Università
della
Svizzera
italiana

Faculty
of Informatics

Institute of
Computational
Science
ICS

Stochastic Methods

Solving optimization problems,

PCA in practice – feature selection, compression

Seminar 3

Spring 2019

Illia Horenko, <u>Lukas Pospisil</u>

Outline

1. More complaints about official tSNE Matlab implementation: magic parameters?

2. Some optimization problems
   - Equality constrained problem
   - Inequality constrained problem

3. PCA in practice:
   - feature selection
   - compression

2.) Some optimization problems

Problem 1: Find the dimensions of the box with largest volume if the total surface area is s>0.

Problem 2:

Find $\arg\min 4x^2 + 10y^2 - 3$ s.t. $x - y \geq 1$

3a) PCA: feature selection

Find optimal basis such that every image can be written as linear combination:

$$= \quad \alpha_1 \boxed{?} + \alpha_2 \boxed{?} + \alpha_3 \boxed{?}$$

(“orthonormal” images)

See Atelier 3

!!!

Find optimal basis such that every image can be written as linear combination:

$$\approx \alpha_1 \boxed{?} + \alpha_2 \boxed{?} + \alpha_3 \boxed{?} + \text{orig. mean}$$

“optimal” approximation

- Matlab is ready for working with images (see “imread”, “imwrite”)
- Instead of working with rectangular images – work with vectorized images (see “reshape”)
- We can use command “pca”, but it is too boring, let’s implement it from the scratch
- Not from the total scratch – use “eig”/”eigs”

PCA algorithm

1. substract mean value from the data
2. construct covariance matrix
3. compute m largest eigenvalues and corresponding orthonormal eigenvectors, construct Q
4. project the data $X_{\text{reduced}} = Q^T X_{\text{original}}$

- Matlab is ready for working with images (see “imread”, “imwrite”)
- Instead of working with rectangular images – work with vectorized images (see “reshape”)
- We can use command “pca”, but it is too boring, let’s implement it from the scratch
- Not from the total scratch – use “eig”/”eigs”

PCA algorithm

“Mean” image

1. substract mean value from the data
2. construct covariance matrix
3. compute m largest eigenvalues and corresponding orthonormal eigenvectors, construct Q
4. project the data $X_{\text{reduced}} = Q^T X_{\text{original}}$

- Matlab is ready for working with images (see “imread”, “imwrite”)
- Instead of working with rectangular images – work with vectorized images (see “reshape”)
- We can use command “pca”, but it is too boring, let’s implement it from the scratch
- Not from the total scratch – use “eig”/”eigs”

PCA algorithm

“Mean” image

1. substract mean value from the data
2. construct covariance matrix
3. compute m largest eigenvalues and corresponding orthonormal eigenvectors, construct Q
4. project the data $X_{\text{reduced}} = Q^T X_{\text{original}}$

Be smart and avoid for-loop

- Matlab is ready for working with images (see “imread”, “imwrite”)
- Instead of working with rectangular images – work with vectorized images (see “reshape”)
- We can use command “pca”, but it is too boring, let’s implement it from the scratch
- Not from the total scratch – use “eig”/”eigs”

PCA algorithm

“Mean” image

1. substract mean value from the data
2. construct covariance matrix
3. compute m largest eigenvalues and corresponding orthonormal eigenvectors, construct Q
4. project the data $X_{\text{reduced}} = Q^T X_{\text{original}}$

Be smart and avoid for-loop

You don’t want to compute all of the eigenvectors!, use “eigs”

m = 1

m = 1

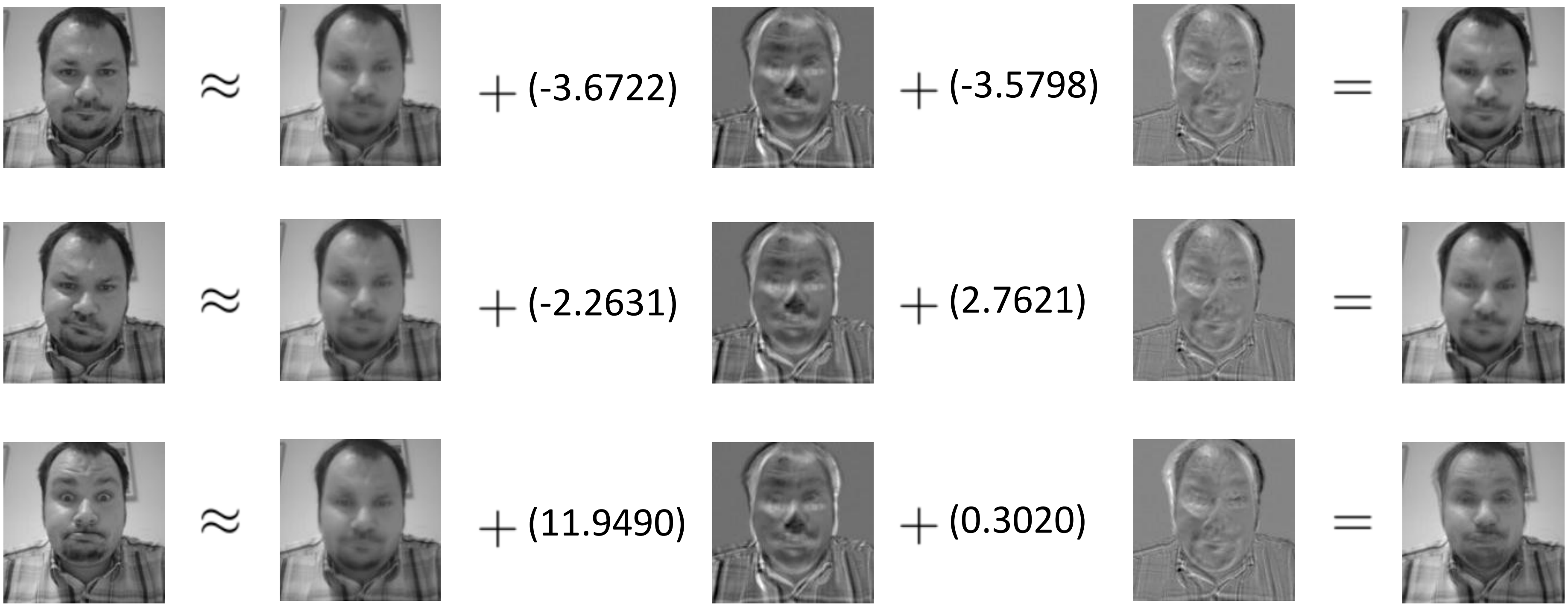
(only the largest deviations from mean are captured)

m = 1

(only the largest deviations from mean are captured)

m = 2

m = 2

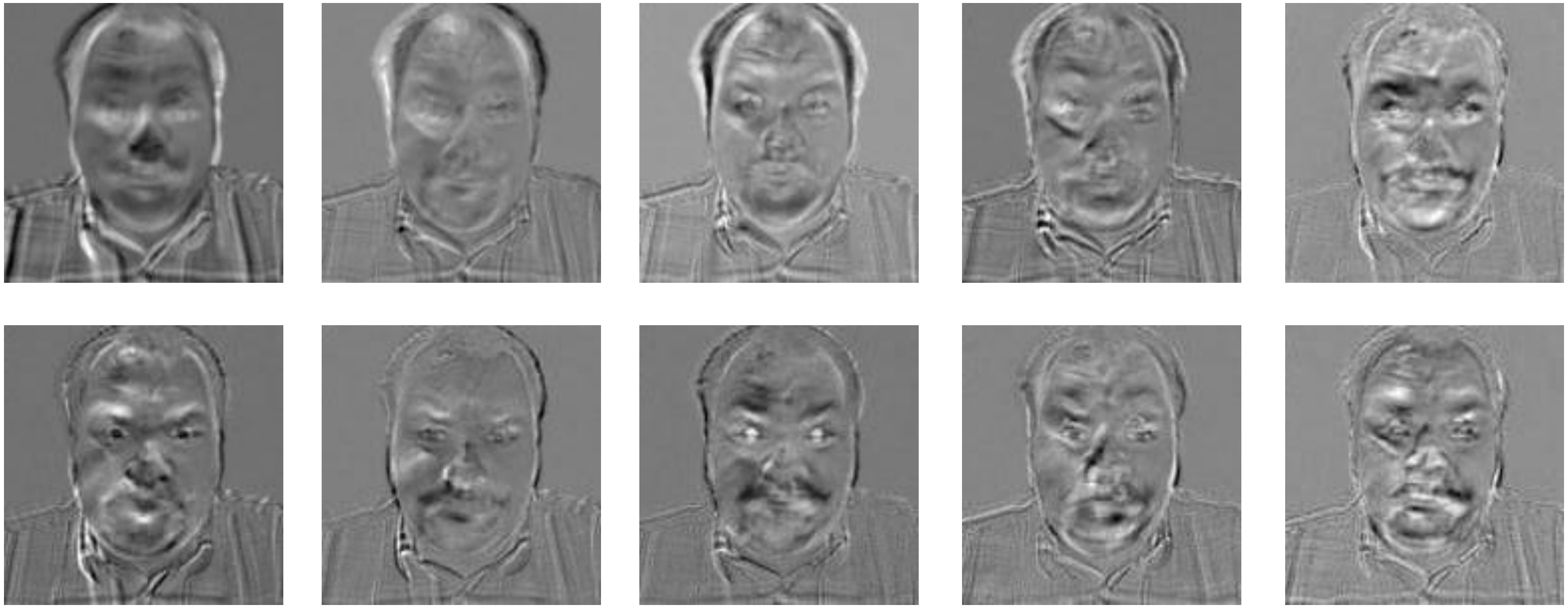


(only the two largest deviations from mean are captured)

m = 2

(only the two largest deviations from mean are captured)

m = 10



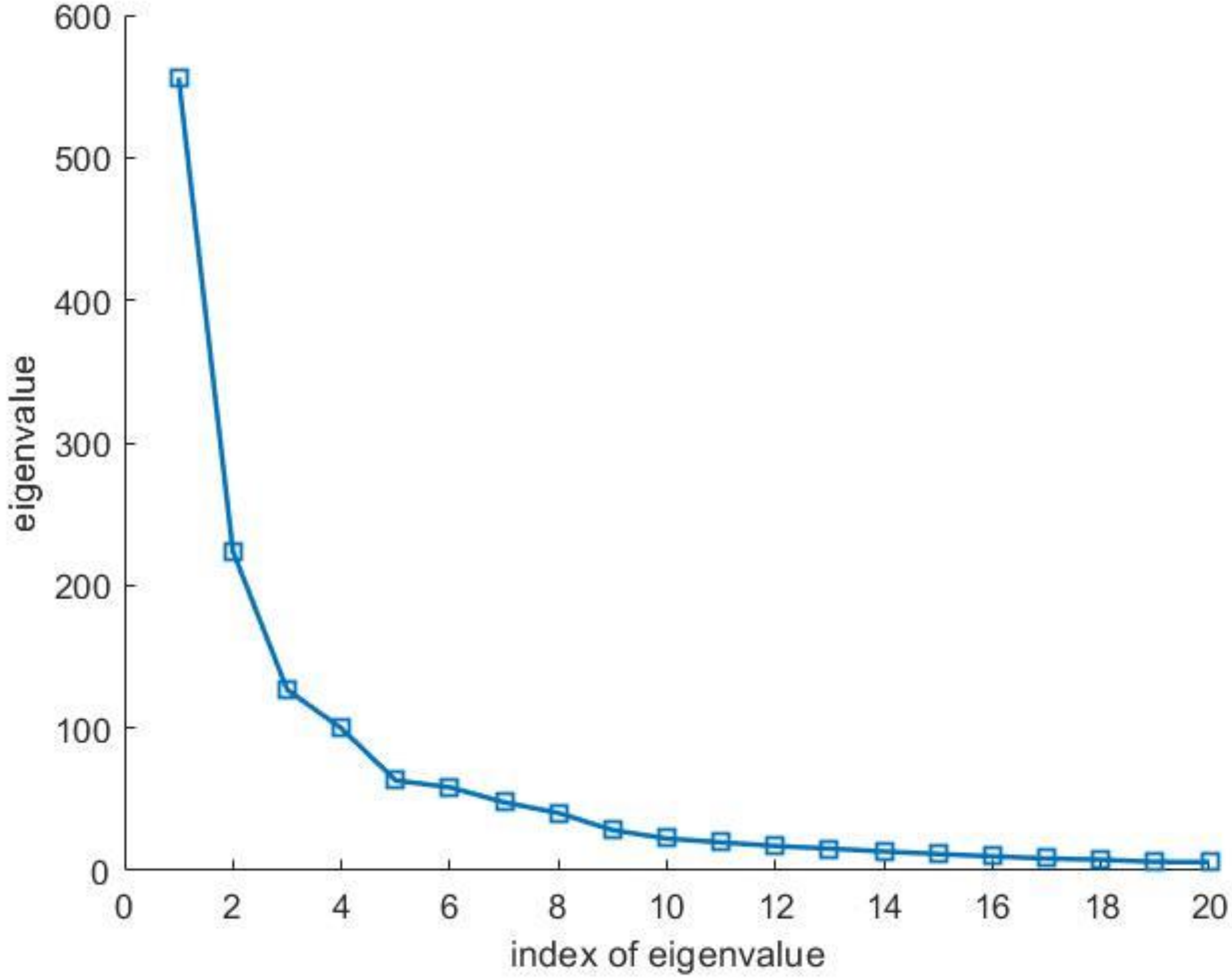
Size of original dataset: T*width*height = 24*140*140 = 470400 values

Size of reduction: $\underbrace{T*m}_{\text{coeffs}} + \underbrace{m*width*height}_{\text{basis vectors}} + \underbrace{width*height}_{\text{mean face}} = 215840$ values

Compression?

See next example

m = 10

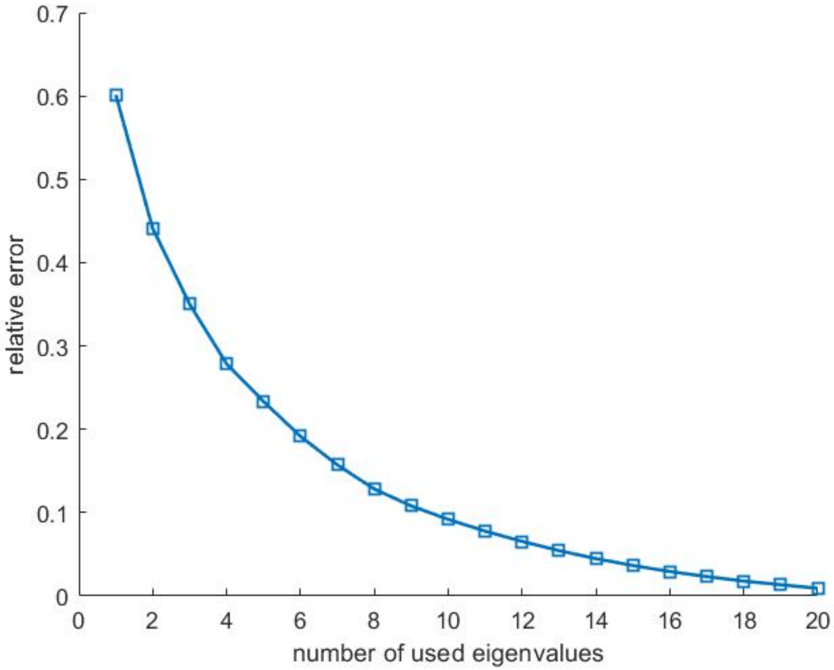


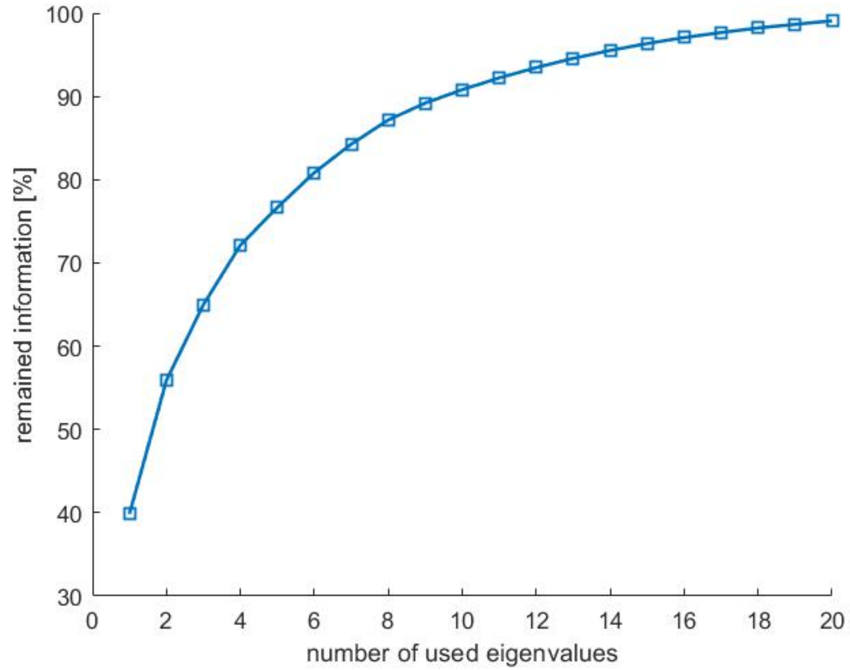
eigenvalue
index of eigenvalue
600
500
400
300
200
100
0
0
2
4
6
8
10
12
14
16
18
20

$$\text{relative error} := \frac{\sum\limits_{i=m+1}^{n} \lambda_i}{\sum\limits_{i=1}^{n} \lambda_i} \qquad \text{remained information in the reduced data} = \left(1 - \frac{\sum\limits_{i=m+1}^{n} \lambda_i}{\sum\limits_{i=1}^{n} \lambda_i}\right) \cdot 100[\%]$$

$$\text{relative error} := \frac{\sum_{i=m+1}^{n} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \qquad \text{remained information in the reduced data} = \left(1 - \frac{\sum_{i=m+1}^{n} \lambda_i}{\sum_{i=1}^{n} \lambda_i}\right) \cdot 100[\%]$$

Hint: use trace instead of computing all eigenvalues!





- Method is completely knowledge free
- PCA can distiguish what is important and what is redundant
- By rearranging pixels column by column to a 1D vector, relations of a given pixel to pixels in neighboring rows are not taken into account.
- Another disadvantage is in the global nature of the representation; small change or error in the input images influences the whole eigen-representation.

More info: google “eigenface”

3b) PCA: compression

(motivated by previous example)

This is Illia.

Illia is too big.

$3 \times 1024 \times 1024 = 3145728$ values

Let’s compress him.

(find optimal reduced space and the representation of projection of this image onto this space)

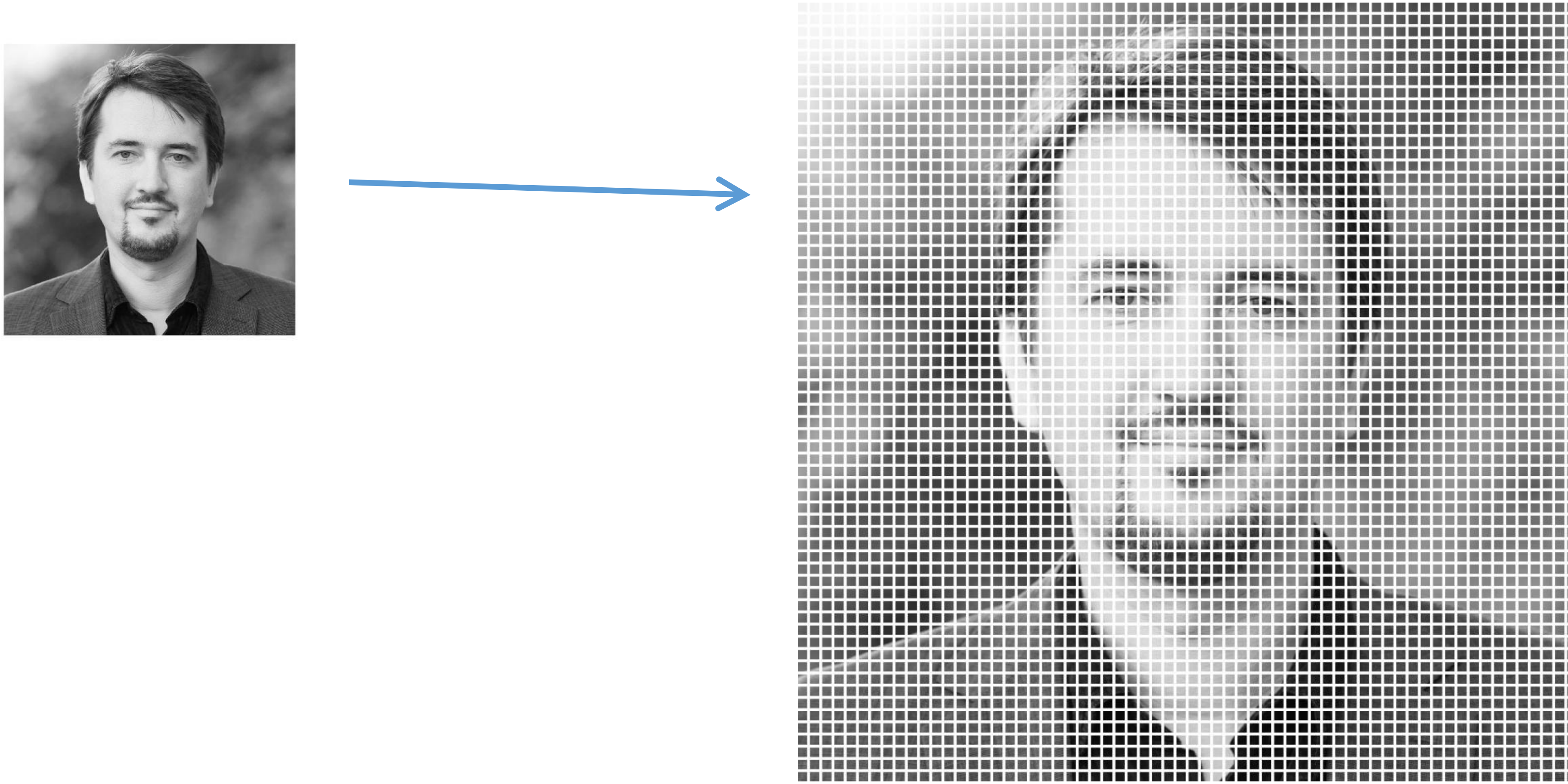
(note: actually this is JPEG image and it is already compressed)

red

green

blue

We will deal with each channel individually.



- Divide the original 1024 x 1024 image into patches:
- Each patch is an instance that contains 16x16 pixels on a grid
- Patch = face in previous example
- Consider each as a 256-D vector and compress it using PCA

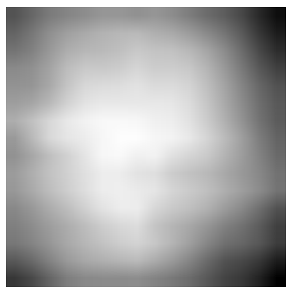
- $T = 4096$, $n = 256$

m = 1

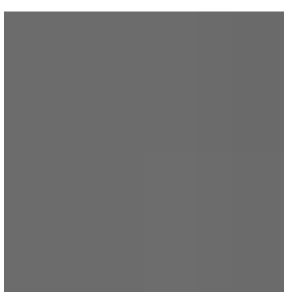
R mean

R eigenvector

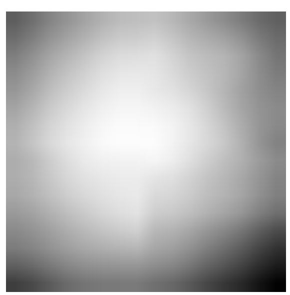
G mean



G eigenvector



B mean



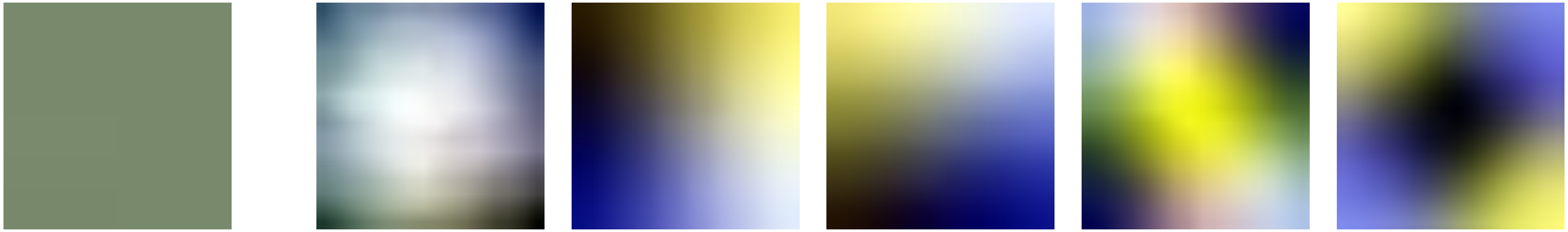
B eigenvector

RGB mean

RGB eigenvector

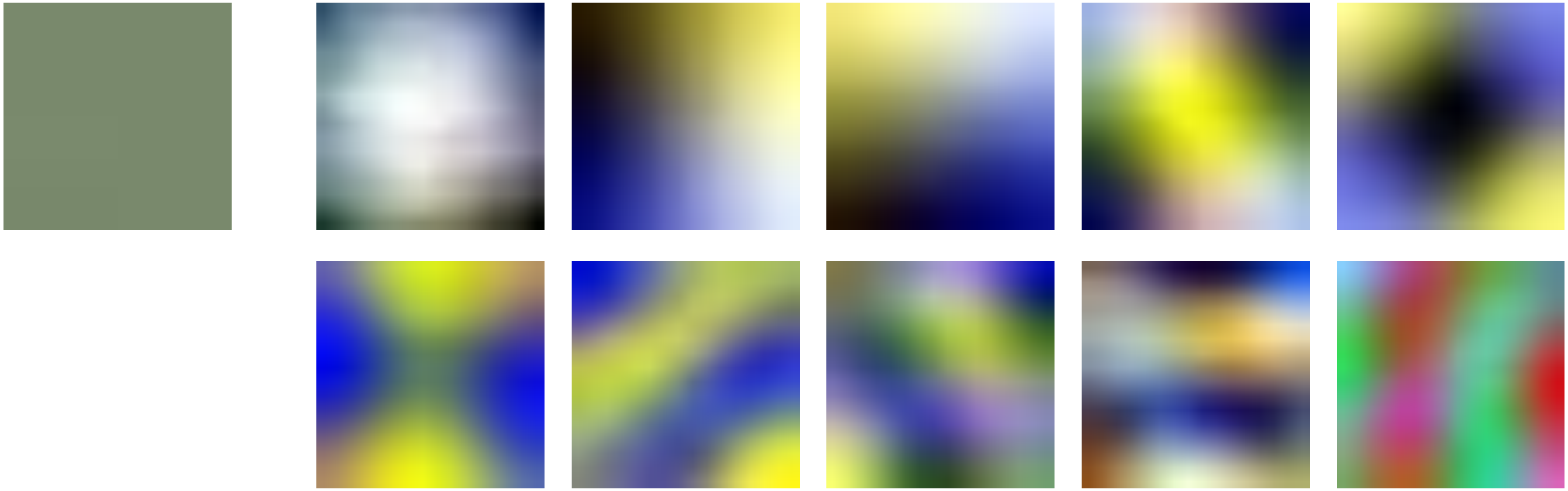
m = 1

m = 5



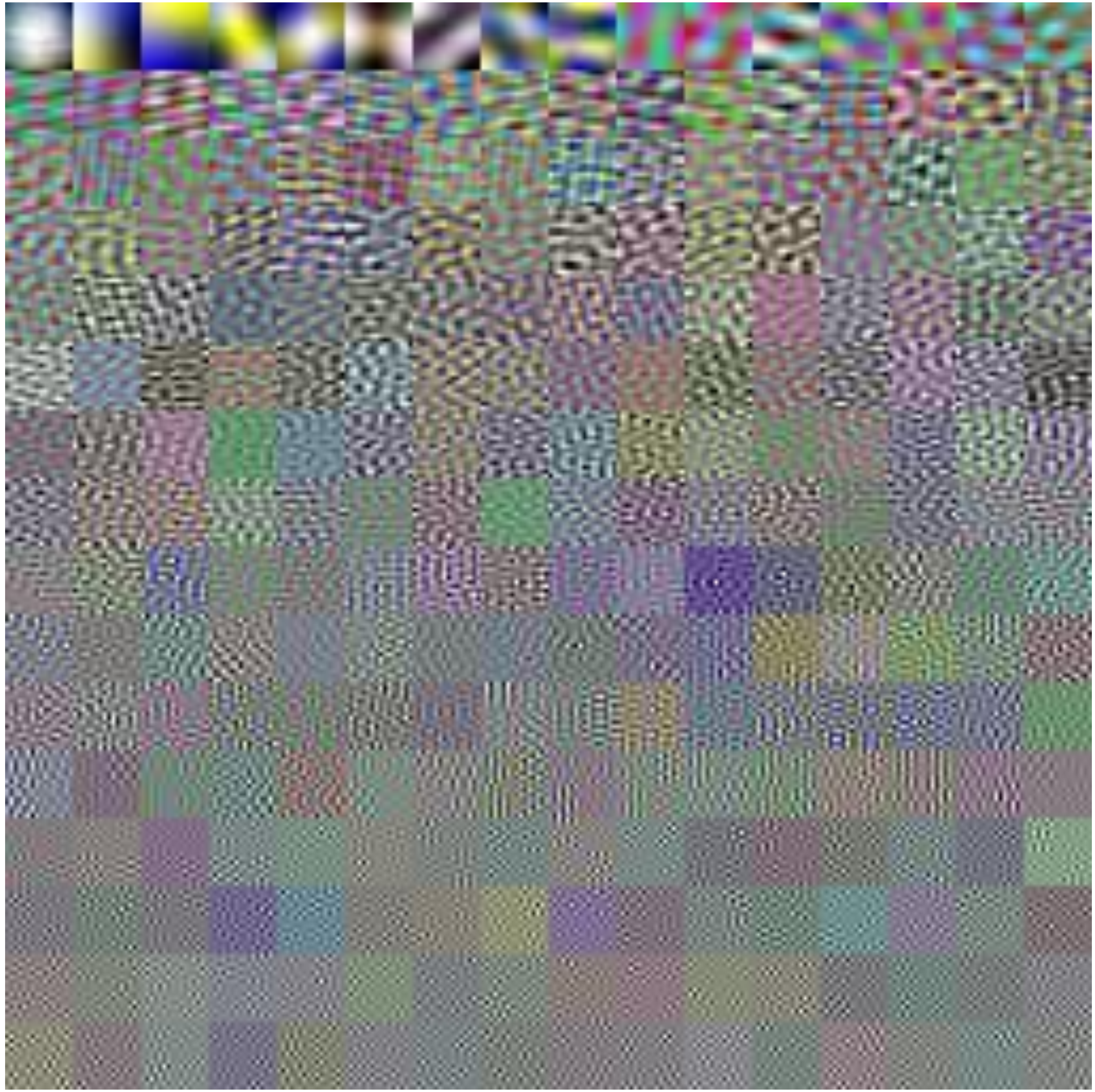
m = 5

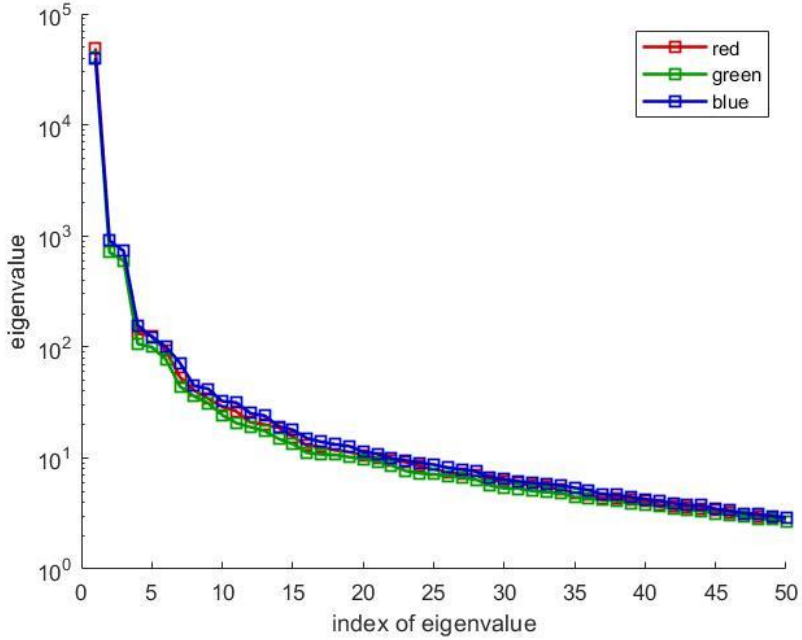
m = 10



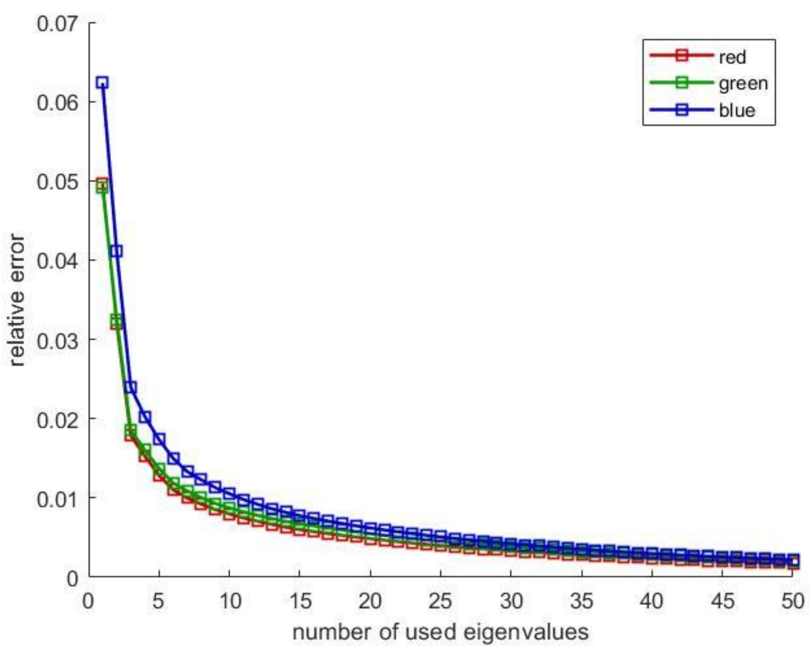
m = 10

m = 256

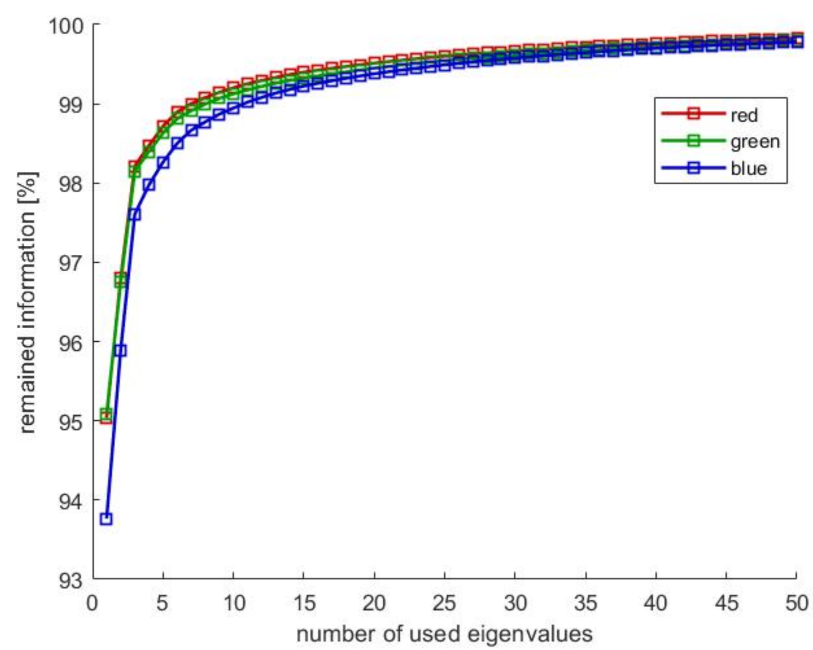




red
green
blue
eigenvalue
10^0
10^1
10^2
10^3
10^4
10^5
0 5 10 15 20 25 30 35 40 45 50
index of eigenvalue



red
green
blue
relative error
0 0.01 0.02 0.03 0.04 0.05 0.06 0.07
0 5 10 15 20 25 30 35 40 45 50
number of used eigenvalues



red
green
blue
remained information [%]
93 94 95 96 97 98 99 100
0 5 10 15 20 25 30 35 40 45 50
number of used eigenvalues

$$\text{Compression Ratio} = \frac{\text{Uncompressed Size}}{\text{Compressed Size}}$$

3 x 1024 x 1024 = 3145728 values

$$(\underbrace{\text{nmb\_of\_patches*m}}_{\substack{\text{coeffs}\\\text{(reduced data)}}} + \underbrace{\text{m*patch\_size* patch\_size}}_{\text{basis vectors}} + \underbrace{\text{patch\_size*patch\_size}}_{\text{mean}}) * 3$$