Matrix Factorization and Collaborative Filtering

Kyle Kastner
Advanced Topics in Pattern Recognition
UT San Antonio
Spring 2013

Topics

- Basic concepts
- Sample use cases
- Singular Value Decomposition (SVD)
- Low Rank SVD
- Matrix Factorization
  - Probabilistic Matrix Factorization (PMF)
  - Kernelized Probabilistic Matrix Factorization (KPMF)

Basic Concepts

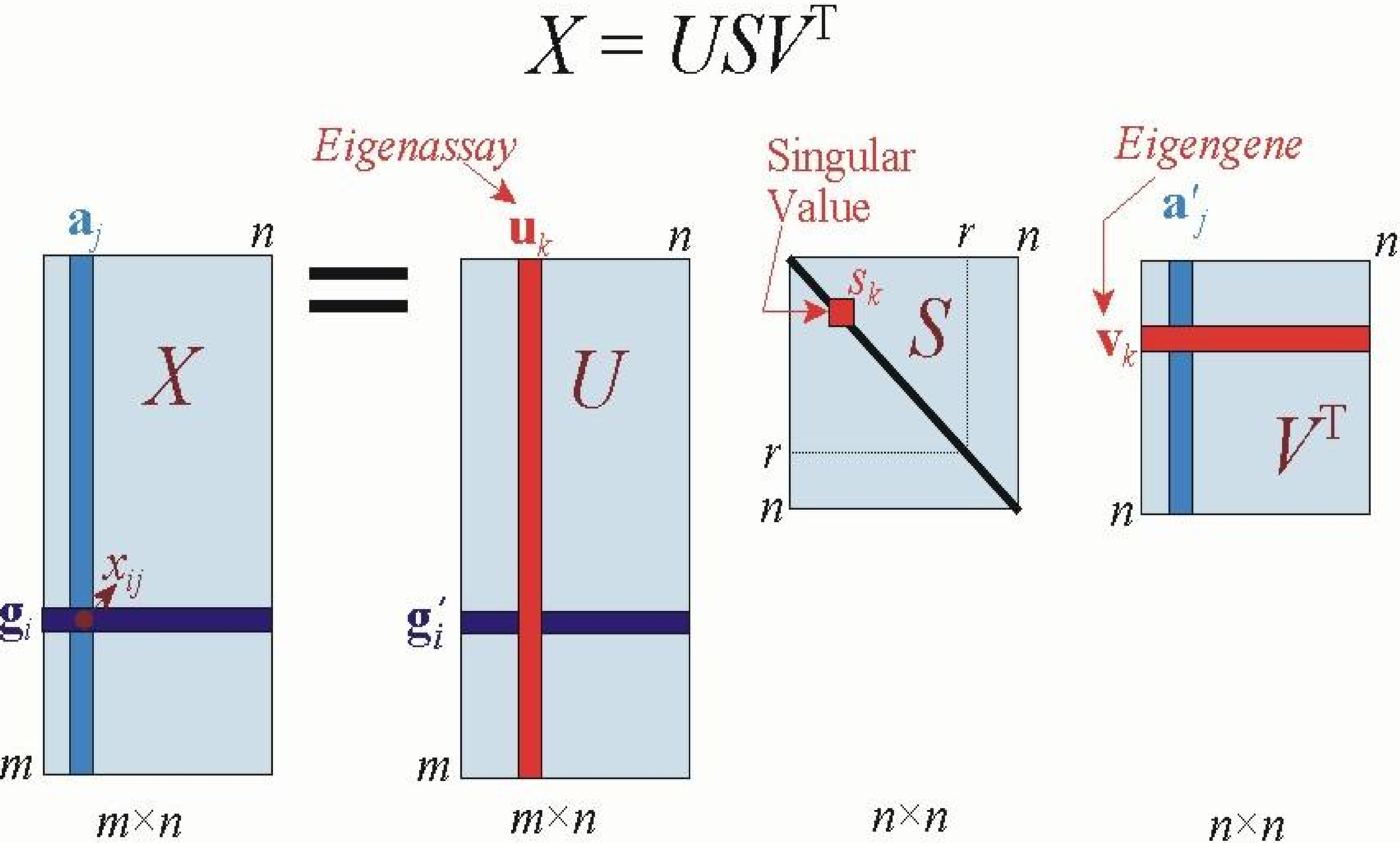
- Large matrix of information, X
- Break X into smaller matrices
- Smaller matrix values chosen to minimize some cost function
- Typical goal is to minimize reconstruction error
- Extension of concept behind linear discriminants, PCA, etc.

Sample Use Cases

- Recommendation systems [1]
    - Netflix
    - Yelp?
- Compression
- Image recovery [3]
- Ecology [3]
- Musical similarity [4]
- Basis in graphical models
- Extends to hierarchical ("deep") learning [3]

SVD

- Break matrix X into U,S,V submatrices
- X can be reconstructed from these
- Many libraries exist for accurate SVD calculation
- There are also "sparse" SVD algorithms
- Computationally light (comparatively)



Visual representation of SVD calculation [2]

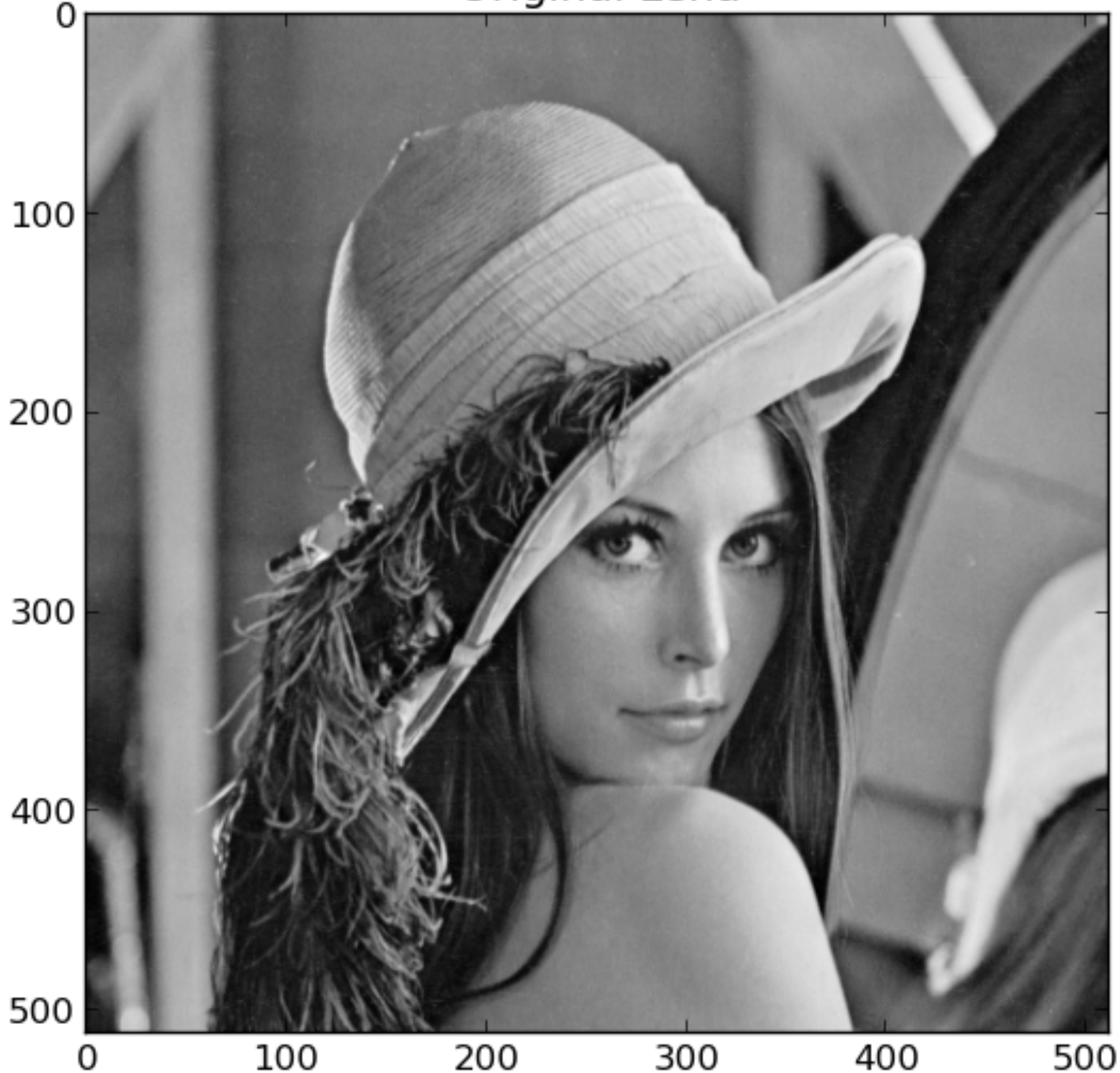
SVD

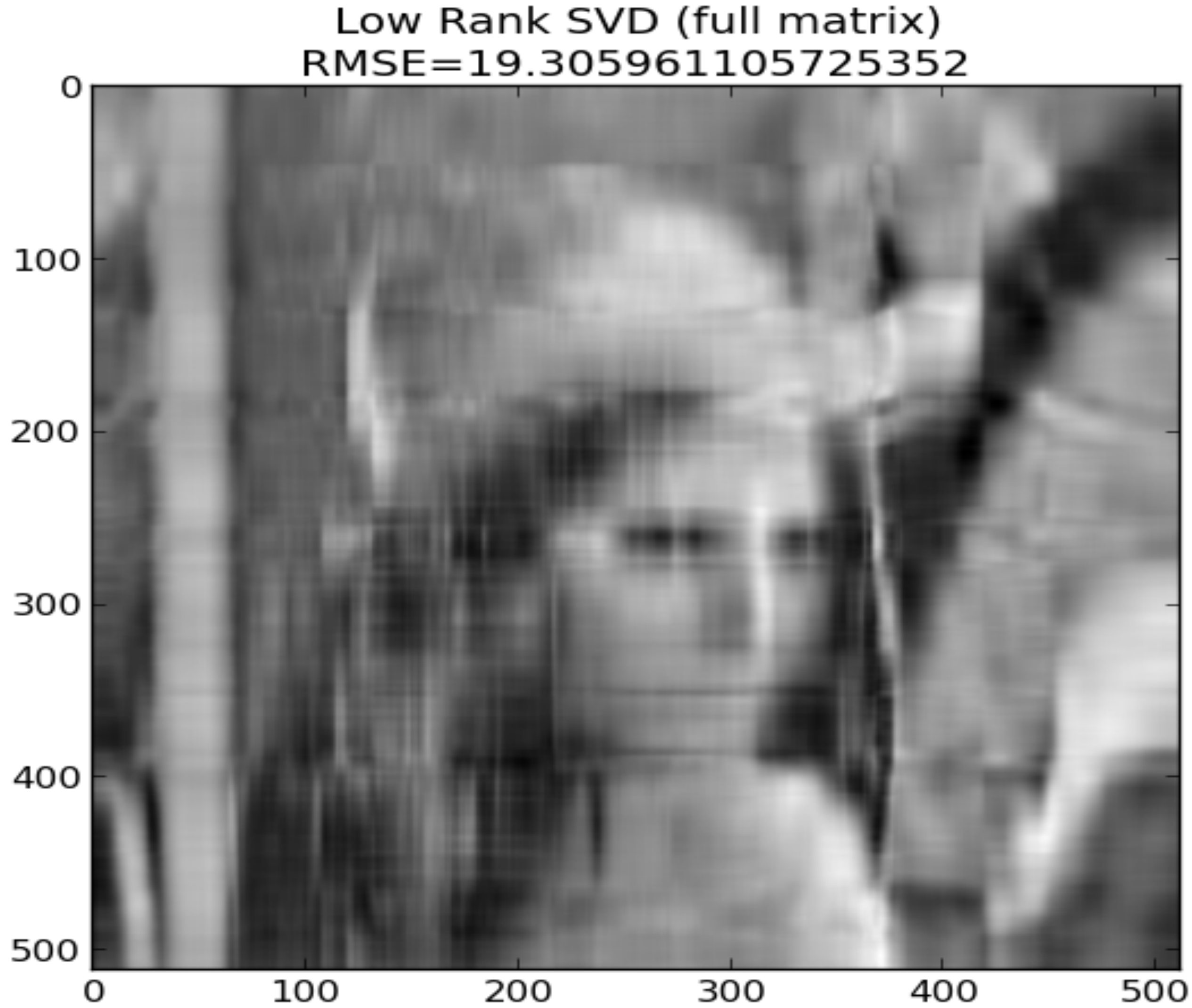
- Python

```
U,S,VT = np.linalg.svd(X, full_matrices=False)
X_ = np.zeros((len(U), len(VT)))
for i in xrange(approx):
    X_ += S[i]*np.outer(U.T[i],VT[i])
```

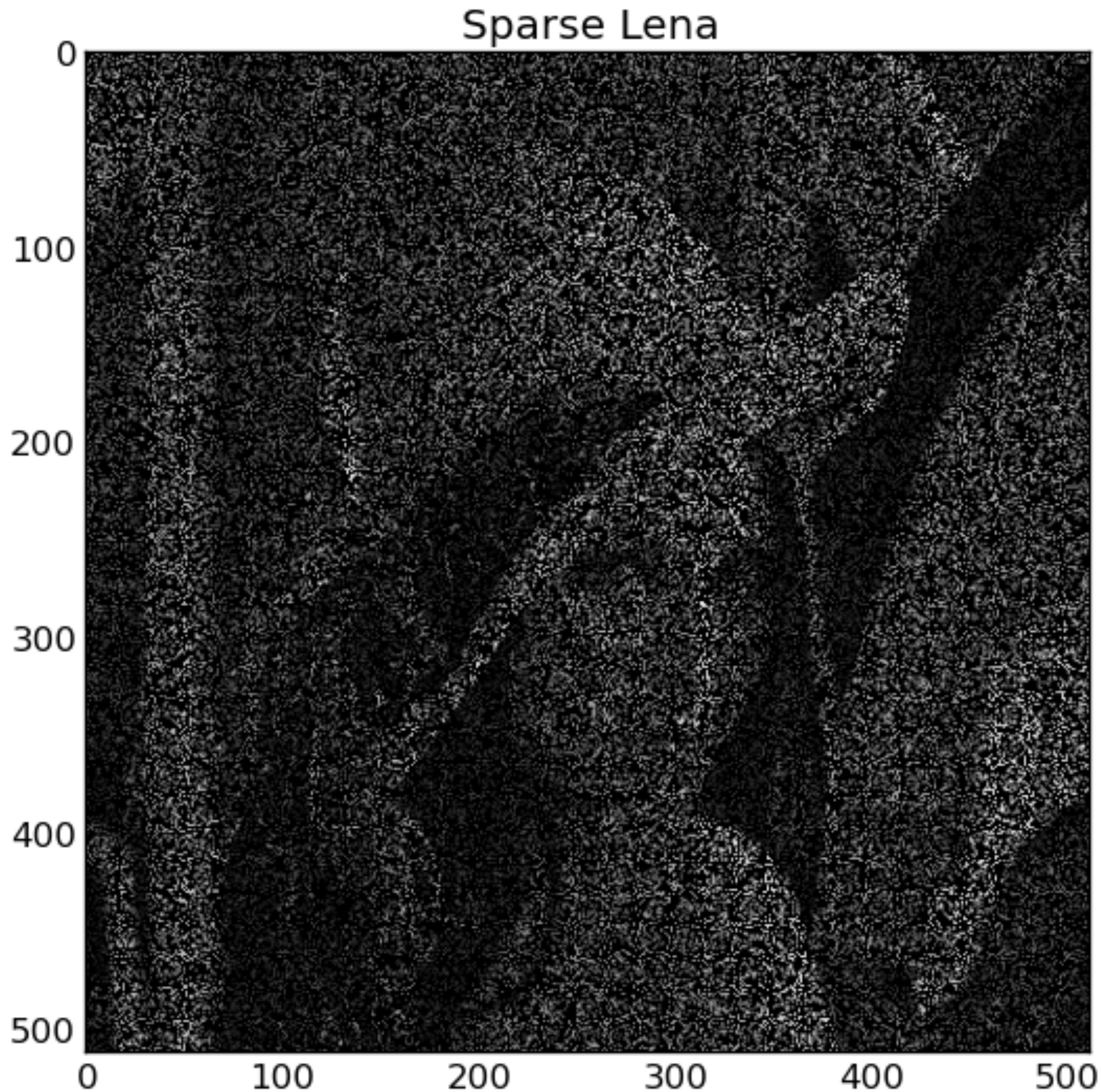
- Standard SVD is just 1 line
- Other two perform low rank approximation

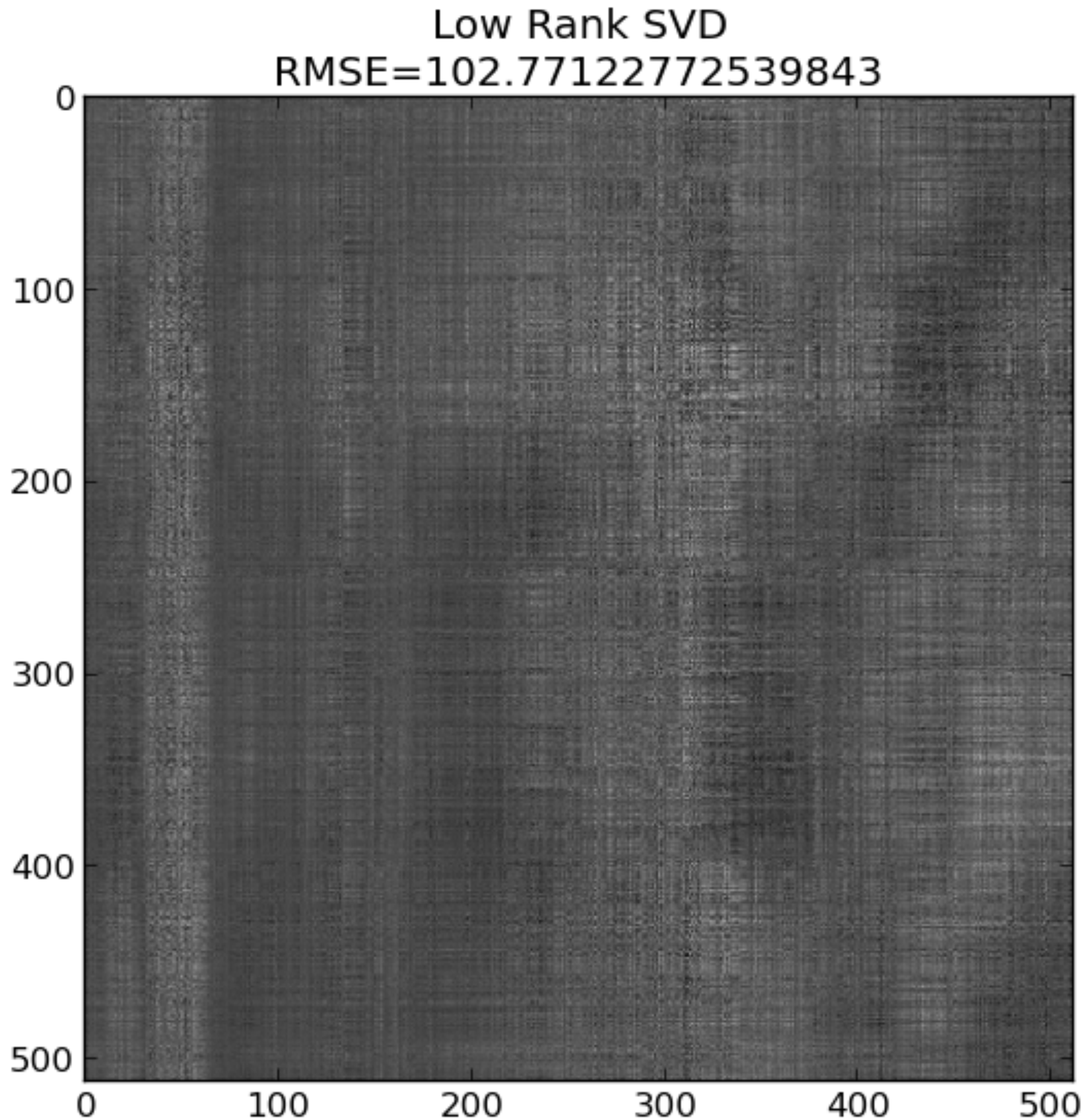
Original Lena





Approximation with 21 factors, 7 iterations will be used for all results


Sparse Lena
0
100
200
300
400
500
0
100
200
300
400
500


Low Rank SVD
RMSE=102.77122772539843
0
100
200
300
400
500
0
100
200
300
400
500

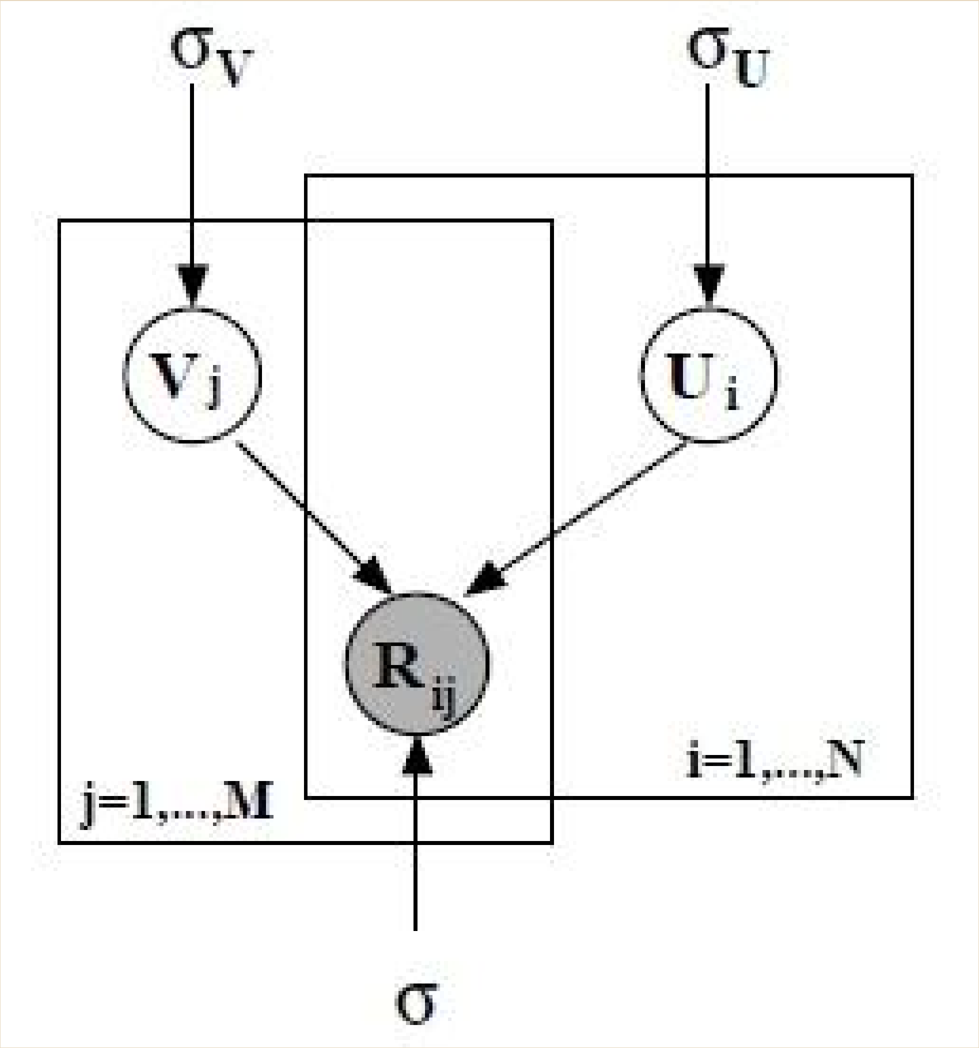
PMF

- Developed in response to Netflix Prize
- Sparse matrix decomposition technique - Netflix data 98%+ empty
- Minimize:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^T V_j)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \|U_i\|_{Fro} + \frac{\lambda_V}{2}\sum_{j=1}^{M} \|V_j\|_{Fro}$$

[1]

- Used for predicting new movie ratings based on "latent" movie and user factors (U, V)



Graphical model for PMF [1]

PMF

- Minimize using gradient descent:

$$V_{i,j} = V_{i,j} - \alpha \frac{\delta}{\delta V}$$

$$U_{i,j} = U_{i,j} - \alpha \frac{\delta}{\delta U}$$

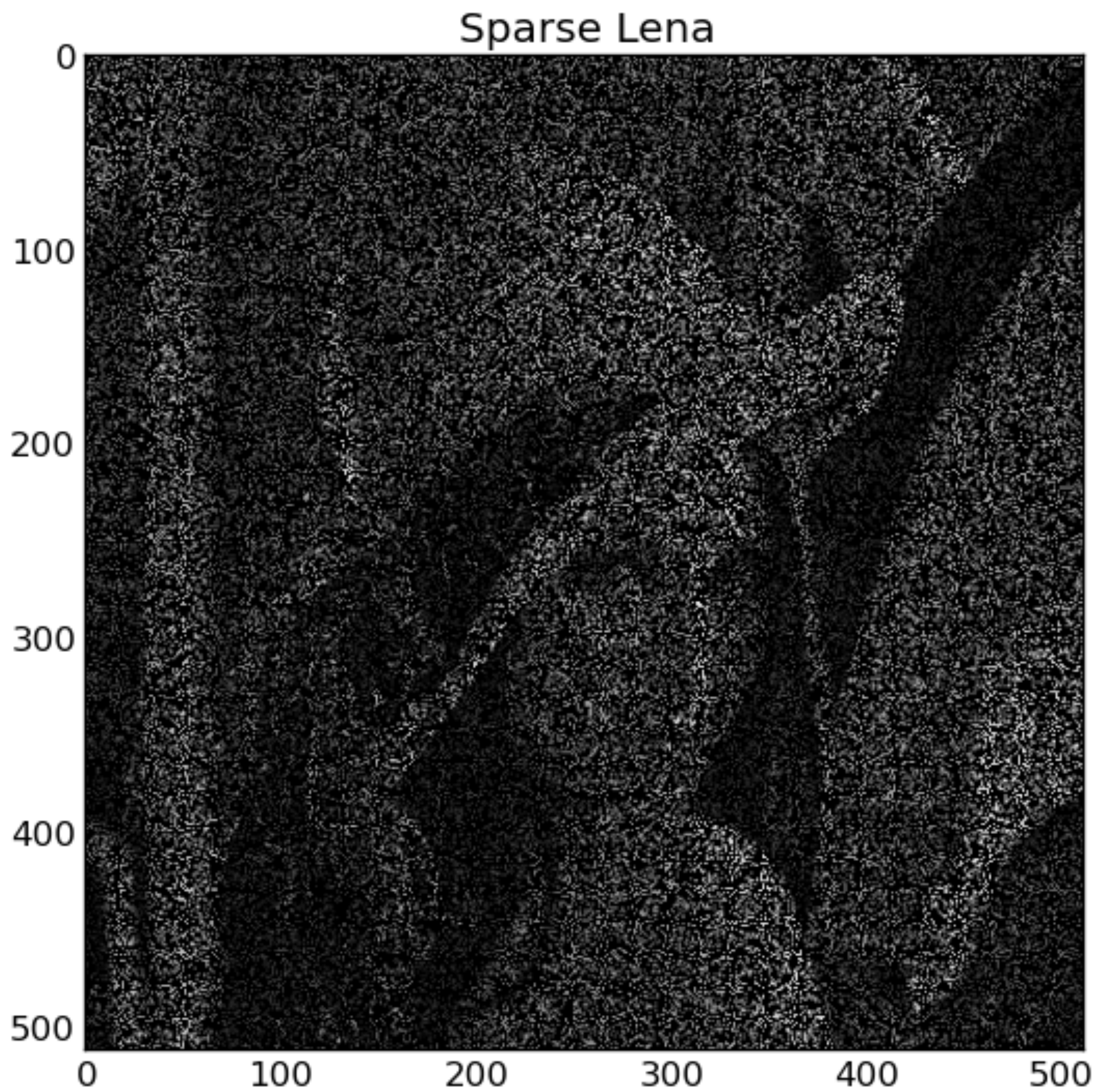
$$\frac{\delta}{\delta U} = \sum_{i=1}^{N} \sum_{j=1}^{M} I_{i,j}(R_{i,j} - U_i^T V_j) V_j + \lambda_U \sum_{i=1}^{N} U_i$$

$$\frac{\delta}{\delta V} = \sum_{i=1}^{N} \sum_{j=1}^{M} I_{i,j}(R_{i,j} - U_i^T V_j) U_i^T + \lambda_V \sum_{j=1}^{M} V_j$$

PMF

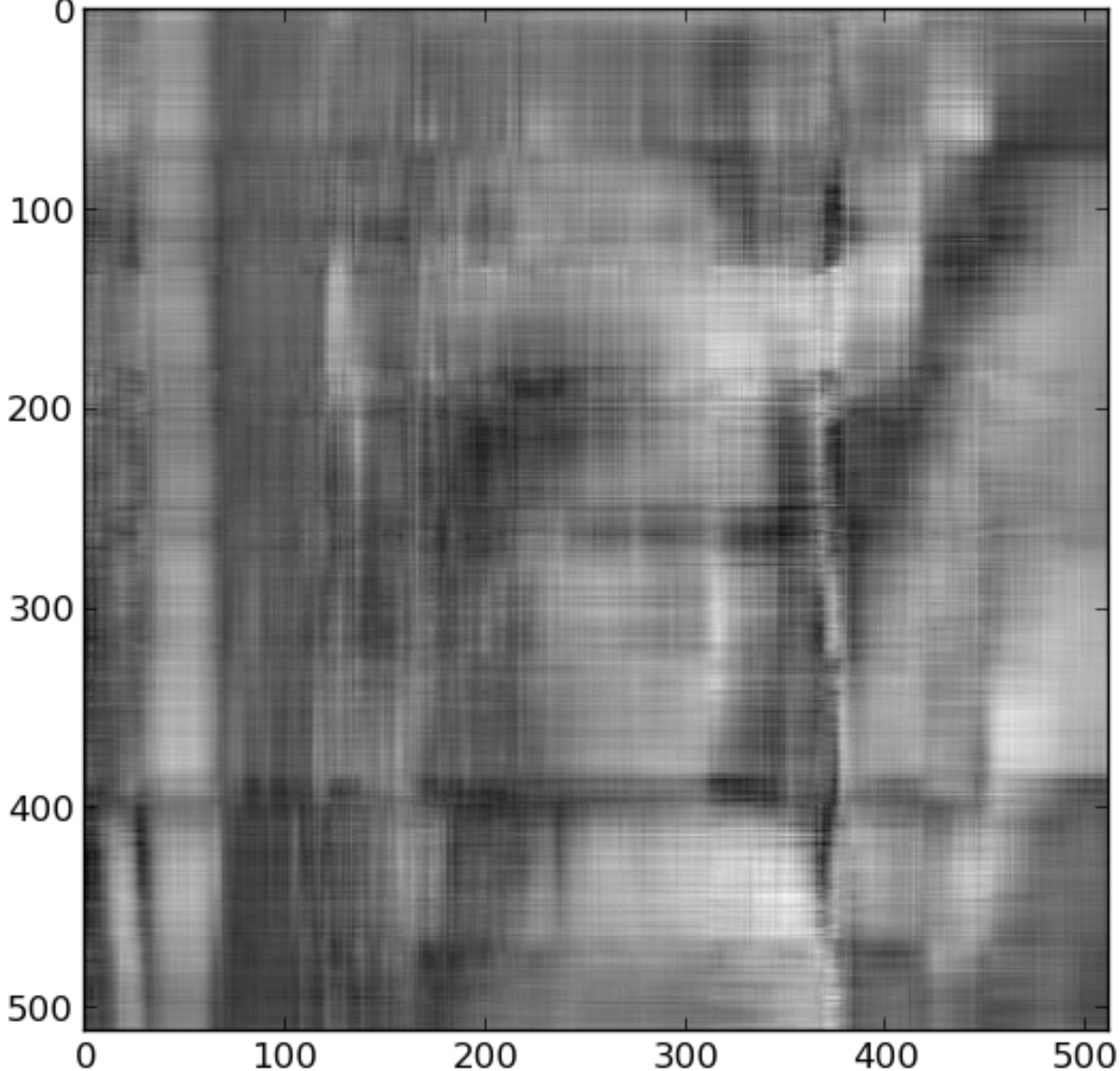
- Python

```
a = alpha
b = lamda_j = lambda_v
for i in range(N):
    for j in range(M):
        if I[i,j] > 0:
            e = A[i,j] - np.dot(U[i,:],V[:,j])
            U[i,:] = U[i,:] + a*(e*V[:,j] - b*U[i,:])
            V[:,j] = V[:,j] + a*(e*U[i,:] - b*V[:,j])
```



Sparse Lena
0
100
200
300
400
500
0
100
200
300
400
500

PMF, λ=0.5
RMSE=46.431530935357046



KPMF

- Uses connectivity matrix to add "side information"
- Originally focused on social recommendations
- Also has uses in image damage correction

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{i,j}(R_{i,j} - U_i V_j^T)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} U_i^T S_U U_i + \frac{\lambda_V}{2}\sum_{j=1}^{M} V_j^T S_V V_j$$

[5]

KPMF

- Minimize using gradient descent

$$V_{i,j} = V_{i,j} - \alpha \frac{\delta}{\delta V}$$

$$U_{i,j} = U_{i,j} - \alpha \frac{\delta}{\delta U}$$

$$\frac{\delta}{\delta V} = \sum_{i=1}^{N} \sum_{j=1}^{M} I_{i,j}(R_{i,j} - U_i^T V_j) U_i + \sum_{j=1}^{M} S_V V_j$$

$$\frac{\delta}{\delta U} = \sum_{i=1}^{N} \sum_{j=1}^{M} I_{i,j}(R_{i,j} - U_i^T V_j) V_j + \sum_{i=1}^{N} S_U U_i$$

KPMF

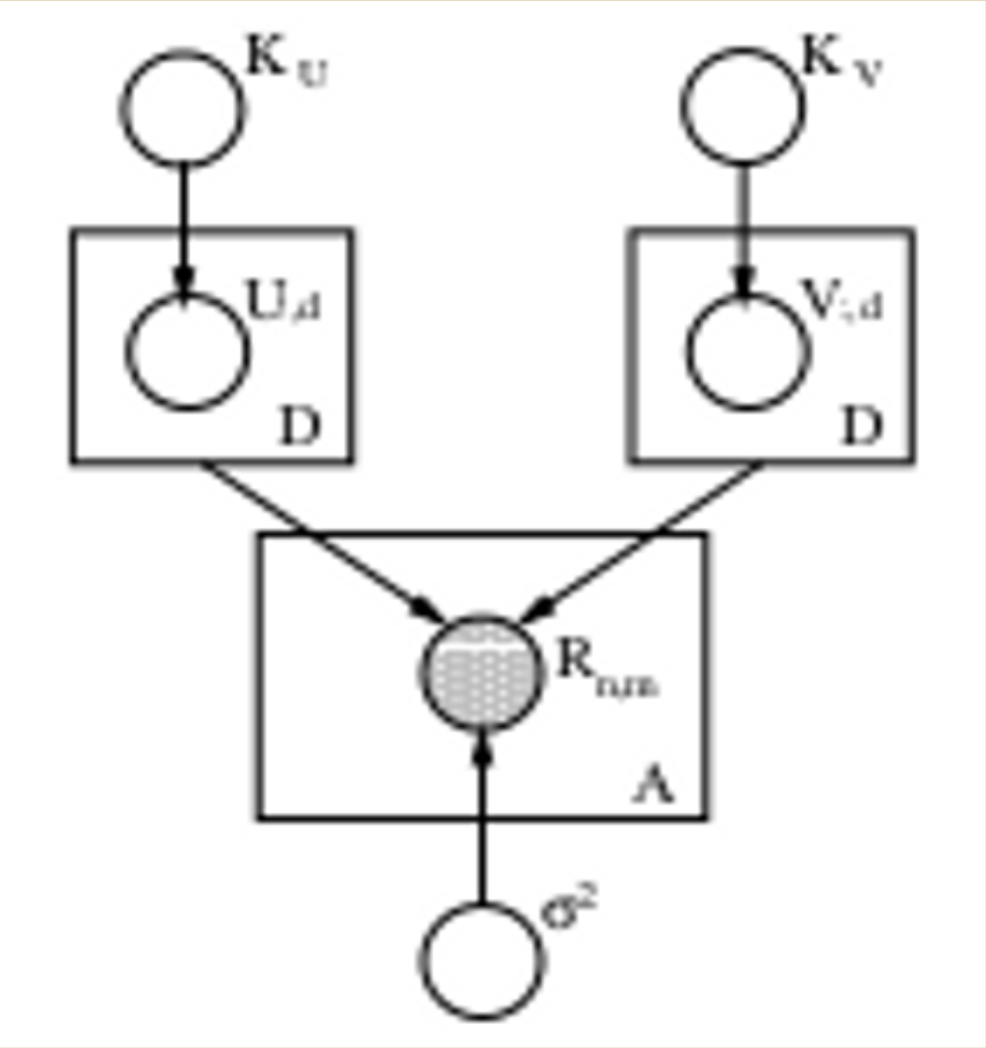
- Kernel is generated by Laplacian of adjacency matrix and an equation (D is [illegible]trix of A)

$$L = D - A$$

- Diffusion equation

$$e^{-\beta L}$$

- Adjacency matrix is a band matrix with adjustable bandwidth for diffusion

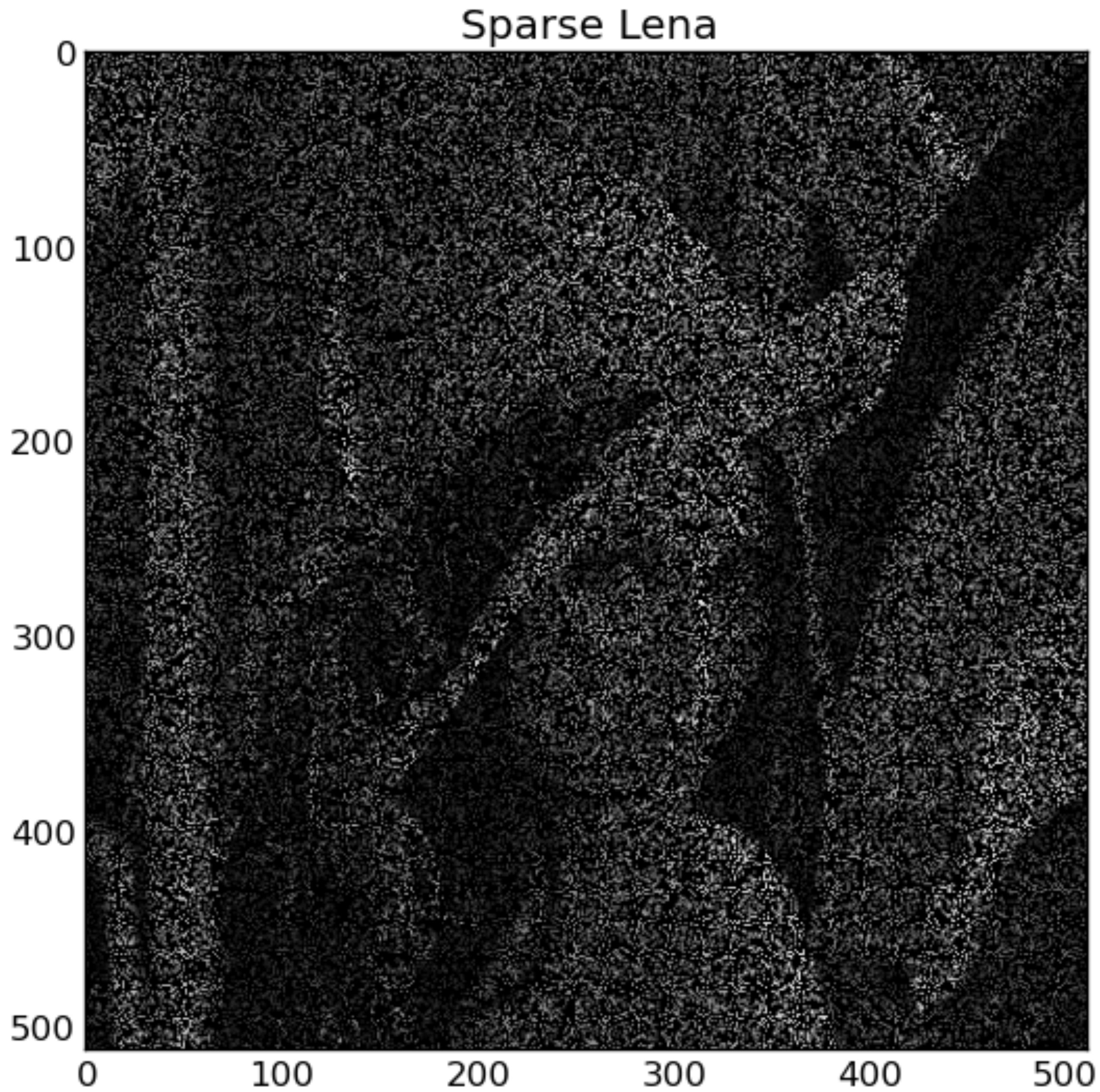


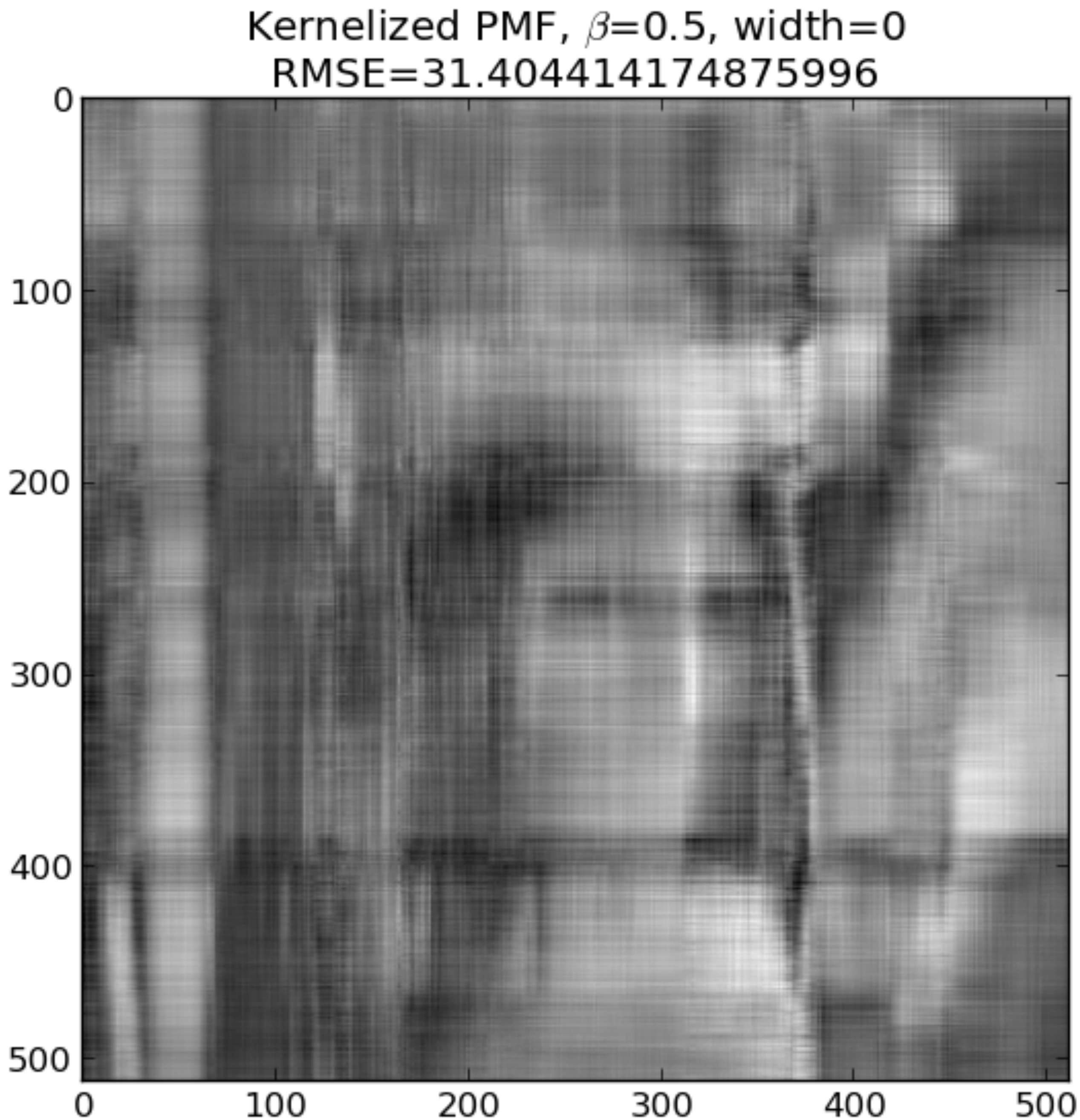
Graphical model for KPMF [5]

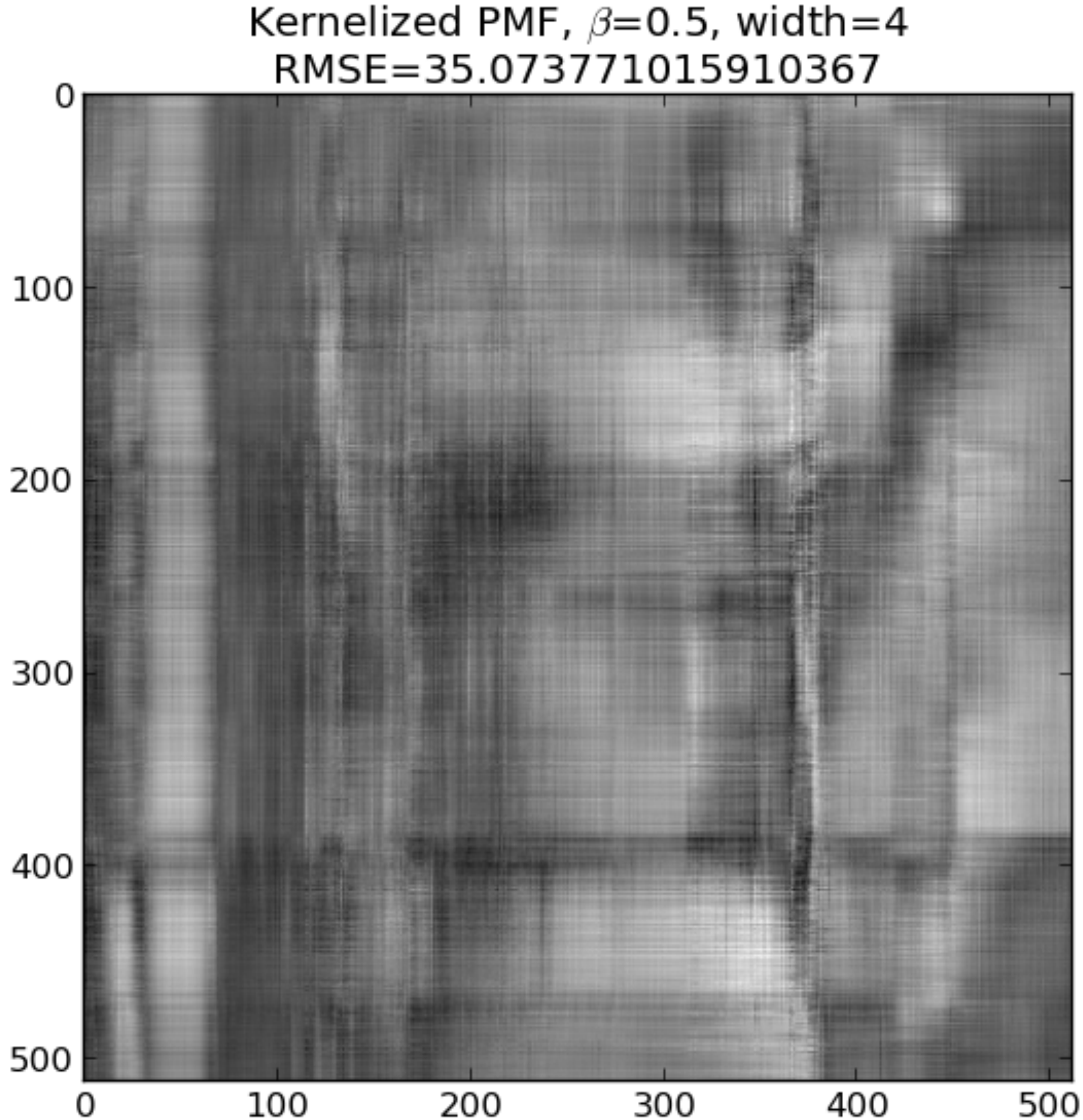
KPMF

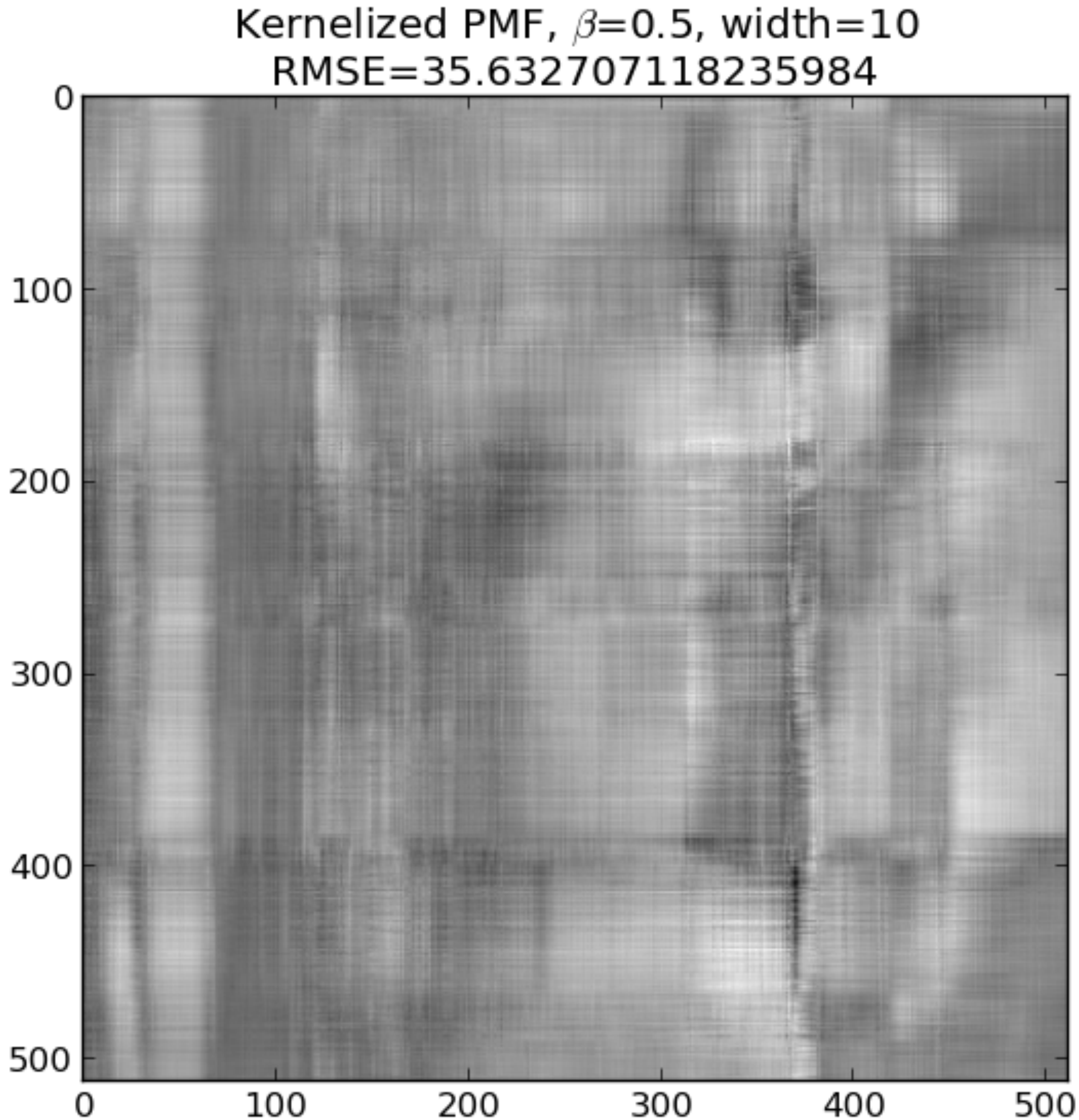
- Python

```
for r in range(R):
    for i in range(N):
        for j in range(M):
            if I[i,j] > 0:
                e = A[i,j] - np.dot(U[i,:],V[:,j])
                U[i,:] = U[i,:] + l*(e*V[:,j] - np.dot
(SU[i,:],U))
                V[:,j] = V[:,j] + l*(e*U[i,:] - np.dot
(V,SV[:,j]))
```

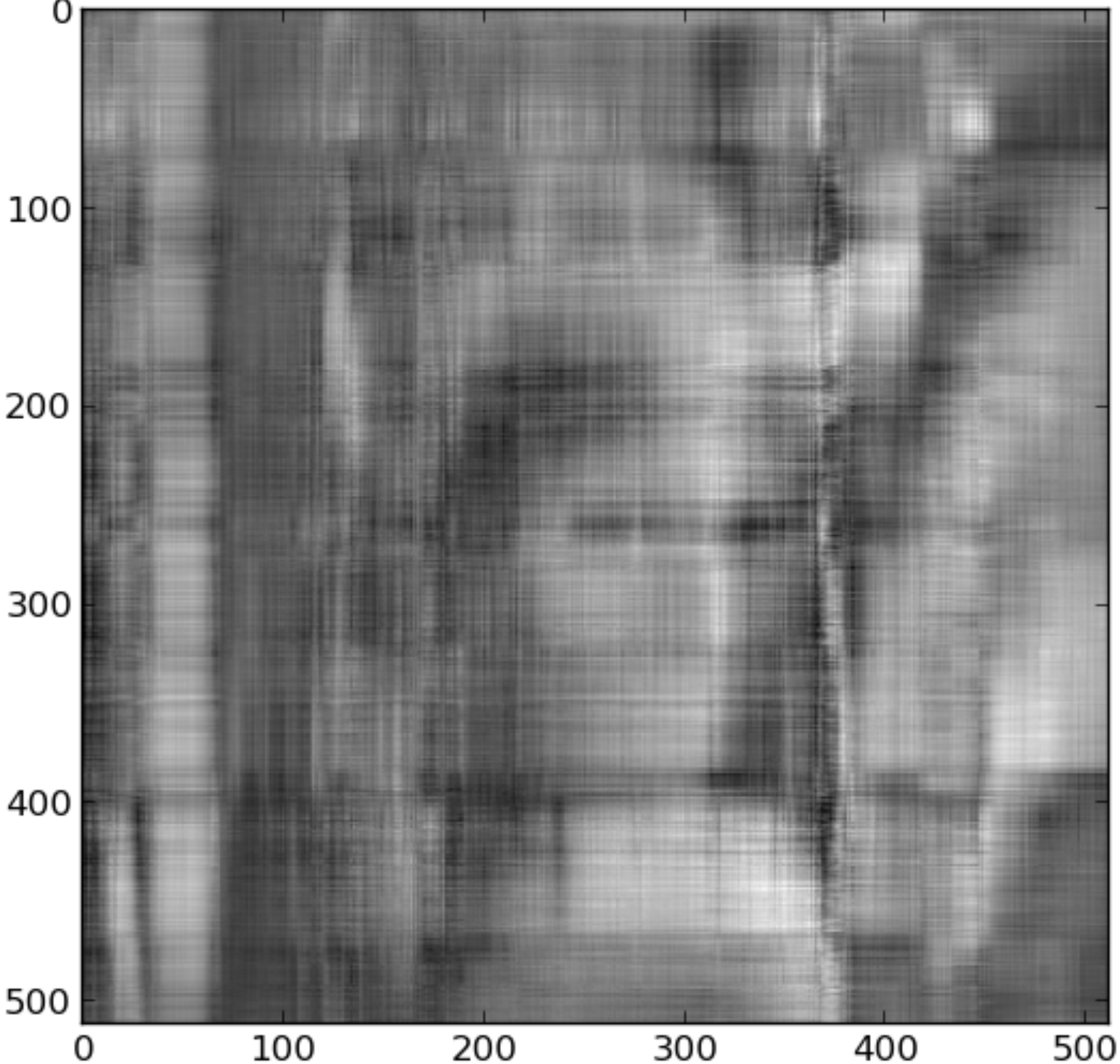

Sparse Lena
0
100
200
300
400
500
0
100
200
300
400
500

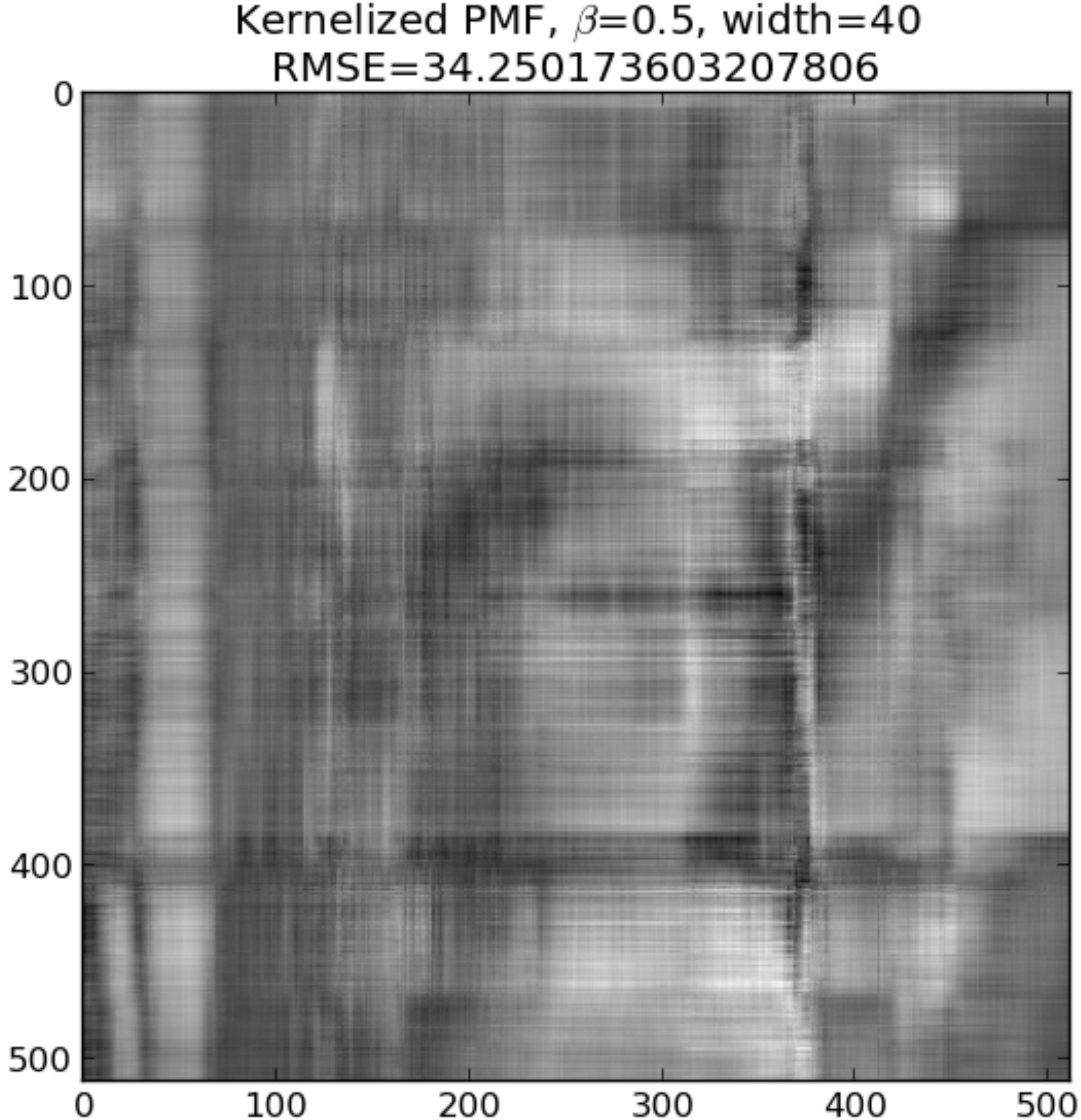

Kernelized PMF, β=0.5, width=0
RMSE=31.404414174875996
0
100
200
300
400
500
0
100
200
300
400
500

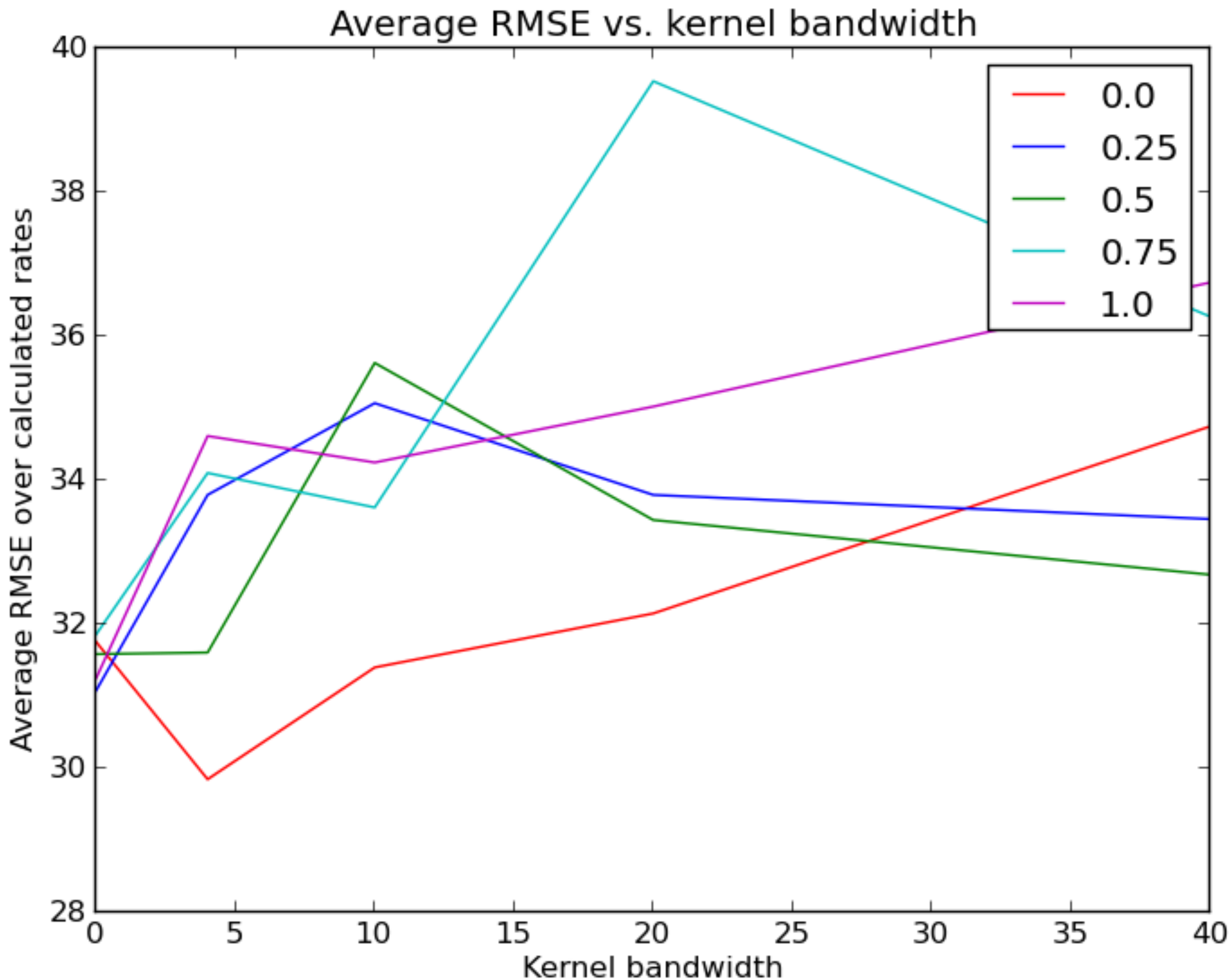

Kernelized PMF, β=0.5, width=4
RMSE=35.073771015910367
0
100
200
300
400
500
0
100
200
300
400
500

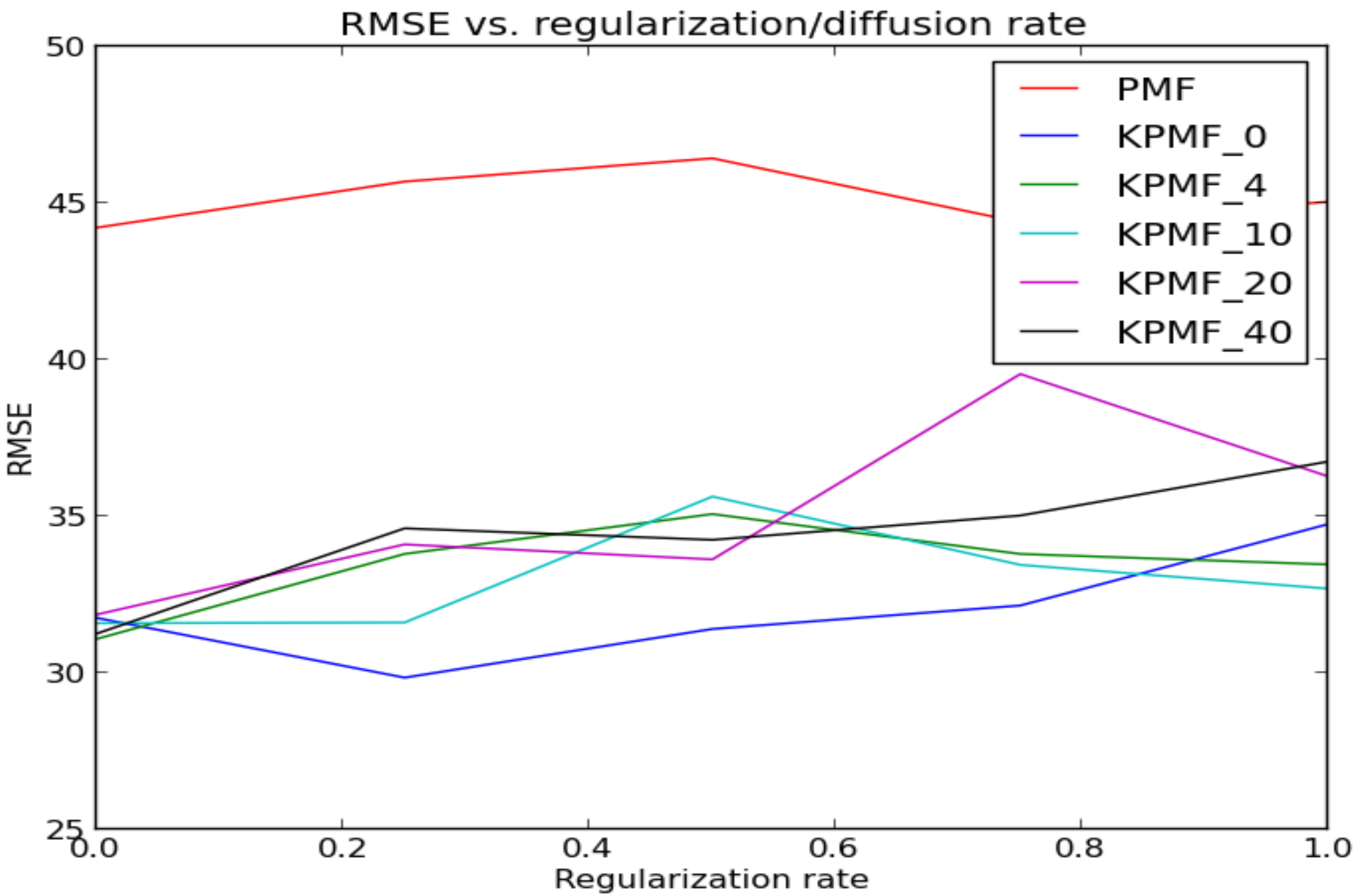

Kernelized PMF, β=0.5, width=10
RMSE=35.632707118235984
0
100
200
300
400
500

Kernelized PMF, β=0.5, width=20
RMSE=33.625623938106301




Kernelized PMF, β=0.5, width=40
RMSE=34.250173603207806
0
100
200
300
400
500
0
100
200
300
400
500


Average RMSE vs. kernel bandwidth
Average RMSE over calculated rates
Kernel bandwidth
0.0
0.25
0.5
0.75
1.0
28
30
32
34
36
38
40
0
5
10
15
20
25
30
35
40



Overall results

Conclusions

- SVD was designed for full matrix decomposition
- SVD can work on sparse data, but tricky
- PMF is a simple, efficient method
- KPMF adds flexibility but also computational complexity
- Choose appropriate model! Not all data can be "linked" in adjacency matrix
- Different kernels provide research opportunities
- Leveraging additional information increases prediction power

Other Factorization Techniques

- Constrained PMF
- Bayesian Probabilistic Matrix Factorization
- Robust PCA
- Robust SVD
- Parametric PMF (PPMF)
- NNMF methods (Non Negative Matrix Factorization)

Questions?

Sources

[1]http://www.cs.toronto.edu/~rsalakhu/papers/nips07_pmf.pdf

[2]http://public.lanl.gov/mewall/kluwer2002.html

[3]ftp://ftp.dca.fee.unicamp.

br/pub/docs/vonzuben/ia900_1s12/notas_de_aula/notas_de_aula_Parte12.pdf

[4]http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.2702&rep=rep1&type=pdf

[5]http://www.ece.duke.edu/~lcarin/kpmf_sdm_final.pdf