Recommendation Systems

REVIEW

Today …

- Recommendation Systems
- Gradient Descent

Recommendation Systems

Utility Matrix

- Users (rows) & Items (columns)
- Matrix entries are scores/ratings by user for the item
  - Boolean
  - Ordered set
  - Real
- Matrix is sparse

- **Goal of recommendation systems**
  - Predict the blank entries of the utility matrix
  - Not necessary to predict every entry
    - predict some high entries

Recommendation Systems

- Two major approaches
  - **Content based systems** – similarity of item properties
    - Depending on the properties of movies you have watched, suggest movies with the same properties – genre, director, actors etc.
  - **Collaborative filtering** – relationship between users and items
    - Find users with a similar ‘taste’
    - Recommend items preferred by similar users

Collaborative Filtering

- Instead of using an item-profile vector use the column in the utility matrix
  - Item defined by which users have bought/rated the item
- Instead of using an user-profile vector use the row in the utility matrix
  - User defined by what items they have bought/liked
- Users similar if their vectors are close using some metric
  - Jaccard, cosine
- Recommendations based on finding similar users and recommending items liked by similar users

Duality of Similarity

- Two approaches estimate missing entries of the utility matrix
  - Find **similar users** and average their ratings for the particular item
  - Find **similar items** and average user's ratings for those items
- Considerations
  - Similar users: only find similar users once, generate rankings on demand
  - Similar items: need to find similar items for all items
    - Is more reliable in general

Clustering users and items

- In order to deal with the sparsity of the utility matrix
- Cluster items
  - New utility matrix has entries with average rating that the user gave to items in the cluster
  - Use this utility matrix to …
- Cluster users
  - Matrix entry → average rating that the users gave
- Recurse
  - Until matrix is sufficiently dense

Estimating entries in the original utility matrix

- Find to with clusters the user (U) and item (I) belong, say C and D
- If an entry exists for row C and column D, use that for the UI entry of the original matrix
- If the CD entry is blank, then find similar item (clusters) and estimate the value for the CD entry and consequently that for the UI entry of the original matrix.

Dimensionality reduction

- Assume use can approximate M using matrices U, V

- $M \rightarrow n \times m$
- $M = UV$
- $U \rightarrow n \times d, \quad V \rightarrow d \times m$

- M is sparse, U, V are dense

Optimization

- Consider the misfit in $||M - UV||$
- Ideally want it to be zero
  - Minimize the misfit as much as possible
  - $nd + dm$ unknowns

Gradient Descent

Given a multivariate function $F(x)$, at point p

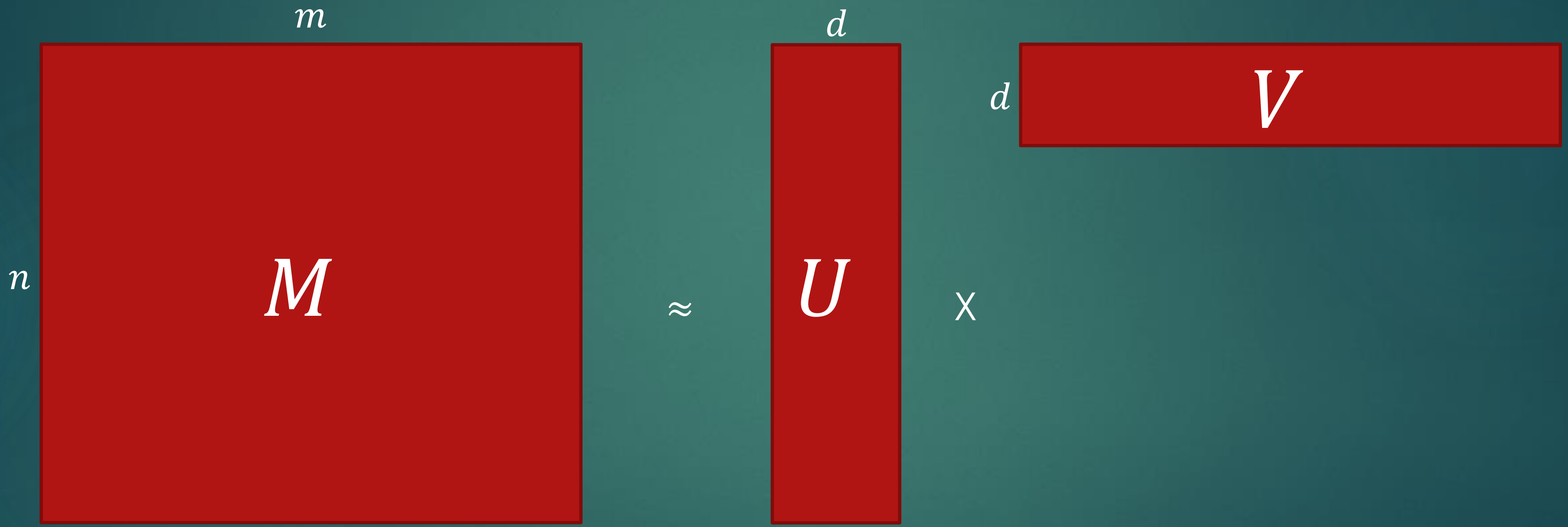
then, $F(b) < F(a)$, where

$$b = a - \gamma \nabla F(a)$$

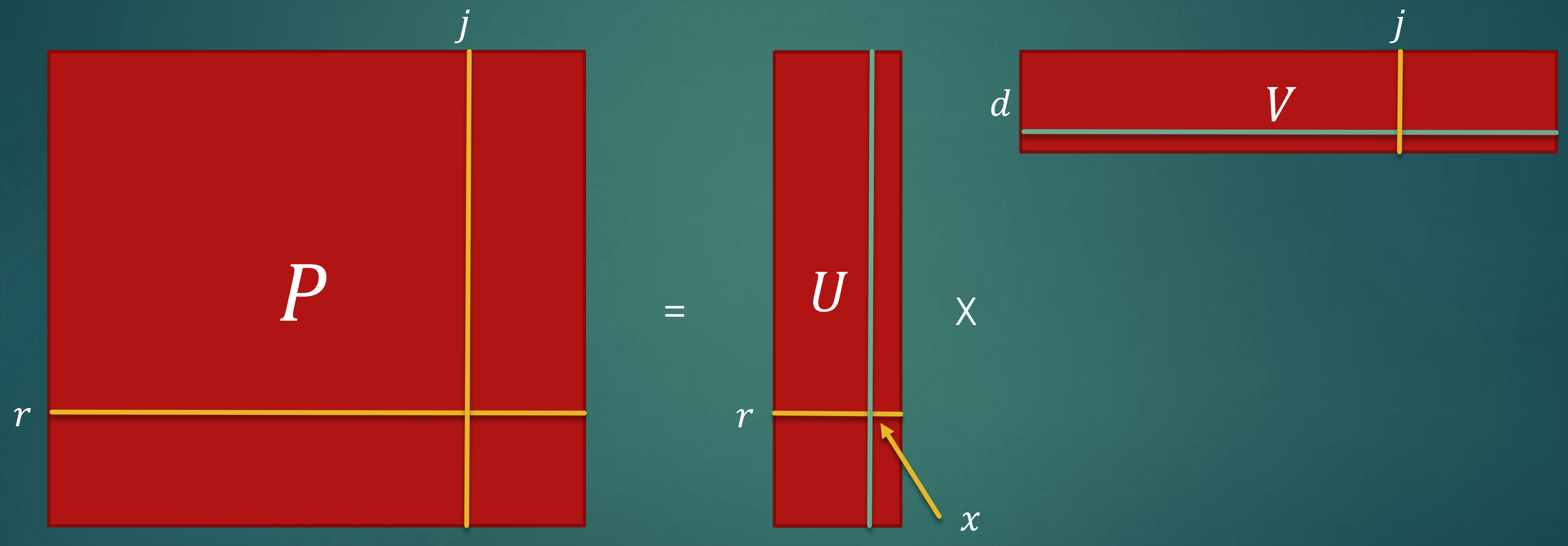
for some sufficiently small γ

$$\min \ ||M - UV||$$

UV Decomposition



UV Decomposition



$$p_{rj} = \sum_{k=1}^{d} u_{rk} v_{kj} = \sum_{k \neq s} u_{rk} v_{kj} + x v_{sj}$$

UV Decomposition

- $M, U, V,$ and $P = UV$
- Let us optimize for $x = u_{rs}$

$$p_{rj} = \sum_{k=1}^{d} u_{rk} v_{kj} = \sum_{k \neq s} u_{rk} v_{kj} + x v_{sj}$$

$$\mathcal{C} = \sum_{j} \left(m_{rj} - p_{rj}\right)^2 = \sum_{j} \left(m_{rj} - \sum_{k \neq s} u_{rk} v_{kj} - x v_{sj}\right)^2$$

UV Decomposition

- First order optimality → $\partial \mathcal{C}/\partial x = 0$

$$\mathcal{C} = \sum_j \left(m_{rj} - p_{rj}\right)^2 = \sum_j \left(m_{rj} - \sum_{k \neq s} u_{rk} v_{kj} - x v_{sj}\right)^2$$

$$\frac{\partial \mathcal{C}}{\partial x} = \sum_j -2 v_{sj} \left(m_{rj} - \sum_{k \neq s} u_{rk} v_{kj} - x v_{sj}\right) = 0$$

$$x = \frac{\sum_j v_{sj}\left(m_{rj} - \sum_{k \neq s} u_{rk} v_{kj}\right)}{\sum_j v_{sj}^2}$$

UV Decomposition

- Choose elements of U and V to optimize
  - In order
  - Some random permutation
  - Iterate
- Correct way
  - Use expression to compute $\partial \mathcal{C} / \partial \boldsymbol{x}$ at current estimate
    - Expensive when number of unknowns is large $(2\ n\ d)$
  - Use traditional gradient descent

Stochastic Gradient Descent

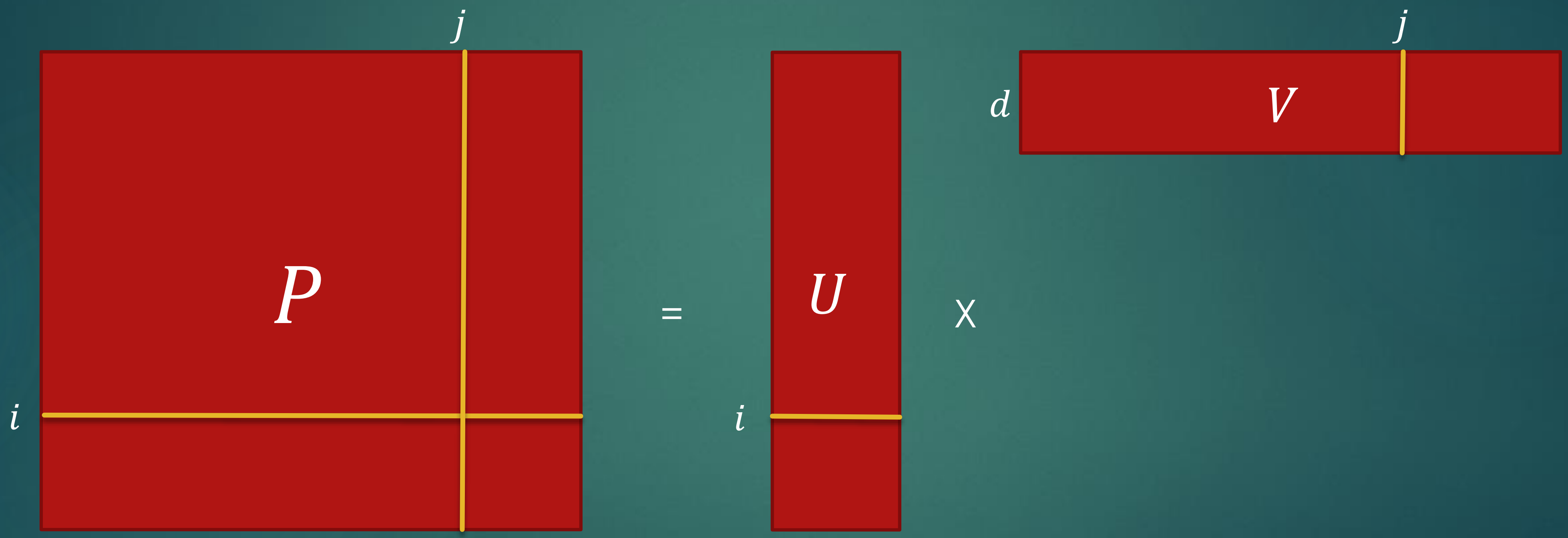
In cases where the objective function $\mathcal{C}(w)$ can be written in terms of local costs

$$\mathcal{C}(w) = \sum_n \mathcal{C}_i(w)$$

For the case of UV decomposition,

$$\mathcal{C} = \sum_{i,j} c(M_{ij}, U_{i*}, V_{*j})$$

Stochastic Gradient Descent



$$c\left(M_{ij}, U_{i*}, V_{*j}\right) = \left(m_{ij} - p_{rj}\right)^2 = \left(m_{ij} - \sum_{k=1}^{d} u_{ik} v_{kj}\right)^2$$

Stochastic Gradient Descent

- Traditional gradient descent

$$w \leftarrow w - \lambda \sum_{n} \nabla C_i(w)$$

- In Stochastic GD, approximate true gradient by a single example:

$$w \leftarrow w - \lambda \nabla C_i(w)$$

Stochastic Gradient Descent

- Input: samples Z, initial values $\boldsymbol{U}_0, \boldsymbol{V}_0$
- while not converged do
  - Select a sample $(i, j) \in Z$ uniformly at random
    - $\boldsymbol{U'}_{i*} \leftarrow \boldsymbol{U}_{i*} - \lambda_n N \frac{\partial}{\partial \boldsymbol{U}_{i*}} c(\boldsymbol{M}_{ij}, \boldsymbol{U}_{i*}, \boldsymbol{V}_{*j})$
    - $\boldsymbol{V}_{*j} \leftarrow \boldsymbol{V}_{*j} - \lambda_n N \frac{\partial}{\partial \boldsymbol{V}_{*j}} c(\boldsymbol{M}_{\boldsymbol{ij}}, \boldsymbol{U}_{i*}, \boldsymbol{V}_{*j})$
    - $U_{i*} \leftarrow U'_{i*}$

Stochastic Gradient Descent

$$\frac{\partial}{\partial \boldsymbol{U}_{i*}} c(\boldsymbol{M}_{ij}, \boldsymbol{U}_{i*}, \boldsymbol{V}_{*j})$$

$$\frac{\partial}{\partial \boldsymbol{U}_{i*}} \left(m_{ij} - \sum_{k=1}^{d} u_{ik} v_{kj} \right)^2$$

$$\frac{\partial c}{\partial \boldsymbol{U}_{ik}} = -2 v_{kj} \left(m_{ij} - \sum_{k=1}^{d} u_{ik} v_{kj} \right)$$

$$\frac{\partial c}{\partial \boldsymbol{U}_{i*}} = -2 \left(m_{ij} - \boldsymbol{U}_{i*} \cdot \boldsymbol{V}_{*j} \right) \boldsymbol{V}_{*j}$$