Probabilistic Matrix Factorization

Piyush Rai
IIT Kanpur

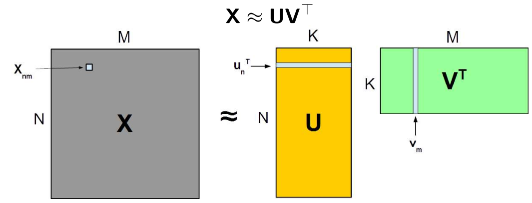
Probabilistic Machine Learning (CS772A)

Feb 8, 2016

Matrix Factorization

- Given a matrix $\mathbf{X}$ of size $N \times M$, approximate it via a low-rank decomposition

$$\mathbf{X} \approx \mathbf{U}\mathbf{V}^\top$$

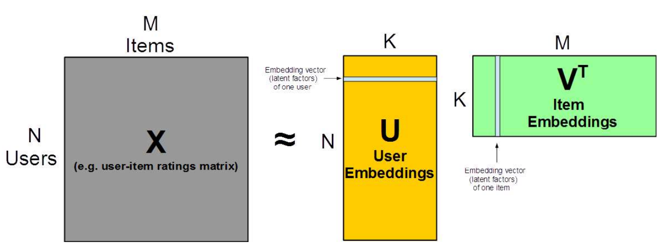


- Each entry of $\mathbf{X}$ can be written as

$$X_{nm} \approx \boldsymbol{u}_n^\top \boldsymbol{v}_m = \sum_{k=1}^{K} u_{nk} v_{mk}$$

- Note: $K \ll \min\{M, N\}$
- $\mathbf{U}$: $N \times K$ row latent factor matrix, $\boldsymbol{u}_n$: $K \times 1$ latent factors of row n
- $\mathbf{V}$: $M \times K$ column latent factor matrix, $\boldsymbol{v}_m$: $K \times 1$ latent factors of column m
- $\mathbf{X}$ may have missing entries

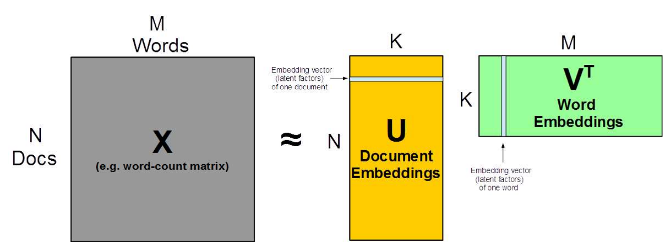
Matrix Factorization: Examples and Applications



Some applications:

- Learning embeddings from dyadic/relational data (each matrix entry is a dyad, e.g., user-item rating, document-word count, user-user link, etc.). Thus it also performs dimensionality reduction.

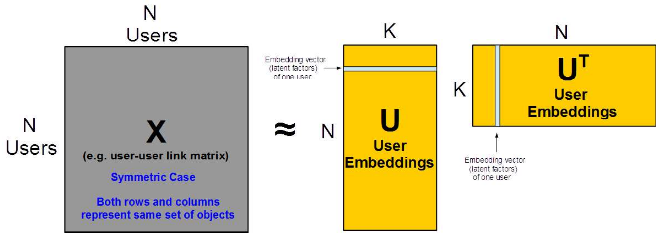
Matrix Factorization: Examples and Applications



Some applications:

- Learning embeddings from dyadic/relational data (each matrix entry is a dyad, e.g., user-item rating, document-word count, user-user link, etc.). Thus it also performs dimensionality reduction.

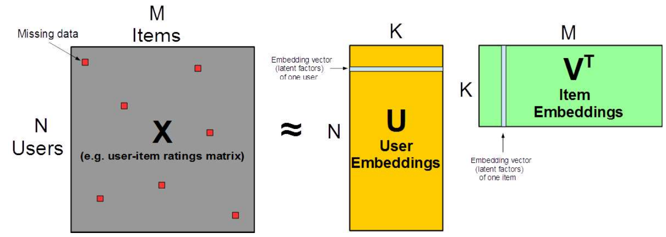
Matrix Factorization: Examples and Applications



Some applications:

- Learning embeddings from dyadic/relational data (each matrix entry is a dyad, e.g., user-item rating, document-word count, user-user link, etc.). Thus it also performs dimensionality reduction.

Matrix Factorization: Examples and Applications

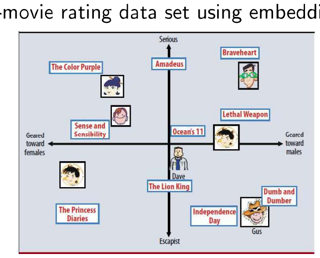


Some applications:

- Learning embeddings from dyadic/relational data (each matrix entry is a dyad, e.g., user-item rating, document-word count, user-user link, etc.). Thus it also performs dimensionality reduction.
- Matrix Completion, i.e., predicting missing entries in $\mathbf{X}$ via the learned embeddings (useful in recommender systems/collaborative filtering - Netflix Prize competition, link prediction in social networks, etc.): $X_{nm} \approx \boldsymbol{u}_n^\top \boldsymbol{v}_m$

Interpreting the Embeddings

- The embeddings/latent factors/latent features can be given interpretations (e.g., as genres if the matrix $\mathbf{X}$ represents a user-movie rating matrix case)
- A cartoon illustation of matrix factorization based embeddings (or "generes") learned from a user-movie rating data set using embedding dimension $K = 2$

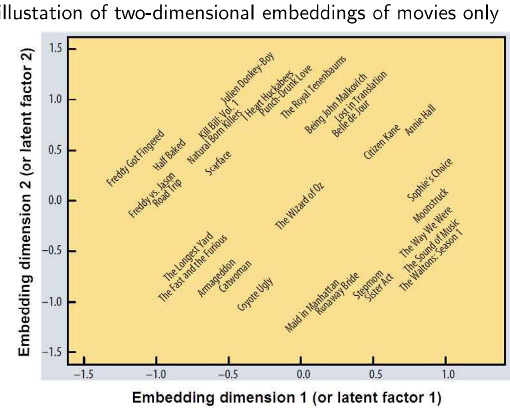


- Similar things (users/movies) get embedded nearby in the embedding space (two things will be deemed similar if their embeddings are similar). Thus useful for computing similarities and/or making recommendations

Picture courtesy: Matrix Factorization Techniques for Recommender Systems: Koren *et al*, 2009

Interpreting the Embeddings

- Another illustation of two-dimensional embeddings of movies only

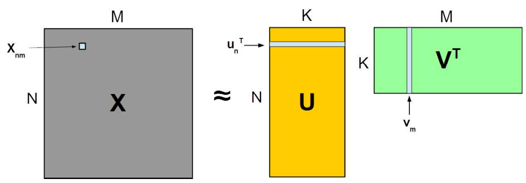


- Similar movies get embedded nearby in the embedding space

Picture courtesy: Matrix Factorization Techniques for Recommender Systems: Koren *et al*, 2009

Matrix Factorization

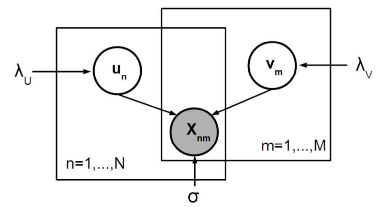
- Recall our model $\mathbf{X} \approx \mathbf{U}\mathbf{V}^\top$ or $\mathbf{X} = \mathbf{U}\mathbf{V}^\top + \mathbf{E}$ where $\mathbf{E}$ is the noise matrix
- Goal: learn $\mathbf{U}$ and $\mathbf{V}$, given a subset Ω of $\mathbf{X}$ (let's call it $\mathbf{X}_\Omega$)
- Some notations:
  - $\Omega = \{(n,m)\}$: X_{nm} is observed
  - $\Omega_{\boldsymbol{u}_n}$: column indices of observed entries in rows n
  - $\Omega_{\boldsymbol{v}_m}$: row indices of observed entries in column m



Probabilistic Matrix Factorization

- Assuming latent factors $\boldsymbol{u}_n$, $\boldsymbol{v}_m$ and each matrix entry X_{nm} to be real-valued

$$\begin{aligned} \boldsymbol{u}_n &\sim \mathcal{N}(\boldsymbol{u}_n|\mathbf{0}, \lambda_U^{-1}\mathbf{I}_K), & n = 1,\ldots,N \\ \boldsymbol{v}_m &\sim \mathcal{N}(\boldsymbol{v}_n|\mathbf{0}, \lambda_V^{-1}\mathbf{I}_K), & m = 1,\ldots,M \\ X_{nm} &\sim \mathcal{N}(X_{nm}|\boldsymbol{u}_n^\top\boldsymbol{v}_m, \sigma^2), & \forall (n,m) \in \Omega \end{aligned}$$



- This is also equivalent to $X_{nm} = \boldsymbol{u}_n^\top\boldsymbol{v}_m + \epsilon_{nm}$ where the noise/residual

$$\epsilon_{nm} \sim \mathcal{N}(0, \sigma^2)$$

Probabilistic Matrix Factorization

- Our basic model

$$\begin{aligned} \boldsymbol{u}_n &\sim \mathcal{N}(\boldsymbol{u}_n|\mathbf{0}, \lambda_U^{-1}\mathbf{I}_K), & n = 1,\ldots,N \\ \boldsymbol{v}_m &\sim \mathcal{N}(\boldsymbol{v}_n|\mathbf{0}, \lambda_V^{-1}\mathbf{I}_K), & m = 1,\ldots,M \\ X_{nm} &\sim \mathcal{N}(X_{nm}|\boldsymbol{u}_n^\top\boldsymbol{v}_m, \sigma^2), & \forall (n,m) \in \Omega \end{aligned}$$

- Note: Many variations possible, e.g., adding row/column biases (a_n, b_m), rows/column features ($\mathbf{X}^U$, $\mathbf{X}^V$); will not consider those here

$$X_{nm} = \mathcal{N}(X_{nm}|\boldsymbol{u}_n^\top\boldsymbol{v}_m + a_n + b_m + \beta_U^\top\boldsymbol{x}_n^U + +\beta_V^\top\boldsymbol{x}_m^V, \sigma^2)$$

- Note: Gaussian assumption on X_{nm} may not be appropriate if data is not real-valued, e.g., is binary/counts/ordinal (but it still works well nevertheless)
- Likewise, if we want to impose specific constraints on the latent factors (e.g., non-negativity, sparsity, etc.) then Gaussians on $\boldsymbol{u}_n$, $\boldsymbol{v}_m$ are not appropriate
- Here, we will only focus on the Gaussian case (leads to a simple algorithm)

Parameter Estimation via MAP

- Let's do MAP estimation (recall, we have priors on the latent factors)
- Log-posterior $\log p(\mathbf{X}_\Omega, \mathbf{U}, \mathbf{V}) = \log p(\mathbf{X}_\Omega|\mathbf{U}, \mathbf{V})p(\mathbf{U})p(\mathbf{V})$ is given by

$$\begin{aligned} \mathcal{L} &= \log p(\mathbf{X}_\Omega|\mathbf{U}, \mathbf{V}) + \log p(\mathbf{U}) + \log p(\mathbf{V}) \\ &= \log \prod_{(n,m)\in\Omega} p(X_{nm}|\boldsymbol{u}_n, \boldsymbol{v}_m) + \log \prod_{n=1}^{N} p(\boldsymbol{u}_n) + \log \prod_{m=1}^{M} p(\boldsymbol{v}_m) \end{aligned}$$

- With Gaussian likelihood and priors, ignoring the constants, we have

$$\mathcal{L} = \sum_{(n,m)\in\Omega} -\frac{1}{2\sigma^2}(X_{nm} - \boldsymbol{u}_n^\top\boldsymbol{v}_m)^2 - \sum_{n=1}^{N}\frac{\lambda_U}{2}||\boldsymbol{u}_n||^2 - \sum_{m=1}^{M}\frac{\lambda_V}{2}||\boldsymbol{v}_m||^2$$

- Can solve for row and column latent factors $\boldsymbol{u}_n, \boldsymbol{v}_m$ in an alternating fashion

Solving for Row Latent Factors

- The (negative) log-posterior

$$\mathcal{L} = \sum_{(n,m)\in\Omega} \frac{1}{2\sigma^2}(X_{nm} - \boldsymbol{u}_n^\top \boldsymbol{v}_m)^2 + \sum_{n=1}^{N} \frac{\lambda_U}{2}||\boldsymbol{u}_n||^2 + \sum_{m=1}^{M} \frac{\lambda_V}{2}||\boldsymbol{v}_m||^2$$

- For row latent factors $\boldsymbol{u}_n$ (with all column factors fixed), the objective will be

$$\mathcal{L}_{\boldsymbol{u}_n} = \sum_{m\in\Omega_{\boldsymbol{u}_n}} \frac{1}{2\sigma^2}(X_{nm} - \boldsymbol{u}_n^\top \boldsymbol{v}_m)^2 + \frac{\lambda_U}{2}\boldsymbol{u}_n^\top \boldsymbol{u}_n$$

- Taking derivative w.r.t. $\boldsymbol{u}_n$ and setting to zero, we get

$$\boldsymbol{u}_n = \left(\sum_{m\in\Omega_{\boldsymbol{u}_n}} \boldsymbol{v}_m \boldsymbol{v}_m^\top + \lambda_U \sigma^2 \mathbf{I}_K\right)^{-1} \left(\sum_{m\in\Omega_{\boldsymbol{u}_n}} X_{nm}\boldsymbol{v}_m\right)$$

- Note: with $\mathbf{V}$ fixed, we can solve for all $\boldsymbol{u}_n$ $(n = 1, \ldots, N)$ in parallel

Solving for Column Latent Factors

- The (negative) log-posterior

$$\mathcal{L} = \sum_{(n,m)\in\Omega} \frac{1}{2\sigma^2}(X_{nm} - \boldsymbol{u}_n^\top \boldsymbol{v}_m)^2 + \sum_{n=1}^{N} \frac{\lambda_U}{2}||\boldsymbol{u}_n||^2 + \sum_{m=1}^{M} \frac{\lambda_V}{2}||\boldsymbol{v}_m||^2$$

- For column latent factors $\boldsymbol{v}_m$ (with all row factors fixed), the objective will be

$$\mathcal{L}_{\boldsymbol{v}_m} = \sum_{n\in\Omega_{\boldsymbol{v}_m}} \frac{1}{2\sigma^2}(X_{nm} - \boldsymbol{u}_n^\top \boldsymbol{v}_m)^2 + \frac{\lambda_V}{2}\boldsymbol{v}_m^\top \boldsymbol{v}_m$$

- Taking derivative w.r.t. $\boldsymbol{v}_m$ and setting to zero, we get

$$\boldsymbol{v}_m = \left(\sum_{n\in\Omega_{\boldsymbol{v}_m}} \boldsymbol{u}_n \boldsymbol{u}_n^\top + \lambda_V \sigma^2 \mathbf{I}_K\right)^{-1} \left(\sum_{n\in\Omega_{\boldsymbol{u}_m}} X_{nm}\boldsymbol{u}_n\right)$$

- Note: with $\mathbf{U}$ fixed, we can solve for all $\boldsymbol{v}_m$ $(m = 1, \ldots, M)$ in parallel

The Complete Algorithm

- Input: Partially complete matrix $\mathbf{X}_\Omega$
- Initialize the column latent factors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_M$ randomly, e.g., from the prior, i.e., $\boldsymbol{v}_n \sim \mathcal{N}(0, \lambda_V^{-1}\mathbf{I}_K)$
- Iterate until converge
  - Update each row latent factor $\boldsymbol{u}_n$, $n = 1, \ldots, N$ (can be in parallel)

$$\boldsymbol{u}_n = \left(\sum_{m\in\Omega_{\boldsymbol{u}_n}} \boldsymbol{v}_m \boldsymbol{v}_m^\top + \lambda_U \sigma^2 \mathbf{I}_K\right)^{-1} \left(\sum_{m\in\Omega_{\boldsymbol{u}_n}} X_{nm}\boldsymbol{v}_m\right)$$

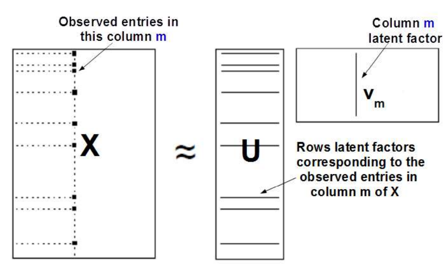
  - Update each column latent factor $\boldsymbol{v}_m$, $m = 1, \ldots, M$ (can be in parallel)

$$\boldsymbol{v}_m = \left(\sum_{n\in\Omega_{\boldsymbol{v}_m}} \boldsymbol{u}_n \boldsymbol{u}_n^\top + \lambda_V \sigma^2 \mathbf{I}_K\right)^{-1} \left(\sum_{n\in\Omega_{\boldsymbol{u}_m}} X_{nm}\boldsymbol{u}_n\right)$$

- Final prediction for any entry: $X_{nm} = \boldsymbol{u}_n^\top \boldsymbol{v}_m$

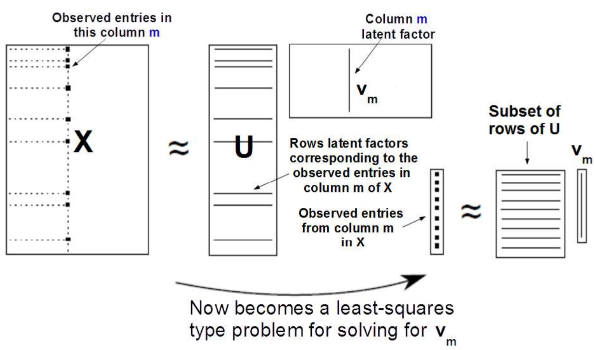
Matrix Factorization as Linear Regression

Suppose we are solving for the column latent factor $\boldsymbol{v}_m$ (with $\mathbf{U}$ fixed)



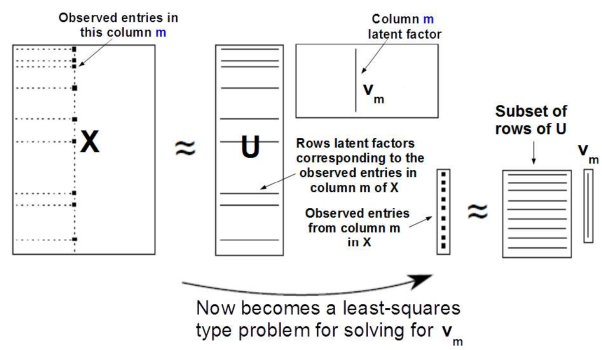
Matrix Factorization as Linear Regression

Suppose we are solving for the column latent factor $\boldsymbol{v}_m$ (with $\mathbf{U}$ fixed)



Matrix Factorization as Linear Regression

Suppose we are solving for the column latent factor $\boldsymbol{v}_m$ (with $\mathbf{U}$ fixed)



Likewise, solving for each row latent factor $\boldsymbol{u}_n$ is a least-squares regression problem

Matrix Factorization as Linear Regression

- A very useful way to think about matrix factorization
- Can modify the regularized least-squares like objective

$$\arg\min_{\boldsymbol{u}_n} \sum_{m \in \Omega_{\boldsymbol{u}_n}} \frac{1}{2\sigma^2}(X_{nm} - \boldsymbol{u}_n^\top \boldsymbol{v}_m)^2 + \frac{\lambda_U}{2}\boldsymbol{u}_n^\top \boldsymbol{u}_n$$

  .. and replace it by other loss functions and regularizers
- Can easily extend the model in various ways, e.g.
  - Handle other types of entries in the matrix $\mathbf{X}$, e.g., binary, counts, etc. (by changing the loss function or the likelihood function term)
  - Impose constraints on the latent factors (by changing the regularizer or prior on latent factors)