Foundations of Data Science

Lecture 6: Matrix Decomposition

MING GAO

DaSE@ECNU
(for course related communications)
mgao@sei.ecnu.edu.cn

Oct. 9, 2016

Outline

Dimensionality reduction

Motivation

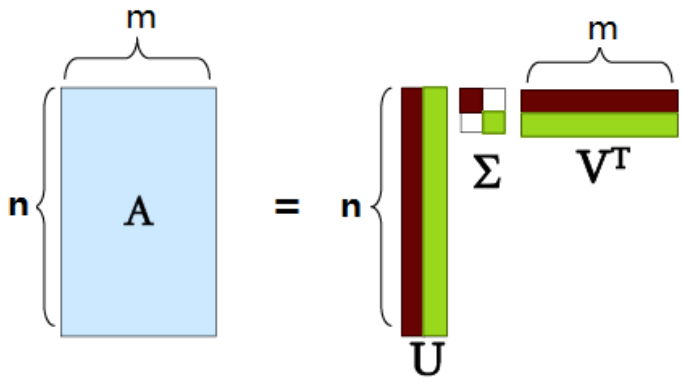
- High-dimension means many features.
  - Netflix: 480K users and 177K movies
  - Documents VS. words: thousands of words and billions of documents
  - Taobao: millions of users and millions of products
- Dimensionality reduction or compression
  - Discover hidden correlations, concepts or topics due to objects that occur commonly together.
  - Remove redundant and noisy features, not all features are useful
  - Easier storage and processing of the data

Outline

SVD

Definition

Any real $m \times n$ matrix A can be decomposed uniquely as $A_{[n\times m]} \sim U_{[n\times r]} D_{[r\times r]} V^T_{r\times m}$ where U, V are orthogonal matrix, D is a diagonal matrix (non-negative real values called singular values).

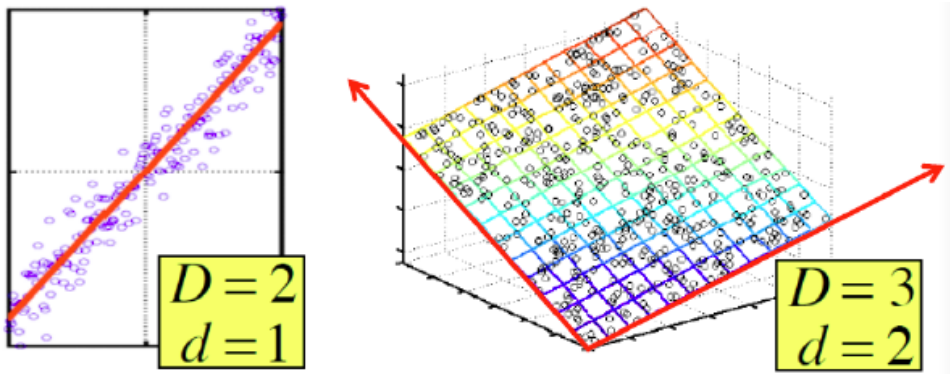


- A: an input $n \times m$ data matrix, e.g., n users and m items.
- U: left singular vectors in a $n \times r$ matrix, e.g., n users and r interests.
- D: a $r \times r$ diagonal matrix, e.g., strength of each interest.
- V: right singular vectors in a $r \times m$ matrix, e.g., r interests and m

SVD cont.

Properties

- It is always possible to decompose a real matrix A into $A = UDV^T$
  - U, D, V are unique, and D is a diagonal matrix.
  - U, V are column orthogonal (i.e., $U^T U = I$ and $V^T V = I$)
  - Entries of D are positive and sorted in decreasing order $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$.
- Axes of this subspace are effective representation of the data.



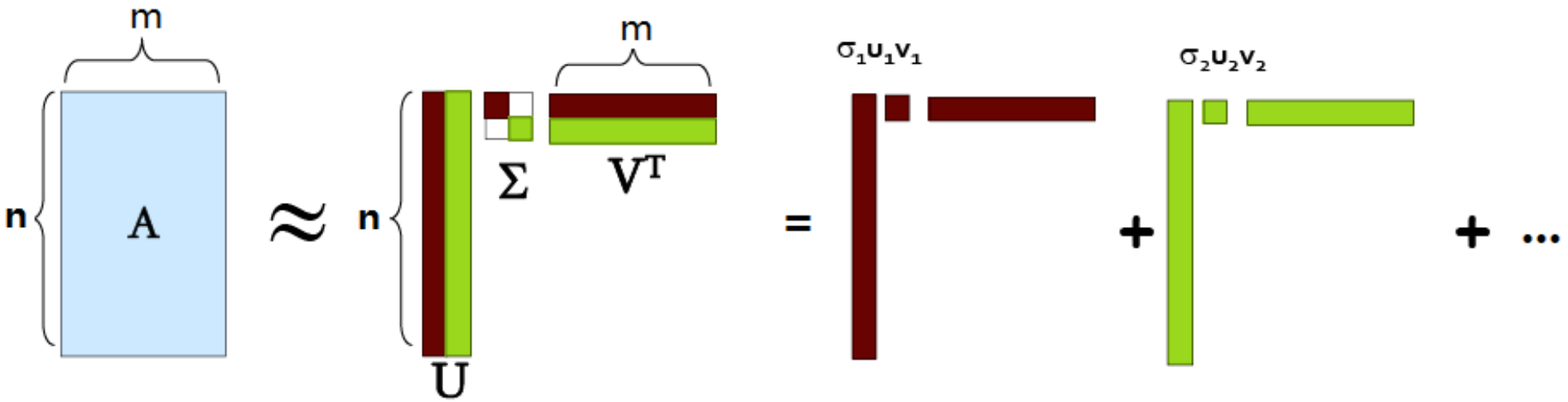
Assumptions

- Data lies on or near a low $d-$dimensional subspace.
- Axes of this subspace are effective representation of the data.

Methodology of decomposition

Diagonalization

- $AA^T = UDV^T VDU^T = UD^2U^T$, where $D = diag(\sigma_1^2, \sigma_2^2, \cdots, \sigma_n^2)$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$.
- $A^T A = VDU^T UDV^T = VD^2V^T$
- $A = UDV^T$. If $U = (\mathbf{u}_1\ \mathbf{u}_2\ \cdots \mathbf{u}_n)$ and $U = (\mathbf{v}_1\ \mathbf{v}_2\ \cdots \mathbf{v}_n)$, then $A = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$.



Methodology of dimensionality reduction

$$A = \begin{bmatrix} u_1 & \cdots & u_k & u_{k+1} & \cdots & u_m \end{bmatrix} \left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma_k & \\ \hline & 0 & & 0 \end{array}\right] \left[\begin{array}{c} v_1^T \\ \vdots \\ v_k^T \\ \hline v_{k+1}^T \\ \vdots \\ v_n^T \end{array}\right]$$

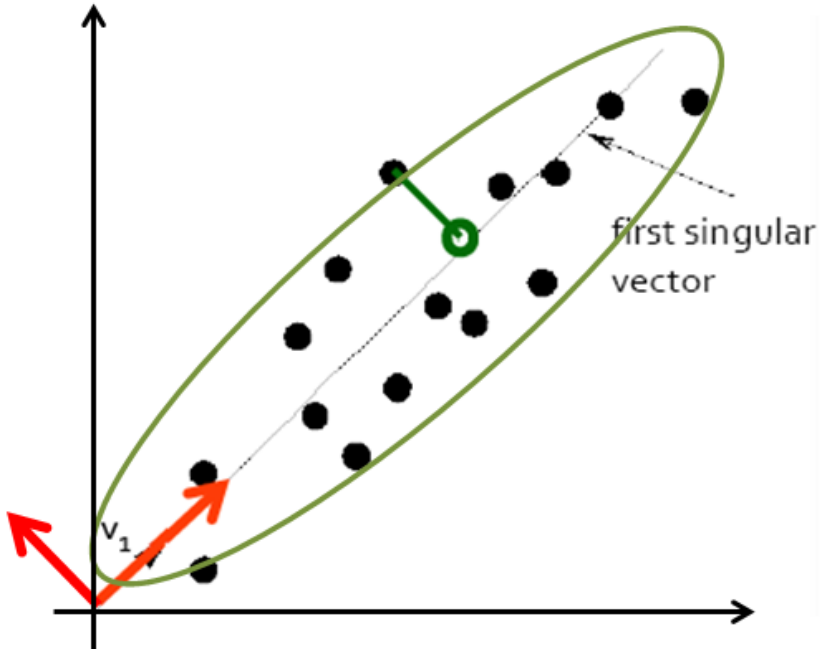
$$A = \begin{bmatrix} u_1 & \cdots & u_k \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_k^T \end{bmatrix}$$

Criteria

- $k = \arg\min_r \left\{ \frac{\sum_{i=1}^r \sigma_i}{\sum_{i=1}^n \sigma_i} > 90\% \right\}$, $A \approx \sum_{i=1}^k \sigma_i \mathbf{u}_i \mathbf{v}_i^T$
- For example

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \approx \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

SVD interpretation



Explanation

- SVD gives best axis to project entities, where "best" means to minimize sum of squares of projection errors.
- SVD also gives the minimum reconstruction errors.

Applications of SVD

Applications

- A square matrix A is nonsingular (i.e., $\sigma_i \neq 0$ for all i)
  - If A is a nonsingular matrix, then its inverse is given by $A^{-1} = V^T D^{-1} U$.
  - If A is singular or ill-conditioned, then we can use SVD to approximate its inverse by the following matrix: $A^{-1} = (UDV^T)^{-1} \approx VD_0^{-1}U^T$, where t is a small threshold and $D_0^{-1} = \begin{cases} \frac{1}{\sigma_i}, & \text{if } \sigma_i > t; \\ 0, & \text{otherwise.} \end{cases}$
- Consider linear system $Ax = b$, where $A \in \mathbb{R}^{n \times m}$. If $A^T A$ is ill-conditioned (small changes in b can lead to relatively large changes in the solution x) or singular, $x \approx VD_0^{-1}U^T b$.
- SVD can be helpful to similarity query or join.
- Data compression and anomaly detection.

Outline

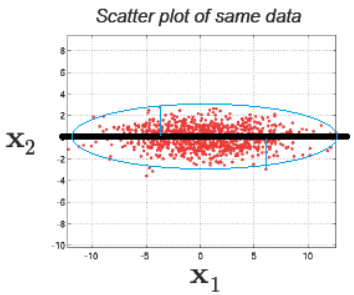
PCA: an important application of SVD

Motivation

PCA: Principle Component Analysis

- Problems arise when performing data mining or machine learning in a high-dimensional space (e.g., curse of dimensionality).
- Significant improvements can be achieved by first mapping the data into a lower-dimensionality space.
- Preserve as much information as possible

$$x = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_N \end{bmatrix} --> reduce\ dimensionality --> y = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_K \end{bmatrix} \quad (K << N)$$



Goals

- Find a good representation for features (what?)
- Reduce redundancy in the data (how?)

PCA cont.

Criteria of good representation for features

- Minimize relation of the different dimensions.
- Keep the dimension as low as possible.

The best low-dimensional space

- It also gives best axis to project data, where "best" means to minimize sum of squares of projection errors.
- It also gives the minimum reconstruction errors.
- It can be determined by the "best" eigenvectors of the covariance matrix of x (i.e., the eigenvectors corresponding to the "largest" eigenvalues, also called "principal components").

Methodology

Methodology

Suppose $x_1, x_2, \cdots, x_n$ are $d \times 1$ vectors, then

1. $\overline{x} = \sum_{i=1}^{n} x_i$.
2. Subtract the mean: $y_i = x_i - \overline{x}$.
3. Form the matrix $A = [y_1\ y_2\ \cdots y_n]$ ($d \times n$ matrix), then compute

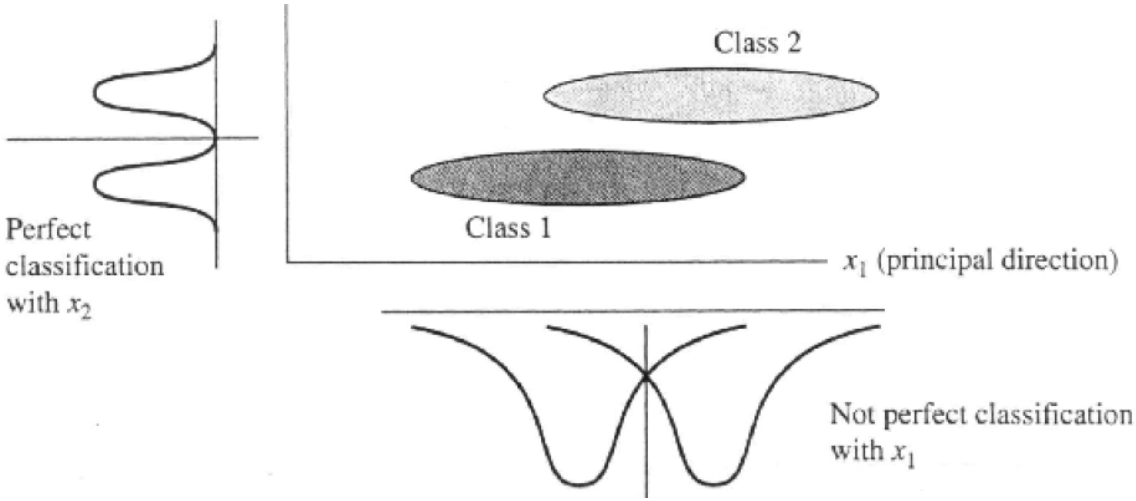
$$C = \frac{1}{n}\sum_{i=1}^{n} y_i y_i^T = AA^T (d \times d \text{matrix})$$

4. Compute the eigenvalues of $C : \lambda_1 > \lambda_2 > \cdots > \lambda_d$, and corresponding eigenvectors $u_1, u_2, \cdots, u_d$.
5. Keep only the terms corresponding to the k largest eigenvalues: $\widehat{x} - \overline{x} = \sum_{i=1}^{k} b_i u_i$, i.e., the representation of $\widehat{x} - \overline{x}$ into the basis $u_1, u_2, \cdots, u_k$.

Information loss

Analysis

- $\widehat{x} - \overline{x} = \sum_{i=1}^{k} b_i u_i$, i.e., $\widehat{x} = \sum_{i=1}^{k} b_i u_i + \overline{x}$
- It can be shown that the low-dimensional basis based on principal components minimizes the reconstruction error: $e = \|x - \widehat{x}\|$
- It can be shown that the error is equal to $e = \sum_{i=k+1}^{d} \lambda_i$.
- PCA is not always an optimal dimensionality-reduction procedure, e.g., classification problem.



SVD: Pros & Cons

Pros

- Optimal low-rank approximation in L_2 norm.
- There are many implementations, such as LINPACK, Matlab, SPlus, Mathematica...

Cons

- Conventional SVD is undefined for incomplete matrices.
- The complexity of computing SVD is $O(nm^2)$ or $O(n^2m)$. Less work if we want first k singular vecotrs or matrix is sparse.
- We need an approach that can simply ignore missing values and reduce the complexity.

Matrix factorization

Motivation

How can one determine the factor matrices P and Q, so that the fully specified matrix R matches PQ^T as closely as possible?

Solution

Formulate an optimization problem as:

$$\text{Minimize } J = \frac{1}{2}\|R - PQ^T\|^2$$
$$s.t. \text{ No constraints on } P \text{ and } Q$$

where $\|\cdot\|^2$ represents the squared Frobenius norm of the matrix.

- Thus, the objective function is equal to the sum of the squares of the entries in the residual matrix $R - PQ^T$.
- This objective function can be viewed as a quadratic loss function, which quantifies the loss of accuracy in estimating the matrix R with the use of low-rank factorization.

Matrix factorization

Definition

Given a set of users U, and a set of items D, let $R \in \mathbb{R}^{|U| \times |D|}$ be the rating matrix. The matrix factorization is to find two matrices $P \in \mathbb{R}^{|U| \times K}$ and $Q \in \mathbb{R}^{|D| \times K}$ such that $R \approx PQ^T = \widehat{R}$,where K denotes the dimensionality of latent features.

- Each row of P would represent the strength of the associations between a user and the features.
- Each row of Q would represent the strength of the associations between an item and the features.
- Now, we have to find a way to obtain P and Q.

Problem formulation

Formal definition

$$J = \min_{q^*, p^*} \frac{1}{2} \sum_{(u,i) \in \mathcal{K}} (r_{ui} - p_u^T q_i)^2,$$

where r_{ui} is the known rating of user u for item i, $\widehat{u}_{ui} = p_u^T q_i$ is the predicted rating given by user u for item i, and the set of all user-item pair (u, i), which are observed in R, be denoted by $\mathcal{K}$, i.e., $\mathcal{K} = \{(u, i) | r_{ui} \text{ is observed}\}$.

- The error between the predicted rating and the real rating, can be calculated by the following equation for each user-item pair: $e_{ui}^2 = (r_{ui} - \widehat{r}_{ui})^2 = (r_{ui} - \sum_{k=1}^{d} p_{uk} q_{ki})^2$.
- Now, we have to find a way to obtain P and Q.

Outline

Batch gradient descent algorithm

Estimate parameters iteratively

Let's start at $P^{(0)}$ and $Q^{(0)}$.

- $\frac{\partial}{\partial p_{uk}} e_{ui}^2 = -(r_{ui} - \widehat{r}_{ui}) q_{ki}$ and $\frac{\partial}{\partial q_{ki}} e_{ui}^2 = -(r_{ui} - \widehat{r}_{ui}) p_{uk}$.
- Update rules:
  - $p_{uk}^{(t+1)} \leftarrow p_{uk}^{(t)} + \alpha \sum_{i:(u,i)\in\mathcal{K}} e_{ui}^{(t)} q_{ki}^{(t)}$.
  - $q_{ki}^{(t+1)} \leftarrow q_{ki}^{(t)} + \alpha \sum_{u:(u,i)\in\mathcal{K}} e_{ui}^{(t)} p_{uk}^{(t)}$.
  - Where $e_{ui}^{(t)} = r_{ui} - {p_u^{(t)}}^T q_i^{(t)}$
- Subsequently, the updates can be computed as follows:
  - $P^{(t+1)} \leftarrow P^{(t)} + \alpha E^{(t)} Q^{(t)}$.
  - $Q^{(t+1)} \leftarrow Q^{(t)} + \alpha {E^{(t)}}^T P^{(t)}$.

Stochastic gradient descent algorithm

SGDA decomposes the errors into smaller components associated with the errors in individual observed entries rather than all entries.

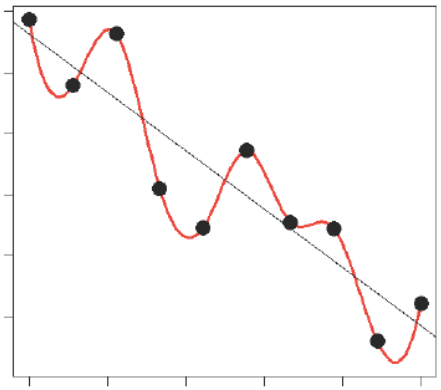
Different strategy to update parameters

Let's start at $P^{(0)}$ and $Q^{(0)}$.

- Recall that $\frac{\partial}{\partial p_{uk}} e_{ui}^2 = e_{ui} q_{ki}$ and $\frac{\partial}{\partial q_{ki}} e_{ui}^2 = e_{ui} p_{uk}$.
- Update rules:
  - $p_{uk}^{(t+1)} \leftarrow p_{uk}^{(t)} + \alpha e_{ui}^{(t)} q_{ki}^{(t)}$ for $\forall u \in \{1, 2, \cdots, K\}$.
  - $q_{ki}^{(t+1)} \leftarrow q_{ki}^{(t)} + \alpha e_{ui}^{(t)} p_{uk}^{(t)}$ for $\forall i \in \{1, 2, \cdots, K\}$.
  - Where $e_{ui}^{(t)} = r_{ui} - {p_u^{(t)}}^T q_i^{(t)}$
- In practice, faster convergence is achieved by the stochastic gradient descent method as compared to the batch method ($2K$ VS. $mK + nK$ for each iteration).
- Stochastic gradient descent is preferable when the data size is very large and computational time is the primary bottleneck.

Outline

Matrix factorization: regularization



Problem

- The observed set $\mathcal{K}$ of ratings is small, which can cause overfitting (also common in classification problem).
- Regularization is a common approach to address the problem.

Regularization

$$\min_{q^*,p^*} J = \frac{1}{2}\Big[\sum_{(u,i)\in\mathcal{K}} (r_{ui} - q_i^T p_u)^2 + \lambda(\|Q\|^2 + \|P\|^2)\Big],$$

- $\frac{\partial}{\partial p_{uk}} J = -\sum_{i:(u,i)\in\mathcal{K}} (r_{ui} - \hat{r}_{ui}) q_{ki} + \lambda p_{uk}$
- $\frac{\partial}{\partial q_{ki}} J = -\sum_{i:(u,i)\in\mathcal{K}} (r_{ui} - \hat{r}_{ui}) p_{uk} + \lambda q_{ki}.$

Gradient descent algorithm for regularization

Batch gradient descent

- $p_{uk}^{(t+1)} \leftarrow p_{uk}^{(t)} + \alpha(\sum_{i:(u,i)\in\mathcal{K}} e_{ui}^{(t)} q_{ki}^{(t)} - \lambda p_{uk}^{(t)})$.
- $q_{ki}^{(t+1)} \leftarrow q_{ki}^{(t)} + \alpha(\sum_{u:(u,i)\in\mathcal{K}} e_{ui}^{(t)} p_{uk}^{(t)} - \lambda q_{ki}^{(t)})$.
- Where $e_{ui}^{(t)} = r_{ui} - {p_u^{(t)}}^T q_i^{(t)}$

Stochastic gradient descent

- $p_{uk}^{(t+1)} \leftarrow p_{uk}^{(t)} + \alpha(e_{ui}^{(t)} q_{ki}^{(t)} - \lambda p_{uk}^{(t)})$.
- $q_{ki}^{(t+1)} \leftarrow q_{ki}^{(t)} + \alpha(e_{ui}^{(t)} p_{uk}^{(t)} - \lambda q_{ki}^{(t)})$.
- Where $e_{ui}^{(t)} = r_{ui} - {p_u^{(t)}}^T q_i^{(t)}$

Incorporating user and item biases

Loss function

$$J = \frac{1}{2}\Big[\sum_{(u,i)\in\mathcal{K}} (r_{ui} - b_i - b_u - q_i^T p_u)^2 + \lambda(\|Q\|^2 + \|P\|^2 + \|b_u\|^2 + \|b_i\|^2)\Big]$$

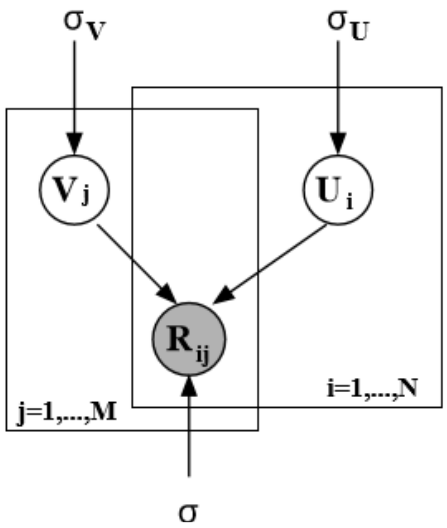
- Instead of having separate bias variables b_u and b_i for users and items, we can increase the size of the factor matrices to incorporate these bias variables.
  - $p_{u(k+1)} = b_u$ and $p_{u(k+2)} = 1, \forall u \in \{1, 2, \cdots, n\}$
  - $q_{i(k+1)} = 1$ and $p_{i(k+2)} = b_i, \forall i \in \{1, 2, \cdots, m\}$
-

$$\min_{q^*,p^*} J = \frac{1}{2}\Big[\sum_{(u,i)\in\mathcal{K}} (r_{ui} - \widetilde{q}_i^T \widetilde{p}_u)^2 + \lambda(\|\widetilde{Q}\|^2 + \|\widetilde{P}\|^2)\Big]$$

$$s.t. (k+2)\text{th column of } \widetilde{P} \text{ contains only 1s}$$

$$(k+1)\text{th column of } \widetilde{Q} \text{ contains only 1s}$$

Probabilistic matrix factorization



Assumption

$$P(r_{ij}|u_i, v_j, \sigma^2) = \left[\mathcal{N}(r_{ij}|u_i^T v_j, \sigma^2)\right]^{I_{ij}}$$

$$P(U|\sigma_U) = \Pi_{i=1}^{N}\mathcal{N}(u_i|0, \sigma_U^2 I), U \in \mathbb{R}^{D\times N}$$

$$P(V|\sigma_V) = \Pi_{j=1}^{M}\mathcal{N}(v_j|0, \sigma_V^2 I), V \in \mathbb{R}^{D\times M}$$

Joint probability

$$P(U, V, R|\sigma^2, \sigma_U^2, \sigma_V^2) =$$

$$\Pi_{i=1}^{n}\Pi_{j=1}^{m}\left[P(r_{ij}|u_i, v_j, \sigma^2)\right]^{I_{ij}}\Pi_{i=1}^{n}P(u_i|\sigma_U^2 I)\Pi_{j=1}^{m}P(v_j|\sigma_V^2 I)$$

Learning parameters

The log of the posterior distribution

$$\ln P(U, V, R|\sigma^2, \sigma_U^2, \sigma_V^2)$$

$$= -\frac{1}{2\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(r_{ij} - u_i^T v_j)^2 - \frac{1}{2\sigma_U^2}\sum_{i=1}^{N} u_i^T u_i - \frac{1}{2\sigma_V^2}\sum_{j=1}^{M} v_j^T v_j$$

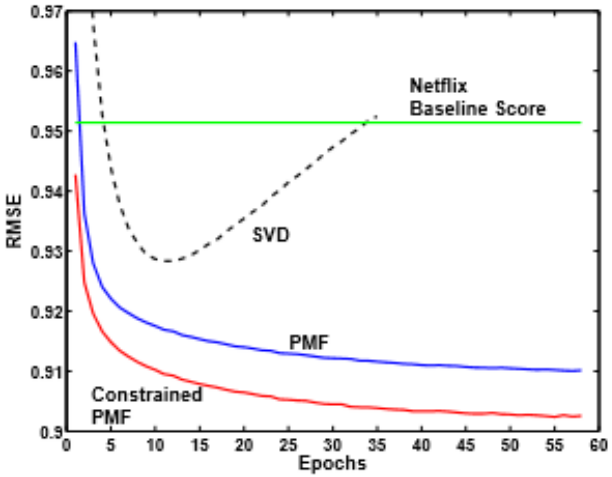
$$- \frac{1}{2}\left((\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij})\ln\sigma^2 + ND\ln\sigma_U^2 + MD\ln\sigma_V^2\right) + C$$

- $$J = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(r_{ij} - u_i^T v_j)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N}\|u_i\|^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M}\|v_j\|^2$$

$$= \frac{1}{2}\sum_{(i,j)\in\mathcal{K}}(r_{ij} - u_i^T v_j)^2 + \frac{\lambda_U}{2}\|U\|^2 + \frac{\lambda_V}{2}\|V\|^2$$

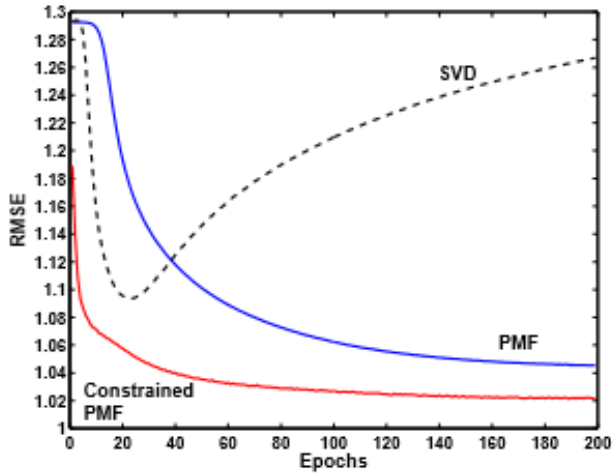
where $\lambda_U = \frac{\sigma_U^2}{\sigma^2}$ and $\lambda_V = \frac{\sigma_V^2}{\sigma^2}$.

Applications



Experiment setup

It illustrates the results on the Netflix dataset with 30-dimen. feature vectors.



Experiment setup

It illustrates the results on the Toy dataset with 30-dimen. feature vectors.

Non-negative matrix factorization

Formulation

Define an optimization problem as:

$$\text{Minimize } J = \frac{1}{2}\|R - UV^T\|^2$$
$$s.t. U \geq 0$$
$$V \geq 0$$

- Users specify a "like" for an item, but no mechanism to specify a "dislike", such as browsing or buy behaviors, Web-pages clicking, and Facebook liking, etc.
- Many attributes of entities are non-negative.
  - Pixels of an image
  - Frequencies of words in a document
  - Prices of stocks

Solution for NMF

Iterative algorithm

- $u_{ij}^{(t+1)} \leftarrow \frac{(RV^{(t)})_{ij} u_{ij}^{(t)}}{(U^{(t)} V^{(t)\,T} V^{(t)})_{ij}+\epsilon}$
- $v_{ij}^{(t+1)} \leftarrow \frac{(R^T U^{(t)})_{ij} v_{ij}^{(t)}}{(V^{(t)} U^{(t)\,T} U^{(t)})_{ij}+\epsilon}$, where ϵ is a small term, such as $\epsilon \approx 10^{-9}$.

Regularization

As in the case of other types of matrix factorization, regularization can be used to improve the quality of the underlying solution.

- The basic idea is to add the penalties $\frac{\lambda_1 \|U\|^2}{2} + \frac{\lambda_2 \|V\|^2}{2}$ to the objective function.
- The update rules:
  - $u_{ij}^{(t+1)} \leftarrow \max\left\{ \left\lceil \frac{(RV^{(t)})_{ij} - \lambda_1 u_{ij}^{(t)}}{(U^{(t)} V^{(t)\,T} V^{(t)})_{ij}+\epsilon} \right\rceil u_{ij}^{(t)}, 0 \right\}$
  - $v_{ij}^{(t+1)} \leftarrow \max\left\{ \left\lceil \frac{(R^T U^{(t)})_{ij} - \lambda_2 v_{ij}^{(t)}}{(V^{(t)} U^{(t)\,T} U^{(t)})_{ij}+\epsilon} \right\rceil v_{ij}^{(t)}, 0 \right\}$

Take-home messages

- Singular value decomposition
  - SVD
  - PCA
- Matrix factorization
  - Gradient descent algorithm
  - Regularization
- Probabilistic matrix factorization
- Non-negative matrix factorization