Matrix Factorization

Ziping Zhao

School of Information Science and Technology
ShanghaiTech University, Shanghai, China

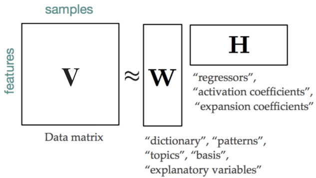
CS182: Introduction to Machine Learning (Spring 2023)
http://cs182.sist.shanghaitech.edu.cn

Outline

Outline

Introduction

- Matrix factorization (MF) (or matrix decomposition) refers to techniques that approximate a matrix by the product of two or more (smaller) matrices.
- PCA as a dimensionality reduction technique may also be seen as a matrix factorization method that is subject to an orthogonality constraint.



- MF is a generalization of many methods (e.g., PCA, SVD, QR, CUR, Truncated SVD, etc.)
- Depending on the applications, other constraints may also be enforced to give different matrix factorization methods.
- Two methods considered here:
  - Non-negative matrix factorization (NMF)
  - Probabilistic matrix factorization (PMF)

Outline

Non-negative Matrix Factorization

- Non-negative matrix factorization (NMF) was first proposed as a method for learning parts-based representations in visual perception tasks, as opposed to other methods such as vector quantization (VQ) and PCA which learn holistic, not parts-based representations.
- NMF algorithms are very useful in a wide variety of machine learning applications as a dimensionality reduction or feature extraction method.
- It enforces non-negativity constraints on the factor matrices, i.e., all entries in the factor matrices are non-negative.
- The non-negativity constraints make the method biologically more plausible, e.g., the firing rates of biological neurons and the synaptic strengths between neurons are non-negative.

Matrix Factorization for Representing Facial Images

- A database consists of m facial images.
- Each image consists of n nonnegative pixels.
- The whole database can be represented by a matrix $\mathbf{V}$ of size $n \times m$.
- $\mathbf{V}$ is approximated by two factor matrices $\mathbf{W}$ (basis elements) and $\mathbf{H}$ (encodings) of sizes $n \times r$ and $r \times m$, respectively:

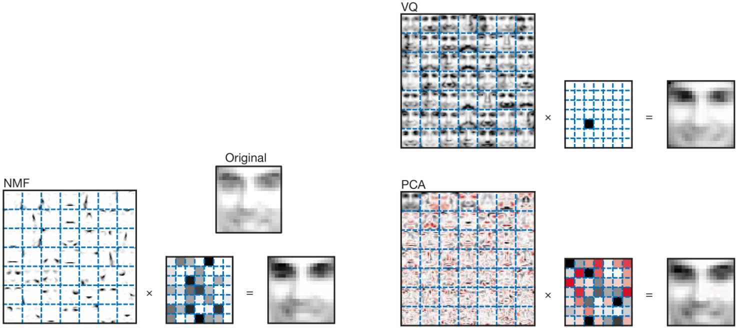
$$\mathbf{V} \approx \mathbf{WH}$$

where the rank r of the factorization is generally chosen such that

$$(n + m)r < nm$$

to give a compressed form of the data in $\mathbf{V}$.

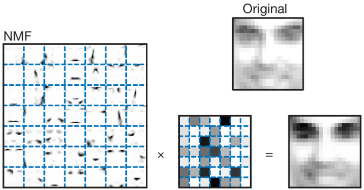
NMF vs. VQ and PCA I



- Each method has learned r $(= 49 = 7 \times 7)$ basis images.
- Each facial image is approximately represented by a linear combination of the r basis images.
- Positive values are shown in black and negative values in red.

NMF vs. VQ and PCA II

- Each of the basis images learned by VQ is a whole face.
- Each of the basis images learned by PCA is called an eigenface. Some eigenfaces resemble distorted versions of whole faces.
- Unlike the holistic basis images learned by VQ and PCA, those learned by NMF are localized features that correspond to parts of faces, i.e., parts-based representations, such as the nose, mouth, moustaches, and eyes of a face.



NMF vs. VQ and PCA III

- The combinations in NMF can only be additive due to the non-negative constraints on the matrices.
- This offers an intuitive explanation as to why NMF learns a parts-based representation (in contrast to the holistic representations of faces resulting from PCA and VQ), as parts of the face are additively combined to create a whole.
- It is also important to note that the NMF basis images and encodings contain several vanishing coefficients, meaning that the basis images and encodings are sparse, which is crucial for parts-based representation.

NMF Problem Formulation

- Given a non-negative matrix $\mathbf{V}$, find non-negative matrices $\mathbf{W}$ and $\mathbf{H}$ such that

$$\mathbf{V} \approx \mathbf{WH}$$

- Each column vector $\mathbf{v}$ of $\mathbf{V}$ is approximated by the corresponding column vector $\mathbf{h}$ of $\mathbf{H}$:

$$\mathbf{v} \approx \mathbf{Wh}$$

  I.e., each vector $\mathbf{v}$, corresponding to one facial image, is approximated by a linear combination of the columns of $\mathbf{W}$ where the components of $\mathbf{h}$ are the weights of the linear combination.

Cost Functions for Optimization Problem

- The optimization problem is to minimize the difference between $\mathbf{V}$ and $\mathbf{WH}$ which is quantified by some cost function.
- Two choices:
  - Euclidean distance between $\mathbf{A}$ and $\mathbf{B}$ (as Frobenius norm of $\mathbf{A} - \mathbf{B}$):

$$\|\mathbf{A} - \mathbf{B}\|_F^2 = \sum_{ij} (A_{ij} - B_{ij})^2$$

  - Variant of Kullback-Leibler (KL) divergence of $\mathbf{A}$ with respect to $\mathbf{B}$:

$$D(\mathbf{A} \parallel \mathbf{B}) = \sum_{ij} \left(A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij} \right)$$

    which reduces to the KL divergence when $\sum_{ij} A_{ij} = \sum_{ij} B_{ij} = 1$. This objective can be derived by treating NMF like a probabilistic generative model.
- Both cost functions have a lower bound of zero and are equal to zero if and only if $\mathbf{A} = \mathbf{B}$.

Optimization Problem I

- Optimization problem:

$$\begin{aligned} \underset{\mathbf{W},\mathbf{H}}{\text{minimize}} \quad & \|\mathbf{V} - \mathbf{WH}\|_F^2 \\ \text{subject to} \quad & \mathbf{W}, \mathbf{H} \geq \mathbf{0} \end{aligned} \qquad \text{or} \qquad \begin{aligned} \underset{\mathbf{W},\mathbf{H}}{\text{minimize}} \quad & D(\mathbf{V} \parallel \mathbf{WH}) \\ \text{subject to} \quad & \mathbf{W}, \mathbf{H} \geq \mathbf{0} \end{aligned}$$

- As opposed to the unconstrained problem which can be solved efficiently using the SVD, NMF is NP-hard in general.
- Issues:
  - It is not guaranteed to find a single unique decomposition (in general, there might be many schemes for defining sets of basis elements). It can be tackled using other priors on $\mathbf{W}$ and $\mathbf{H}$ and adding proper regularization terms in the objective function.
  - It is hard to know how to choose the factorization rank r.
- The objective functions are convex in $\mathbf{W}$ only or $\mathbf{H}$ only, but not convex in both variables together.

Optimization Problem II

- Gradient methods such as gradient descent and conjugate gradient may be applied to find local minima for the optimization problem, but their convergence is very sensitive to the choice of step size.
- Almost all NMF algorithms use a two-block coordinate descent scheme (exact or inexact), that is, they optimize alternatively over one of the two factors, $\mathbf{W}$ or $\mathbf{H}$, while keeping the other fixed, since the subproblem in one factor is convex.
- More precisely, it is a nonnegative least squares (NNLS) problem, a type of constrained least squares problems.
- Many algorithms exist to solve the NNLS problem; and NMF algorithms based on two-block coordinate descent differ by which NNLS algorithm is used.
- Some NNLS algorithms that can be used include multiplicative updates, alternating nonnegative least squares, hierarchical alternating least squares, etc.

Multiplicative Update Rules

- The Euclidean distance $\|\mathbf{V} - \mathbf{WH}\|_F^2$ is nonincreasing under the following multiplicative update rules:

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(\mathbf{W}^T\mathbf{V})_{a\mu}}{(\mathbf{W}^T\mathbf{WH})_{a\mu}} \qquad W_{ia} \leftarrow W_{ia} \frac{(\mathbf{VH}^T)_{ia}}{(\mathbf{WHH}^T)_{ia}}$$

  The Euclidean distance is invariant under these updates if and only if **W** and **H** are at a stationary point of the distance.

- The divergence $D(\mathbf{V} \parallel \mathbf{WH})$ is nonincreasing under the following multiplicative update rules:

$$H_{a\mu} \leftarrow H_{a\mu} \frac{\sum_i W_{ia} V_{i\mu}/(\mathbf{WH})_{i\mu}}{\sum_k W_{ka}} \qquad W_{ia} \leftarrow W_{ia} \frac{\sum_\mu H_{a\mu} V_{i\mu}/(\mathbf{WH})_{i\mu}}{\sum_v H_{av}}$$

  The divergence is invariant under these updates if and only if **W** and **H** are at a stationary point of the divergence.

Convergence Guarantee

- Each update involves a multiplicative factor which is equal to one (i.e., no update) when $\mathbf{V} = \mathbf{WH}$. Thus perfect reconstruction is necessarily a fixed point of the update rules.
- Convergence proofs for the two cost functions exist and are similar to that for the EM algorithm by making use of an auxiliary function.
- Using the multiplicative update rules, locally optimal solutions to the optimization problem are guaranteed.

Algorithm Initializations

- You can initialize $\mathbf{W}$ and $\mathbf{H}$ randomly, but there are also alternate strategies designed to give better initial estimates in the hope of converging more rapidly to a good solution:
  - Using some clustering method, and make the cluster means of the top r clusters as the columns of $\mathbf{W}$, and $\mathbf{H}$ as a scaling of the cluster indicator matrix (which elements belong to which cluster).
  - Finding the best rank-r approximation of $\mathbf{V}$ using SVD and using this to initialize $\mathbf{W}$ and $\mathbf{H}$.
  - Picking r columns of $\mathbf{V}$ and just using those as the initial values for $\mathbf{W}$.

More Applications Based on NMF

- The applications of NMF for feature extraction are not just limited to facial images in image processing.
- It can be used in text mining for semantic analysis on text from the Encyclopedia, which is called latent semantic indexing or latent semantic analysis.
  - $\mathbf{V}$ is a sample of m documents each using a bag of words representation with n words, each factor may be one topic or concept written using a certain subset of words and each document is a certain combination of such factors.
  - NMF identifies topics and simultaneously classifies the documents among these different topics.
- It can also be used for hyperspectral imaging, calcium imaging (of neurons), biological data mining, etc.

Outline

Recommendation Systems

- Recommendation systems (a.k.a. recommender systems) predict the preferences of users for items (e.g., books, movies) and make recommendations accordingly.
- Examples of recommendation systems:
  - Ordering news articles to online newspaper readers based on prediction of reader interests.
  - Ordering suggestions to customers of an online retailer about what they might like to buy based on their past history of purchases and/or product searches.
- Two main approaches:
  - Content-based systems recommend items based on properties of the items.
  - Collaborative filtering (CF) systems recommend items based on similarity measures between users and/or items.

Collaborative Filtering

- The goal of CF is to infer user preferences for items given a large but incomplete collection of preferences for many users.
- An illustrative example:
  - Suppose we infer from data that most users who like 'Star Wars' also like 'Lord of the Rings' and dislike 'Dune'.
  - Then, if a user watched and liked 'Star Wars', we would recommend him/her 'Lord of the Rings' but not 'Dune'.
- Preferences can be explicit or implicit:
  - Explicit preferences: e.g., ratings given to items by users
  - Implicit preferences: e.g., which items were rented or bought by users.

Content-based vs. CF Approaches

- The content-based approach makes recommendations based on item content.
  - E.g., for a movie: genre, actors, director, length, language, etc.
  - Can be used to recommend new items for which no ratings are available yet.
  - Does not perform as well as CF in most cases.
- The CF approach does not look at item content.
  - Preferences are inferred from rating patterns alone.
  - Cannot recommend new items – they all look the same to the system.
  - Very effective when a sufficient amount of data is available.
- It is possible to combine the two approaches in different ways.

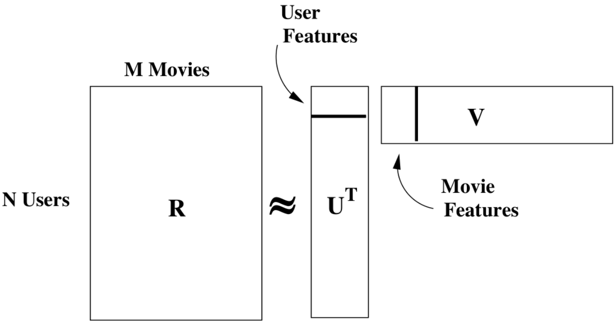
CF as Matrix Completion

- CF can be viewed as a matrix completion problem.

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
Alice	***				*****	
Bob	**	****			*	*****
Charles				****		****
David		****	****		*****	
Eva	*****			***		**
Fred	*	*****	****		****	

- Problem: given a user-item rating matrix with only a small subset of entries present, fill in (some of) the missing entries.
- An effective way to solve the matrix completion problem is by matrix factorization, which approximates the rating matrix by the product of two smaller matrices.

Matrix Factorization: Notation



- Suppose we have N users and M movies.
- Let $\mathbf{R}$ be a rating matrix of integer rating values from 1 to K with R_{ij} denoting the rating of user i for movie j, and $\mathbf{U} \in \mathbb{R}^{D\times N}$ and $\mathbf{V} \in \mathbb{R}^{D\times M}$ be the latent user feature matrix and latent movie feature matrix, respectively.

A Non-probabilistic View

- To predict the rating given by user i to movie j, we simply compute the dot product between the corresponding latent feature vectors $\mathbf{U}_i$ and $\mathbf{V}_j$ for user i and movie j, respectively:

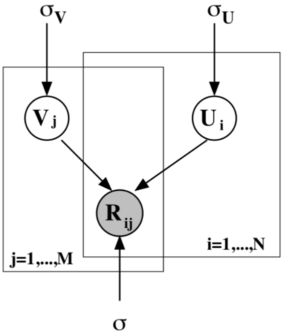
$$\hat{R}_{ij} = \mathbf{U}_i^T \mathbf{V}_j = \sum_k U_{ik} V_{jk}$$

- Intuition: for each user, we predict a movie rating by giving the movie feature vector to a linear model.
  - The movie feature vector can be viewed as the input.
  - The user feature vector can be viewed as the weight vector.
  - The predicted rating is the output.
  - Unlike in linear regression where the inputs are fixed and the weights are learned, we learn both the inputs and the weights here (by minimizing the squared error).
  - Note that the model is symmetric in the users and movies.

PMF: Generative Model I

- Probabilistic matrix factorization (PMF) is a simple probabilistic linear model with Gaussian observation noise.
- Given the feature vectors for user i and movie j, the distribution of the corresponding rating is:

$$p(R_{ij} \mid \mathbf{U}_i, \mathbf{V}_j, \sigma^2) = \mathcal{N}(R_{ij} \mid \mathbf{U}_i^T \mathbf{V}_j, \sigma^2)$$

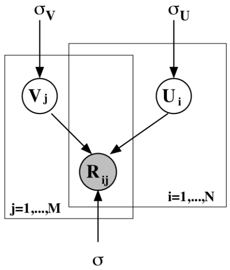


PMF: Generative Model II

- Distribution over all observed ratings:

$$p(\mathbf{R} \mid \mathbf{U}, \mathbf{V}, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij} \mid \mathbf{U}_i^T \mathbf{V}_j, \sigma^2) \right]^{\mathbf{I}_{ij}}$$

where $\mathbf{I}_{ij} = 1$ if user i has rated movie j and is 0 otherwise.



- The user and movie feature vectors are given zero-mean spherical Gaussian priors:

$$p(\mathbf{U} \mid \sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{U}_i \mid \mathbf{0}, \sigma_U^2 \mathbf{I}), \qquad p(\mathbf{V} \mid \sigma_V^2) = \prod_{j=1}^{M} \mathcal{N}(\mathbf{V}_j \mid \mathbf{0}, \sigma_V^2 \mathbf{I})$$

PMF: Learning I

- MAP estimation: maximizing the log-posterior over the user and movie features with fixed hyperparameters.
- Equivalent optimization problem (minimizing the sum of squared errors with quadratic regularization terms):

$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} \mathbf{I}_{ij}(R_{ij} - \mathbf{U}_i^T \mathbf{V}_j)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \|\mathbf{U}_i\|_2^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \|\mathbf{V}_i\|_2^2$$

where $\lambda_U = \sigma^2/\sigma_U^2$ and $\lambda_V = \sigma^2/\sigma_V^2$.

PMF: Learning II

- The error function E is convex in $\mathbf{U}$ only or $\mathbf{V}$ only, but not convex in both $\mathbf{U}$ and $\mathbf{V}$ together.
- Block coordinate descent alternates between two convex optimization subproblems:
  - Minimizing E over $\mathbf{U}$ with $\mathbf{V}$ fixed
  - Minimizing E over $\mathbf{V}$ with $\mathbf{U}$ fixed.
- The algorithm takes time linear in the number of observed ratings.

Slight Variation

- Using a simple linear-Gaussian model as above can lead to predicted ratings outside the valid range.
- To overcome this problem, the dot product of the user and movie feature vectors is passed through the logistic function

$$g(x) = \frac{1}{1 + \exp(-x)}$$

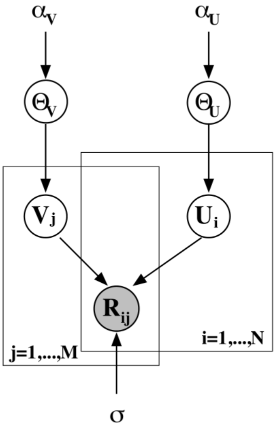
- Distribution over all observed ratings:

$$p(\mathbf{R} \mid \mathbf{U}, \mathbf{V}, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij} \mid g(\mathbf{U}_i^T \mathbf{V}_j), \sigma^2) \right]^{\mathbf{I}_{ij}}$$

- The ratings $1, \ldots, K$ are linearly mapped to the interval $[0, 1]$ so that the range of valid rating values matches the range of predictions made by the model.

Extension with Automatic Complexity Control I

- Model complexity is controlled by the noise variance σ^2 and the parameters of the priors σ_U^2 and σ_V^2 in standard PMF.
- Approach: find MAP estimates for the hyperparameters $\mathbf{\Theta}_U$ and $\mathbf{\Theta}_V$ after introducing priors for them.



Extension with Automatic Complexity Control II

- Learning: find MAP estimates of the parameters and hyperparameters by maximizing the log-posterior:

$$\begin{aligned}
&\log p(\mathbf{U}, \mathbf{V}, \sigma^2, \mathbf{\Theta}_U, \mathbf{\Theta}_V \mid \mathbf{R}) \\
=&\log p(\mathbf{R} \mid \mathbf{U}, \mathbf{V}, \sigma^2) + \log p(\mathbf{U} \mid \mathbf{\Theta}_U) + \log p(\mathbf{V} \mid \mathbf{\Theta}_V) \\
&+ \log p(\mathbf{\Theta}_U) + \log p(\mathbf{\Theta}_V) + \mathit{const}.
\end{aligned}$$

- Essentially this is like choosing the hyperparameters λ_U and λ_V in the standard PMF automatically.
- Automatic selection of the hyperparameter values works considerably better than the manual approach using a validation set.