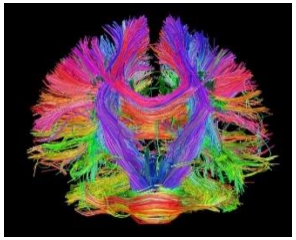
    

CS-570
Statistical Signal Processing

Lecture 7: Dictionary Learning

Spring Semester 2019

Grigorios Tsagkatakis

Today’s Objectives

NO CLASS ON WEDNESDAY

Topics:

- Dictionary Learning

Disclaimer: Material used:

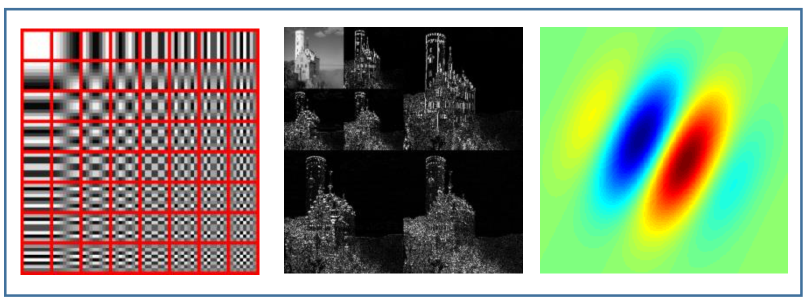
Zhang, Zheng, et al. "A survey of sparse representation: algorithms and applications." *IEEE access* 3 (2015): 490-530.

Sparse Signal Modeling

Key idea $\min \|\mathbf{y} - \mathbf{D}\mathbf{s}\|_2 \quad \text{s.t.} \quad \|\mathbf{s}\|_0 \leq K$

$\text{Greedy} \qquad \|\mathbf{s}\|_1$

Dictionary learning



$$\min \|\mathbf{Y} - \mathbf{D}\mathbf{S}\|_F$$
$$\text{s.t.} \quad \|\mathbf{S}_i\|_1 \leq K, \|\mathbf{D}_i\|_2 \leq 1$$

Transform

- DFT
- DCT
- Wavelets

Learned from examples

- KSVD

A special type of dictionary: two-ortho case

- Motivation for over-complete dictionary: many signals are mixtures of diverse phenomena; no single basis can describe them well

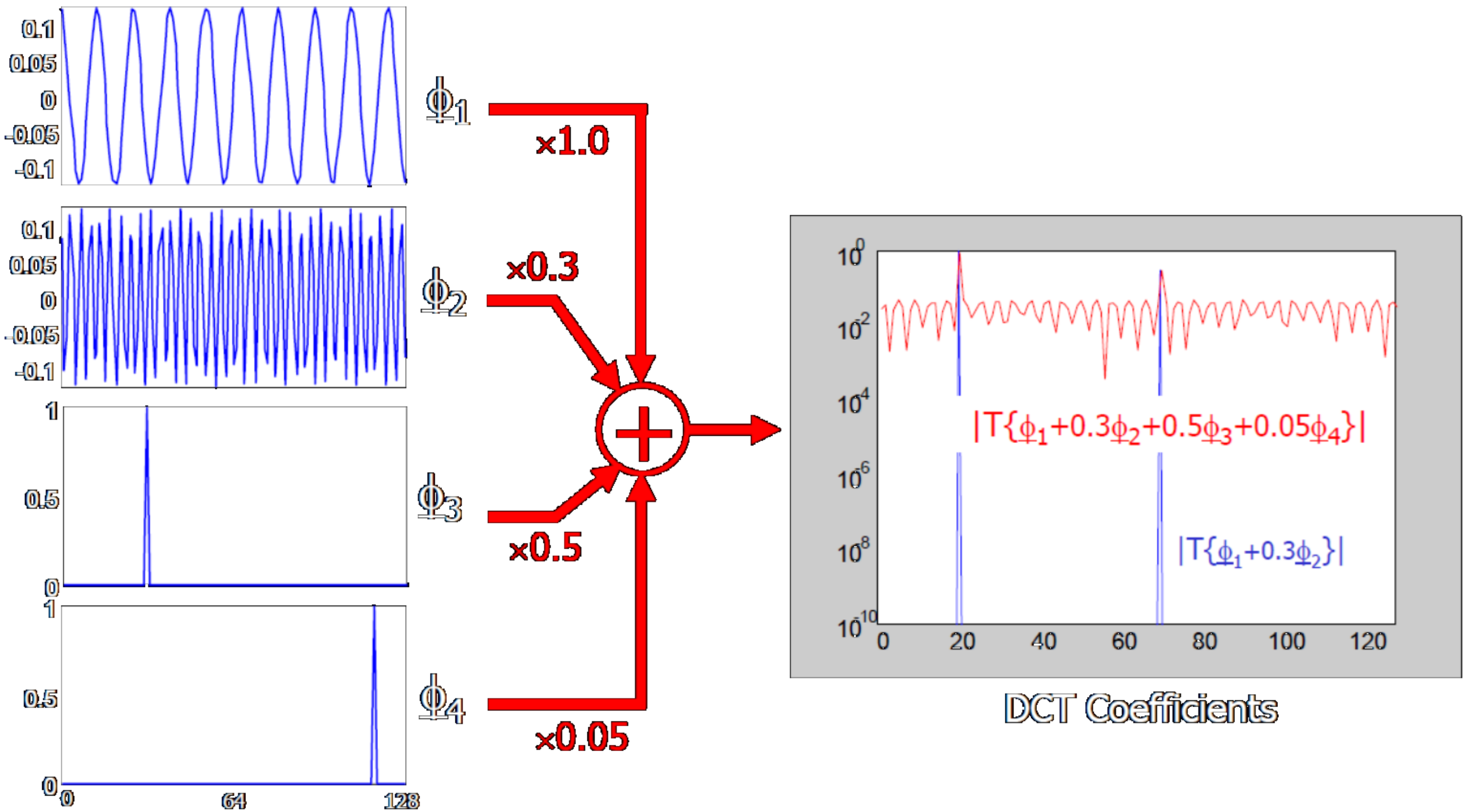
Two-ortho case: A is a concatenation of 2 orthonormal matrices

$$\boldsymbol{A} = [\boldsymbol{\Psi}, \boldsymbol{\Phi}] \qquad \text{where } \boldsymbol{\Psi}\boldsymbol{\Psi}^* = \boldsymbol{\Psi}^*\boldsymbol{\Psi} = \boldsymbol{\Phi}\boldsymbol{\Phi}^* = \boldsymbol{\Phi}^*\boldsymbol{\Phi} = \boldsymbol{I}$$

- A classical example: A = [I,F] (F : Fourier matrix)

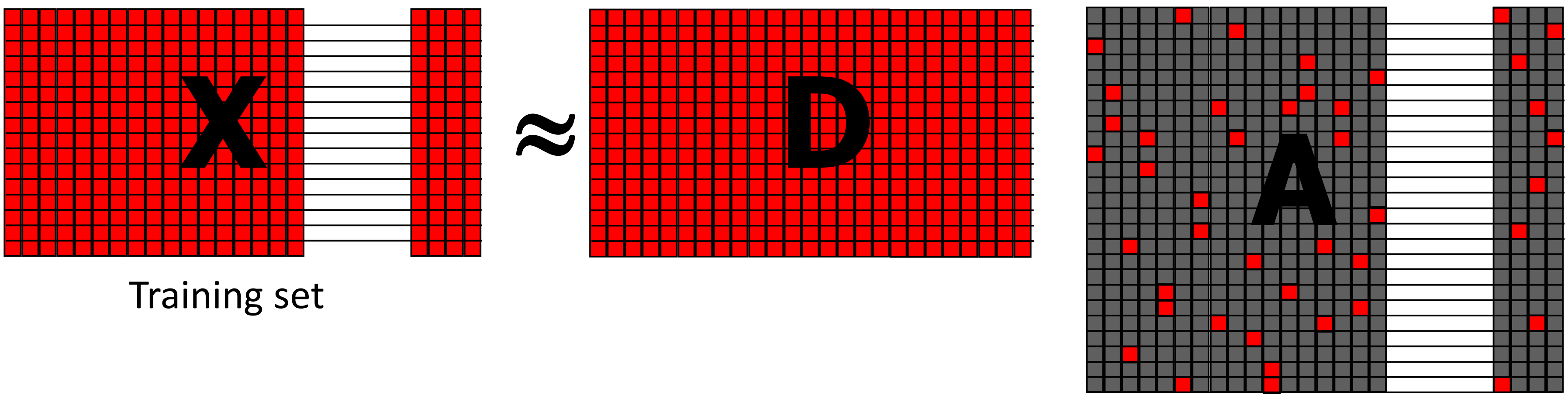
representing a signal y as a superposition of spikes and sinusoids

Example



0.1
0.05
0
-0.05
-0.1
ϕ_1
×1.0
ϕ_2
×0.3
1
0.5
0
ϕ_3
×0.5
ϕ_4
×0.05
0
64
128
10^0
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
0
20
40
60
80
100
120
$|T\{\phi_1+0.3\phi_2+0.5\phi_3+0.05\phi_4\}|$
$|T\{\phi_1+0.3\phi_2\}|$
DCT Coefficients

Dictionary learning



$$\underset{\mathbf{D},\mathbf{A}}{\text{Min}} \sum_{j=1}^{P} \left\| \mathbf{D}\underline{\alpha}_j - \underline{x}_j \right\|_2^2 \quad \text{s.t. } \forall j, \left\| \underline{\alpha}_j \right\|_0^0 \le L$$

Each example is a linear combination of atoms from **D**

Each example has a sparse representation with no more than L atoms

Divergence – Matrix Rank

The **rank** of a matrix *M* is the size of the largest collection of linearly independent *columns of M* (the **column rank**) or the size of the largest collection of linearly independent *rows of M* (the **row rank**)

- Row Echelon Form

$$\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} R_2 \to 2r_1 + r_2 \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 3 \\ 3 & 5 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 3 \\ 3 & 5 & 0 \end{bmatrix} R_3 \to -3r_1 + r_3 \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 3 \\ 0 & -1 & -3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 3 \\ 0 & -1 & -3 \end{bmatrix} R_3 \to r_2 + r_3 \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 0 \end{bmatrix} R_1 \to -2r_2 + r_1 \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

Rank=2

A matrix is in **row echelon form** if

(i) all nonzero rows are above any rows of all zeroes

(ii) The leading coefficient of a nonzero row is always strictly to the right of the leading coefficient of the row above it

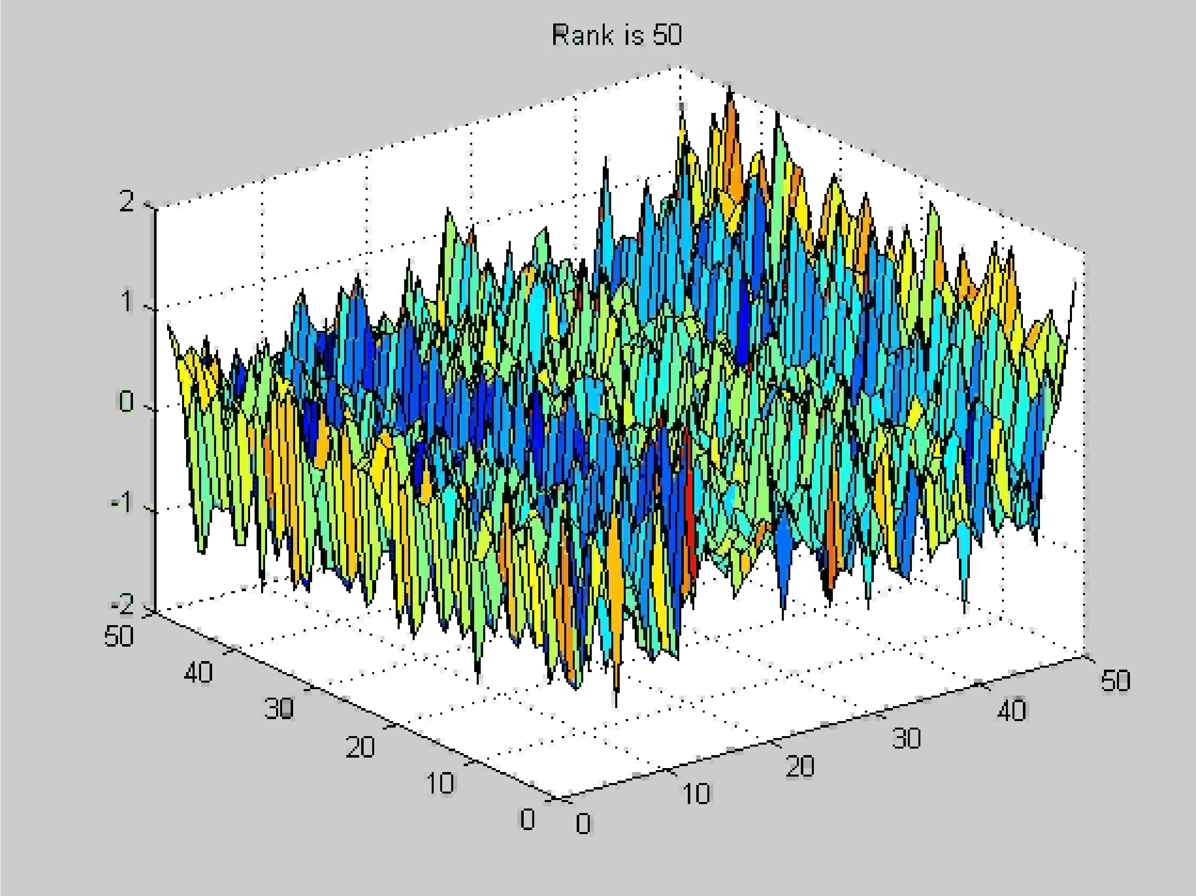
Matrix Rank

- The rank of an $m \times n$ matrix is a nonnegative integer and cannot be greater than either m or n. That is, $\mathrm{rank}(M) \le \min(m, n)$.
- A matrix that has a rank as large as possible is said to have **full rank**; otherwise, the matrix is **rank deficient**.

$$\mathrm{rank}(AB) \le \min(\mathrm{rank}\ A, \mathrm{rank}\ B).$$

$$\mathrm{rank}(A^T A) = \mathrm{rank}(AA^T) = \mathrm{rank}(A) = \mathrm{rank}(A^T)$$

Matrix Rank



Singular Value Decomposition (SVD)

Given any $m \times n$ matrix **M**, algorithm to find matrices **U**, **Σ**, and **V** such that $\mathbf{M} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^{T}$

- **U**: left singular vectors (orthonormal)
- **Σ**: diagonal containing singular values
- **V**: right singular vectors (orthonormal)

$$M = \underbrace{U}_{m \times m}\underbrace{\Sigma}_{m \times n}\underbrace{V^{T}}_{V \text{ is } n \times n}$$

$$\begin{pmatrix} \\ \\ M \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ \mathbf{U} \\ \\ \\ \end{pmatrix} \begin{pmatrix} s_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & s_n \end{pmatrix} \begin{pmatrix} & & \\ & \mathbf{V} & \\ & & \end{pmatrix}^{\mathrm{T}}$$

Singular Value Decomposition (SVD)

<u>Properties</u>

- The s_i are called the singular values of **M**
- If **M** is singular, some of the s_i will be 0
- In general *rank*(**M**) = number of nonzero s_i
- SVD is mostly unique (up to permutation of SV)

M-term approximation

- SVD can be used to compute optimal **low-rank approximations**.
- Approximation problem: Find ***A*$_k$** of rank ***k*** such that

$$A_k = \min_{X:rank(X)=k} \|A - X\|_F$$

Frobenius norm

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

A_k and X are both $m \times n$ matrices.

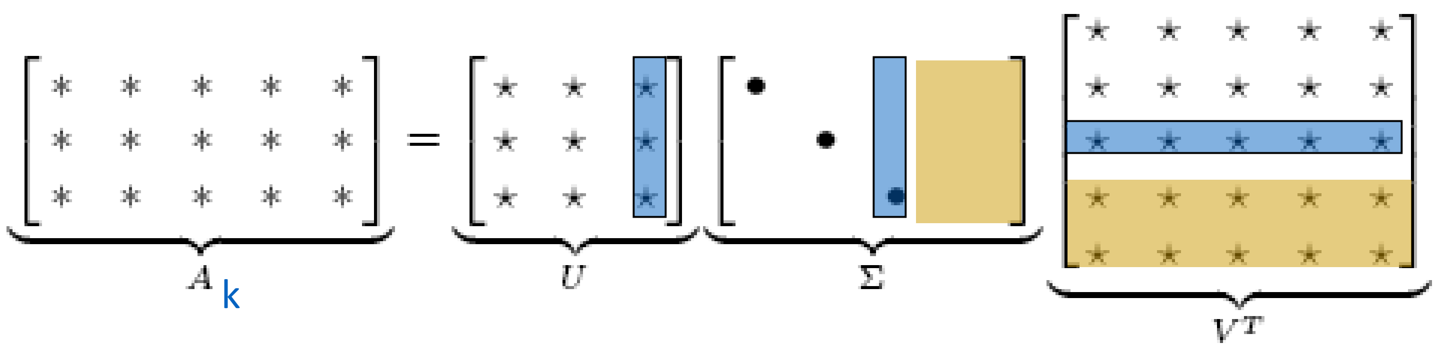
Typically, want $k << r$.

Low-rank Approximation

- Solution via SVD

$$A_k = U \operatorname{diag}(\sigma_1, \ldots, \sigma_k, \underbrace{0, \ldots, 0}) V^T$$

set smallest r-k
singular values to zero



$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

column notation: sum of rank 1 matrices

Method of optimal directions (K. Engan and S. Husoy 1999)

$$\min_{\boldsymbol{D}\in\mathcal{D},\, \boldsymbol{X}\in\mathcal{X}_\Omega} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2 .$$

MOD: least squares

1. Fix $\boldsymbol{D}$, solve for $\boldsymbol{X}$:

$$\min_{\boldsymbol{X}\in\mathcal{X}_\Omega} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2 .$$

2. Fix $\boldsymbol{X}$, solve for $\boldsymbol{D}$:

$$\min_{\boldsymbol{D}} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2 .$$

3. (Optional) Normalization:

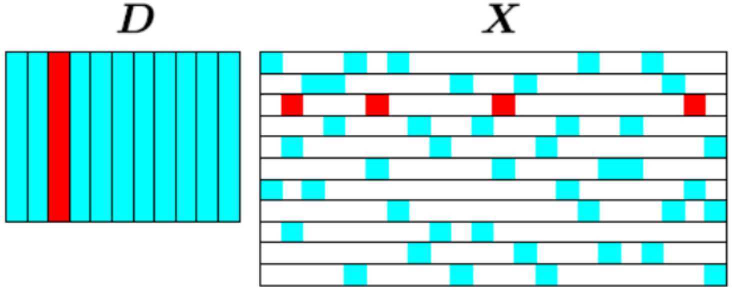
$$\boldsymbol{D}_{:,i} = \boldsymbol{D}_{:,i} / \|\boldsymbol{D}_{:,i}\|_2 .$$

The K-SVD (M. Aharon, et al. 2006)

$$\min_{\boldsymbol{D}\in\mathcal{D},\ \boldsymbol{X}\in\mathcal{X}_\Omega} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 \,.$$

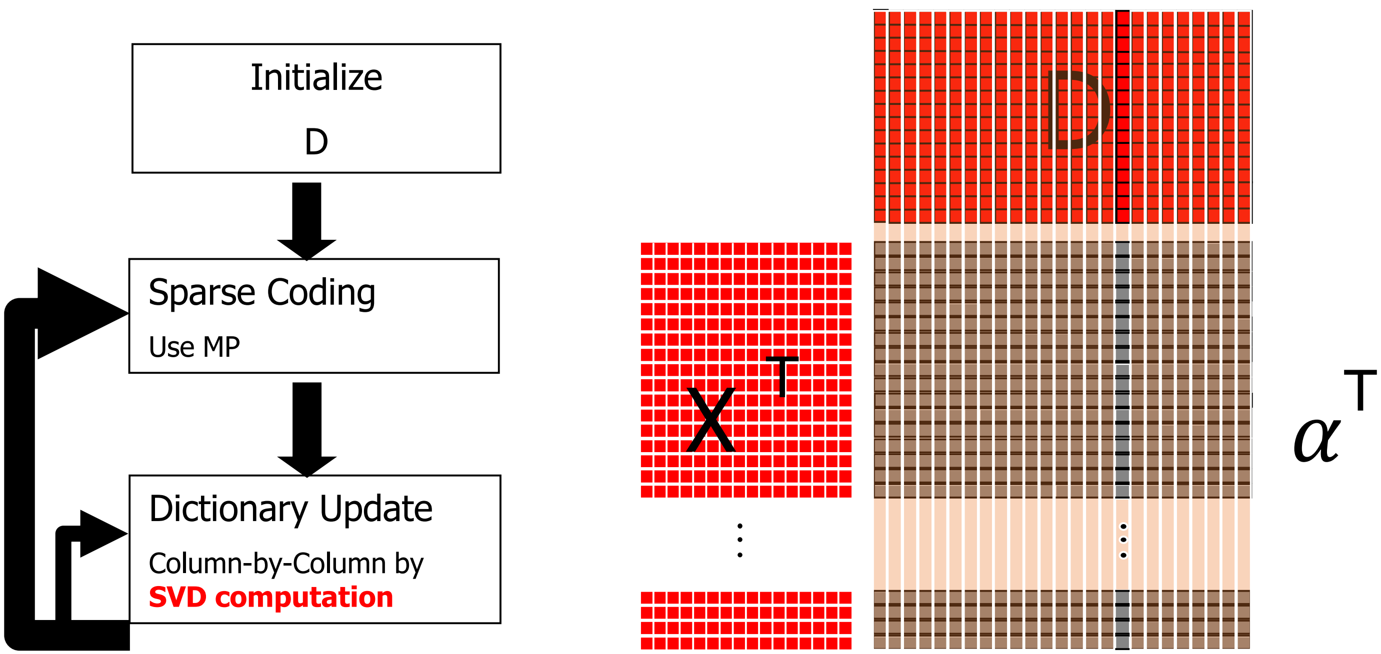
For each column:

Update: this column in $\boldsymbol{D}$ & the corresponding row in $\boldsymbol{X}$.



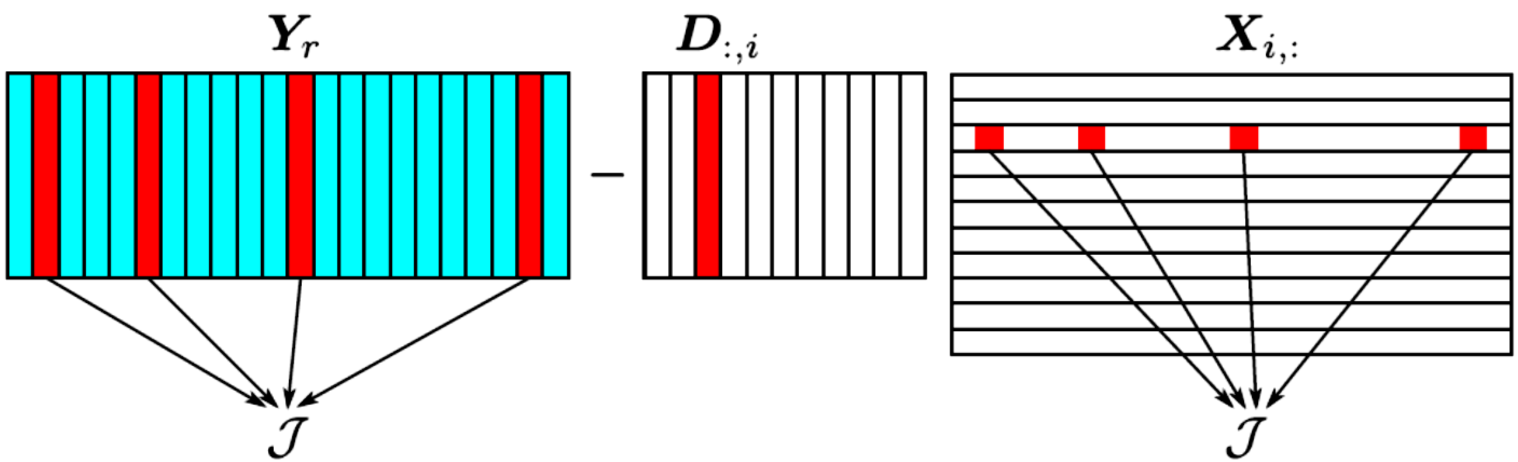
Fix: other columns in $\boldsymbol{D}$ & the corresponding rows in $\boldsymbol{X}$.

K-SVD algorithm



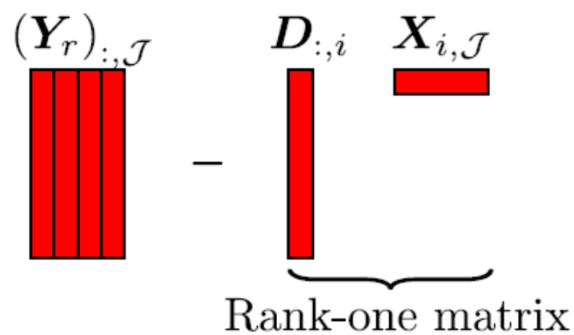
K-SVD details

$$
\begin{aligned}
&\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|^2 \\
&= \|\boldsymbol{Y} - \boldsymbol{D}_{:,j\neq i}\boldsymbol{X}_{j\neq i,:} - \boldsymbol{D}_{:,i}\boldsymbol{X}_{i,:}\|^2 \\
&= \|\boldsymbol{Y}_r - \boldsymbol{D}_{:,i}\boldsymbol{X}_{i,:}\|^2 \\
&= \left\|(\boldsymbol{Y}_r)_{:,\mathcal{J}} - \boldsymbol{D}_{:,i}\boldsymbol{X}_{i,\mathcal{J}}\right\|^2 + c
\end{aligned}
$$



K-SVD details

$$
\begin{aligned}
&\|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|^2 \\
&= \|\boldsymbol{Y}-\boldsymbol{D}_{:,j\neq i}\boldsymbol{X}_{j\neq i,:}-\boldsymbol{D}_{:,i}\boldsymbol{X}_{i,:}\|^2 \\
&= \|\boldsymbol{Y}_r-\boldsymbol{D}_{:,i}\boldsymbol{X}_{i,:}\|^2 \\
&= \left\|(\boldsymbol{Y}_r)_{:,\mathcal{J}}-\boldsymbol{D}_{:,i}\boldsymbol{X}_{i,\mathcal{J}}\right\|^2 + c
\end{aligned}
$$



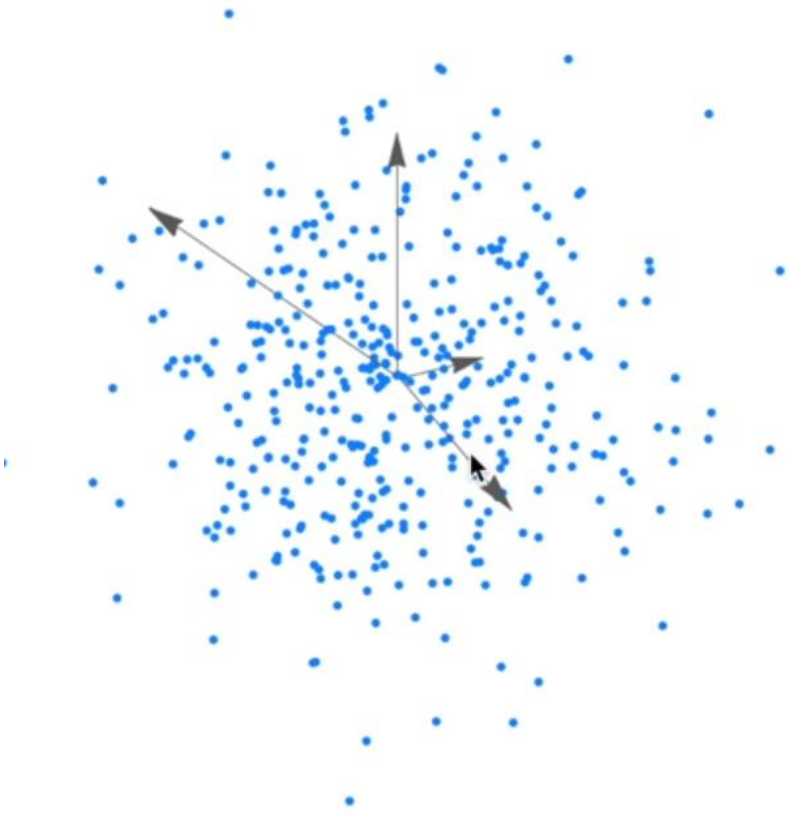
SVD: optimal rank-one matrix approximation.

$$
\begin{aligned}
\boldsymbol{A} &= \sum \lambda_i \boldsymbol{u}_i \boldsymbol{v}_i^T \qquad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \\
&\approx \lambda_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T
\end{aligned}
$$

K-SVD algorithm

Here is three-dimensional data set, spanned by over-complete dictionary of four vectors.

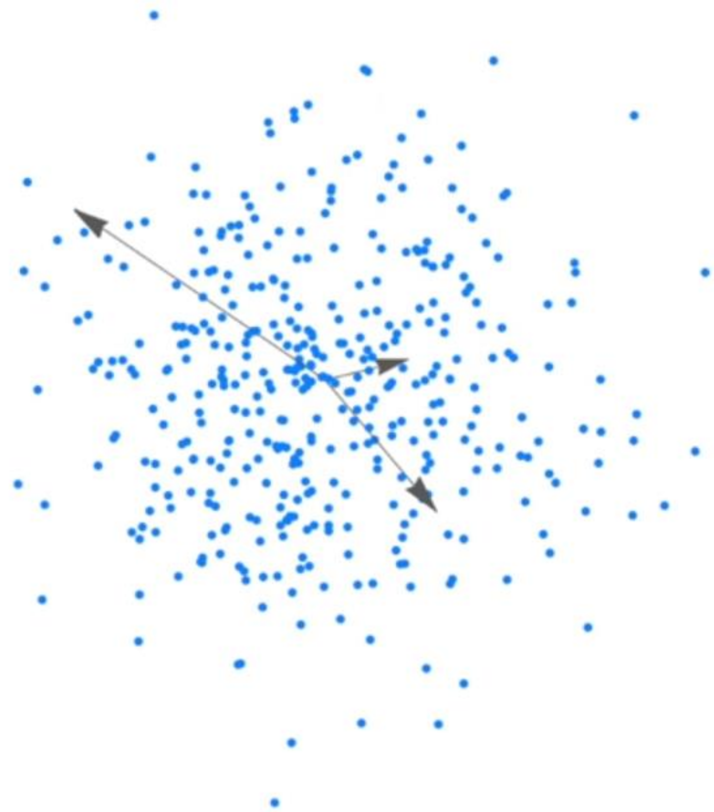
What we want is to update each of these vector to better represent the data.



K-SVD algorithm

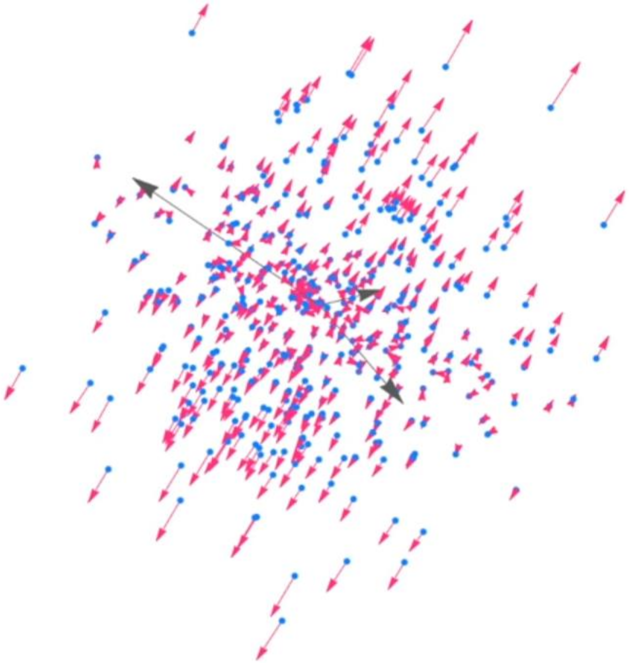
1. Remove one of these vector

   If we do sparse coding using only three vectors, from the dictionary, we cannot perfectly represent the data.



K-SVD algorithm

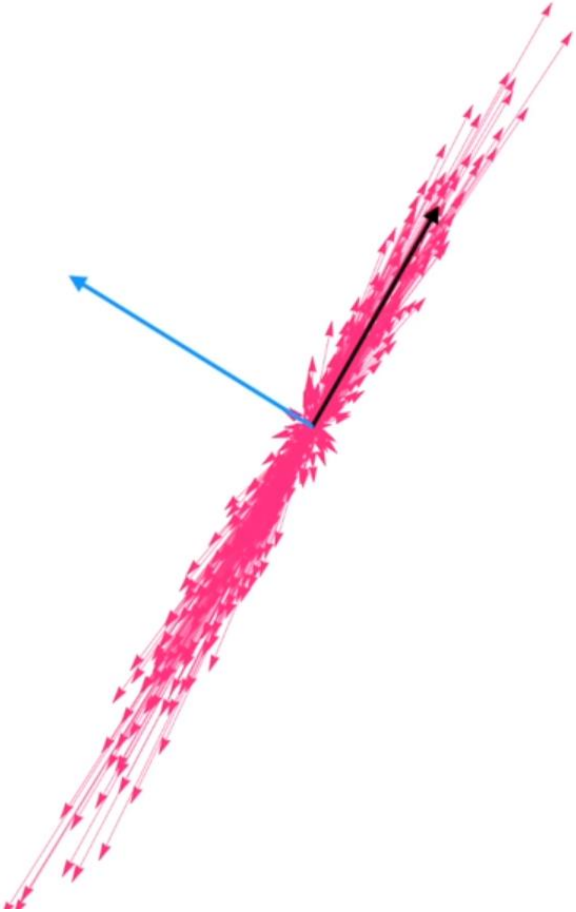
2. Find approximation error on each data point



K-SVD algorithm

2. Find approximation error on each data point

K-SVD algorithm



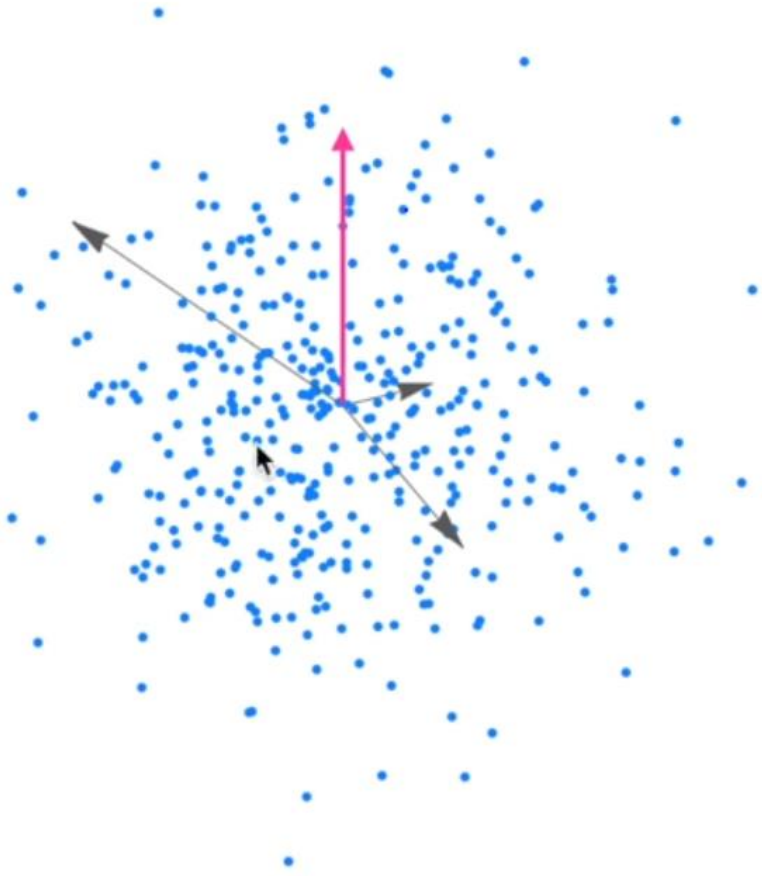
3. Apply SVD on error matrix

The SVD provides us a set of orthogonal basis vector sorted in order of decreasing ability to represent the variance error matrix.

K-SVD algorithm

3. Replace the chosen vector with the first eigenvector of error matrix.

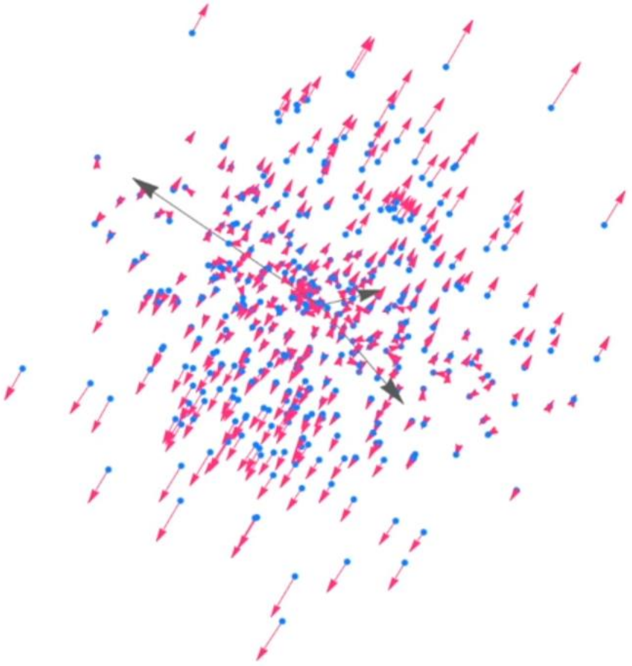
4. Do the same for other vectors.



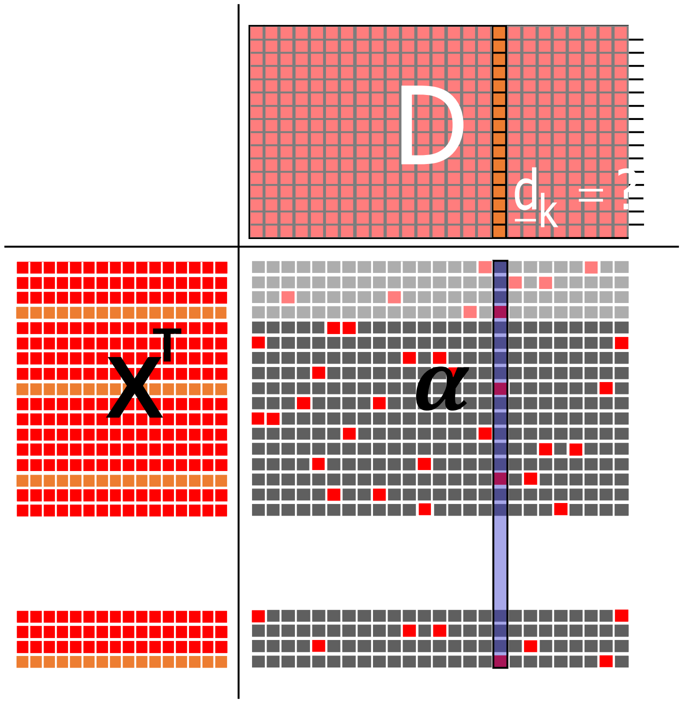
K-SVD algorithm

But, there is not all, but a few data points using the chosen vector.

Then, it is not necessary to calculate error for all data points, but instead a few of them that are using the chosen vector.



K-SVD algorithm



Spring Semester 2019

K-SVD algorithm

1. Initialize the dictionary randomly
2. Using any pursuit algorithm to find a sparse coding α, for the input data X using dictionary D.
3. Update D:
   a. Remove a basis vector d_k
   b. Compute the approximation error E_k on data points that were actually using d_k
   c. Take SVD of E_k
   d. Update d_k.
4. Repeat to step 2 until convergence.

Example

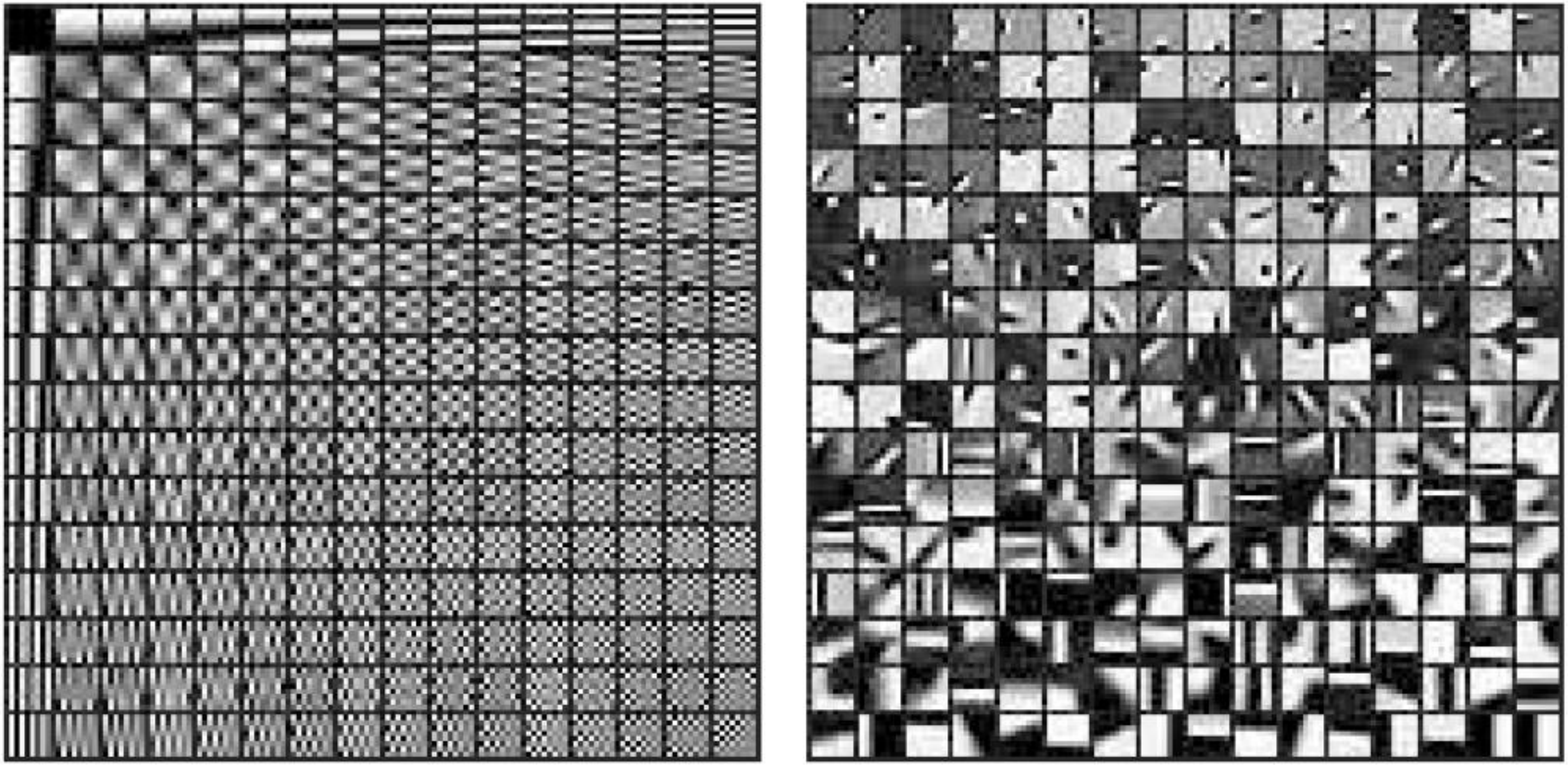
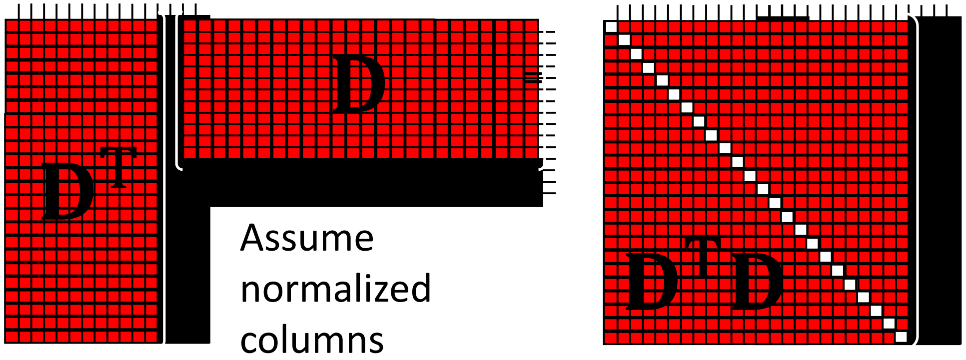


Fig. 2. Left: Overcomplete DCT dictionary. Right: Globally trained dictionary.

The Mutual Coherence

Compute



- The Mutual Coherence $\mu(\mathbf{D})$ is the largest off-diagonal entry in absolute value
- Other ways to characterize the dictionary
  - Restricted Isometry Property - RIP,
  - Exact Recovery Condition - ERC,
  - Spark

Basis pursuit sucess

Theorem: Given a noisy signal $y = \mathbf{D}\alpha + v$ where $\|v\|_2 \leq \varepsilon$ and α is sufficiently sparse,

$$\|\boldsymbol{\alpha}\|_{\mathbf{0}} < \frac{\mathbf{1}}{\mathbf{4}}\left(\mathbf{1} + \frac{\mathbf{1}}{\boldsymbol{\mu}}\right)$$

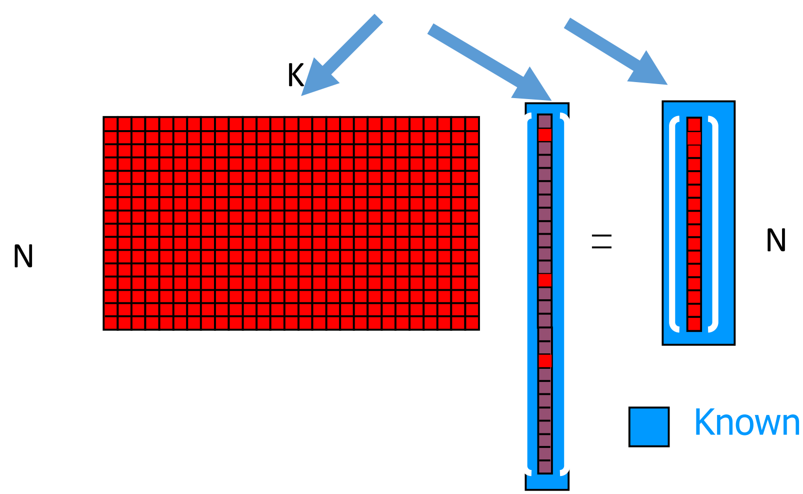
then Basis-Pursuit: $\min_\alpha \ \|\alpha\|_1 \ \ \text{s.t.} \ \ \|\mathbf{D}\alpha - y\|_2 \leq \varepsilon$

leads to a stable result: $\|\hat{\alpha} - \alpha\|_2^2 \leq \frac{4\varepsilon^2}{1-\mu(4\|\alpha\|_0 - 1)}$

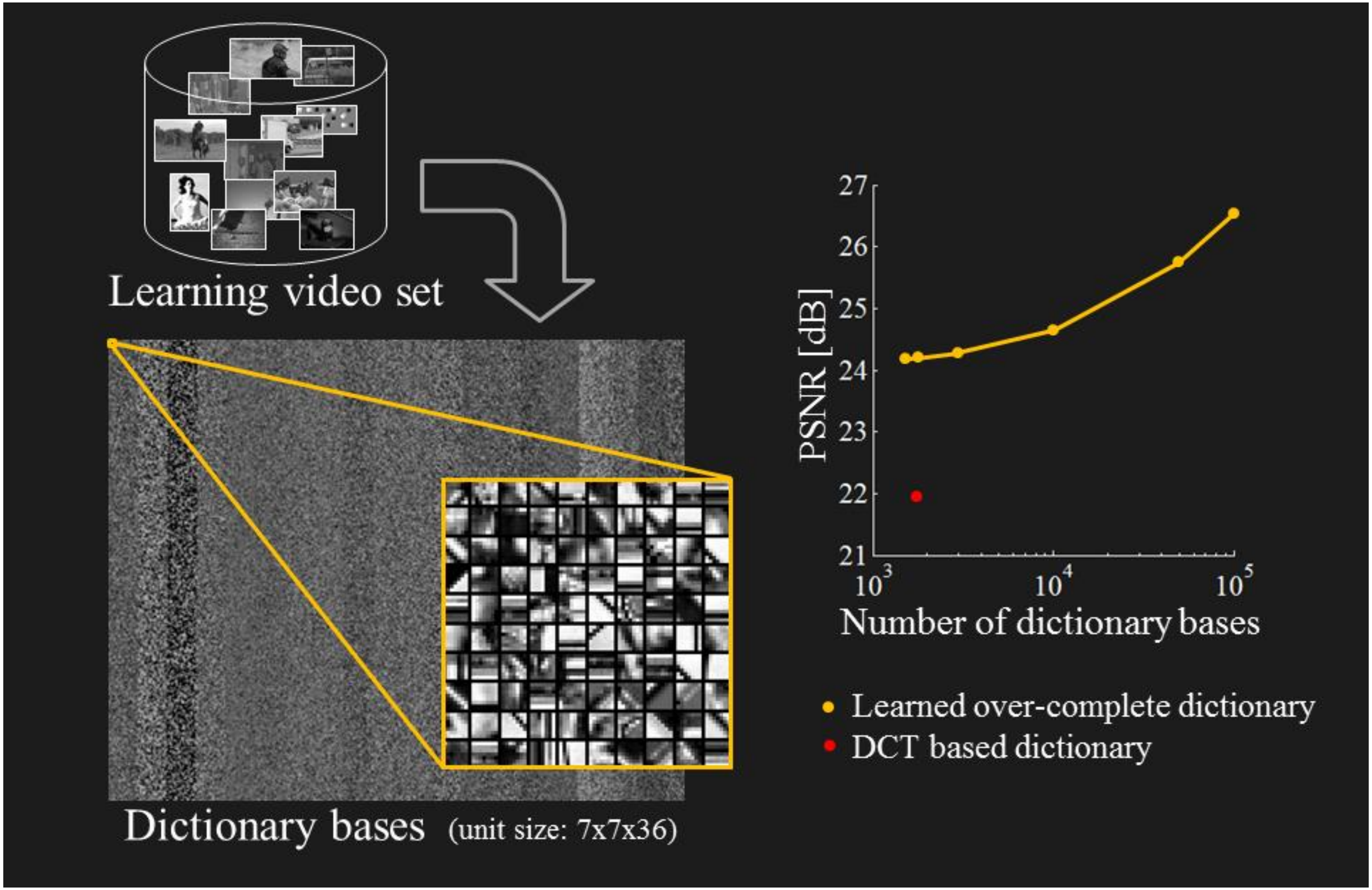
Dictionary Learning

- How to correctly choose the basis for representing the data ?

$$D\alpha = x$$



Accuracy increases with dictionary size



Y. Hitomi, J. Gu, M. Gupta, T. Mitsunaga, and S. K. Nayar. Video from a single coded exposure photograph using a learned over-complete dictionary. In IEEE Intl. Conf. Computer Vision, 2011

Denoising

Image Denoising [E. & Aharon ('06)]

The MAP estimator for denoising this image patch is built by solving

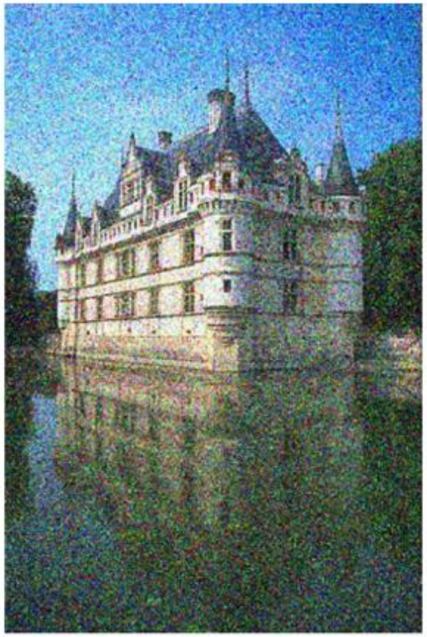
$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_0 \text{ subject to } \|\mathbf{D}\boldsymbol{\alpha} - \mathbf{y}\|_2^2 \leq T \quad \longrightarrow \quad \hat{\mathbf{x}} = \mathbf{D}\hat{\boldsymbol{\alpha}}$$

De-noising

- Learn a patch dictionary.
- For each patch, compute the sparse representation then use it to reconstruct the patch.

$$x^* = \arg\min_{x} \|x\|_1 + \lambda \|Ax - b\|_1$$

$$b = Ax^*$$



What data to train on?

Option 1:

- ❑ Use a database of images,
- ❑ Works fine (~0.5-1dB below the SotA).

Option 2:

- ❑ Use the corrupted image itself !!
- ❑ Simply sweep through all patches of size N-by-N (overlapping blocks),
- ❑ Image of size 1000^2 pixels ~10^6 examples to use – more than enough.
- ❑ This works much better!

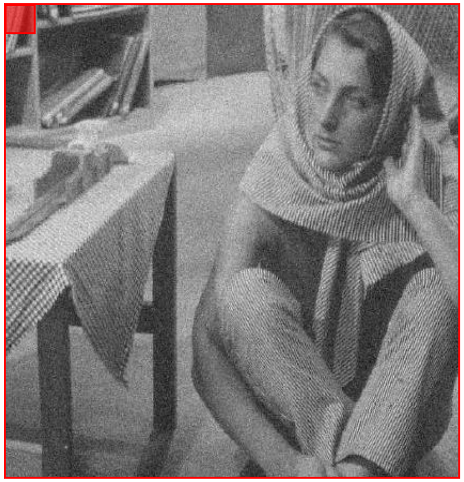


Fig. 6. Example of the denoising results for the image “Barbara” with $\sigma = 20$—the original, the noisy, and two restoration results.