}$ and $G \in R^{N \times k}$, i.e.,

$$B_{ui} = F_u G_i^T = \sum_{s=1}^{k} F_{us} G_{is}$$

where F_u and G_i are the u-th row and i-th row of F and G

- Let O be the index set with observed A_{ui}, we replace $\sum_{u,i}(A_{ui} - B_{ui})^2$ with

$$\sum_{(u,i)\in O} (A_{ui} - B_{ui})^2 = \sum_{(u,i)\in O} (A_{ui} - \sum_{s=1}^{k} F_{us} G_{is})^2$$

Best Rank-k Approximation: Incomplete Data

Optimization problem

- Find $F \in R^{M \times k}$ and $G \in R^{N \times k}$ so as to minimize

$$\mathcal{E}(F, G) = \sum_{(u,i) \in O} (A_{ui} - F_u G_i^T)^2$$

- Let $\mathcal{S}_O$ be a binary matrix, $\odot$ indicates component-wise multiplication

$$\min_{F,G} \mathcal{E}(F, G) = \min_{F,G} \|\mathcal{S}_O \odot (A - FG^T)\|_F^2$$

Regularized SVD

- Without controlling the size of the F and G leads to overfitting
- Adding *regularization* terms, the objective function we want to minimize is

$$E(F,G) = \frac{1}{2} \sum_{(u,i)\in O} (A_{ui} - \sum_{s=1}^{k} F_{us} G_{is})^2 + \frac{\tilde{\lambda}}{2} \sum_{u,s} U_{us}^2 + \frac{\tilde{\lambda}}{2} \sum_{i,s} V_{is}^2$$

- $\tilde{\lambda}$ the regularization parameter

Gradient Descent

- Minimization problem,

$$\min_{x \in R^D} F(x)$$

- Iterative methods starting with an initial guess x_0,

$$x_{i+1} = x_i - \alpha_i \nabla F(x_i)$$

where ∇F is the *gradient* of F

Gradient Descent

- Consider a single term from $E(F, G)$,

$$E_{ui}(F, G) = \frac{1}{2}(A_{ui} - \sum_{s=1}^{k} F_{us} G_{is})^2 + \frac{\lambda}{2} \sum_{u,s} F_{us}^2 + \frac{\lambda}{2} \sum_{i,s} G_{is}^2$$

- Take derivative w.r.t. F_{us},

$$\frac{\partial E_{ui}(F, G)}{\partial F_{us}} = (\sum_{s=1}^{k} F_{us} G_{is} - A_{ui}) G_{is} + \lambda F_{us} = -R_{ui} G_{is} + \lambda F_{us}$$

Iterative Scheme

- Notice that if $F(x) = F_1(x) + \cdots + F_s(x)$, then

$$\nabla F(x) = \nabla F_1(x) + \cdots + \nabla F_s(x)$$

- We also update the iterates one component at a time

Algorithm: Pseudo-Code

For Each Iteration

```
For each (u, i) ∈ O
    Compute the current estimate Â_ui = F_u G_i^T
    Compute the current error R_ui = A_ui − Â_ui
    For each s = 1, ..., k
        F_us ← F_us + μ(R_ui G_is − λF_us)
        G_is ← G_is + μ(R_ui F_us − λG_is)
```

Computational cost: $O(|O|k)$

Storage: $O(|O| + (M + N)K)$

Several Issues

- Choice of step length/learning rate μ, and choice of regularization parameter λ
  — Adaptive regularization: λ dependent on iteration number
- Choice of K
- Multiple local minimizers, choice of initial values
- The data $\{A_{ui}, (u,i) \in O\}$ can *NOT* fit into the existing memory: out of core implementation
- Multiple relations: ordering of the updates
- Parallel implementation
  — Trade-off between communication latency and convergence rate

Netflix Matrix Example

- $k = 96$
- $\mu = 0.001$
- $\lambda = 0.02$

Several Extensions

- BASELINE PREDICTOR for A_{ui}: linear regression on six features
  — empirical probabilities of each rating $1 - 5$ for user u
  — mean rating for movie i, after subtracting mean rating of each user
- CLIPPING: After learning of each feature, the predictions is clipped to range 1-5

Improved Regularized SVD

- New prediction formula

$$A_{ui} = \alpha_u + \beta_i + F_u G_i^T$$

- Reducing number of parameters: $O((M+N) \times k)$
  — Suppose I_u the set of items u rated
  — Assumption: $F_{us} = \sum_{i \in I_u} G_{is}$
  — New formula,

$$A_{ui} = \alpha_u + \beta_i + \sum_{s=1}^{k} G_{is} \sum_{j \in I_u} G_{js}$$

  — Number of parameters: $O(N \times k)$

Experimental Results

Predictor	Test RMSE with BASIC	Test RMSE with BASIC and RSVD2	Cumulative test RMSE
BASIC	.9826	.9039	.9826
RSVD	.9094	.9018	.9094
RSVD2	.9039	.9039	.9018
KMEANS	.9410	.9029	.9010
SVD_KNN	.9525	.9013	.8988
SVD_KRR	.9006	.8959	.8933
LM	.9506	.8995	.8902
NSVD1	.9312	.8986	.8887
NSVD2	.9590	.9032	.8879
SVD_KRR * NSVD1	—	—	.8879
SVD_KRR * NSVD2	—	—	.8877

SVD via Lanczos Bidigonalization

- Bidiagonalization: dense matrices,

$$A = UBV^T, \quad B = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & \beta_{n-1} \\ & & & & \alpha_n \end{bmatrix}$$

- The above can be computed using Householder transformations (Golub-Kahan algorithm)
- Then QR algorithm applied to B reduces it to diagonal form

Golub-Kahan-Lanczos Bidigonalization

- From $A = UBV^T$,

$$AV = UB, \quad A^T U = VB^T$$

Consider the k-columns of both sides,

$$Av_k = \alpha_k u_k + \beta_{k-1} v_{k-1}, \quad A^T u_k = \alpha_k v_k + \beta_{k+1} v_{k+1}$$

or

$$\alpha_k u_k = Av_k - \beta_{k-1} v_{k-1}, \beta_{k+1} v_{k+1} = A^T u_k - \alpha_k v_k$$

and

$$\alpha_k = \|Av_k - \beta_{k-1} v_{k-1}\|_2, \quad \beta_{k+1} = \|A^T u_k - \alpha_k v_k\|_2$$

- Start with unit v_1 and $\beta_0 = 0$

Golub-Kahan-Lanczos Bidigonalization

- After k steps,

$$AV_k = U_k B_k, \quad A^T U_k = V_k B_k^T + \beta_{k+1} v_{k+1} e_k^T$$

- Compute the SVD of $B_k = P_k S_k Q_k^T$, singular values of S_k, approximate singular values of A
- $U_k P_k$ and $V_k Q_k$ give approximate singular vectors,

$$A \approx (U_k P_k) S_k (V_k Q_k)^T$$

- Computational bottleneck: matrix-vector multiplication with A and A^T
- Re-orthogonalization

Partial SVD by Random Projection/Sampling

- $A \in R^{m \times n}$ and a given ℓ
  1. Draw $\Omega \in R^{n \times \ell}$ iid standard Gaussian
  2. Form $Y = A\Omega \in R^{m \times \ell}$
  3. Compute an orthonormal basis Q of Y
  4. Compute $B = Q^T A$
  5. Compute the SVD of $B = U_B \Sigma V^T$
     — Through eigen-decomposition of BB^T for example
  6. Then $A \approx (UU_B)\Sigma V^T$

Partial SVD by Random Projection: Error Bounds

- Let $Y = A\Omega$ and P_Y orthogonal projection, $\ell = k + p$,

$$\mathcal{E}\|(I - P_Y)A\|_F \leq \left(1 + \frac{k}{p-1}\right)^{1/2} \|A - A_k\|_F$$

  where A_k best rank-k approximation of A.
- p is called oversampling factor