Matrix Factorization

Used for:

- dim. red
- imputation of missing data
- denoising by "borrowing strength"
- finding "archetypes" of "latent dims"

- looking to find a low rank matrix st. loss is minimized (for real valued matrices, sq. loss typically used). Using sq. loss is same as minimizing Frob. norm.

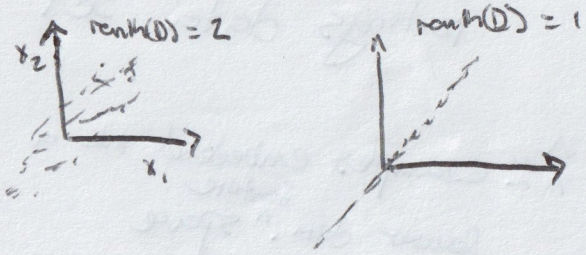
eg: PCA

$$\hat{D} = \underset{\tilde{D} \in \mathbb{R}^{m \times n}, \operatorname{rank}(\tilde{D}) = k}{\operatorname{argmin}} \| D - \tilde{D} \|_F^2$$

where $D \in \mathbb{R}^{m \times n}$ is the design matrix

rank(D) : dim. of subspace spanned by rows in D

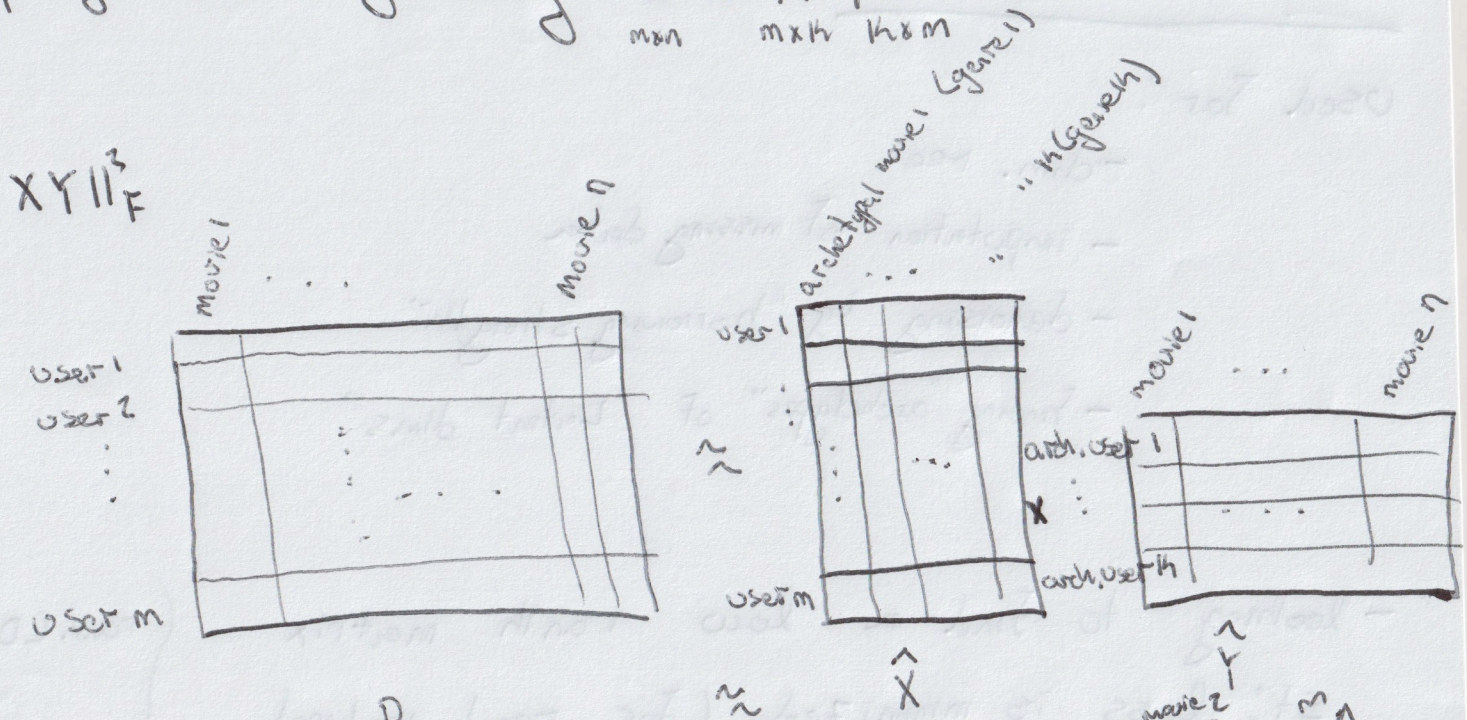
eg



rank constraint can be implicitly encoded by letting $\tilde{D}_{m \times n} = X_{m \times k} Y_{k \times n}$

$$\Rightarrow \hat{X}, \hat{Y} = \underset{X \in \mathbb{R}^{m \times k},\ Y \in \mathbb{R}^{k \times n}}{\arg\min} \|D - XY\|_F^2$$

$$\hat{D} = \hat{X}\hat{Y}$$



eg: a user x movie ratings data set

$\hat{X} \approx$ examples embedded into a lower dim. feature space

$\hat{Y} \approx$ features " " lower dim. example space

users characterized by movie ratings

users characterized by genre ratings

movies characterized by arch. user ratings

- in general, we can add regularization to keep generalization error low as well as summing over only observed data in a sparse dataset, as well as add desired constraints (eg. non-negativity of $\hat{D}$)

$$\hat{X}, \hat{Y} = \underset{\substack{X \in \mathbb{R}^{k \times m} \\ Y \in \mathbb{R}^{k \times n} \\ \text{constraints}}}{\operatorname{argmin}} \sum_{(i,j) \in \Omega} (D_{ij} - x_i^T y_j)^2 + \lambda \sum_{i=1}^{m} r_x(x_i) + \gamma \sum_{j=1}^{n} r_y(y_j)$$

x_i: rows of X; y_j: cols of Y

$r_x : \mathbb{R}^k \to \mathbb{R}_+$ (reg. on rows of X)

$r_y : \mathbb{R}^k \to \mathbb{R}_+$ (reg. on cols of Y)

$\Omega = \{(i,j)$ indices of D which are observed$\}$

$\hat{D}_{ij} = \hat{x}_i^T \hat{y}_j$

- the optimization is usually solved w/ Alternating minimization (hold X constant and optimize over Y, hold Y constant and optimize over X...), similar to coordinate descent

or

SGD (each iteration takes $O(\text{batch size})$)

vs.

(each iteration takes $O(|\Omega|)$ for Alt. min.)

- general recipe for Matrix Fact.
  - choose loss function
  - choose regularization
  - impose any constraints
  - choose optimization algo.

Generalized Low Rank Models

- is applicable to design matrices w/ any data type (not just reals) (eg: booleans, ordinals, categoricals)
- can embed the data into $\mathbb{R}^n$ by finding the "best" rank k real valued matrix
- can also project ^(matrix) back into original data types if desired

let $\mathcal{F}_j$ = set of values that col. j of D can take on (eg: $\mathcal{F}_j = \{-1, 1\}$ for booleans)

and let $L_j : \mathcal{F}_j \times \mathbb{R} \to \mathbb{R}_+$ be an appropriate loss function for that data type. Then, under a GLM

$$\hat{X}, \hat{Y} = \underset{\substack{X \in \mathbb{R}^{m \times k} \\ Y \in \mathbb{R}^{k \times n}}}{\operatorname{argmin}} \sum_{(i,j) \in \Omega} L_j(D_{ij}, X_i^T y_j) + \lambda \sum_{i=1}^{m} r_x(x_i) + \gamma \sum_{j=1}^{n} r_y(y_j)$$

constraints

- then $\hat{X}\hat{Y}$ is the "best" low rank representation of the data in $\mathbb{R}^n$. $\hat{X}$ is the "best" representation of the examples in $\mathbb{R}^k$

↪

- if desired, to express the data in the original data types

$$\hat{D}_{ij} = \underset{d \in \mathcal{F}_j}{\operatorname{argmin}} \; L_j(d, \hat{X}_j^T, \hat{q}_j)$$

- that is, we choose the value of $\hat{D}_{ij}$ s.t. the loss between this choice and the real-valued, low rank approx^ of the original design matrix is minimized

Possible loss functions

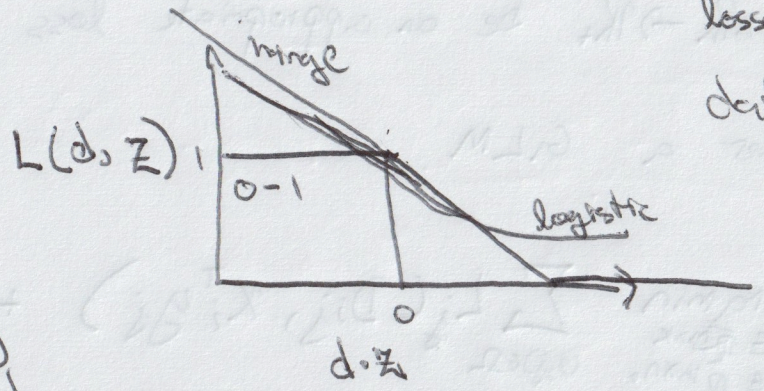
Boolean: hinge, logistic

categorical: log loss / cross entropy

ordinal: encode levels to numbers and use ordinal hinge-loss

$d \in \{-1, 1\}$; $z \in \mathbb{R}$

losses for Boolean data $\{-1, 1\}$



$$L_{hinge}(d, z) = \max(0, 1 - dz)$$

$$L_{logistic}(d, z) = \log(1 + e^{-dz})$$