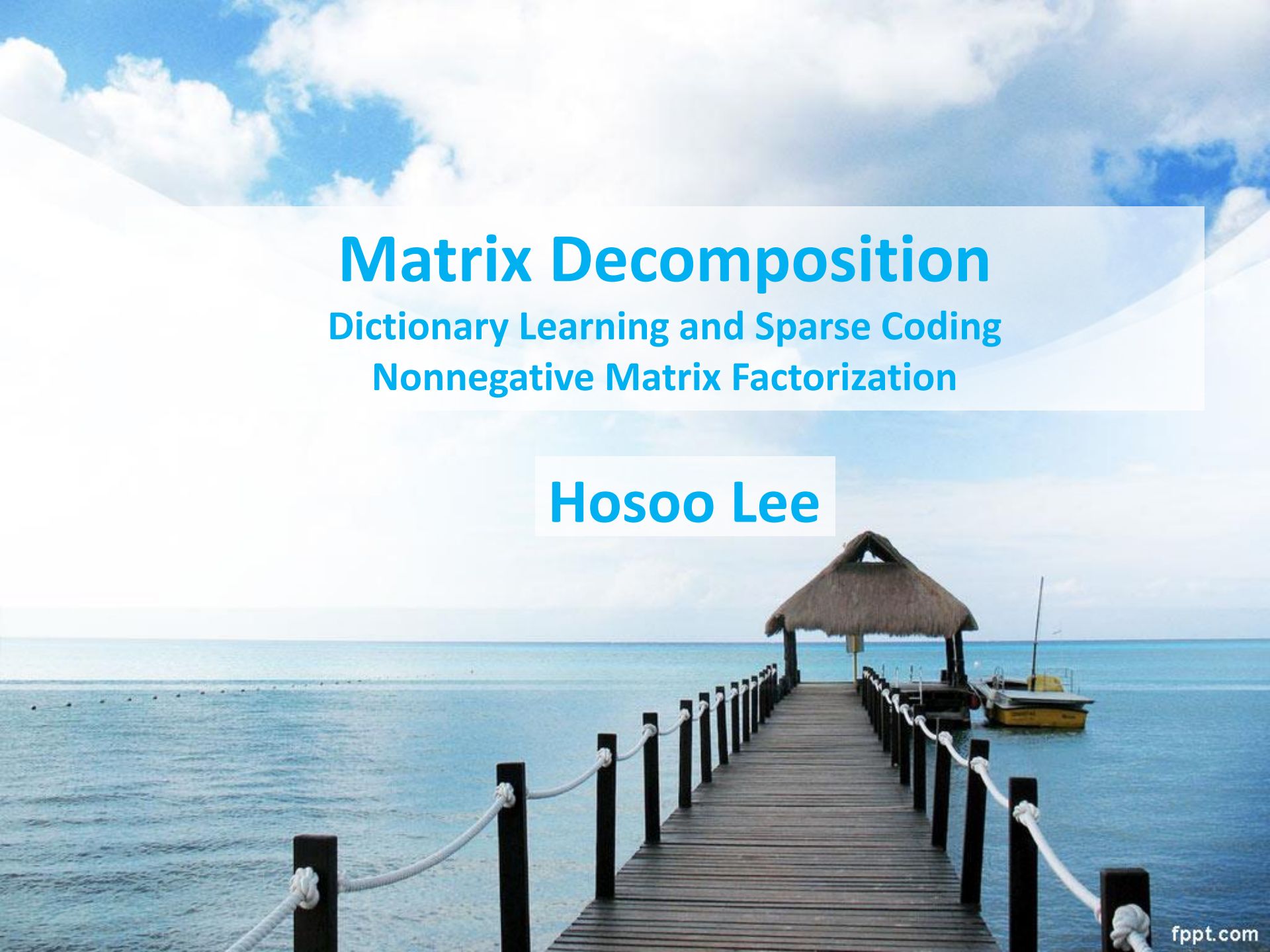
Matrix Decomposition

Dictionary Learning and Sparse Coding
Nonnegative Matrix Factorization

Hosoo Lee



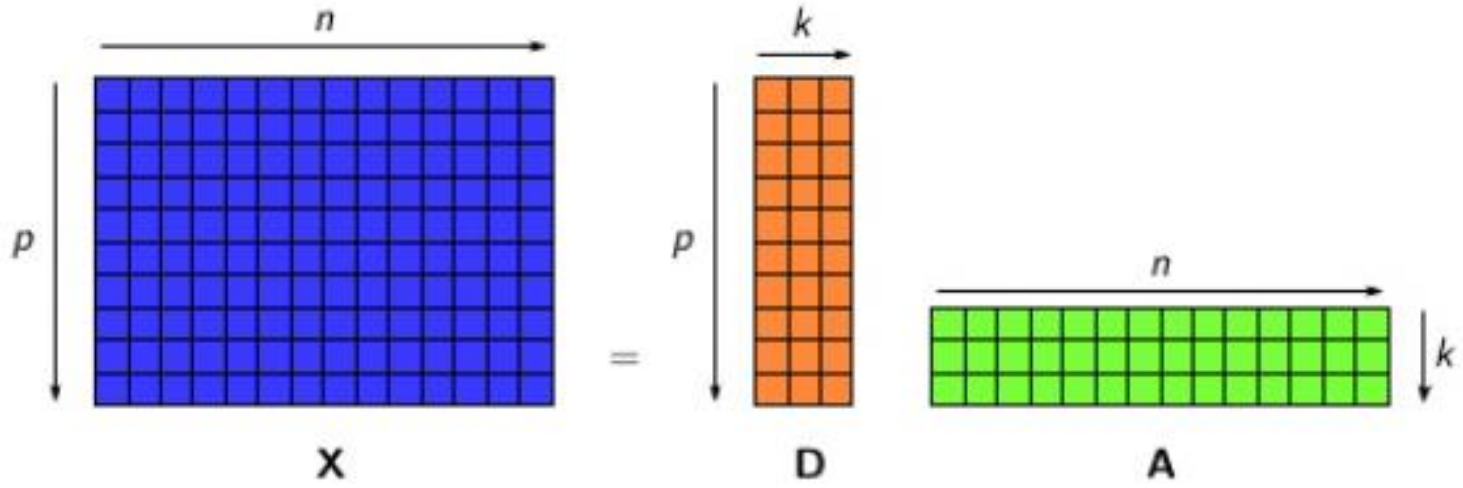
Matrix Decomposition

Matrix Decomposition

PCA(Principal component analysis),
SVD(Singular-value decomposition),
NMF(Non-negative matrix factorization)

LDA(Linear discriminant analysis),
DL(Dictionary Learning),
Sparse Coding,
…

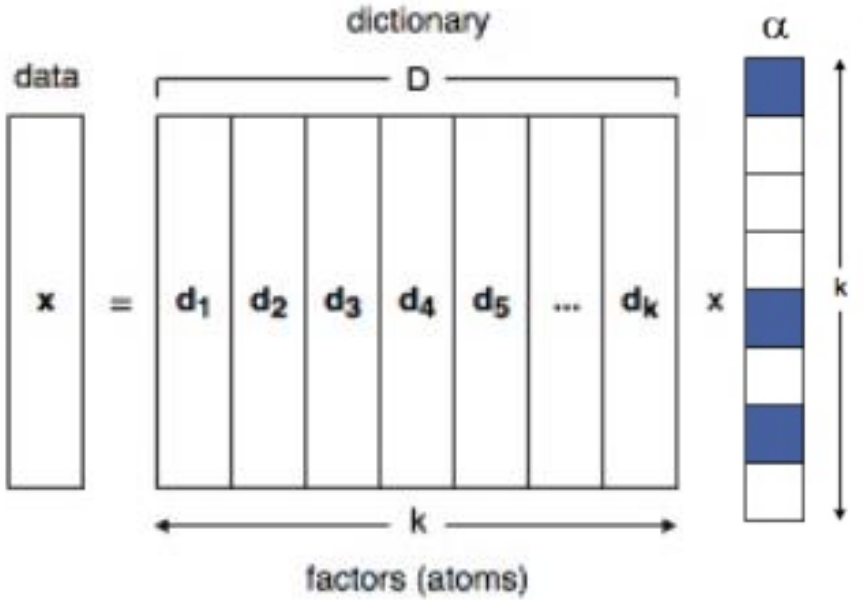
What Is Matrix Decomposition?



- We wish to decompose the matrix X by writing it as a product of two or more matrices:

$$X_{p\times \mathrm{n}} = D_{p\times k}\, A_{k\times n}$$

Matrix factorization



$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} a \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ b \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ c \end{pmatrix} = a\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + b\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + c\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\begin{pmatrix} a+b \\ 2a+b \\ 2b \end{pmatrix} = \begin{pmatrix} a \\ 2a \\ 0 \end{pmatrix} + \begin{pmatrix} b \\ b \\ 2b \end{pmatrix} = a\begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} + b\begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 0 & 2 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix}$$

Why We Need Matrix Decomposition?

$$(x_1, x_2, \dots, x_n) = \boldsymbol{D_{p\times k}}(a_1, a_2, \dots a_n)$$

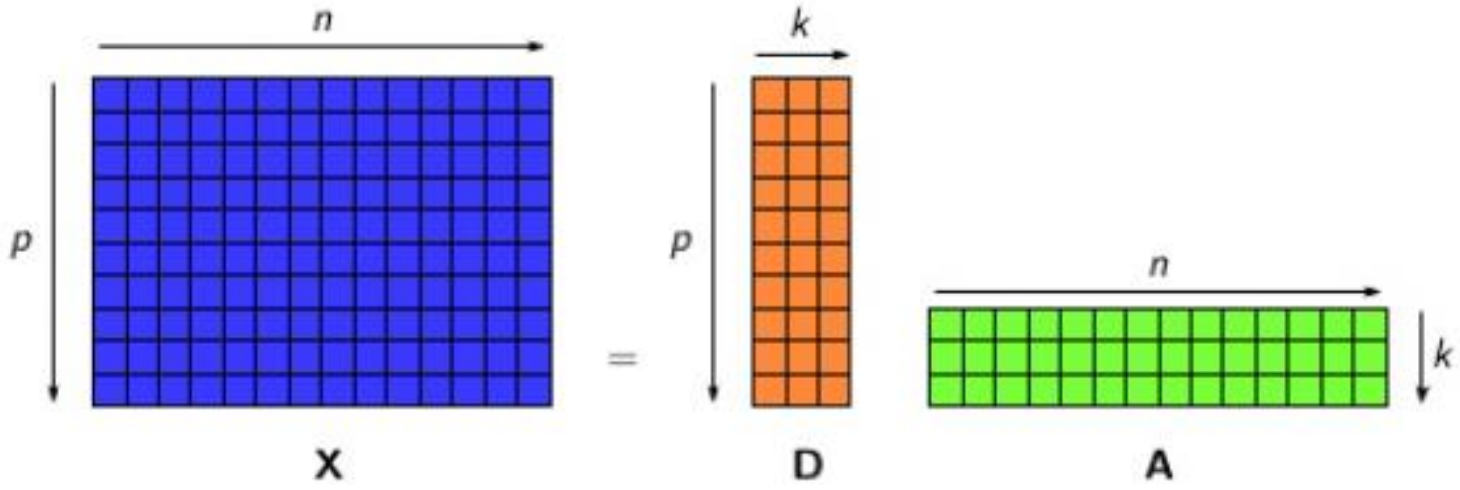
$$\boldsymbol{X_{p\times n}} = \boldsymbol{D_{p\times k}}\, \boldsymbol{A_{k\times n}}$$

We wish to find a set of new basis $\boldsymbol{D}$ to represent data samples $\boldsymbol{X}$, and $\boldsymbol{X}$ will become $\boldsymbol{A}$ in the new space.

In general, $\boldsymbol{D}$ captures the common features in $\boldsymbol{X}$,
while $\boldsymbol{A}$ carries specific characteristics of the original samples.

- In PCA: $\boldsymbol{D}$ is eigenvectors of covariance matrix
- In SVD: $\boldsymbol{D}$ is right (column) eigenvectors
- In LDA: $\boldsymbol{D}$ is discriminant directions

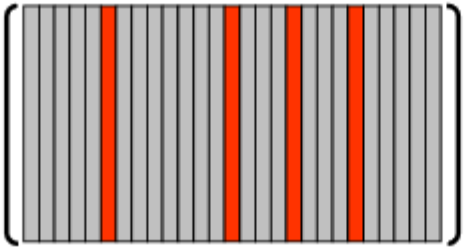
Matrix factorization



$p \geq k$

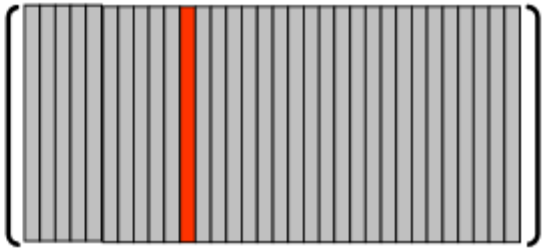
PCA

$p < k$



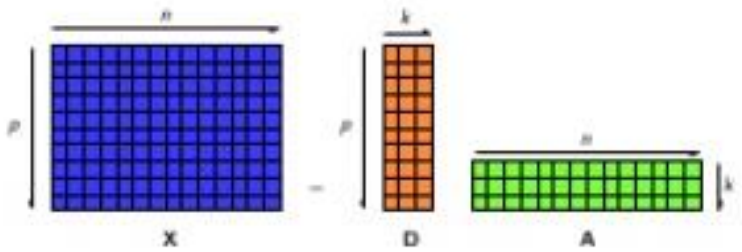
Sparse Coding

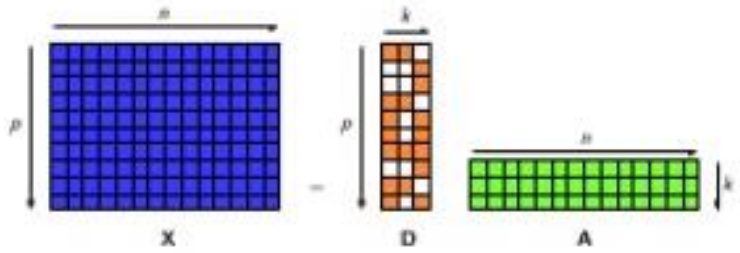
$p \ll k$



Clustering

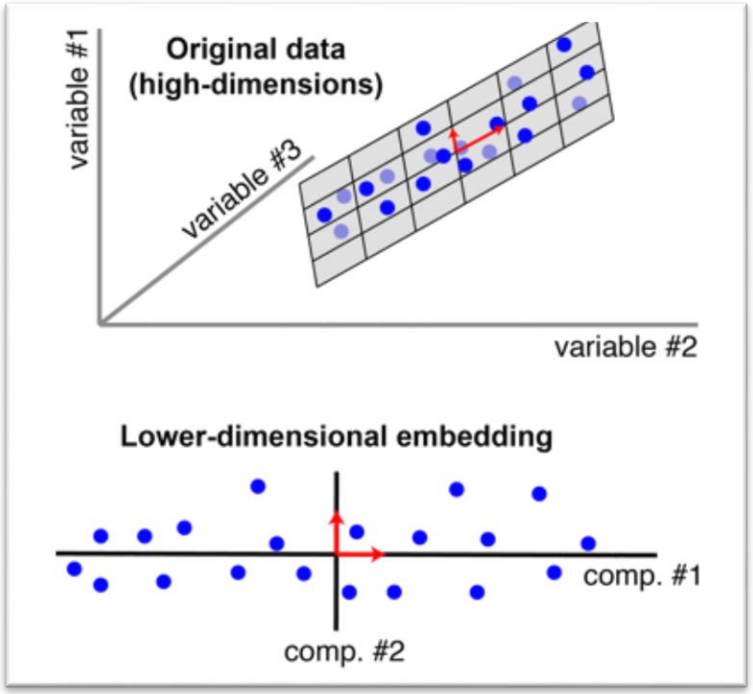
Low rank factorization

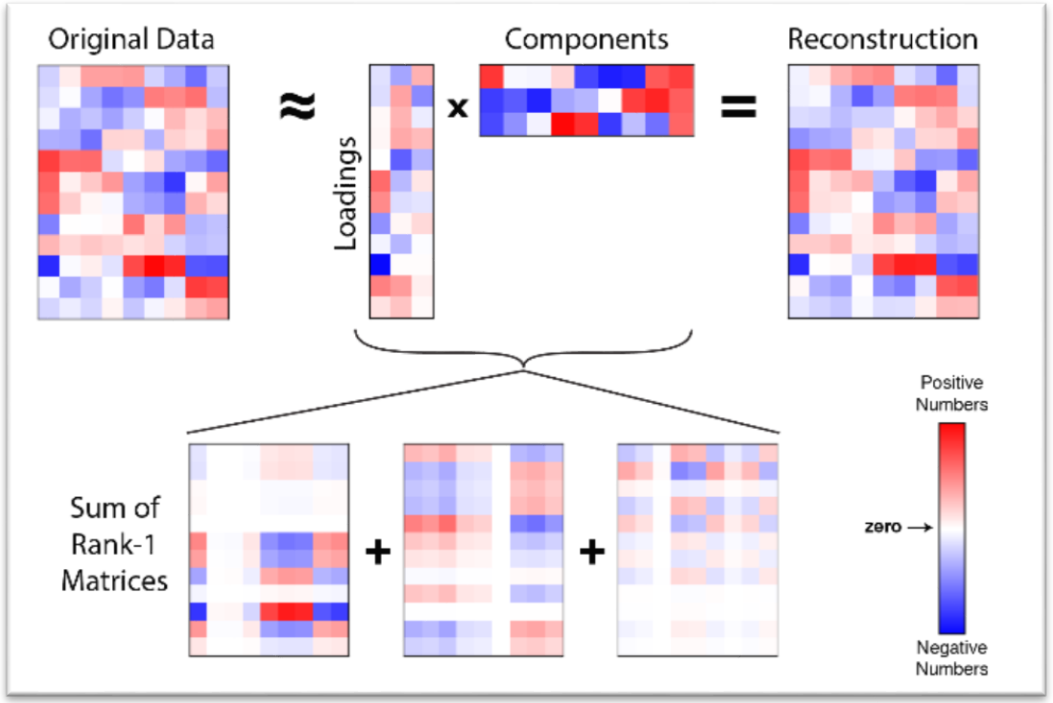




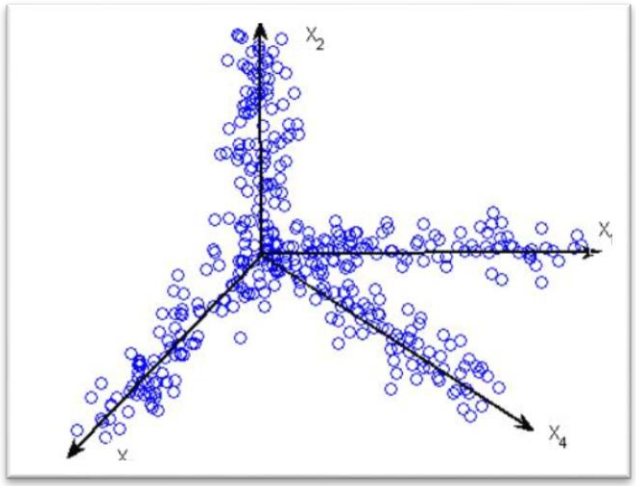
Low rank factorization : $k < p$

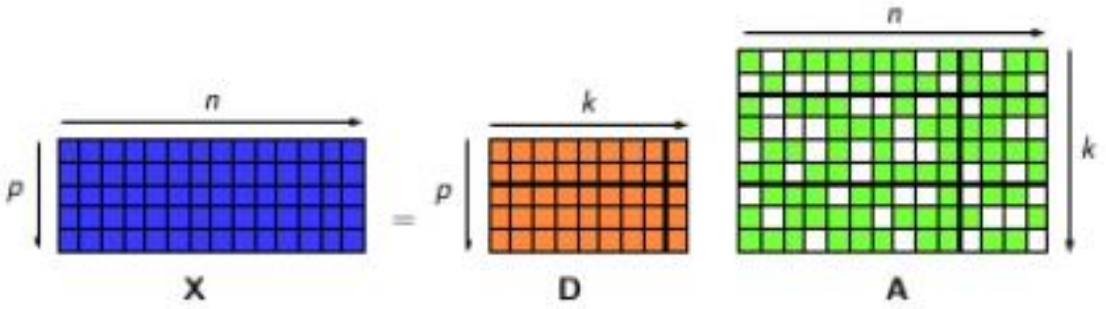
… with optional sparse factors



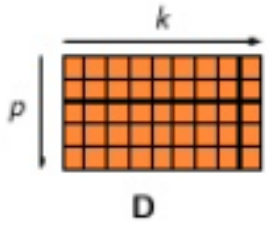


Matrix factorization





Overcomplete dictionary learning $k >> p$



Dictionary matrix

$$D = [d_1\ d_2\ \ldots . d_k] \in R^{p \times k}$$

atoms

Sparse Coding

Overcomplete basis

Complete

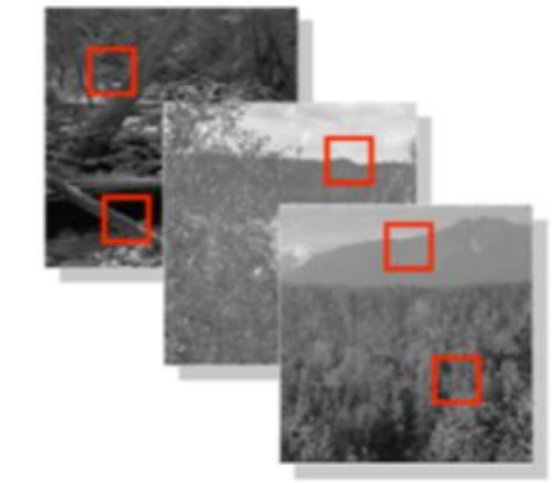
A system $\{y_i\}$ in a Banach space X is $complete$ if every element in *X* can be approximated by linear combinations of elements in $\{y_i\}$

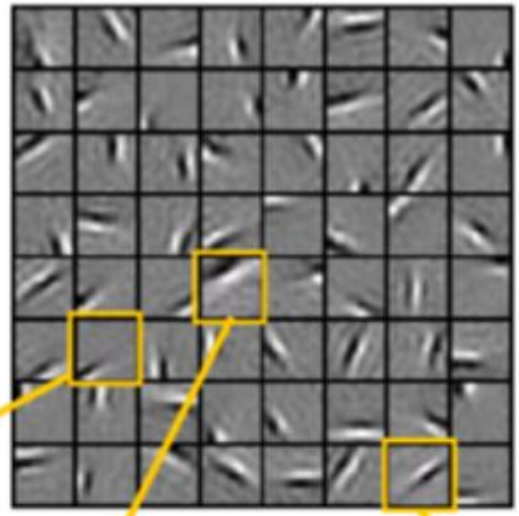
Overcomplete

If removal of an element from $\{y_i\}$ results in a complete system

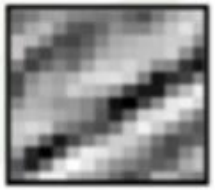
$$\Phi = [e_1|e_2|e_3] = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}$$

Basis for 8×8 fixel image





Test Example



$\approx 0.8\times$

ϕ_{36}

$+ \ 0.3\times$

ϕ_{42}

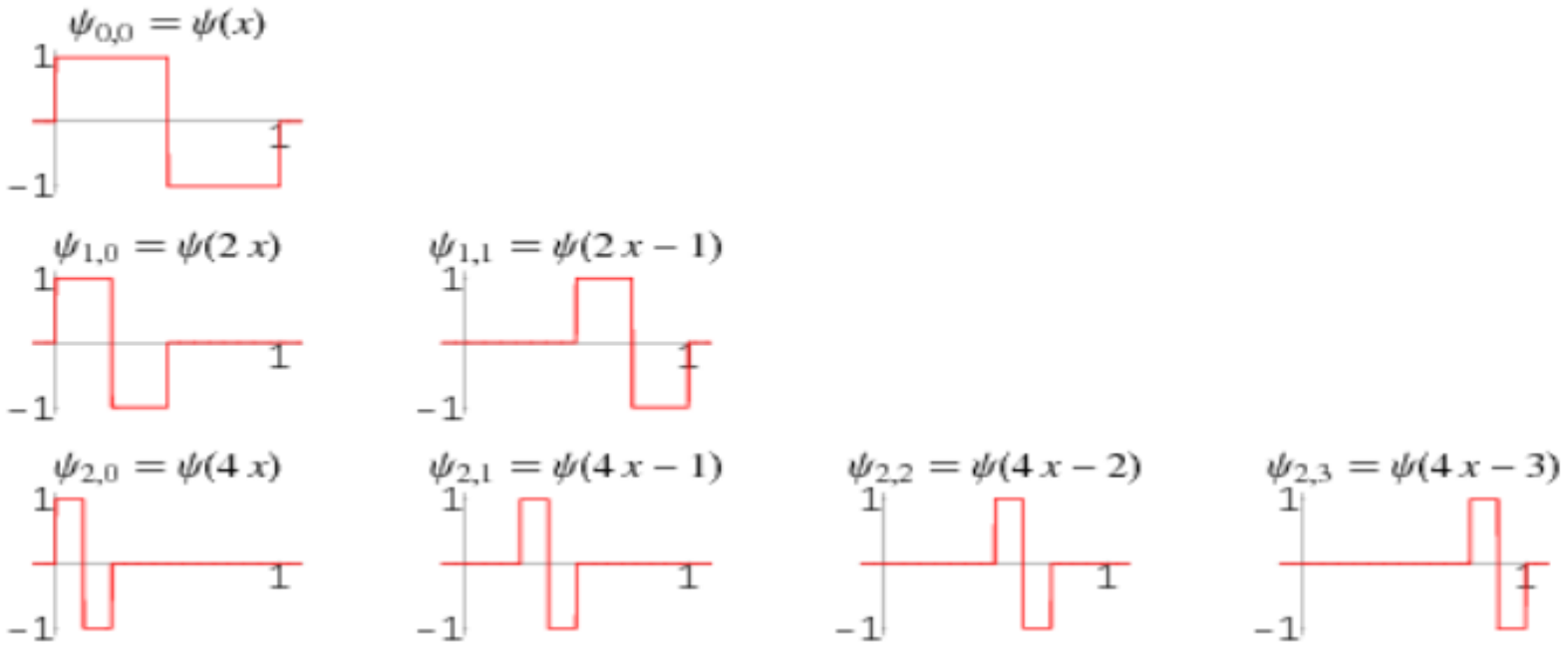
$+ \ 0.5\times$

ϕ_{63}

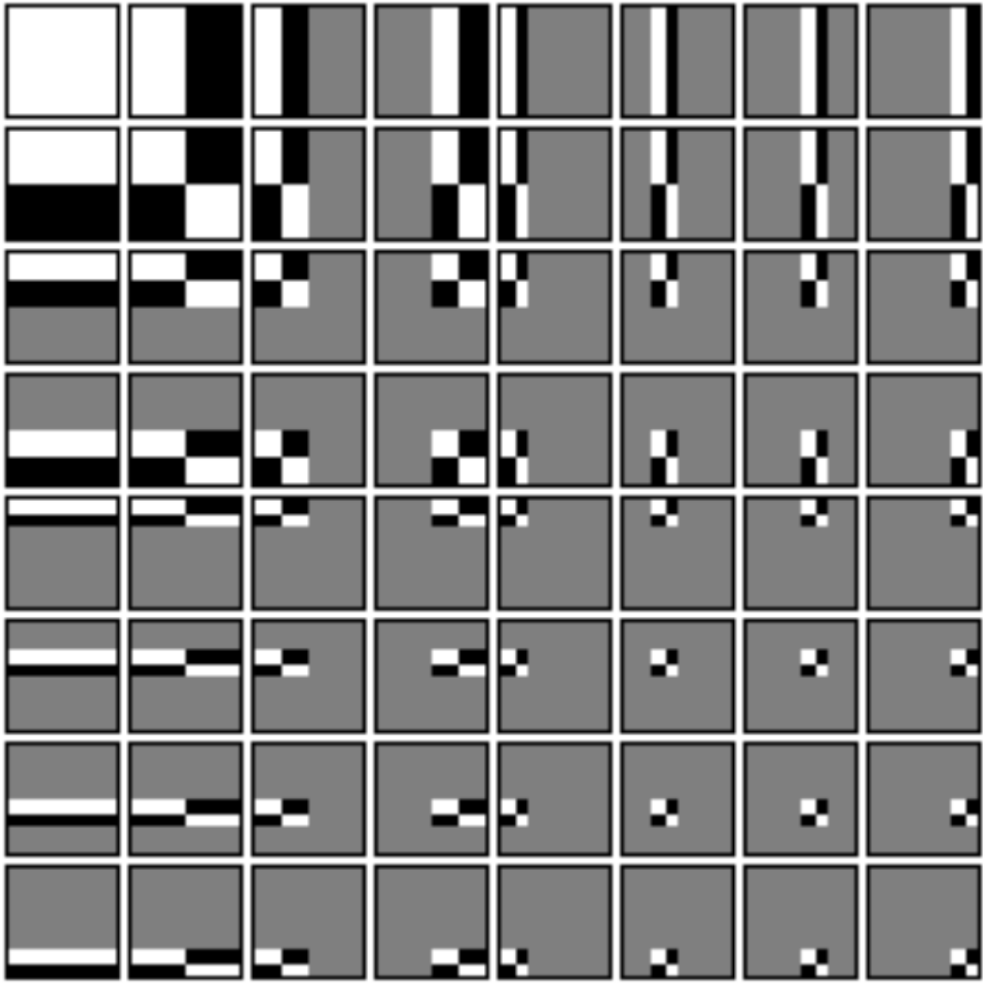
$$[\alpha_1, \ldots, \alpha_{64}] = [0, \ldots, 0.8, \ldots, 0.3, \ldots, 0.5, \ldots, 0]$$

Haar Wavelet Basis

$$\psi(x) = \begin{cases} 1 & \leq x < \frac{1}{2} \\ -1 & \frac{1}{2} \leq x < 1 \\ 0 & otherwise \end{cases} \quad \text{and} \quad \psi_{jk}(x) = \psi(2^j x - k)$$

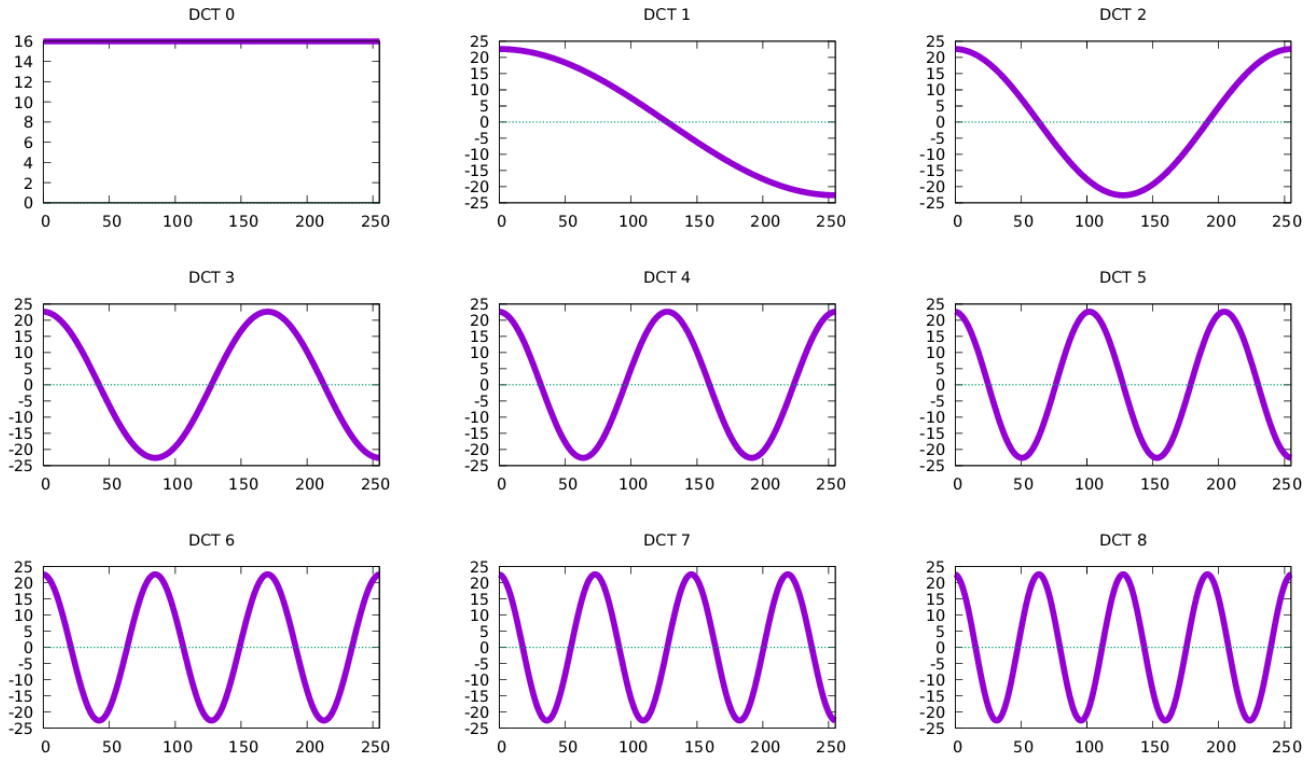


Haar Wavelet Basis



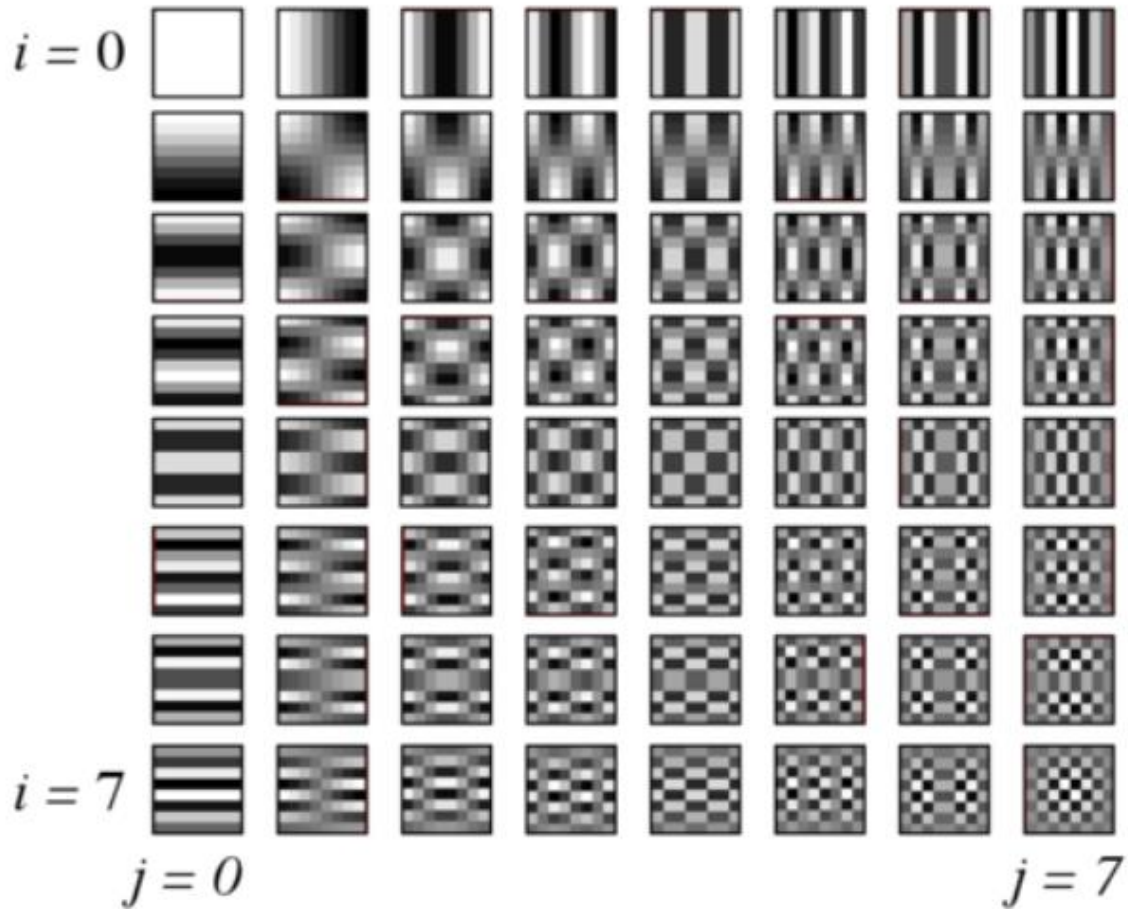
Discrete cosine transform

$$basis[i] = \cos\left[\pi \frac{i}{N}\left(x + \frac{1}{2}\right)\right]$$



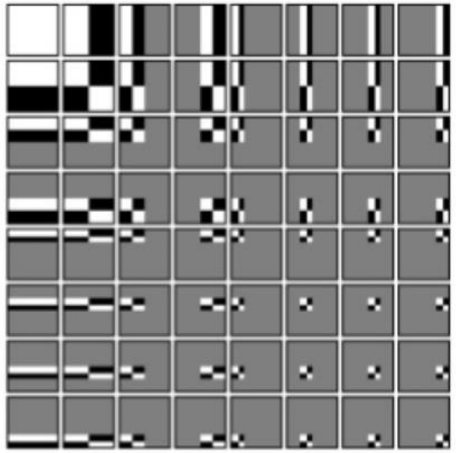
Discrete cosine transform

$$basis[i,j] = \cos\left[\pi \frac{i}{N}\left(x + \frac{1}{2}\right)\right] \times \cos\left[\pi \frac{j}{N}\left(y + \frac{1}{2}\right)\right]$$



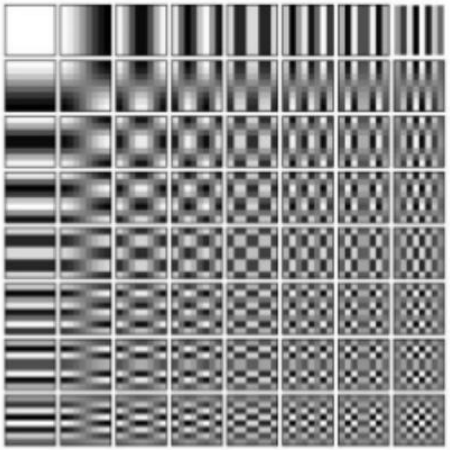
Basis for 8×8 fixel image

Haar Wavelet Basis



Sharp edge

Discrete Cosine Transform

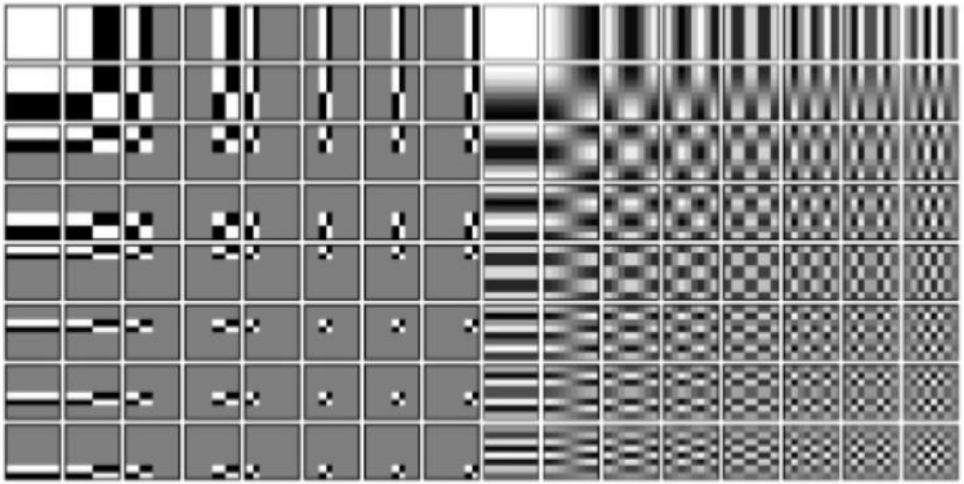


Smooth signal

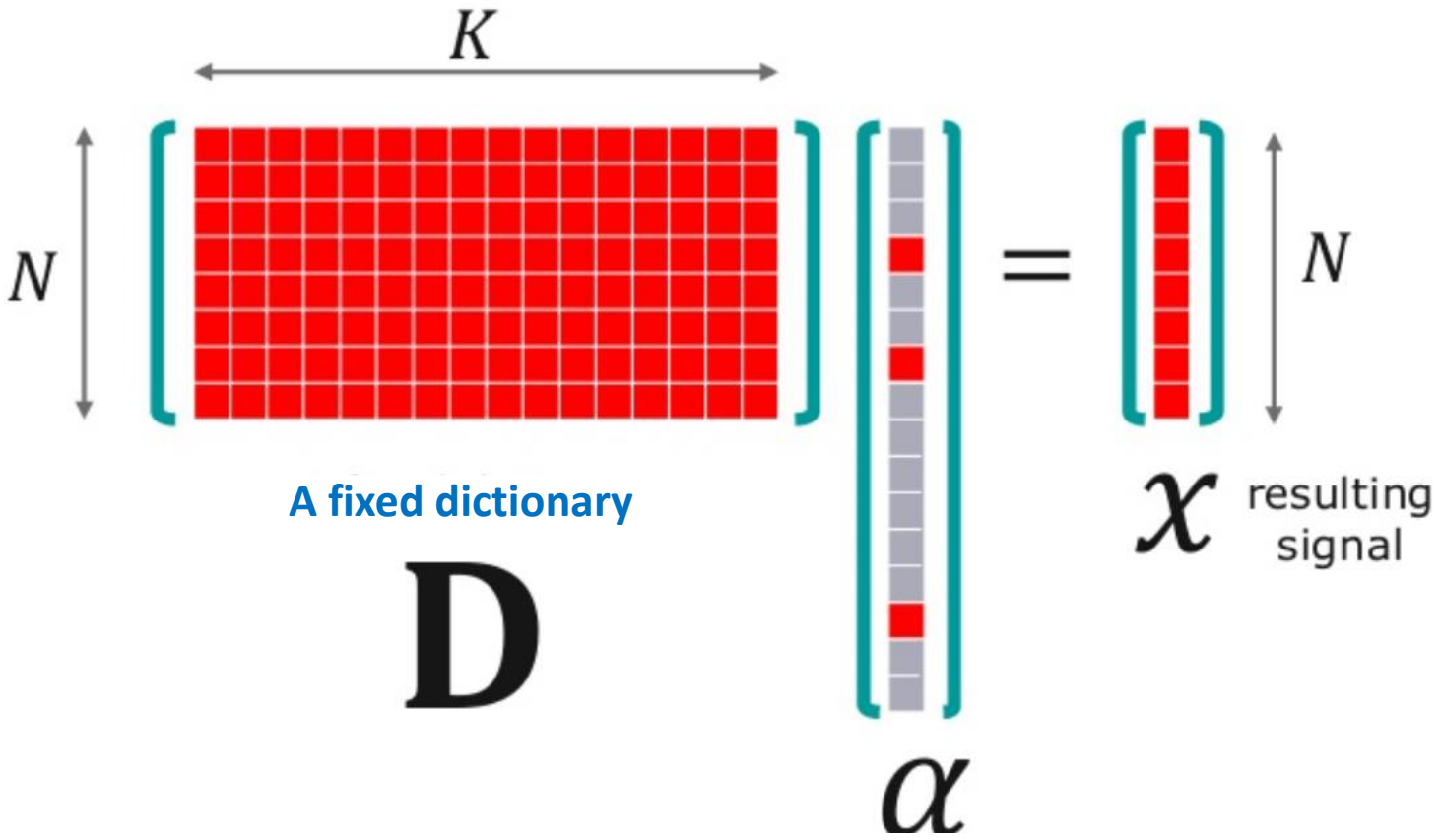
Dictionary for 8×8 fixel image

Mixed Dictionaries

- A dictionary of Harr +DCT gives the best of both worlds



The Sparse Signal Model



How do we pick which coefficients to use?

Solving for Sparsity

What is the minimum number of coefficients we can use?

1. Sparsity Constrained

$$\alpha = \arg\min_{\alpha} \|D\alpha - x\|_2^2 \quad s.t. \quad \|\alpha\|_0 \leq K$$

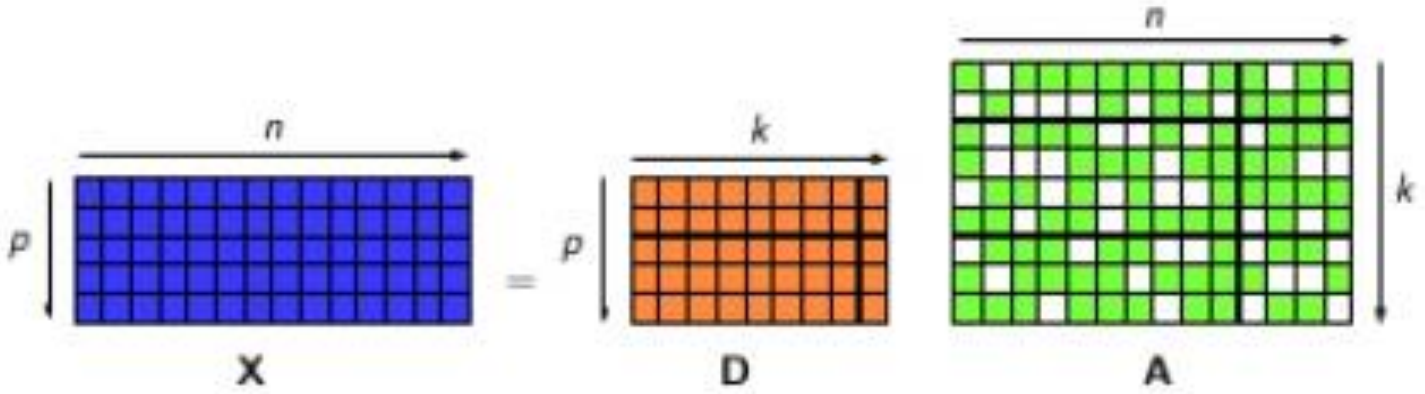
2. Error Constrained

$$\alpha = \arg\min_{\alpha} \|\alpha\|_0 \quad s.t. \quad \|D\alpha - x\|_2^2 \leq \epsilon$$

$||a||_p = \left(\sum_k a_k^p\right)^{1/p}$

$||a||_0 := \lim_{p\to 0} ||a||_p$ =(the number of nonzero elements of a)

Dictionary Learning



Dictionary Learning

Model :

$$\min_{D,A} \|X - DA\|_F^2 \quad \text{subject to } \forall i, \|a_i\|_0 \leq s$$

where

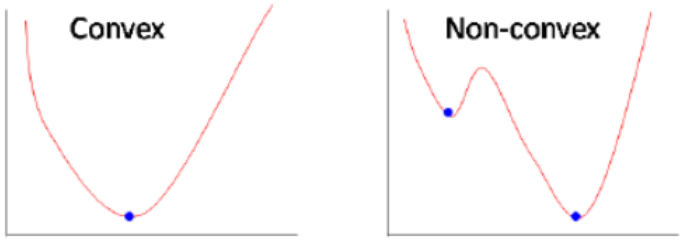
- $X = [x_1 \ \ldots x_p] \in \mathrm{R}^{N \times p}$ represents p training data set, (known)
- $D \in \mathrm{R}^{N \times k}$ is a designed dictionary matrix with k atoms,
- $A = [a_1 \ \ldots a_p] \in \mathrm{R}^{k \times p}$ is a sparse representations matrix with a sparsity level *s*.

Dictionary Learning

Model :

$$\min \|X - DA\|_F^2 \quad \text{subject to } \forall i, \|x_i\|_0 \leq s$$

- Problem is not convex



Eg. $f(x,y) = (2 - xy)^2$

$f(1,3) = f(3,1) = (2-3)^2 = 1$

$f(2,2) = (2-4)^2 = 4$

- Local optimum may not correspond to global optimum
- Generally little hope to find global optimum

Dictionary Learning

Model :

$$\min \|X - DA\|_F^2 \quad \text{subject to } \forall i, \|x_i\|_0 \leq s$$

- Problem is not convex
- But: Problem is biconvex
  - For fixed D, $f(A) = ||X - DA||_F^2$ is convex

$$||A||_F^2 = \sum_j \sum_i a_{ij}^2 = \sum_j \sum_i (A^T)_{ji}(A)_{ij} = \sum_j (A^T A)_{jj} = Tr(A^T A)$$

$$||X - DA||_F^2 = Tr\,(X - DA)^T (X - DA)$$

$$= \cdots = Tr(X^T X) - Tr(X^T DA) - Tr(A^T D^T X) + Tr(A^T D^T DA)$$

  - For fixed A, $f(D) = ||X - DA||_F^2$ is convex
  - Allows for efficient algorithms

Dictionary Learning Algorithm

- Two steps

(i) **Sparse coding:**

Given the dictionary D, producing sparse representations matrix A

$$\min_{A} \|X - DA\|_F^2$$

(ii) **Dictionary update** : (the main innovation)

Given the sparse representations A, updating dictionary D

$$\min_{D} \|X - DA\|_F^2$$

Sparse Coding

- Fix D,
- Aim to find the best coefficient matrix A,

$$\min_{A} \|X - DA\|_F^2 \quad \text{s.t.} \quad \forall i, \|a_i\|_1 \leq S$$

$$\min_{a_i} \|x_i - Da_i\|_F^2 \quad \text{s.t.} \quad \forall i, \|a_i\|_1 \leq S, \ i = 1, \ldots, p$$

(Ordinary Sparse Coding Problem)

- Use an approximate solving methods.

  Orthogonal Matching Pursuit (OMP)

  Basis Pursuit (BP)

  Focal Under-determined System Solver (FOCUSS)

Dictionary Update Stage

- **Goal: search for a better dictionary *D***

$$\min_{D} \|X - DA\|_F^2$$

- **K-SVD [Aharon (2006)]**

  Scheme: update one column of *D* at a time,

  - fixing all columns in *D* except one, d_k,
  - update it by optimizing the target function.

For the *k*-th atom,

$$\|X - DA\|_F^2 = \left\| \left(X - \sum_{j \neq k} d_j g_j^T \right) - d_k g_k^T \right\|_F^2 = \|E_k - d_k g_k^T\|_F^2$$

(Singular Value Decomposition)

Singular Value Decomposition

- For an arbitrary matrix $\mathbf{A}$ there exists a factorization (Singular Value Decomposition = **SVD**) as follows:

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathbf{T}} \in \mathbb{R}^{n\times m}$$

- Where
  - (i) $\mathbf{U} \in \mathbb{R}^{n\times k}$ $\boldsymbol{\Sigma} \in \mathbb{R}^{k\times k}$ $\mathbf{V} \in \mathbb{R}^{m\times k}$
  - (ii) $\mathbf{U}'\mathbf{U} = \mathbf{I}$ $\mathbf{V}'\mathbf{V} = \mathbf{I}$ *orthonormal columns*
  - (iii) $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_k),\ \sigma_i \geq \sigma_{i+1}$ *singular values (ordered)*
  - (iv) $k = \mathrm{rank}(\mathbf{A})$

Low-rank Approximation

- SVD can be used to compute optimal **low-rank approximations**.
- Approximation problem:

$$\mathbf{X}^* = \underset{\hat{\mathbf{X}}:\mathrm{rank}(\mathbf{X})=q}{\operatorname{argmin}} \|\mathbf{X} - \hat{\mathbf{X}}\|_F$$

← *Frobenius norm*

$$\|\mathrm{A}\|_F \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}.$$

- Solution via SVD

$$\mathbf{X}^* = \mathbf{U}\,\mathrm{diag}(\sigma_1, \ldots, \sigma_q, \underbrace{0, \ldots, 0}_{\text{set small singular values to zero}})\,\mathbf{V}'$$

$$\mathbf{X}^* = \sum_{r=1}^{q} \sigma_r \mathbf{u}_r \mathbf{v}_r'$$

← *column notation: sum of rank 1 matrices*

K-SVD Algorithm

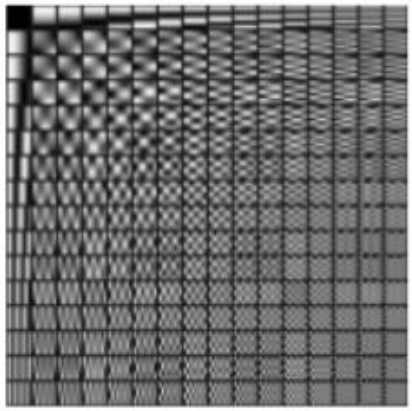
Initialize dictionary $\mathbf{D}$

Sparse Coding

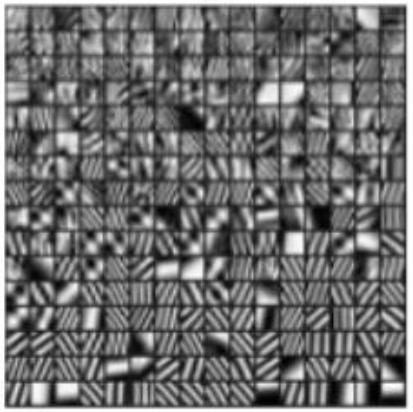
Use **MP or BP, …**

Dictionary update

One atom at a time



D, initialization



D, convergence

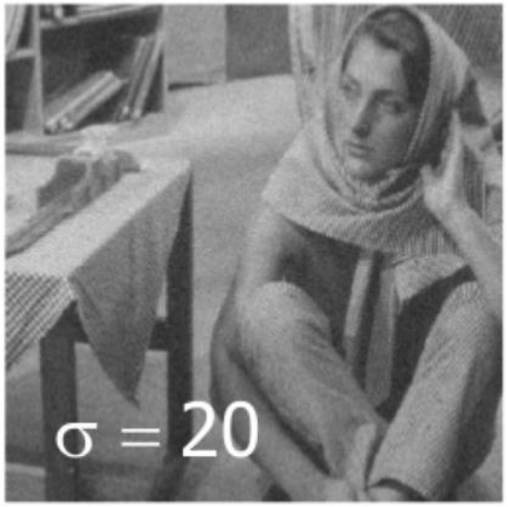
Application of Dictionary Learning

Application of Dictionary Learning

Sparse Denoising

- Uniform noise is incompressible and OMP will reject it
- KSVD can train a denoising dictionary from noisy image blocks

Source



Noisy image

Result 30.829dB

Application of Dictionary Learning

Sparse Denoising

Noisy image: $f = f_0 + \omega$

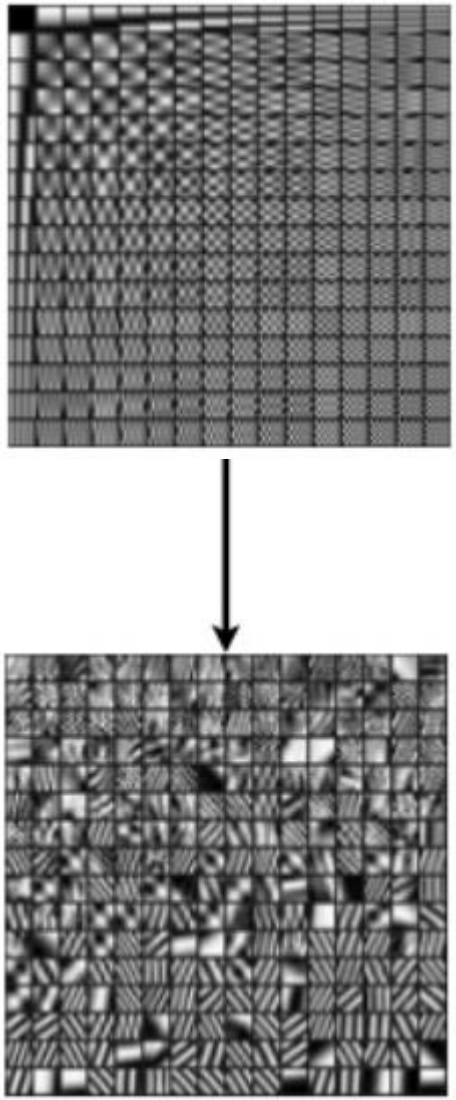
Step 1: Extract patches y_k

Step 2: Dictionary learning

$$\min_{D,(x_k)_k} \sum_k \frac{1}{2}\|y_k - Dx_k\|^2 + \lambda\|x_k\|_1$$

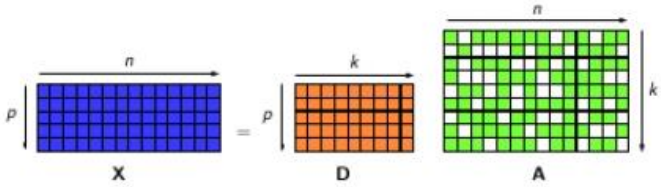
Step 3: Patch averaging $\widetilde{y_k} = Dx_k$

[Aharon & Elad 2006]

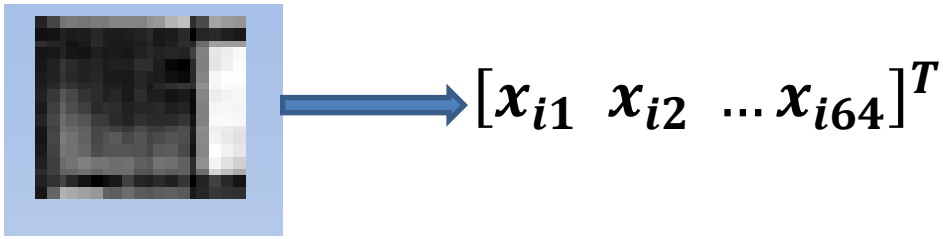


Application of Dictionary Learning

Filling in missing pixels



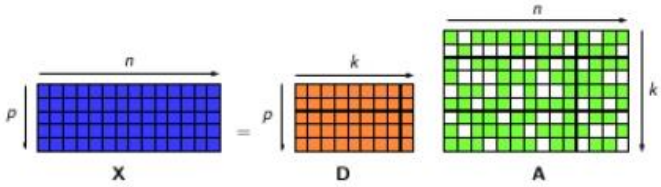
- Training Data:
  - 11,000 examples of 8×8 block patches taken from a data base of face images



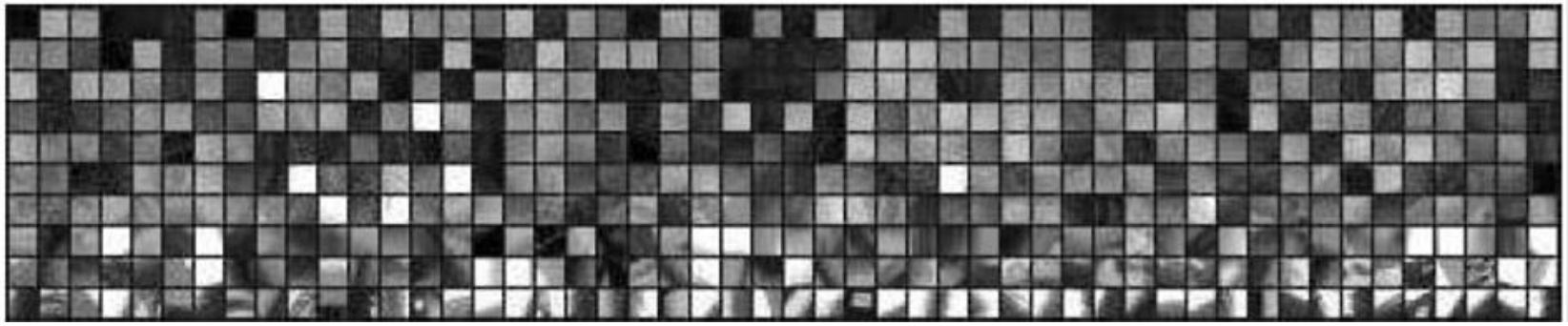
- Every 8×8 block represents a column of $64 \times 11{,}000$ matrix X.

Application of Dictionary Learning

Filling in missing pixels



A random collection of 500 blocks

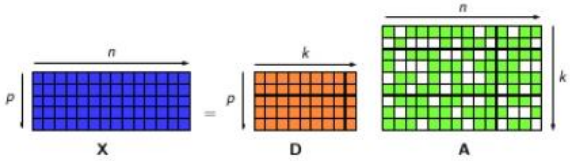


- **Running the K-SVD:**

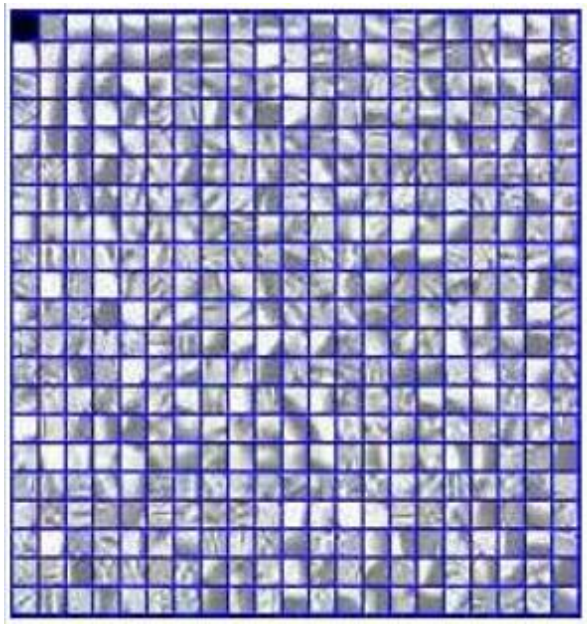
  Size of *D* : 64×441, sparsity of *A* : 10

Application of Dictionary Learning

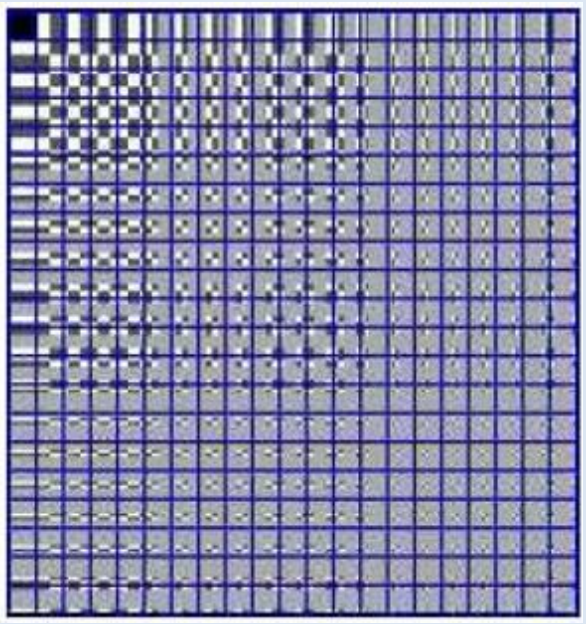
Filling in missing pixels



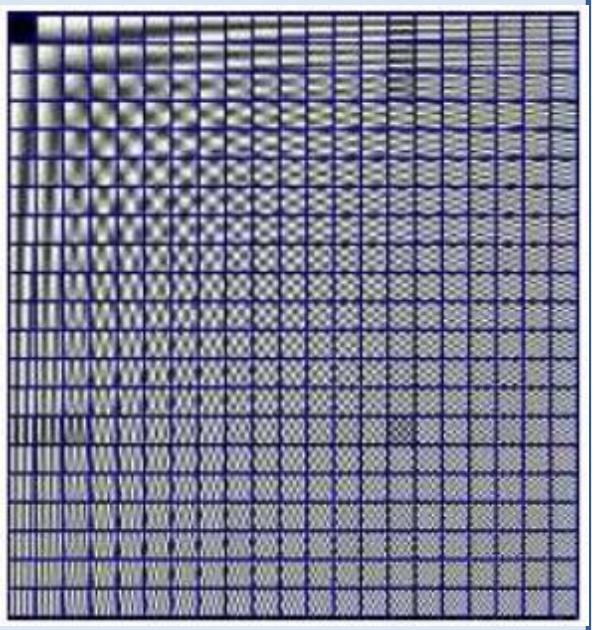
Comparison Dictionaries



Learned dictionary
K-SVD



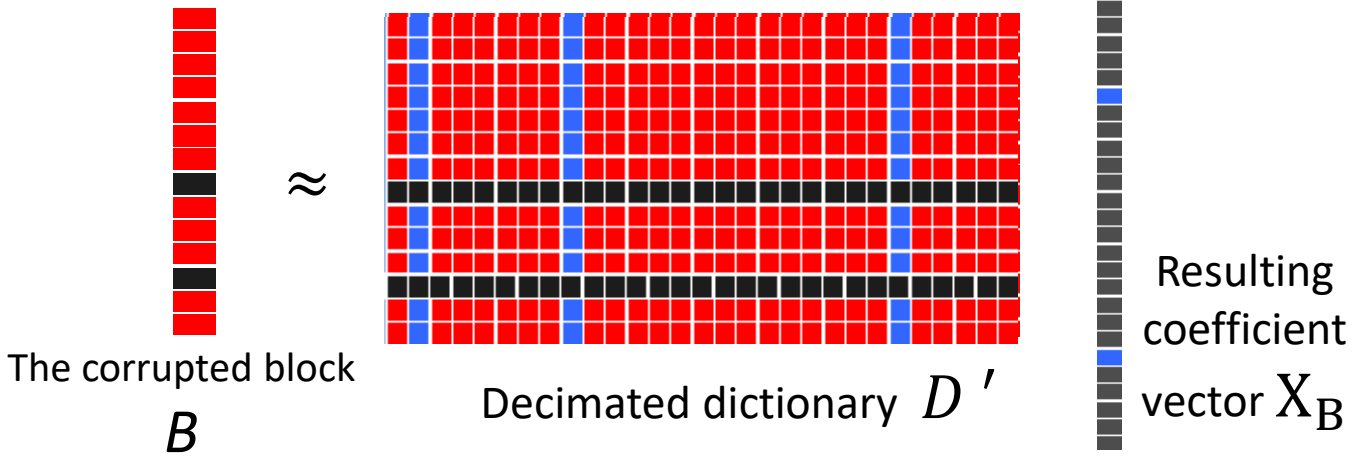
overcomplete Haar dictionary

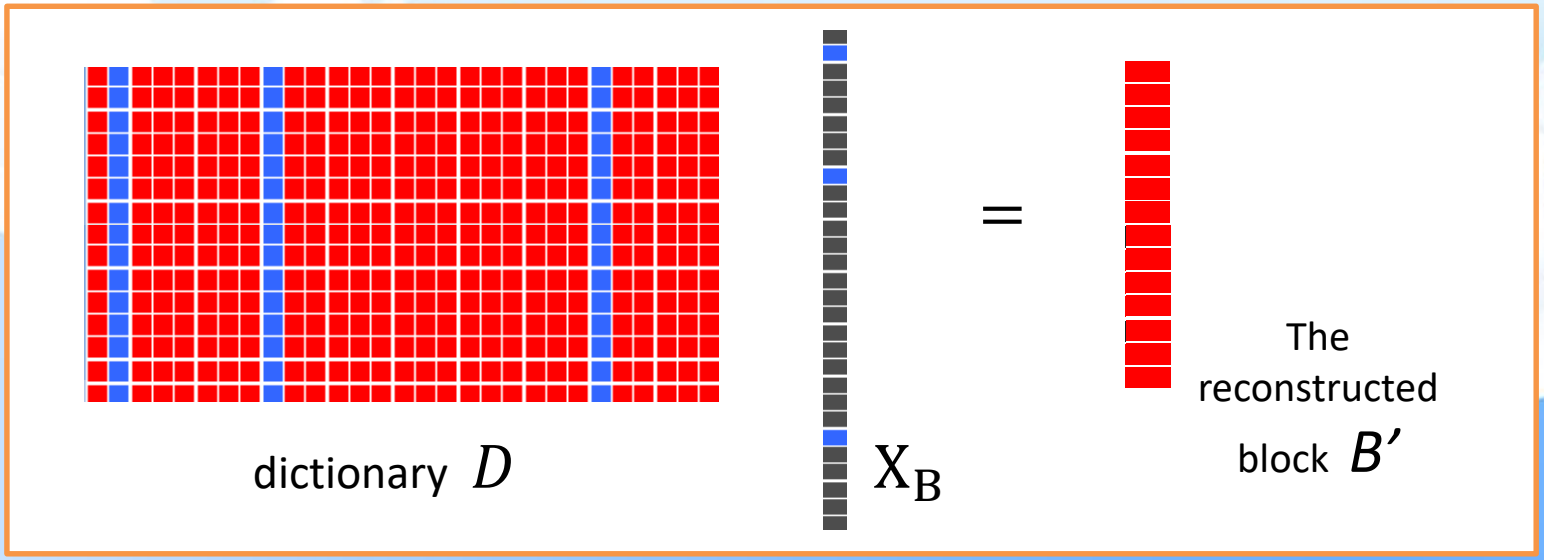


overcomplete DCT dictionary

Application of Dictionary Learning

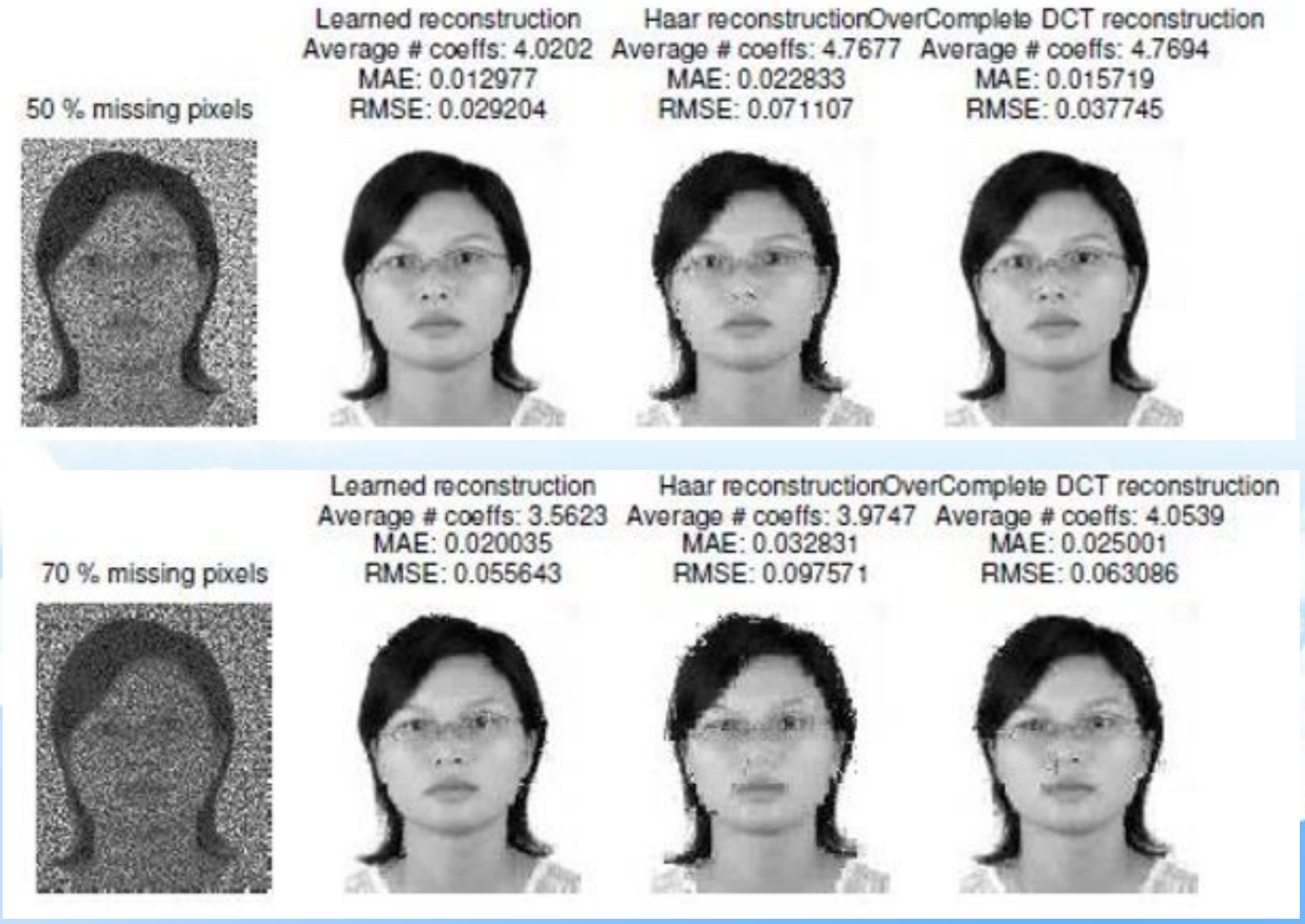
Filling in missing pixels





Application of Dictionary Learning

Filling in missing pixels



Application of Dictionary Learning

Domain Specific Compression

[Elad et al. 2009]

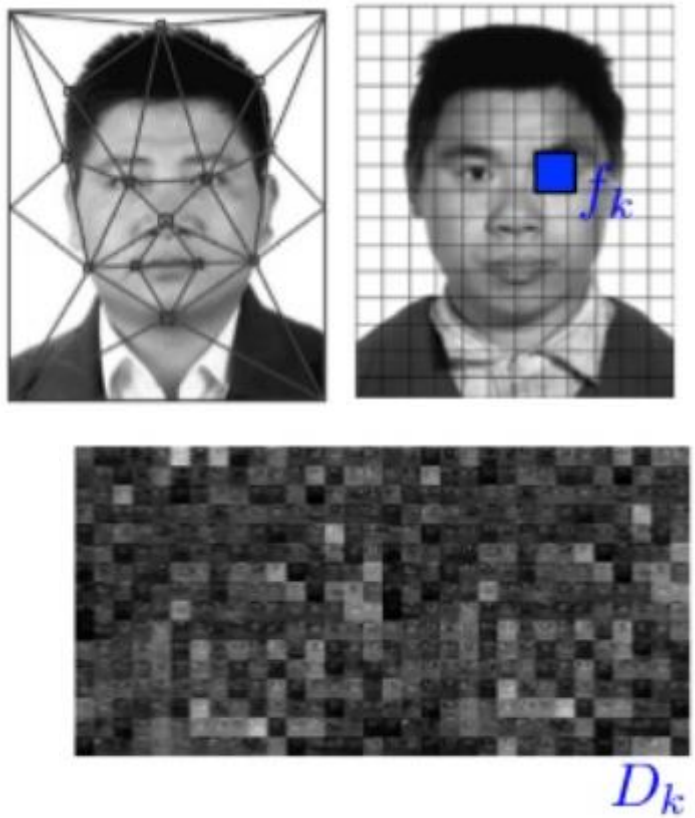
Image registration

Non-overlapping patches $(f_k)_k$

Dictionary learning $(D_k)_k$

Sparse approximation $f_k \approx D_k x_k$

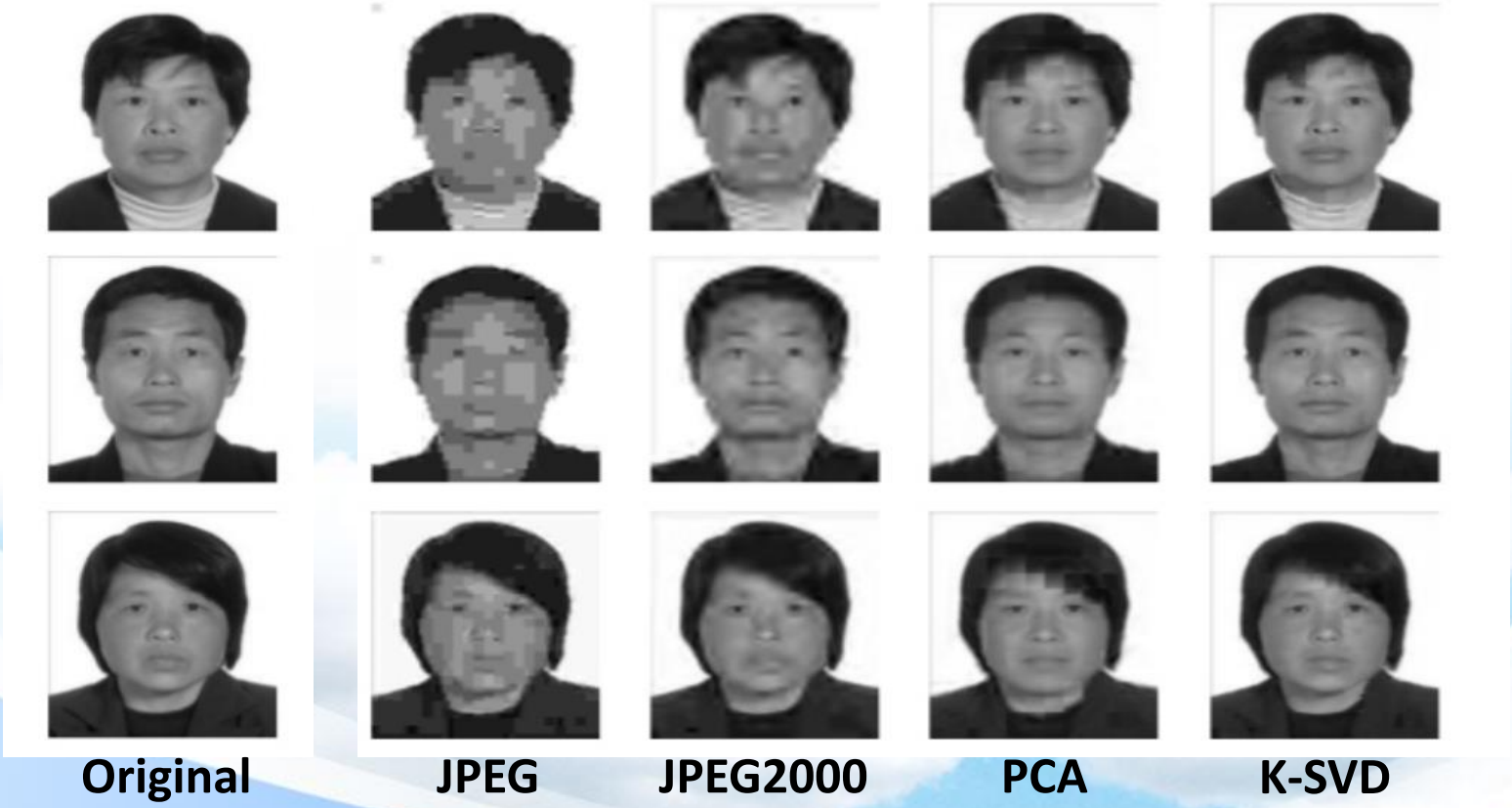
Entopic coding: x_k →file



Application of Dictionary Learning

Domain Specific Compression

Using just 400 bytes per image



Original JPEG JPEG2000 PCA K-SVD

Application of Dictionary Learning

Super Resolution

Application of Dictionary Learning

Super Resolution

The Original

Bicubic Interpolation

SR result

Nonnegative Matrix Factorization (NMF)

Non-Negative Datasets

Some datasets are intrinsically non-negative:

- Counters (e.g., no. occurrences of each word in a text document)
- Quantities (e.g., amount of each ingredient in a chemical experiment)
- Intensities (e.g., intensity of each color in an image)

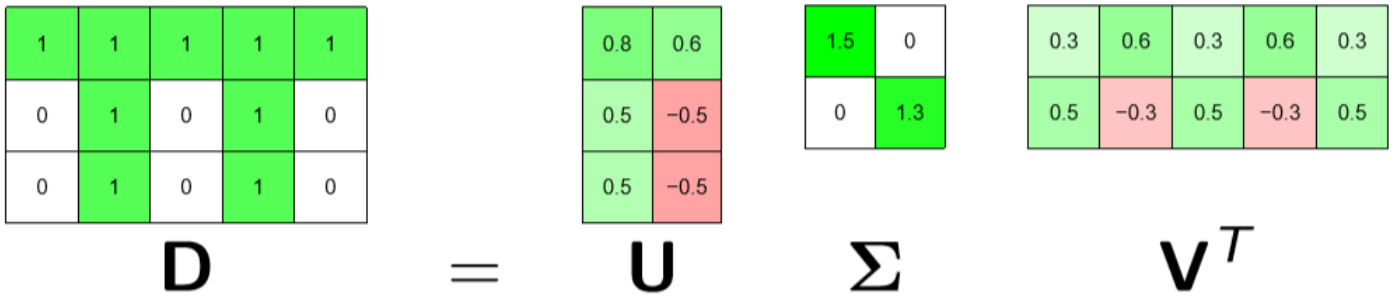
The corresponding data matrix D has only non-negative values.

- Decompositions such as SVD may involve negative values in factors and components
- Negative values describe the absence of something
- Often no natural interpretation

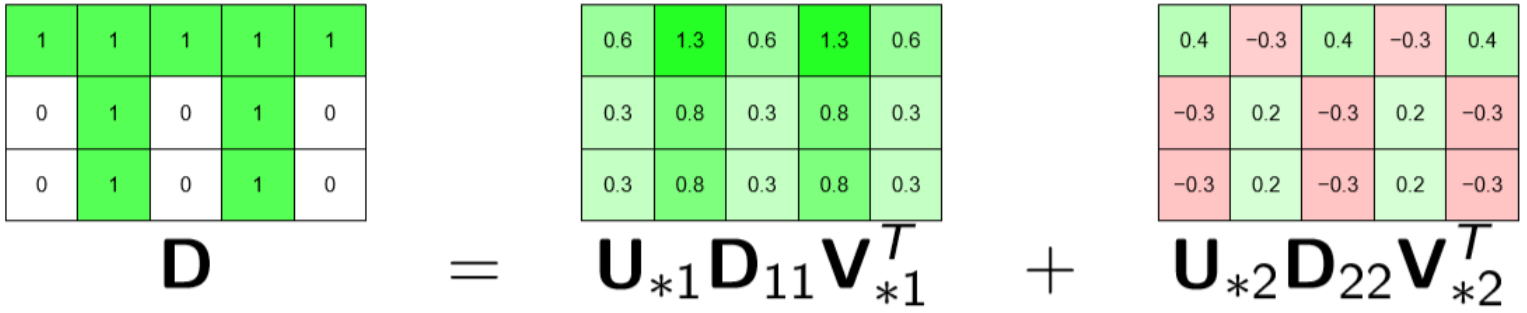
Can we find a decomposition that is more natural to non-negative data?

Example (SVD)

Consider the following “bridge” matrix and its truncated SVD



Here are the corresponding components:



Negative values make interpretation unnatural or difficult.

Non-Negative Matrix Factorization (NMF)

(Basic form of NMF)

Given a non-negative matrix $D \in \mathbf{R}_+^{m \times n}$,

a non-negative matrix factorization of rank k is

$$D \approx LR$$

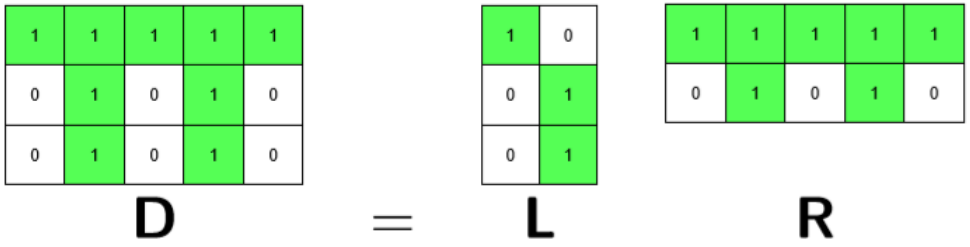
where $L \in \mathbf{R}_+^{m \times r}$ and $R \in \mathbf{R}_+^{r \times n}$ are both non-negative.

- Additive decomposition: factors and components non-negative
  → No cancellation effects
- Smallest r such that $D = LR$ exists is called *non-negative rank of* D

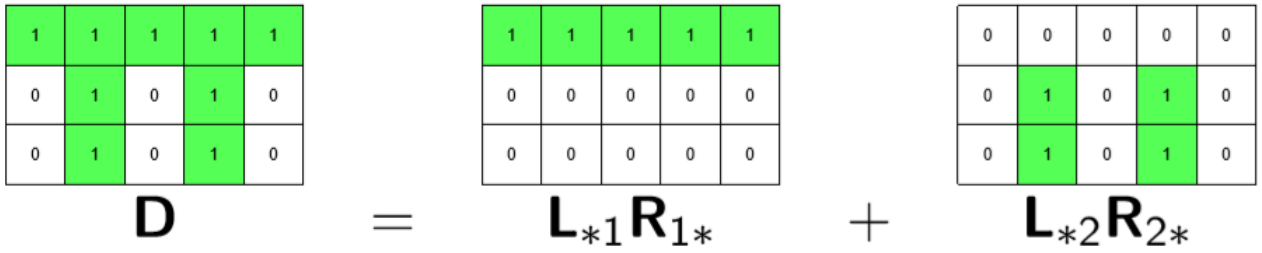
$$\mathrm{rank}(D) \leq \mathrm{rank}_+(D) \leq min\{m, n\}$$

Example (NMF)

Consider the following “bridge” matrix and its rank-2 NMF:

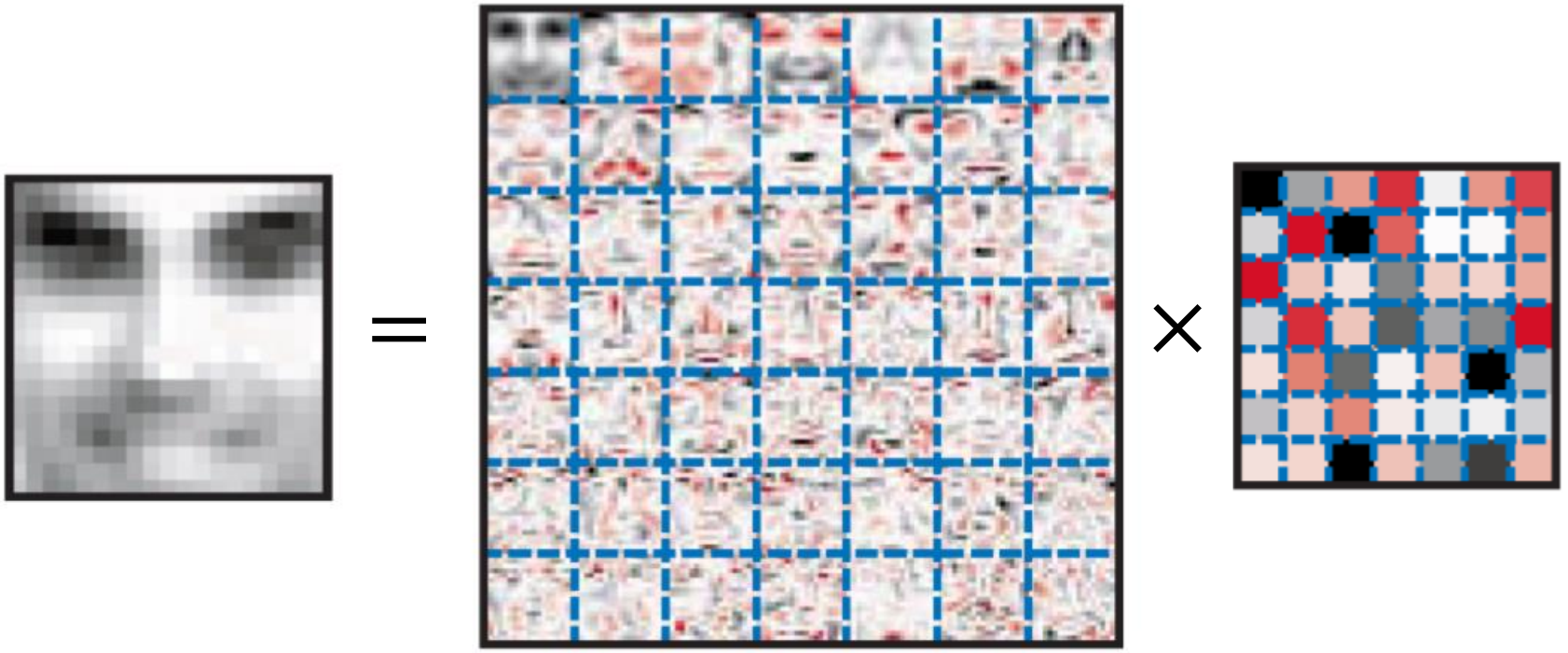


Here are the corresponding components:



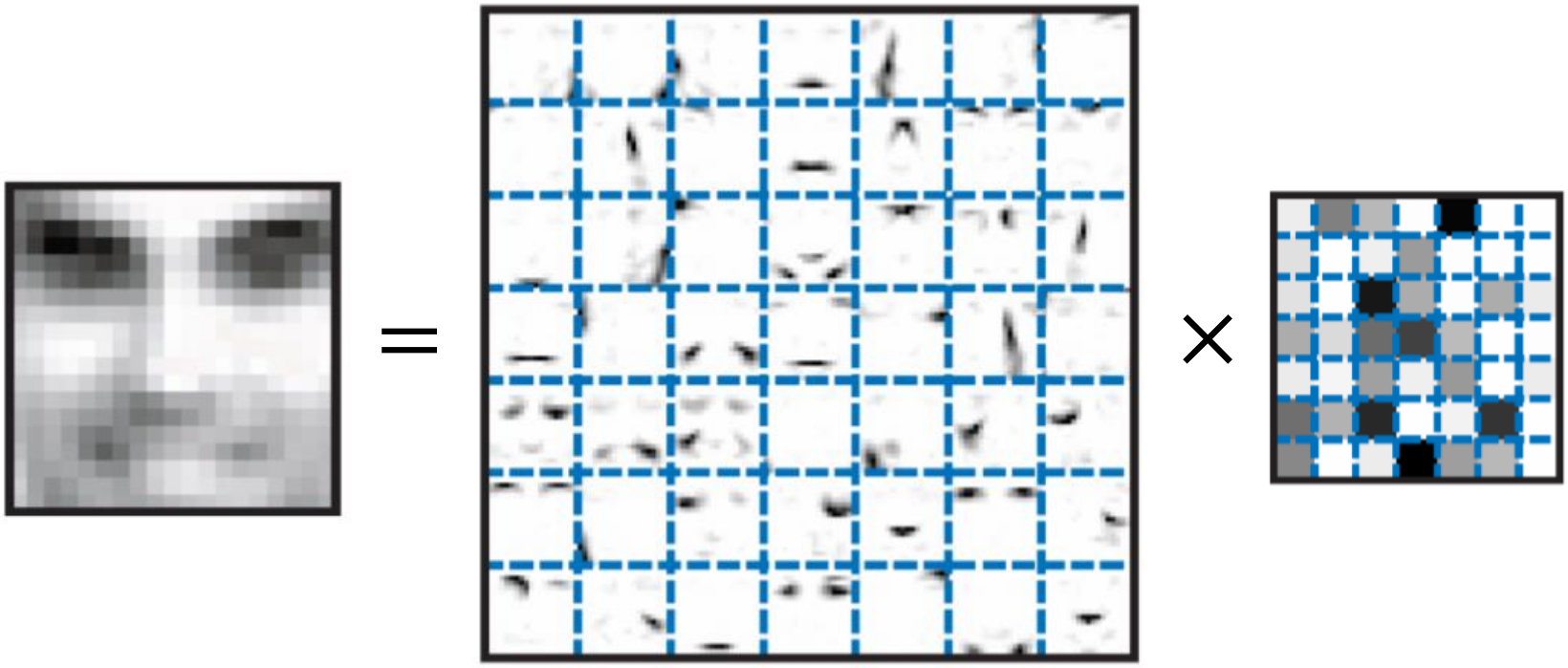
Non-negative matrix decomposition encourage a more natural, part-based representation and (sometimes) sparsity.

Decomposing faces (PCA)



PCA factors are hard to interpret.

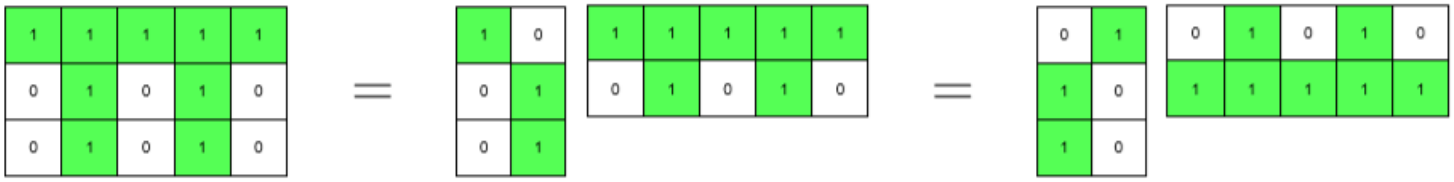
Decomposing faces (NMF)



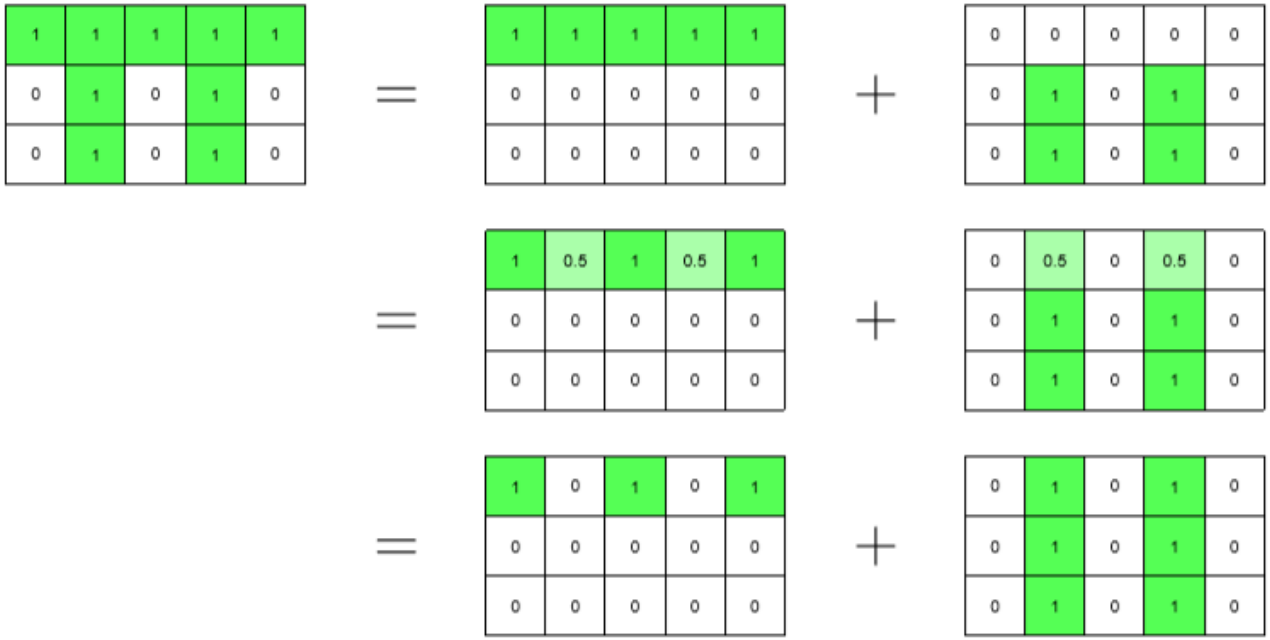
NMF factors correspond to parts of faces.

NMF is not unique

- Factors are not “ordered”



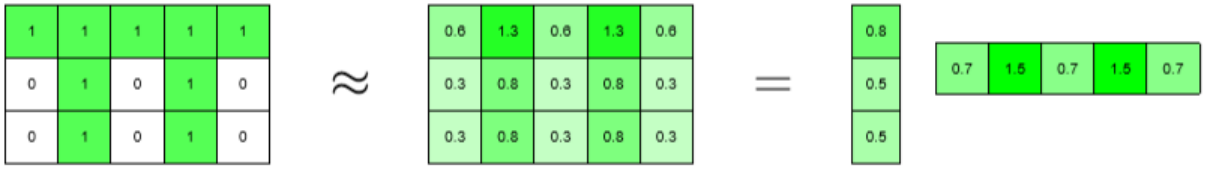
- Factors/components are not unique



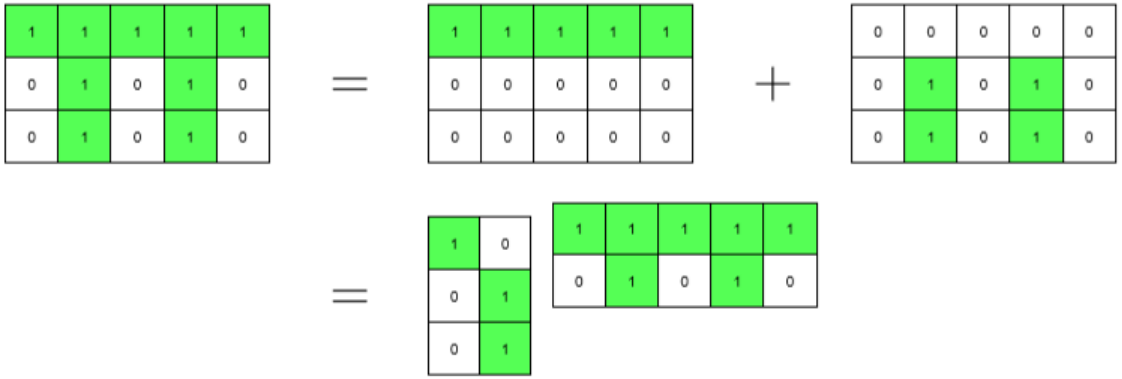
Additional constraints or regularization can encourage uniqueness.

NMF is not hierarchical

- Rank-1 NMF



- Rank-2 NMF



- Best rank-k approximation may differ significantly from best rank- $(k-1)$ approximation
- Rank influences sparsity, interpretability, and statistical fidelity
- Optimum choice of rank is not well-studied

NMF is difficult

We focus on minimizing $L(L, R) = ||D - LR||_F^2$.

- For varying m, n and r problem is NP-hard
- When $\text{rank}(D) = 1$ (or $r = 1$), can be solved in polynomial time
  I. Take first non-zero column of D as $L_{m\times 1}$
  II. Determine $R_{1\times n}$ entry by entry (using the fact that $D_{*j} = LR_{1j}L_{m\times 1}$)
- Problem is not convex
  - Local optimum may not correspond to global optimum
  - Generally little hope to find global optimum
- But: Problem is biconvex
  - For fixed R, $f(L) = ||D - LR||_F^2$ is convex
  - For fixed L, $f(R) = ||D - LR||_F^2$ is convex
  - Allows for efficient algorithms

General framework

Key approach: alternating minimization

1. Pick starting point L_0 and R_0
2. **while** not converged **do**
3. Keep R fixed, optimize L
4. Keep L fixed, optimize R
5. **end while**

- Update steps 3 and 4 easier than full problem
- Also called *alternating projections* or *(block) coordinate descent*

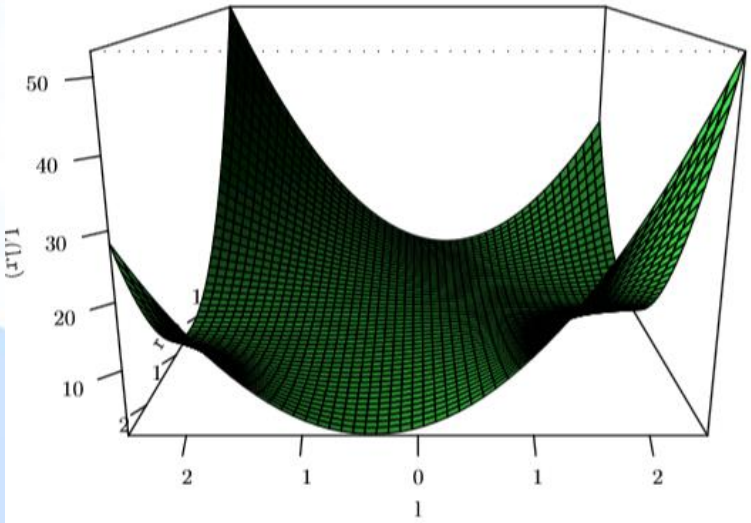
Example

Ignore non-negativity for now. Consider the regularized least-square error:

$$L(L,R) = ||D - LR||_F^2 + \lambda(||L||_F^2 + ||R||_F^2)$$

By setting $m = n = 1$, $D = (1)$ and $\lambda = 0.05$, we obtain

$$f(l,r) = (1 - lr)^2 + 0.05(l^2 + r^2)$$



$$\nabla_l f(l) = -2r(1 - lr) + 0.1l$$
$$\nabla_r f(r) = -2l(1 - lr) + 0.1r$$

Local optima

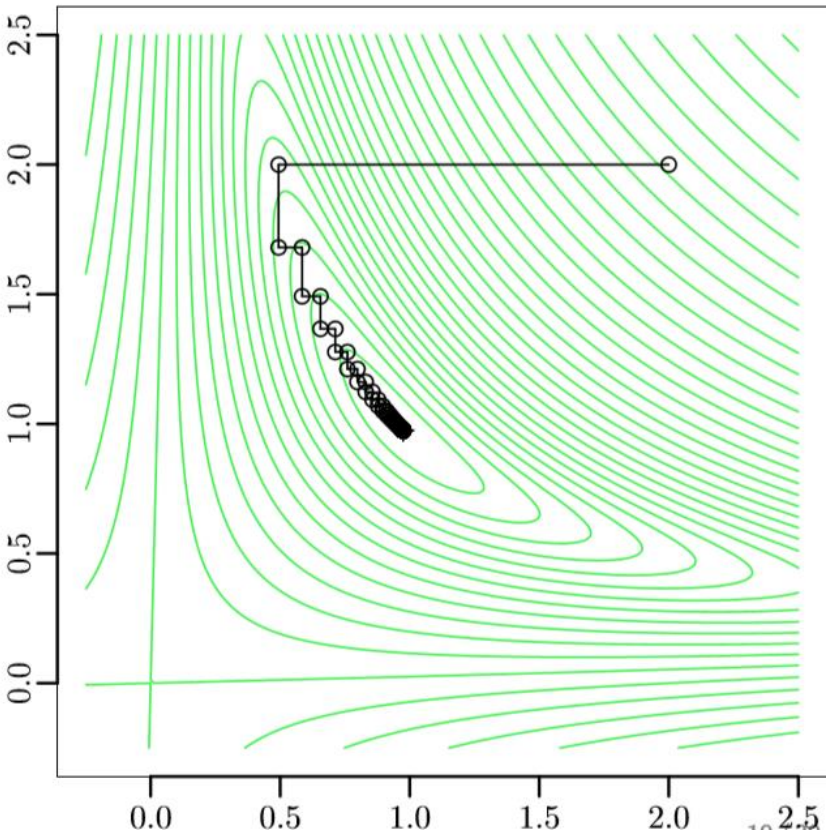
$$\left(\sqrt{\frac{19}{20}}, \sqrt{\frac{19}{20}}\right) \quad \text{and} \quad \left(-\sqrt{\frac{19}{20}}, -\sqrt{\frac{19}{20}}\right)$$

Stationary point (0,0)

Example (ALS, Alternating-Least-Squares)

- $f(l,r) = (1-lr)^2 + 0.05(l^2 + r^2)$
- $l \leftarrow \min_l f(l) = \frac{2r}{2r^2+0.1}$
- $r \leftarrow \min_r f(r) = \frac{2l}{2l^2+0.1}$

Step	l	r
0	2	2
1	0.49	2
2	0.49	1.68
3	0.58	1.68
4	0.58	1.49
⋮	⋮	⋮
100	0.97	0.97

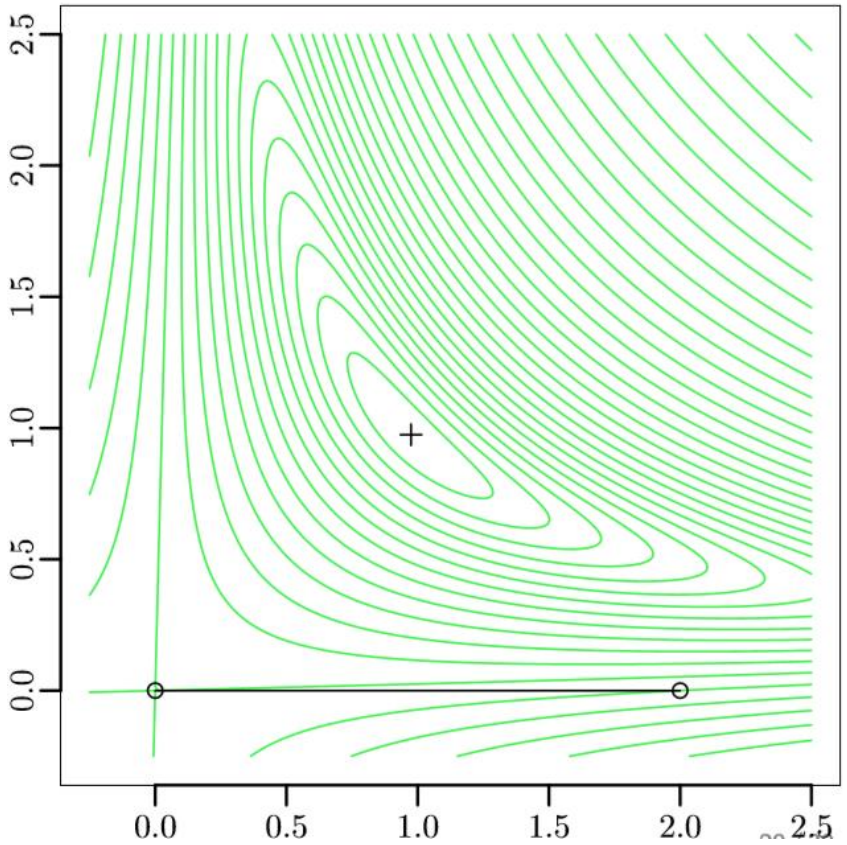


- Converges to local minimum

Example (ALS)

- $f(l,r) = (1 - lr)^2 + 0.05(l^2 + r^2)$
- $l \leftarrow \min_l f(l) = \frac{2r}{2r^2+0.1}$
- $r \leftarrow \min_r f(r) = \frac{2l}{2l^2+0.1}$

Step	l	r
0	2	0
0	0	0
0	0	0
⋮	⋮	⋮



- Converges to stationary point

Alternating non-negative least squares (ANLS)

- Uses non-negative least squares approximation of L and R:

$$\arg\min_{L \in \mathbf{R}_+^{m \times r}} ||D - LR||_F^2 \quad \text{and} \quad \arg\min_{R \in \mathbf{R}_+^{m \times r}} ||D - LR||_F^2$$

- Equivalently: find non-negative least squares solution to $LR = D$
- Common approach: Solve unconstrained least squares problems and "remove" negative values. E.g., when columns (rows) of L (R) are linearly independent, set

$$L = \left[DR^{\dagger}\right]_{\epsilon} \quad \text{and} \quad R = \left[L^{\dagger}D\right]_{\epsilon}$$

where

$R^{\dagger} = R^T(RR^T)^{-1}$ is the right pseudo-inverse of R

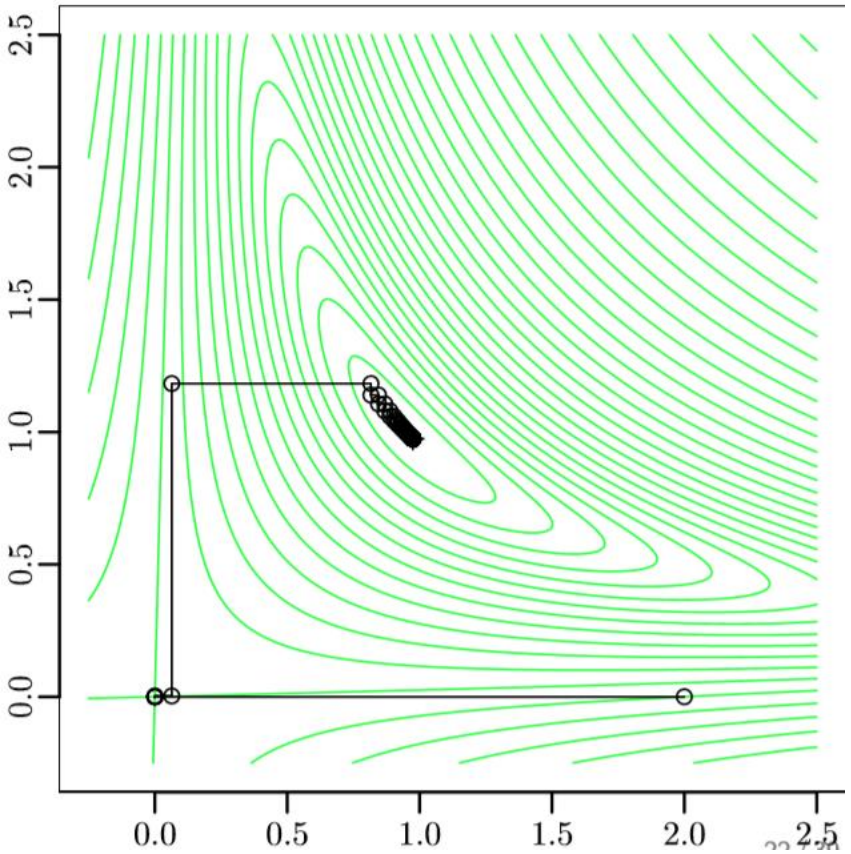
$L^{\dagger} = (L^TL)^{-1}L^T$ is the left pseudo-inverse of L

$[a]_{\epsilon} = \max\{a, \epsilon\}$

Example (ANLS)

- $f(l,r) = (1-lr)^2 + 0.05(l^2+r^2)$ and set $\epsilon = 10^{-9}$
- $l \leftarrow \left[\frac{2r}{2r^2+0.1}\right]_\epsilon$ • $r \leftarrow \left[\frac{2l}{2l^2+0.1}\right]_\epsilon$

Step	l	r
0	2	0
1	$1 \cdot 10^{-9}$	0
2	$1 \cdot 10^{-9}$	$2 \cdot 10^{-8}$
3	$4 \cdot 10^{-7}$	$2 \cdot 10^{-8}$
4	$4 \cdot 10^{-7}$	$8 \cdot 10^{-6}$
⋮	⋮	⋮
100	0.97	0.97



- Converges to local minimum

Application of NMF

Topic modeling

- Consider a document-word matrix constructed from some corpus

$$\tilde{\mathbf{D}} = \begin{array}{c} \\ \text{doc 1} \\ \text{doc 2} \\ \text{doc 3} \\ \text{doc 4} \\ \text{doc 5} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{air} & \text{water} & \text{pollution} & \text{democrat} & \text{republican} \end{array} \\ \begin{pmatrix} 3 & 2 & 8 & 0 & 0 \\ 1 & 4 & 12 & 0 & 0 \\ 0 & 0 & 0 & 10 & 11 \\ 0 & 0 & 0 & 8 & 5 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

- Documents seem to talk about two "topics"
  - Environment (with words air, water, and pollution)
  - Congress (with words democrat and republican)

Can we automatically detect topics in documents?

A probabilistic view point

- Let's normalize such that the entries sum to unity

$$\mathbf{D} = \begin{array}{c} \\ \text{doc 1} \\ \text{doc 2} \\ \text{doc 3} \\ \text{doc 4} \\ \text{doc 5} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{air} & \text{water} & \text{pollution} & \text{democrat} & \text{republican} \end{array} \\ \begin{pmatrix} 0.04 & 0.03 & 0.12 & 0.00 & 0.00 \\ 0.01 & 0.06 & 0.17 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.14 & 0.16 \\ 0.00 & 0.00 & 0.00 & 0.12 & 0.07 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.015 \end{pmatrix} \end{array}$$

- The probability to draw word w from document d is given by

$$P(d, w) = \mathbf{D}_{\mathrm{dw}}$$

- Matrix $\mathbf{D}$ can represent any probability distribution

Probabilistic latent semantic analysis (pLSA)

(pLSA, NMF formulation)

Given a rank r, find matrices L, Σ and R such that

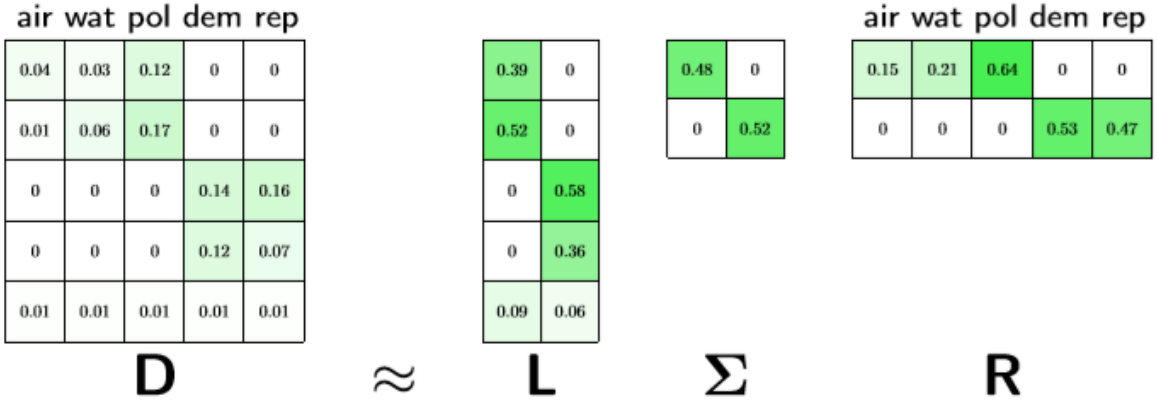
$$D \approx L \, \Sigma \, R$$

where

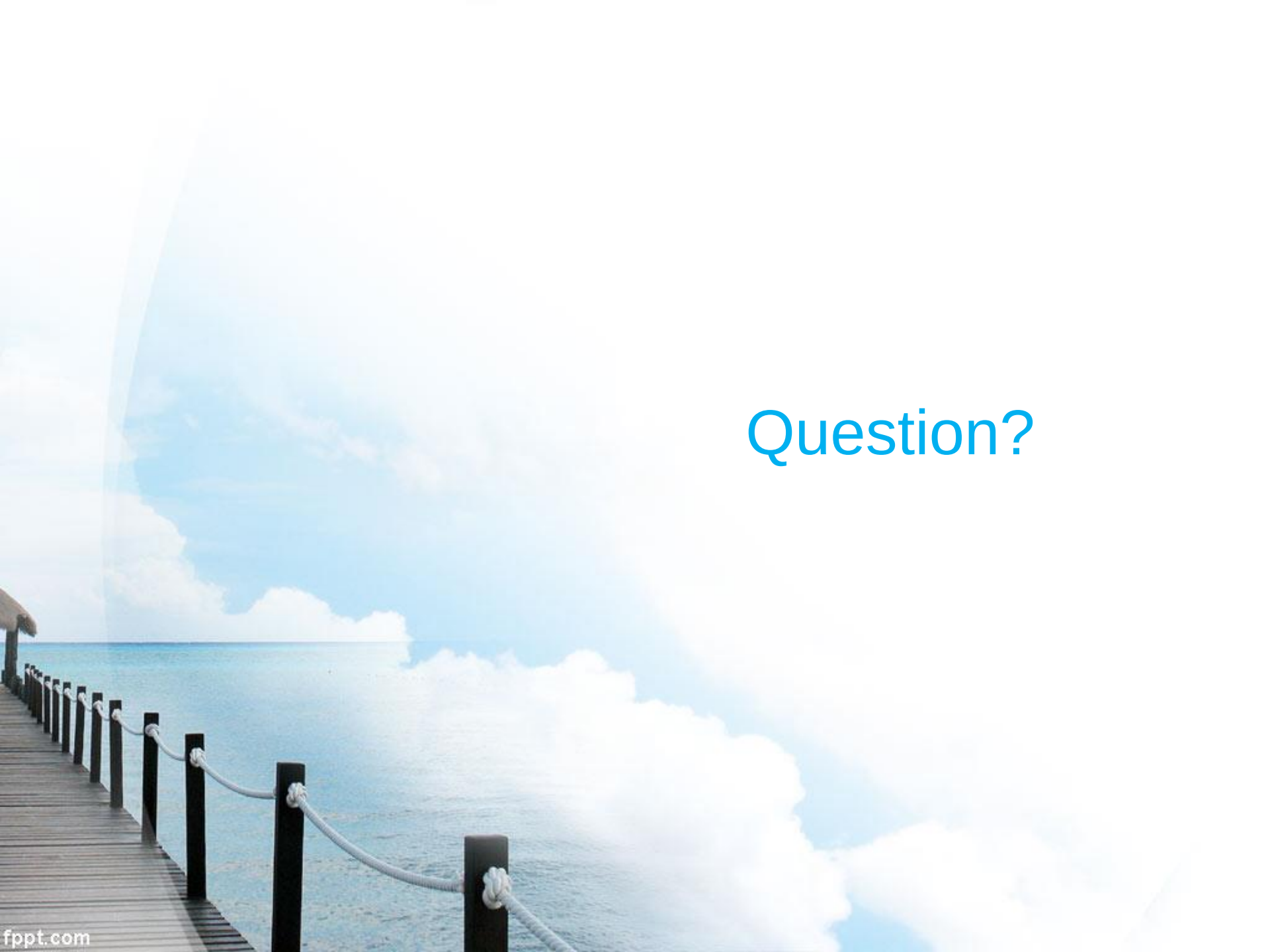
- $L_{m \times r}$ is a non-negative, column-stochastic matrix,
- $\Sigma_{r \times r}$ is a non-negative, diagonal matrix that sums to unity,
- $R_{r \times n}$ is a non-negative, row-stochastic matrix.

Example

- pLSA factorization of example matrix



- Rank r corresponds to number of topics
- Σ_{kk} corresponds to overall frequency of topic k
- L_{dk} corresponds to contribution of document d to topic k
- R_{kw} corresponds to frequency of word w in topic k
- pLSA constraints allow for probabilistic interpretation


Question?
fppt.com