Introduction to Matrix Factorization

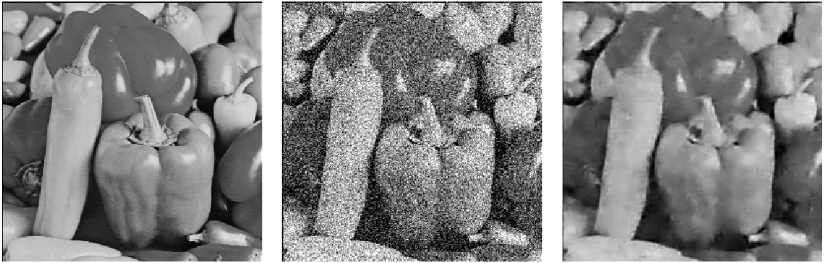
David Rosenberg

New York University

December 26, 2016

The Matrix Denoising (or Smoothing) Problem

- Consider an image as a matrix of gray-scale values:



- Left: **Original**; Middle: **Adds Gaussisan noise**; Right: **Denoised**

Taken from Marc'Aurelio Ranzato's page `http://www.cs.nyu.edu/~ranzato/research/projects.html`.

The Matrix Denoising (or Smoothing) Problem

- M is the **original matrix**
- W is the **noise** added to Z
- We observe $Z = M + W$.
- Problem:
  - Given Z, produce some estimate for M.
- How?

General Approach

- Observe Z (the noisy M matrix).
- We want a matrix $\hat{Z}$ such that

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\left(\hat{z}_{ij}-z_{ij}\right)^2 \text{ is small.}$$

  - But not too small!
  - Don't want to be fitting the noise **(overfitting)**
- Need some way to constrain $\hat{Z}$.
- Different constraints give different algorithms.

Some Constraints for $\hat{M}$ (Quick Look)

1. $\|\hat{Z}\|_{\ell_1} \leqslant c$ (Sparse matrix approximation)
2. $\text{rank}(\hat{Z}) \leqslant c$. (SVD methods)
3. $\|\hat{Z}\|_* \leqslant c$ (**Nuclear norm** $=$ sum of singular values**)**
4. $\hat{Z} = UDV^T$ with $\Phi(u_j) \leqslant c_1, \Phi_2(v_k) \leqslant c_2$ (Penalized SVD)
5. $\hat{Z} = LR^T$ with $\Phi_1(L) \leqslant c_1, \Phi_2(R) \leqslant c_2$ (Max-margin matrix factorization)
6. $\hat{Z} = L + S$ with $\Phi_1(L) \leqslant c_1, \Phi_2(S) \leqslant c_2$ (Additive matrix decompostion)

List from *Statistical Learning with Sparsity* by Hastie et al.

The Frobenius Norm

- **[Squared] Frobenius norm = "sum of squares"** (for a matrix)
- For a matrix $M \in \mathbf{R}^{m \times n}$, the Frobenius norm of M is

$$\|M\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} m_{ij}^2.$$

- The ℓ_2-loss at the matrix level is often written as

$$\|Z - \hat{Z}\|_F^2 \quad = \quad \sum_{i=1}^{m} \sum_{j=1}^{n} (z_{ij} - \hat{z}_{ij})^2$$

General Framework

- Given $Z \in \mathbf{R}^{m \times n}$,
- Find $\hat{Z} \in \mathbf{R}^{m \times n}$ solving the following optimization problem:

$$\begin{array}{ll} \text{minimize} & \|Z - \hat{Z}\|_F^2 \\ \text{such that} & \Phi(\hat{Z}) \leqslant c. \end{array}$$

- Modifications required when
  - Z isn't fully observed, or
  - if we want to apply Φ to factors of $\hat{Z}$

Matrix Notation

- Consider

$$M = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \\ m_{31} & m_{33} \end{pmatrix}$$

- Can also write M in terms of its column vectors: $M = (m_{\cdot 1}, m_{\cdot 2})$.

Matrix-Vector Product

- Consider Mv, where $v = (v_1, v_2)^T$ is a column vector:

$$\begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \\ m_{31} & m_{33} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} v_1 m_{11} + v_2 m_{12} \\ v_1 m_{21} + v_2 m_{22} \\ v_1 m_{31} + v_2 m_{33} \end{pmatrix} = v_1 \begin{pmatrix} m_{11} \\ m_{21} \\ m_{31} \end{pmatrix} + v_2 \begin{pmatrix} m_{12} \\ m_{22} \\ m_{33} \end{pmatrix}$$

- Product is a linear combination of the columns.
- Can write as

$$Mv = v_1 m_{\cdot 1} + v_2 m_{\cdot 2}$$

Matrix-Matrix Product

- Multiplying 2 matrices is a matrix-vector multiply for each column in product

$$
\begin{aligned}
&\begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \\ m_{31} & m_{33} \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \\
= \quad &\left(\begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \\ m_{31} & m_{33} \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{21} \end{pmatrix} ; \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \\ m_{31} & m_{33} \end{pmatrix} \begin{pmatrix} a_{12} \\ a_{22} \end{pmatrix} \right)
\end{aligned}
$$

- From this, easy to see certain results on rank coming up...

Rank of a Matrix

- Let $M \in \mathbf{R}^{m \times n}$ be an $m \times n$ matrix.
- Many equivalent definitions of **rank(M).**
- $\operatorname{rank}(M)$ is
  - max number of linearly independent columns
  - max number of linearly independent rows
  - dimension of column space
  - dimension of row space
  - number of non-zero singular values in SVD
- Largest possible rank for M?

Rank of an outer product

- Suppose we have column vectors $v = (v_1, v_2)^T$ and $w = (w_1, w_2, w_3)^T$.
- Their **outer product** is

$$\begin{aligned} vw^T &= \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \begin{pmatrix} w_1 & w_2 & w_3 \end{pmatrix} \\ &= \begin{pmatrix} v_1 w_1 & v_1 w_2 & v_1 w_3 \\ v_2 w_1 & v_2 w_2 & v_2 w_3 \end{pmatrix} \\ &= \begin{pmatrix} w_1 v \; w_2 v \; w_3 v \end{pmatrix} \end{aligned}$$

- Every column of vw^T is a multiple of v.
- What is the rank of vw^T?

Every Rank 1 Matrix is an Outer Product

- Suppose $M \in \mathbf{R}^{m \times n}$ is a rank 1 matrix.
- Claim: $\exists w \in \mathbf{R}^m, x \in \mathbf{R}^n$ s.t. $M = wx^T$.
- Proof:
  - Rank 1 implies every column is multiple of first column.
  - Let w=first column of M.
  - Take v to be vector of multiples of w required.

SVD for Rank 1 Matrix

- The factorization
$$M = \sigma_1 u v^T,$$
where $\sigma_1 > 0$ and $\|u\| = \|v\| = 1$ is the **singular value decomposition (SVD).**
- Easy derivation from $M = wx^T$.

Rank of matrix product

- Suppose $\text{rank}(M) = r$ and $\text{rank}(N) = s$.
- What is largest possible rank of MN?
- Approach:
  - Every column of MN is a linear combination of columns of M.
  - So $\text{rank}(MN) \leqslant r$.
- Using $\text{rank}(A) = \text{rank}(A^T)$,
  - can apply same argument to $NM \implies \text{rank}(MN) \leqslant s$.
- So

$$\text{rank}\,(MN) \leqslant \min(\text{rank}(M), \text{rank}(N)).$$

Low Rank Matrices

- $M \in \mathbf{R}^{m \times n}$ is **low rank** if $\operatorname{rank}(M) \ll \min(m,n)$.
- (M has relatively few independent columns.)
- Suppose M is 5×4 and $\operatorname{rank}(M) = 2$, then we can **factorize** as:

$$\underbrace{\begin{pmatrix} m_{11} & & & \\ & & & \\ & & & \\ & & & \\ & & & m_{54} \end{pmatrix}}_{5\times 4} = \underbrace{\begin{pmatrix} a_{11} & \\ & \\ & \\ & \\ & a_{52} \end{pmatrix}}_{5\times 2} \underbrace{\begin{pmatrix} b_{11} & & & \\ & & & b_{24} \end{pmatrix}}_{2\times 5}$$

- Proof: Should be clear from discussion above.

Rank-Constrained Matrix Approximation

- Given $Z \in \mathbf{R}^{m \times n}$,
- Find $\hat{Z} \in \mathbf{R}^{m \times n}$ solving the following optimization problem:

$$\begin{array}{ll} \text{minimize} & \|Z - \hat{Z}\|_F^2 \\ \text{such that} & \operatorname{rank}(\hat{Z}) \leqslant c, \end{array}$$

  for $c \in \{1, 2, \ldots, \}$

- The solution to this problem is immediate from the **singular value decomposition** (SVD)

Singular Value Decomposition (SVD)

- If $M \in \mathbf{R}^{m \times n}$ has rank r, then its **singular value decomposition** is

$$M = \sum_{i=1}^{r} \sigma_i u_i v_i^T,$$

  where

  - $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_r > 0$ are called the **singular values**
  - $u_1, \ldots, u_r$ are **orthonormal** and are called the **left singular vectors**
  - $v_1, \ldots, v_r$ are **orthonormal** and are called the **right singular vectors**
- **THEOREM**: **Every matrix has a singular value decomposition.**

SVD: Rank r Approximation Theorem

- Given $Z \in \mathbf{R}^{m \times n}$, find $\hat{Z} \in \mathbf{R}^{m \times n}$ solving the following:

$$\begin{array}{rl} \text{minimize} & \|Z - \hat{Z}\|_F^2 \\ \text{such that} & \operatorname{rank}(\hat{Z}) \leqslant c, \end{array}$$

  for $c \in \{1, 2, \ldots, \operatorname{rank}(Z)\}$.

- **THEOREM:** This optimization problem is solved by

$$\hat{Z} = \sum_{i=1}^{c} \sigma_i u_i v_i^T,$$

  where σ's, u's, and v's are given by the SVD for Z:

$$Z = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

The Netflix Problem

- Partially observed ratings matrix

Movie Ratings	Zora	Sophie	Jordan	Ernie	Christie
Harold and Kumar Escape..	8	4	?	?	4
Ted	?	?	8	10	4
Straight Outta Compton	8	10	?	?	6

Taken from https://analyticsweek.com/content/movie-recommendations-how-does-netflix-do-it-a-9-step-coding-intuitive-guide-into-collaborative-filtering/

The Netflix Problem

- How do we fill in the missing entries of the ratings matrix?
- Let's generalize the problem a bit:
  - m users
  - n movies
  - **ratings matrix** $Z \in \mathbf{R}^{m \times n}$ (an $m \times n$ matrix)
- We observe some entries of Z – how to estimate the rest?

Partial Observations (Notation)

- z_{ij}: rating for user i and movie j
- Observed entries $\mathcal{D} = \{z_{1,2} = 3, z_{2,4} = 0, z_{4,2} = 1, z_{6,2} = 5\}$
  - (This is our **training data**.)
- Ω: set of entries that we observe, e.g.

$$\Omega = \{(1,2), (2,4), (4,2), (6,2)\}$$

- Mean observed rating:

$$\mu = \frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} z_{ij} = \frac{9}{4}$$

The Prediction Problem

- Given a new user/movie pair (i, j),
    - give a prediction $\hat{z}_{ij}$ for the unknown z_{ij}.
- Alternatively,
    - produce a full matrix $\hat{Z} = \left(\hat{z}_{ij}\right) \in \mathbf{R}^{m \times n}$ all at once.
- Equivalent, just two different ways to think about it.

Performance Measure

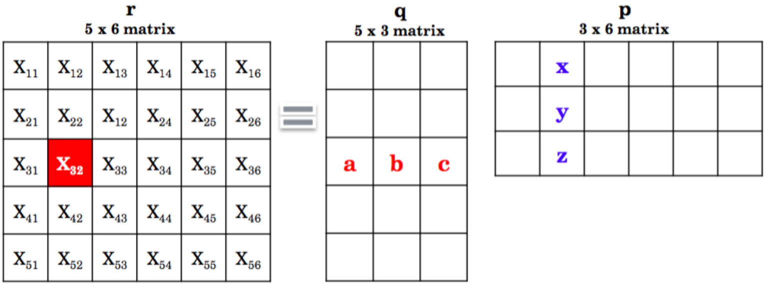
- We predict $\hat{z}_{ij}$.
- Later we get the actual z_{ij}.
- How to evaluate how well we did?
- Need a **loss function**.
- Netflix used ℓ_2-loss (i.e. square loss):

$$\ell(z_{ij}, \hat{z}_{ij}) = (z_{ij} - \hat{z}_{ij})^2$$

- We'll focus on square loss.

More Matrix Multiplication

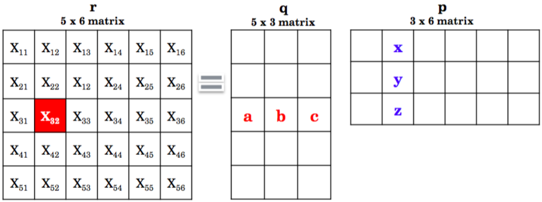
- Sometimes we need the most vanilla description of matrix mult:



$$X_{32} = (a, b, c) \cdot (x, y, z) = a * x + b * y + c * z$$

Alex Lin's "Introduction to Matrix Factorization Methods Collaborative Filtering"

Netflix Matrix Factorization ("Latent Factor Model")



t $\mathbf{X}_{32} = (\mathbf{a}, \mathbf{b}, \mathbf{c}) \cdot (\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathbf{a} * \mathbf{x} + \mathbf{b} * \mathbf{y} + \mathbf{c} * \mathbf{z}$

- Factorization tells us $\text{rank}(r) \leqslant 3$.(clear?)
- x_{32} is rating for user 3 of movie 2.
- Interpretation: we have 3 movie **categories** (called **factors** in more generic contexts)
- $q_{3\cdot}$ gives user 3's weightings to each **category**
- $p_{\cdot 2}$ gives how much each movie belongs to **category** 2

Alex Lin's "Introduction to Matrix Factorization Methods Collaborative Filtering"

Netflix Matrix Factorization: Objective Function

- Find $\hat{Z} = QP^T$ where Q is $m \times r$ and P is $n \times r$.
- Training loss is

$$\frac{1}{|\Omega|} \sum_{(i,j) \in \Omega} \left(z_{ij} - q_{i \cdot} p_{\cdot j} \right)^2$$

- How many parameters? $r(m+n)$
- Minimize this loss using
  - SGD or
  - Alternating least squares

Put in some regularization

- Training loss is

$$\frac{1}{|\Omega|}\sum_{(i,j)\in\Omega}\left(z_{ij}-q_{i\cdot}p_{\cdot j}\right)^2+\sum_{i=1}^{m}\sum_{j=1}^{r}\left(\lambda_1|q_{ij}|+\lambda_2 q_{ij}^2\right)$$

- How many parameters? $r(m+n)$
- First pass: minimize this loss using SGD.

Put in some bias terms

- Training loss is

$$\frac{1}{|\Omega|}\sum_{(i,j)\in\Omega}\left(z_{ij}-\left[q_{i\cdot}p_{\cdot j}+u_i+m_j+c\right]\right)^2+\sum_{i=1}^{m}\sum_{j=1}^{r}\left(\lambda_1|q_{ij}|+\lambda_2 q_{ij}^2\right)$$

- user bias term: u_i
- movie bias term: m_j