Comparison Between Some Matrix Methods with Applications in Pattern Recognition

Elena Pelican [1]

Faculty of Mathematics and Computer Science
"Ovidius" University, Constanta, Romania
{epelican}@univ-ovidius.ro

ALA Conference
May 24-28, 2010, Novi Sad

[1] joint work with L. Grecu

Outline

The Problem

Problem:

Given an image z (the picture of a person/digit), we want to find the closest image from our database $\{v_1, \ldots, v_M\}$.

The Simplest Method - 1

Solution:

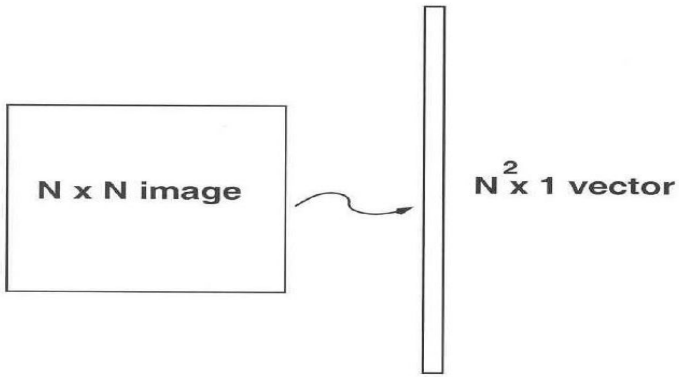
The simplest method (intuitive): is to compare z with every v_i. We have to find out an index $i_0 \in \{1, \ldots, M\}$ such that

$$\|z - v_{i_0}\| = \min_{1 \leq i \leq M} \|z - v_i\| .$$

The Simplest Method - 2

The database

- the pictures have the same resolution $N \times N$;
- every single image is transformed into a vector.

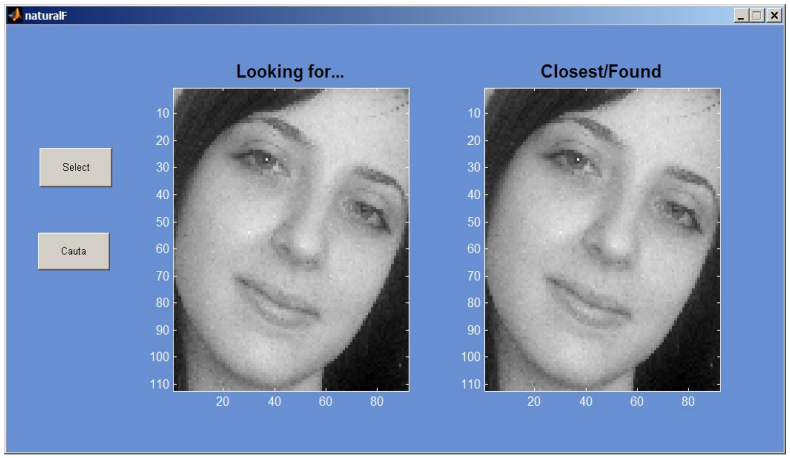


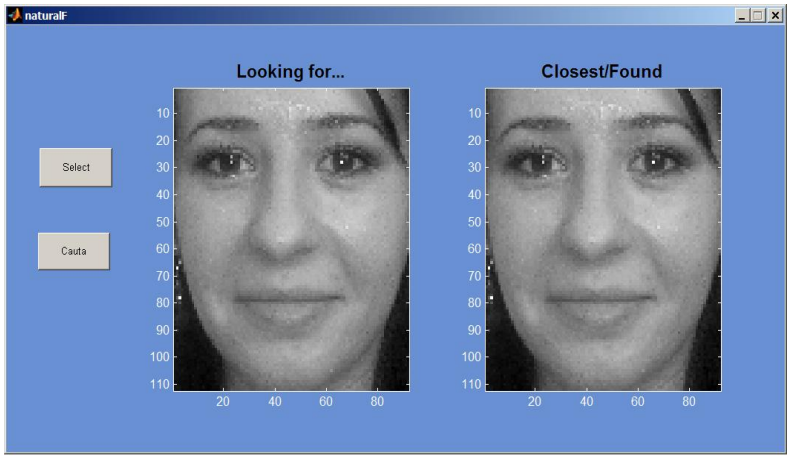
Experiments (The simplest method)

- the faces database contains the pictures of 11 persons with 10 pictures for each person;
- the digit database contains the pictures of all 10 digits with 10 pictures for each digit.

The Simplest Method - 3

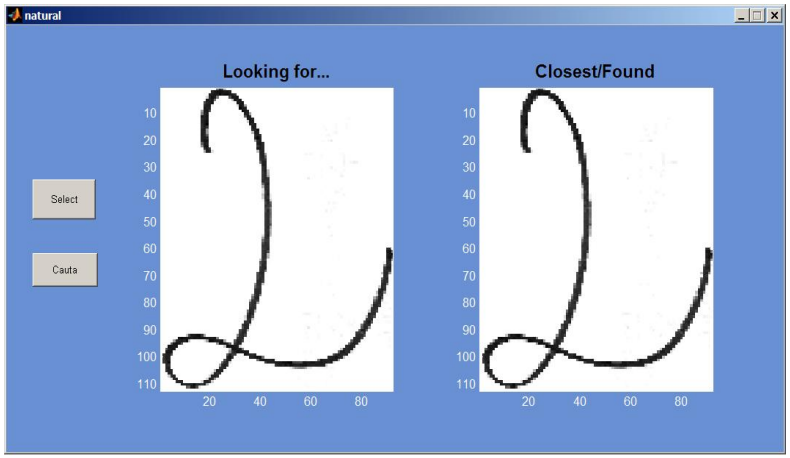
Results for faces (consistent case).

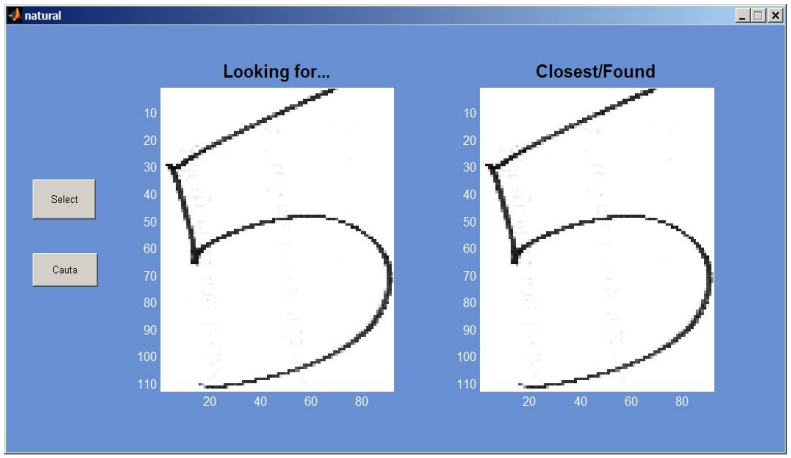




The Simplest Method - 4

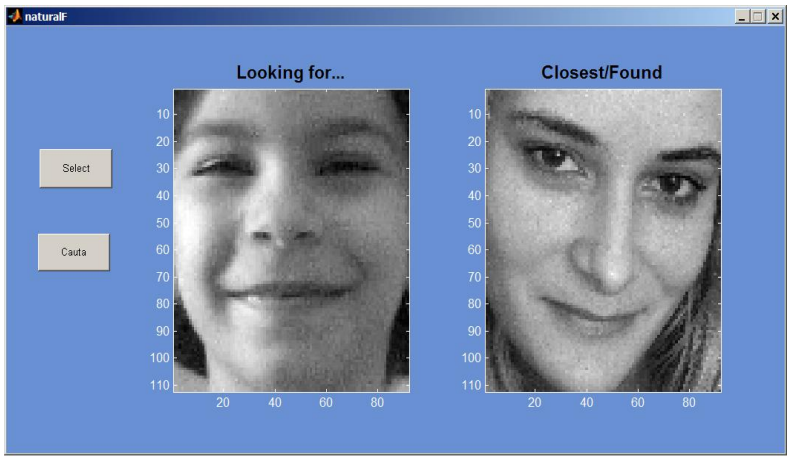
Results for digits (consistent case).





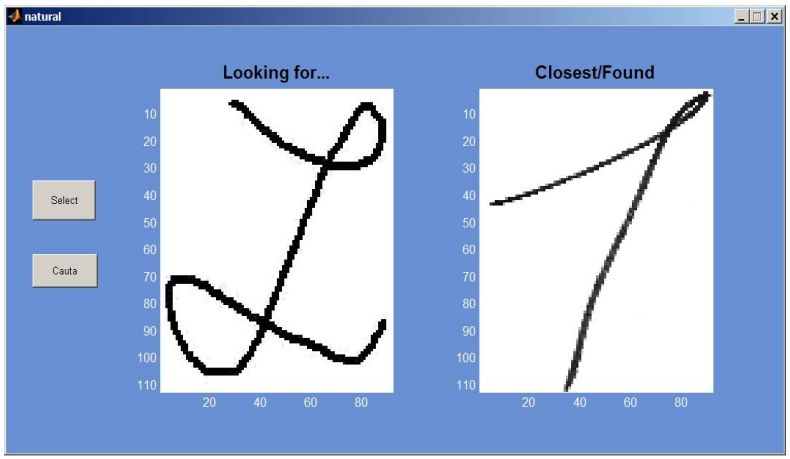
The Simplest Method - 5

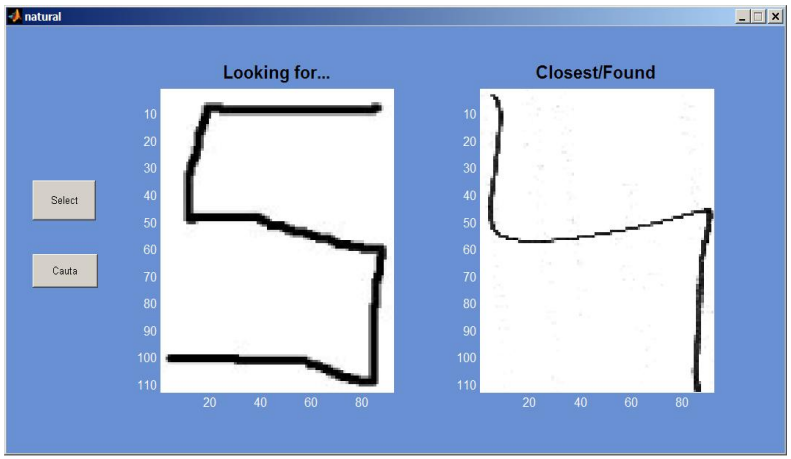
Results for faces (inconsistent case).



The Simplest Method - 6

Results for digits (inconsistent case).





Eigenfaces Algorithm - 1

In Eigenfaces algorithm (proposed by [Turk and Pentland, 1991]) we have to pursue the following steps:

1. The database composed by the images $I_1, I_2, \ldots, I_M$ (all images having the same resolution $N \times N$).
2. Set every single image I_i as a vector Γ_i : $N^2 \times 1$.
3. Compute the average face vector Ψ:
$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i.$$
4. Subtract the average face vector from all vectors.
$$\Phi_i = \Gamma_i - \Psi.$$
5. Compute the covariance matrix C:
$$C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^T = AA^T \ \left(N^2 \times N^2\right).$$
where $A = [\Phi_1 \ \Phi_2 \ \ldots \ \Phi_M] \ \left(N^2 \times M\right)$.
6. Compute the eigenvectors q_i of $C = AA^T$.

Eigenfaces Algorithm - 2

Remarks:

- $C = AA^T \Rightarrow \exists Q$ orthogonal such that
  $Q^T C Q = \text{diag}(\sigma_1, \ldots, \sigma_{N^2}) = D$,
  $\sigma_i \geq 0,\ \sigma_1 \geq \ldots \geq \sigma_r > 0$;
- $Q = \text{col}[q_1, \ldots, q_{N^2}]$;
- $Cq_i = \sigma_i q_i$ şi $\{q_1, \ldots, q_{N^2}\}$ forms an orthonormal basis in $\mathbb{R}^{N^2} \Rightarrow \forall x \in \mathbb{R}^{N^2},\ x = \sum_{i=1}^{N^2} < x, q_i > q_i$.

Note: Similar considerations can be made for $\hat{C} = A^T A$.

Eigenfaces Algorithm - 3

But $C : N^2 \times N^2$, N^2 =resolution $\Rightarrow$ "big"! $\Rightarrow$ the diagonalization $Q^T C Q$ =diag$(\sigma_1, \ldots, \sigma_{N^2})$ can be very expensive.
Solution: From all N^2 eigenvectors $q_1, \ldots, q_{N^2}$ we keep only the first K, corresponding to the K largest eigenvalues $\sigma_1 \geq \ldots \geq \sigma_K > 0$.
We have

$$Cx = \sum_{j=1}^{N^2} x_j C q_j = \sum_{j=1}^{N^2} \sigma_j x_j q_j$$

and

$$Cx = \sum_{j=1}^{N^2} < x, \Phi_j > \Phi_j.$$

So we obtain $Cx = \sum_{j=1}^{N^2} < x, \Phi_j > \Phi_j \approx \sum_{j=1}^{K} \sigma_j x_j q_j.$

Eigenfaces Algorithm - 4

We represent Φ_i in the basis $\{q_1, \ldots, q_{N^2}\}$

$$\Phi_i = \Gamma_i - \Psi = \sum_{j=1}^{N^2} < \Phi_i, q_j > q_j.$$

We approximate $\hat{\Phi}_i \approx \Phi_i$ by

$$\hat{\Phi}_i = \sum_{j=1}^{K} < \Phi_i, q_j > q_j = \sum_{j=1}^{K} w_j^i q_j. \quad (*)$$

The vector $\hat{\Phi}_i$ is represented by the Fourier coefficients vector from (*), i.e.

$$w^i = \begin{bmatrix} w_1^i \\ w_2^i \\ \vdots \\ w_K^i \end{bmatrix}, \in I\!R^K \ i = 1, 2, \ldots, M.$$

Eigenfaces Algorithm - 5

Given a image, Γ, with the same resolution as Γ_i, then we follow the steps:

1. Normalize Γ: $\Phi = \Gamma - \Psi$.
2. Project on the eigenvectors space
$$\hat{\Phi} = \sum_{j=1}^{K} < \Phi, q_j > q_j = \sum_{j=1}^{K} w_j q_j.$$
3. Represent $\hat{\Phi}$ as: $w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{bmatrix}$.
4. Find $i_0 \in \{1, \ldots, M\}$ satisfying $\left\| w - w^{i_0} \right\| = \min_{1 \leq i \leq M} \left\| w - w^i \right\|$.

Experiments with Eigenfaces Algorithm - 1

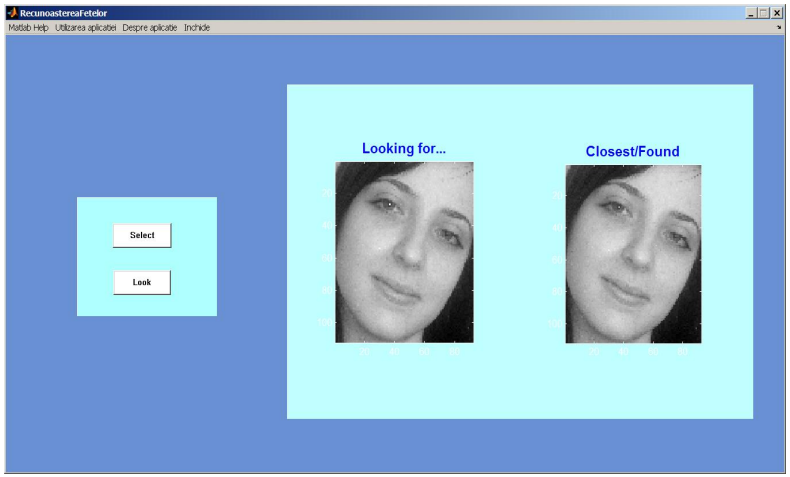
Note. The algorithm works also for images with $N \times N_1$ resolution. In this case we obtain a vector Γ of dimension $NN_1 \times 1$.

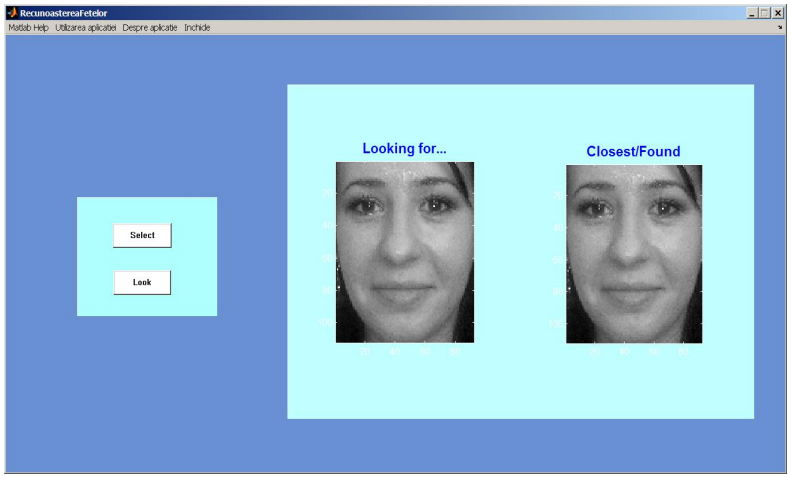
- We have
  - a database with 110 persons $(92\times112)^2$
  - a database with 100 digits (92×112)
- We considered $K = 20$

[2] We also tested on 100×100 and 105×120.

Experiments with Eigenfaces Algorithm - 2

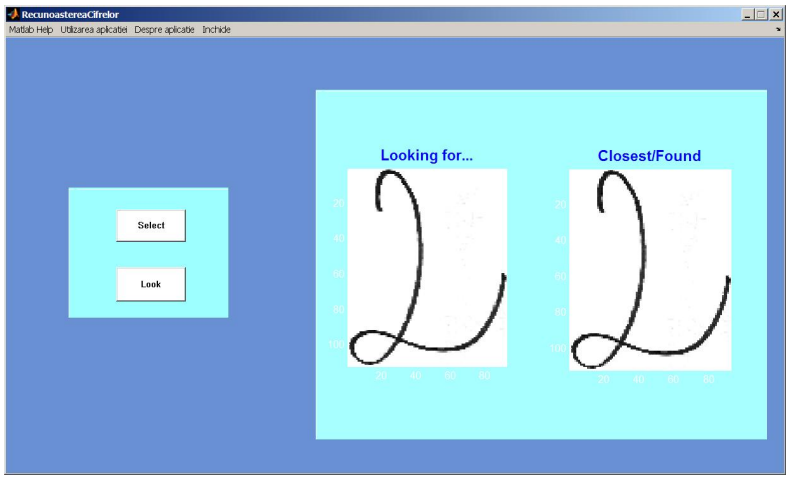
Results for faces (consistent case).

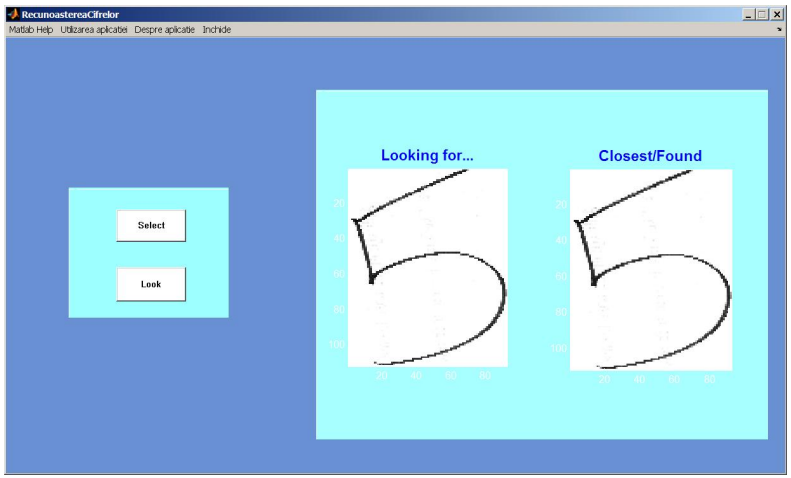




Experiments with Eigenfaces Algorithm - 3

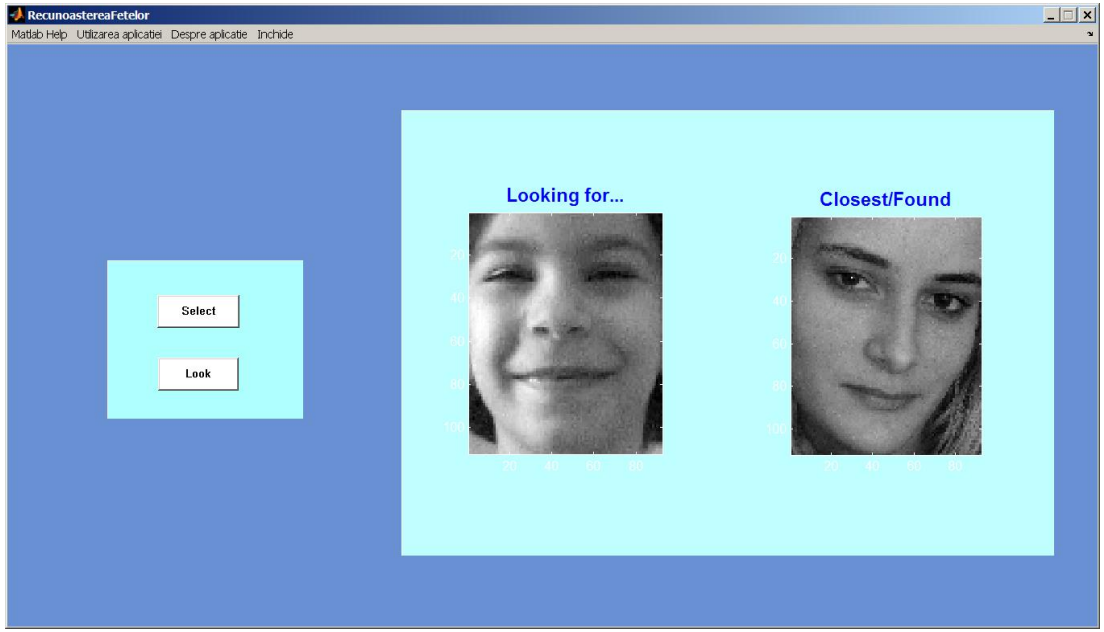
Results for digits (consistent case).





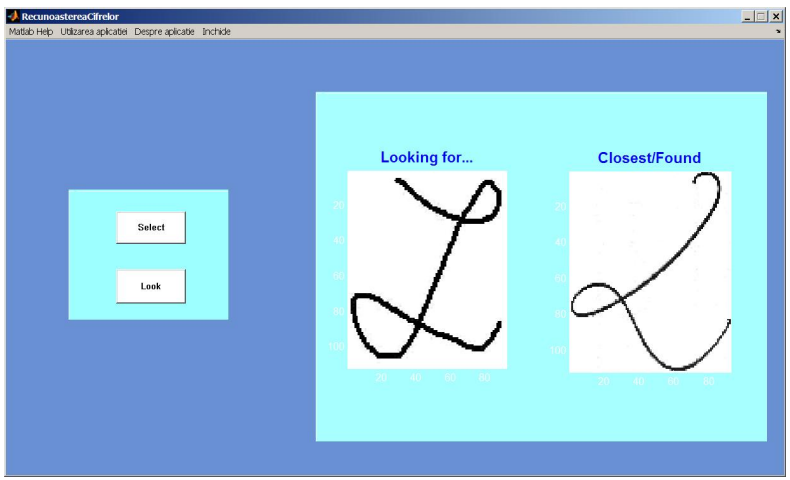
Experiments with Eigenfaces Algorithm - 4

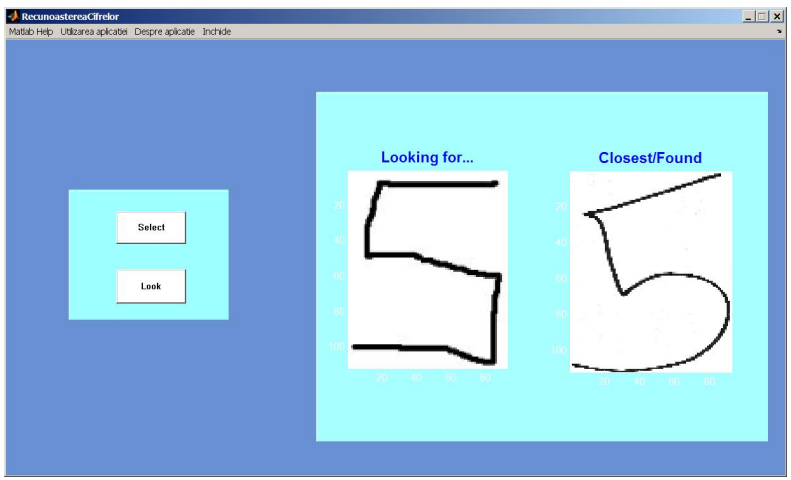
Results for faces (inconsistent case).



Experiments with Eigenfaces Algorithm - 5

Results for digits (inconsistent case).



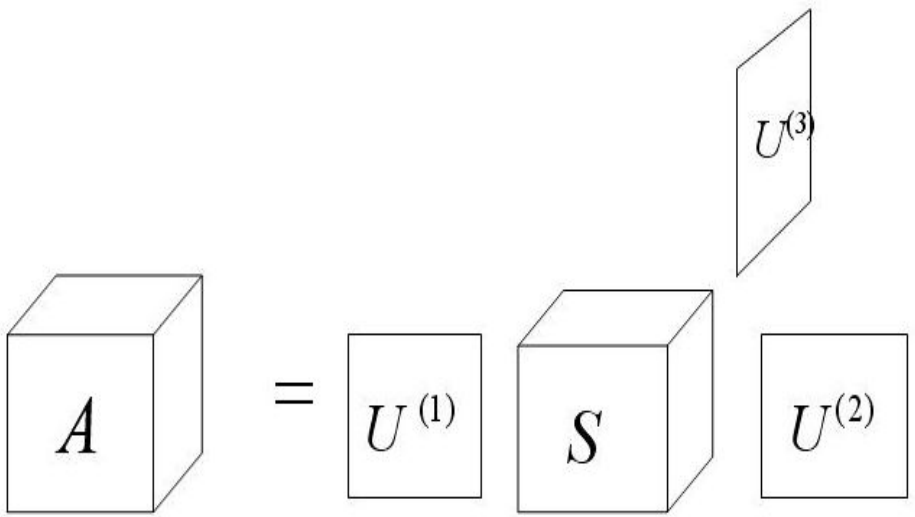


HOSVD (Higher Order SVD)

The tensor $A \in \mathbb{R}^{l \times m \times n}$ can be written as

$$A = S \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)}$$

where $U^{(1)} \in \mathbb{R}^{l \times l}$, $U^{(2)} \in \mathbb{R}^{m \times m}$, $U^{(3)} \in \mathbb{R}^{n \times n}$ are orthogonal matrices. S is a tensor of the same dimensions as A and has the property that any two slices of S are orthogonal. $U^{(i)}$ result from $A_{(i)} = U^{(i)} \Sigma^{(i)} \left(V^{(i)}\right)^T$, $A_{(i)} = unfold_i(A)$.



Notations - 1

Let $A \in \mathbb{R}^{l \times m \times n}$, $U \in \mathbb{R}^{l_0 \times l}$, and $A \times_1 U$ is a tensor of dimension $l_0 \times m \times n$ defined by

$$(A \times_1 U)(j, i_2, i_3) = \sum_{k=1}^{l} u_{j,k} a_{k,i_2,i_3}.$$

Similary we have the 2-mode and 3-mode multiplications of a tensor with a matrix:

$$(A \times_2 U)(i_1, j, i_3) = \sum_{k=1}^{m} u_{j,k} a_{i_1,k,i_3},$$
$$(A \times_3 U)(i_1, i_2, j) = \sum_{k=1}^{n} u_{j,k} a_{i_1,i_2,k}.$$

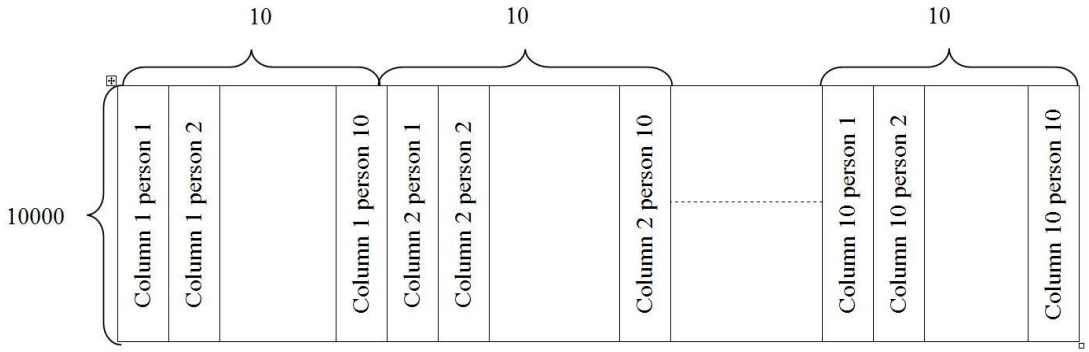
Notations - 2

We have $A_{(i)} = unfold_i(A)$, where

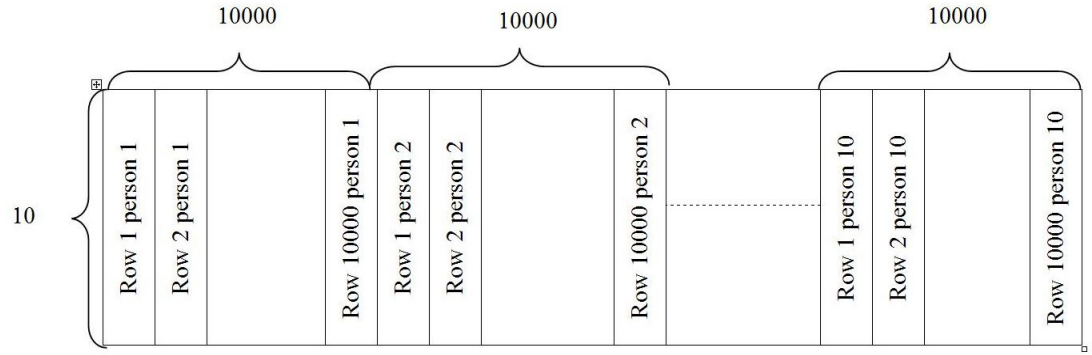
$$
\begin{aligned}
unfold_1\left(A\right) = A_{(1)} &= \left(A\left(:,1,:\right)\ A\left(:,2,:\right)\ \ldots\ A\left(:,m,:\right)\right),\\
unfold_2\left(A\right) = A_{(2)} &= \left(A\left(:,:,1\right)^T\ A\left(:,:,2\right)^T\ \ldots\ A\left(:,:,n\right)^T\right),\\
unfold_3\left(A\right) = A_{(3)} &= \left(A\left(1,:,:\right)^T\ A\left(2,:,:\right)^T\ \ldots\ A\left(l,:,:\right)^T\right).
\end{aligned}
$$

Using these unfoldings we obtain

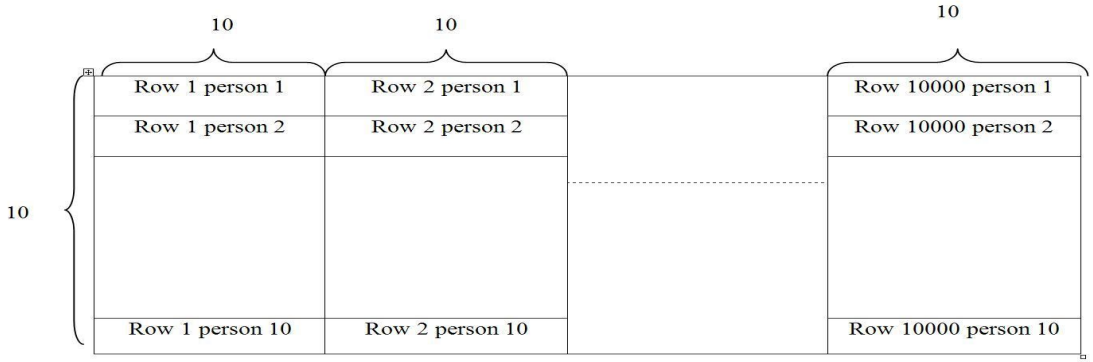
$$
\begin{aligned}
A \times_1 U &= fold_1\left(Uunfold_1\left(A\right)\right),\\
A \times_2 U &= fold_2\left(Uunfold_2\left(A\right)\right),\\
A \times_3 U &= fold_3\left(Uunfold_3\left(A\right)\right).
\end{aligned}
$$



unfolding 1



unfolding 2



unfolding 3

Face Recognition - 1

For our problem, we use the following form of the HOSVD of tensor A

$$A = C \times_p H, \text{ where } C = S \times_i F \times_e G.$$

For a particular expression e we have

$$A(:, e, :) = C(:, e, :) \times_p H,$$

$A(:, e, :) = A_e$ and $C(:, e, :) = C_e$ are matrices. Hence we obtain

$$A_e = C_e H^T, \ e = 1, 2, \ldots, n_e.$$

Face Recognition - 2

The same orthogonal matrix H appears in all n_e relations. If $H^T = \left(h_1 \dots h_{n_p}\right)$ we get

$$a_p^{(e)} = C_e h_p.$$

Let $z \in I\!R^{n_i}$ be the picture of an unknown person, in an unknown expression. The coordinates of z in the e expression basis can be obtained resolving the least squares problem

$$\min_{\alpha_e} \|C_e \alpha_e - z\|_2.$$

Clasification Algorithms - 1

The algorithm (see [Elden, 2007]) is
Algorithm A1

for $e = 1, 2, \ldots, n_e$
    solve $\min_{\alpha_e} \|C_e \alpha_e - z\|_2$
    for $p = 1, 2, \ldots, n_p$
        if $\|\alpha_e - h_p\|_2 < tol$, then is person p and STOP
    end
end

For each image z we have to solve n_e least square problems with $C_e \in I\!R^{n_i \times n_p}$ $\Rightarrow$ lot of time to compute.

Clasification Algorithms - 2

From $C = S \times_i F \times_e G$ we obtain

$$C_e = FB_e,$$

where $B_e \in I\!R^{n_e n_p \times n_p}$, $B_e = (S \times_e G)(:, e, :)$.
$F \in I\!R^{n_i \times n_e n_p}$ and $\hat{F} = \left(FF^{\perp}\right)$. We insert $\hat{F}^T$ inside the norm and we can solve the least squares problems by solving

$$\min_{\alpha_e} \left\| B_e \alpha_e - F^T z \right\|_2, \ e = 1, 2, \ldots, n_e.$$

The algorithm is the following (see [Elden, 2007]).

Clasification Algorithms - 3

Algorithm A2

Compute the QR decomposition of all matrices B_e, $B_e = Q_e R$, with $e = 1, 2, \ldots, n_e$.
Compute $\hat{z} = F^T z$.
for $e = 1, 2, \ldots, n_e$
    solve $R_e \alpha_e = Q_e^T \hat{z}$ for α_e
    for $p = 1, 2, \ldots, n_p$
        if $\|\alpha_e - h_p\|_2 < tol$, then is person p and STOP
    end
end

Tensors and matrices have large dimensions $\Rightarrow$ we can truncate them in such way that the truncated HOSVD can still approximate the tensor A. $F_k = F(:, 1:k)$, with $n_p < k << n_i$ and apply the algorithm A2 for the obtained truncated matrices and tensors.

Experiments with HOSVD - 1

Figure: Consistent case

Experiments with HOSVD - 2

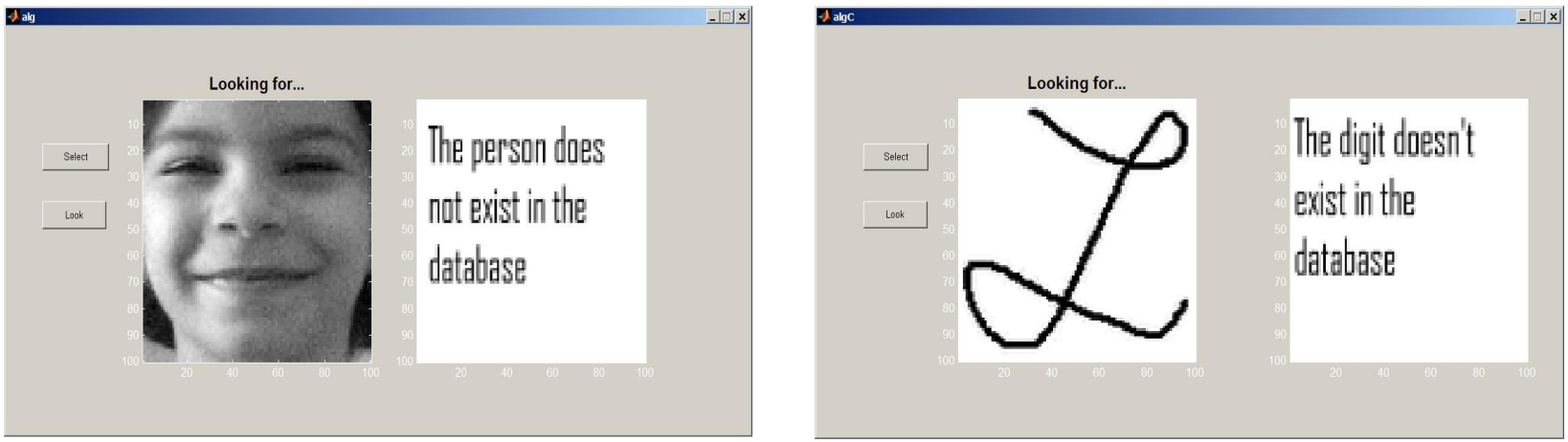


Figure: Inconsistent case

Our Approach - 1

In the previous case, that number k had to be chosen such that is much smaller than n_i, but larger than n_p. We shall make use of the following result (see [Golub and Van Loan, 1996]) in order to compute k from algorithm A2, instead of just choosing a proper k.

Theorem

Let the SVD of $A \in \mathbb{R}^{M \times N}$ *be given by* $A = U\Sigma V^T$. *If* $k < r = rank(A)$ *and*

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

then $\min_{rank(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}$.

Our Approach - 2

The above theorem tells us, that for a given tolerance $\epsilon > 0$ such that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k > \epsilon \geq \sigma_{k+1} \geq \ldots \geq \sigma_p \geq 0,\ p = \min(m, n)$ the matrix A_k given by (1) has the property

$$\|A - A_k\|_2 \leq \epsilon,$$

i.e. $A_k = argmin_{B \in R(k)} \|A - B\|_2$, where $R(k) = \left\{P \in I\!R^{M \times N},\ rank(P) = k\right\}$.

Experiments - 1

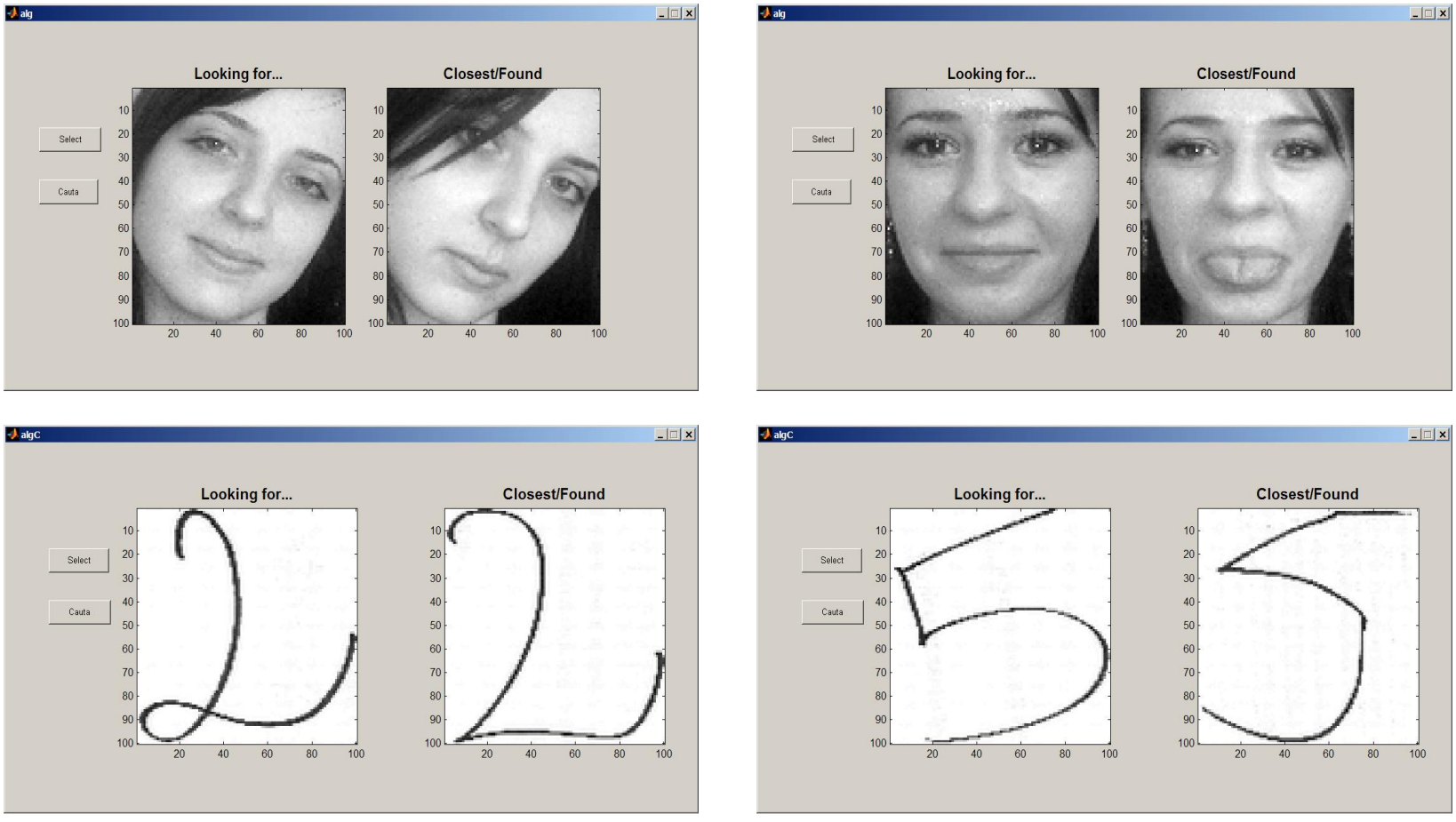


Figure: Consistent case

Experiments - 2

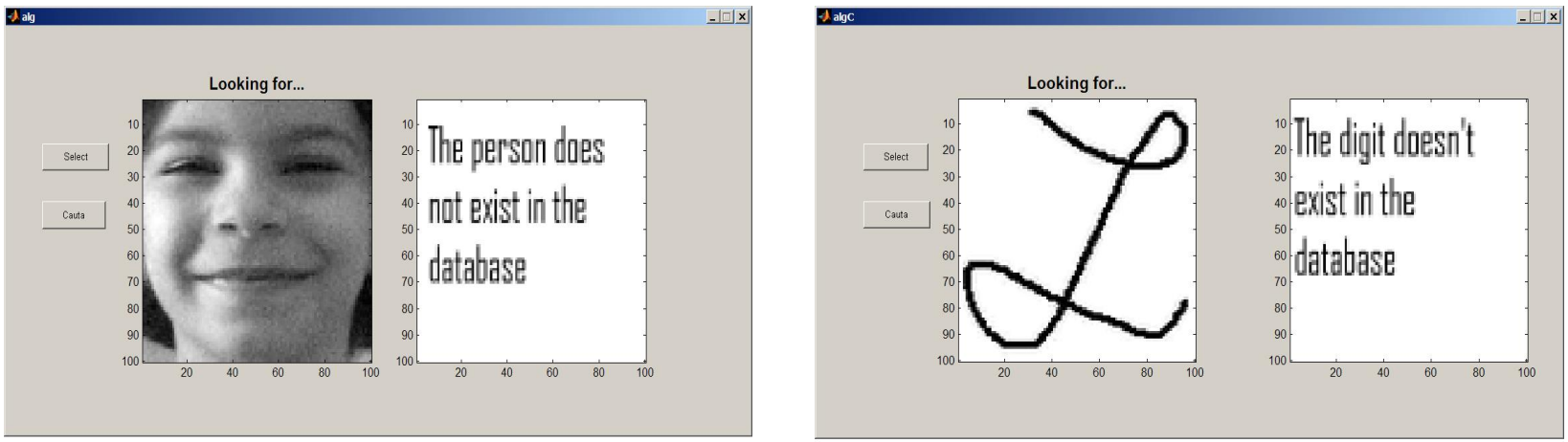


Figure: Inconsistent case

NMF (Nonnegative Matrix Factorization)

Matrix $A \in \mathbb{R}^{n \times m}$ is our face/digit database, a_j (column j) is a face/digit.
NMF aims to find two matrices $W \in \mathbb{R}^{n \times k}$ and $H \in \mathbb{R}^{k \times m}$ such that

$$A \approx WH. \tag{2}$$

After **NMF** we obtain $a_j \approx Wh_j$. We can consider the lines of matrix W as basis images and the vector h_j as the corresponding weight vectors.

NMF - 2

One of the algorithms initially proposed for finding the matrices W and H used the following metric:

$$D_N(A||WH) = \sum_{i,j} \left(a_{ij} \ln \left(\frac{a_{ij}}{\sum_l w_{il} h_{lj}} \right) + \sum_k w_{ik} h_{kj} - a_{ij} \right) \tag{3}$$

as the measure of the cost for factoring A into WH.
The **NMF** factorization is the outcome of optimization:

$$\min_{W,H} D_N(A||WH) \tag{4}$$

subject to $w_{ik} \geq 0,\ h_{kj} \geq 0,\ \sum_i w_{ij} = 1,\ \forall\ j.$

NMF - 3

The update rules for the t-th iteration are given by:

$$h_{kj}^{(t)} = h_{kj}^{(t-1)} \frac{\sum_i w_{ik}^{(t-1)} \frac{a_{ij}}{\sum_l w_{il}^{(t-1)} h_{lj}^{(t-1)}}}{\sum_i w_{ik}^{(t-1)}} \tag{5}$$

and

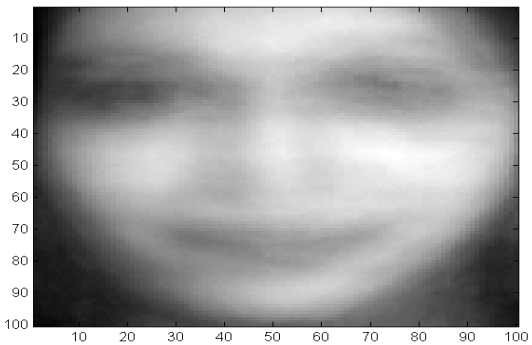
$$w_{ik}^{(t)} = w_{ik}^{(t-1)} \frac{\sum h_{kj}^{(t)} \frac{a_{ij}}{\sum_l w_{il}^{(t-1)} h_{lj}^{(t)}}}{\sum_j h_{kj}^{(t)}}. \tag{6}$$

NMF - 4

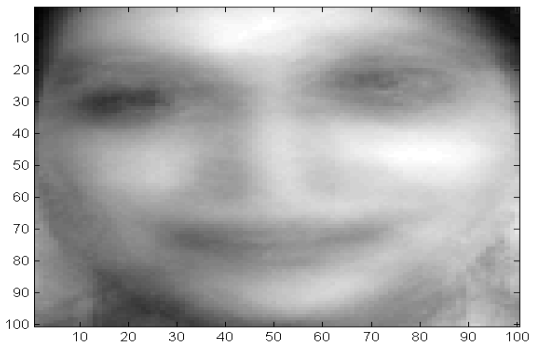
- All **NMF** algorithms are iterative and they are sensitive to the initialization of W and H.
- In all cases, a good initialization can improve the speed accuracy of the algorithms, as it can produce faster convergence to an improved local minimum.
- Nearly all **NMF** algorithms use simple random initialization.

NMF - 5

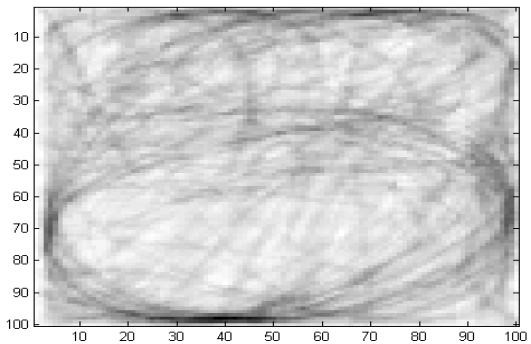
Results of **NMF** reconstruction for faces and digits with (5)-(6) and the initialization: $w_{ij} = \frac{1}{n}$ and H=random(k,m).



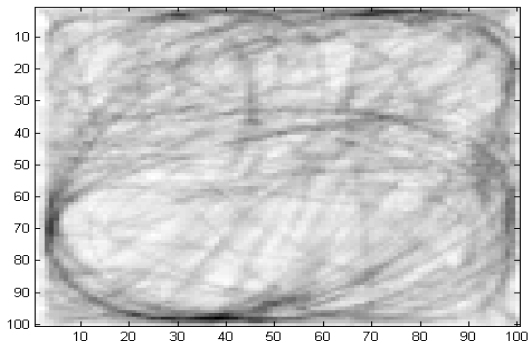
10 iterations



50 iterations



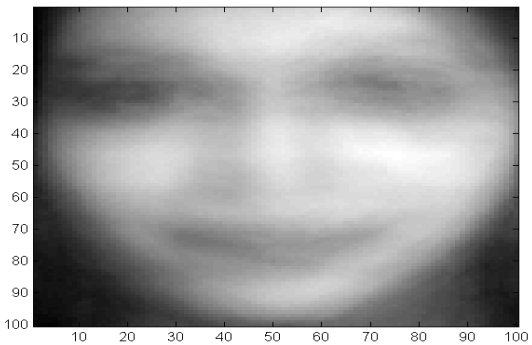
10 iterations



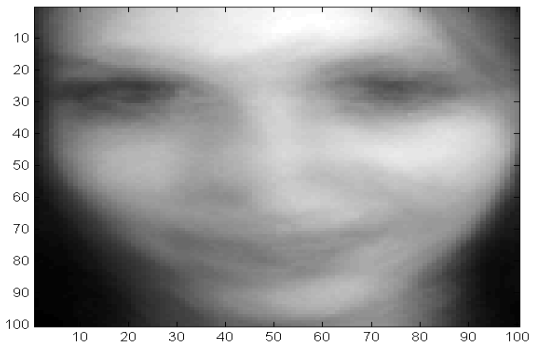
50 iterations

NMF - 6

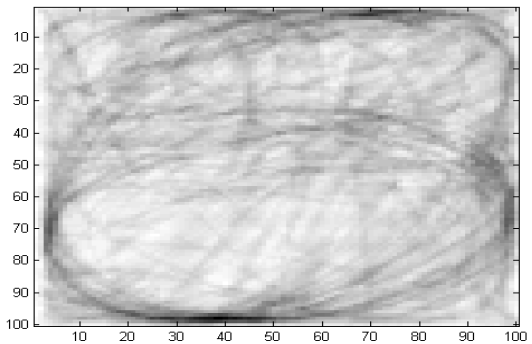
Results of **NMF** for faces and digits with (5)-(6) and the initialization: $w_{ij} = \frac{1}{n}$ and $h_j = W^+ a_j$.



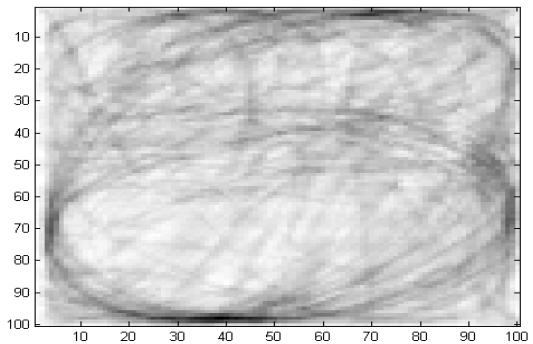
10 iterations



50 iterations



10 iterations



50 iterations

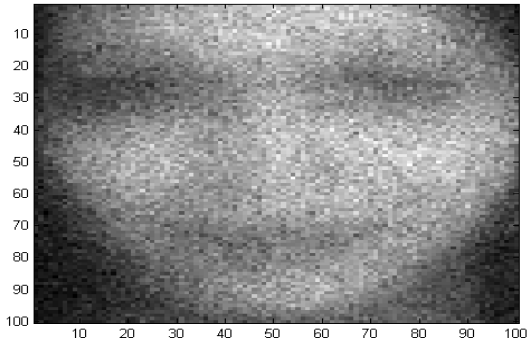
Multiplicative update algorithms for NMF

Algorithm A3 (see [Berry et al, 2007])

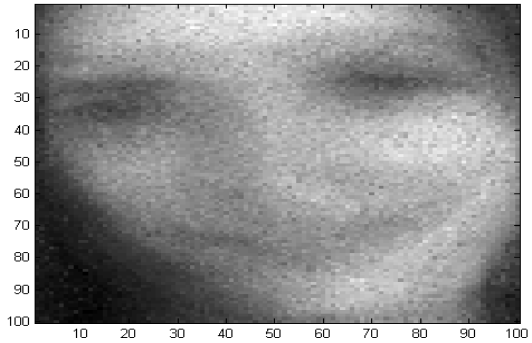
W=random(n,k)
H=random(k,m)
for i=1:maxiter
$H = H. * (W^T A)./(W^T W H + 10^{-9})$
$W = W. * (A H^T)./(W H H^T + 10^{-9})$
end

NMF - 8

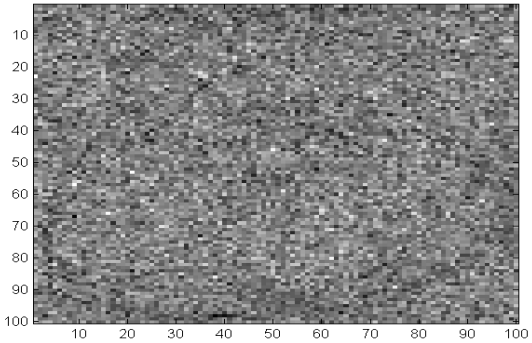
Results of multiplicative update algorithms for NMF for faces and digits



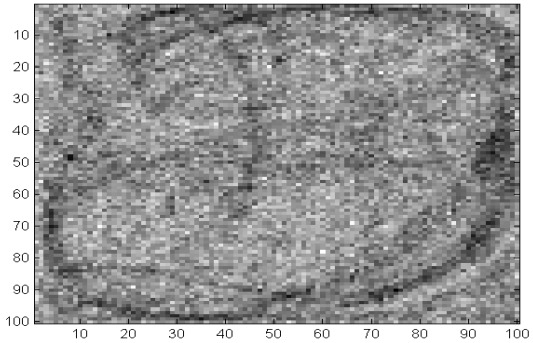
10 iterations



50 iterations



10 iterations



50 iterations

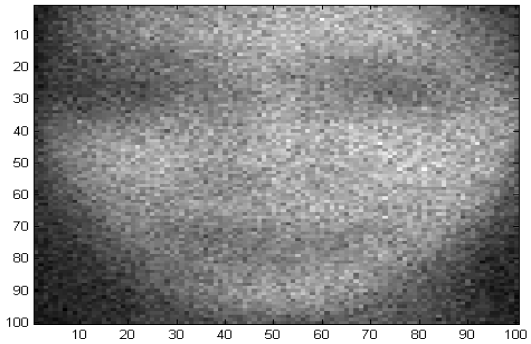
Constraint NMF (CNMF)

Algorithm A4 (see [Berry et al, 2007])

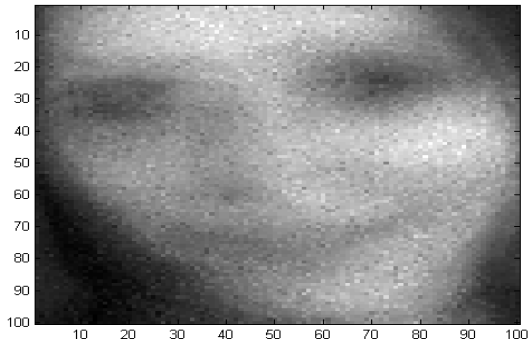
W=random(n,k)
H=random(k,m)
for i=1:maxiter
$H = H.*(W^T A)./(W^T W H + \beta * H + 10^{-9})$
$W = W.*(AH^T)./(WHH^T + \alpha * W + 10^{-9})$
end

NMF - 10

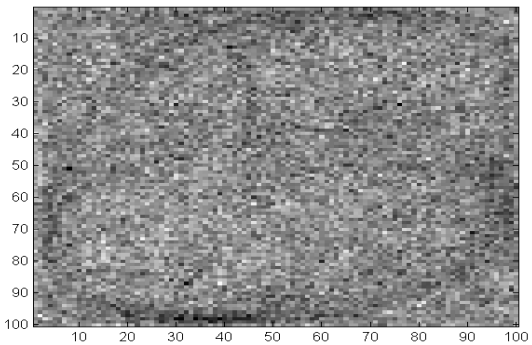
Results of **CNMF** for faces and digits



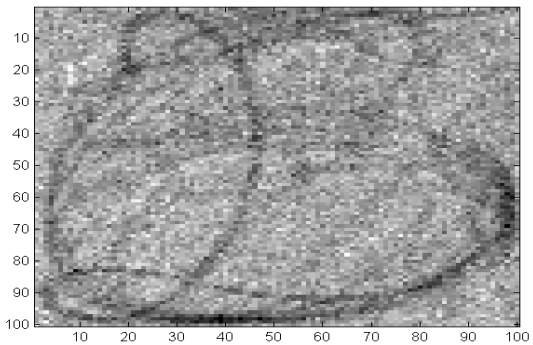
10 iterations



50 iterations



10 iterations



50 iterations

NMF - 11

The **NMF** problem can be written as

$$\min \|A - WH\|_F^2, \text{ where } W \geq 0, \ H \geq 0. \tag{7}$$

At each alternating step in **ACLS** (Alternating Constrained Least Squares) we must solve

$$\min_{h_j} \|a_j - Wh_j\|_2^2 + \lambda_H \|h_j\|_2^2, \text{ where } \lambda_H \geq 0, \ h_j \geq 0. \tag{8}$$

The **ACLS** algorithm is the following.

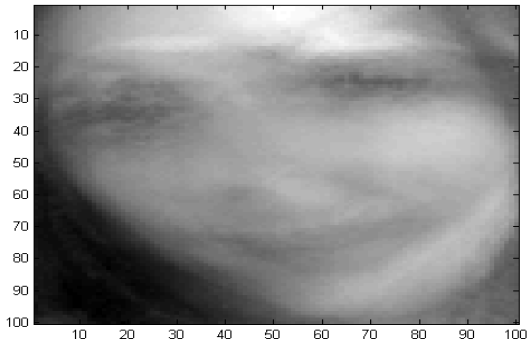
ACLS algorithm

Algorithm A5 (see [Langville et al, 2006])

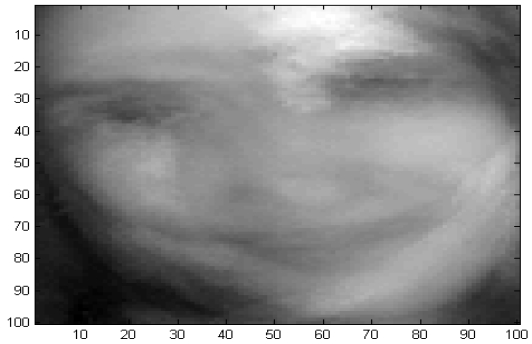
input λ_W, λ_H
W=random(n,k)
for i=1:maxiter
    Solve for H in matrix equation $(W^T W + \lambda_H I)H = W^T A$.
    Set all negative elements in H to 0.
    Solve for W in matrix equation $(HH^T + \lambda_W I)W^T = HA^T$.
    Set all negative elements in W to 0.
end

NMF - 13

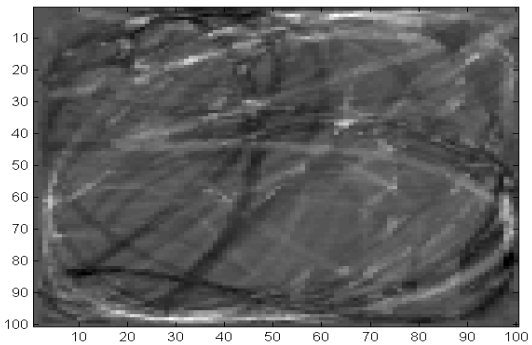
Results of **ACLS** reconstruction for faces and digits



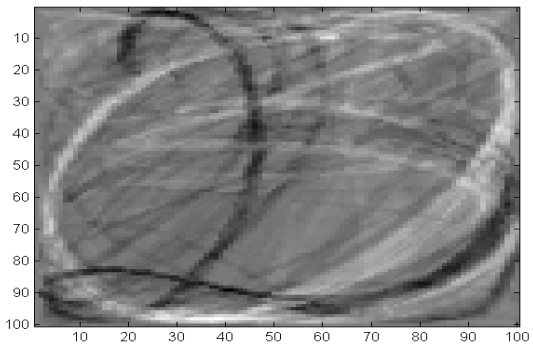
10 iterations



50 iterations



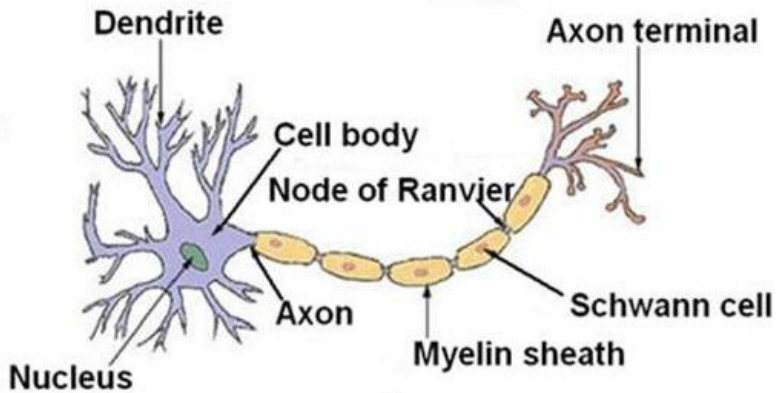
10 iterations



50 iterations

Neural Networks - 1

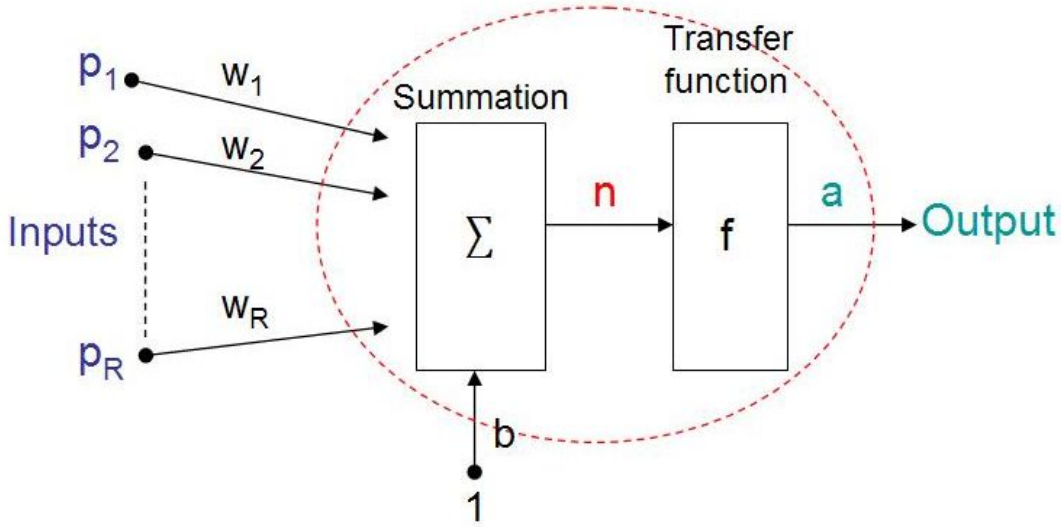
"A neural network is an *interconnected* assembly of simple processing elements, *units or nodes*, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the inter-unit connection strengths, or *weights*, obtained by a process of adaptation to, or *learning* from, a set of training patterns."



Structure of a typical neuron

Humans can recognize faces very easily and we want computers to do too ⇒ the idea of a computer which recognizes faces as human brain. NN have been widely used in pattern recognition applications and it has performed satisfactory in face recognition

Neural Networks - 2



Components of a simple neuron

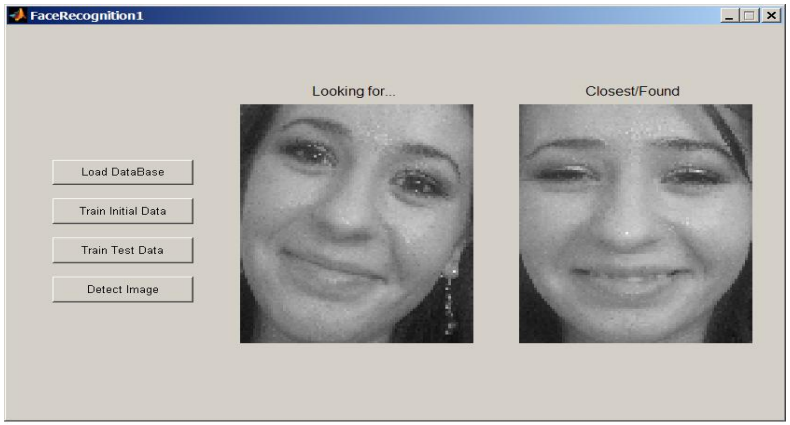
where $a = f(w_1 p_1 + w_2 p_2 + \ldots + w_R p_R + b)$.

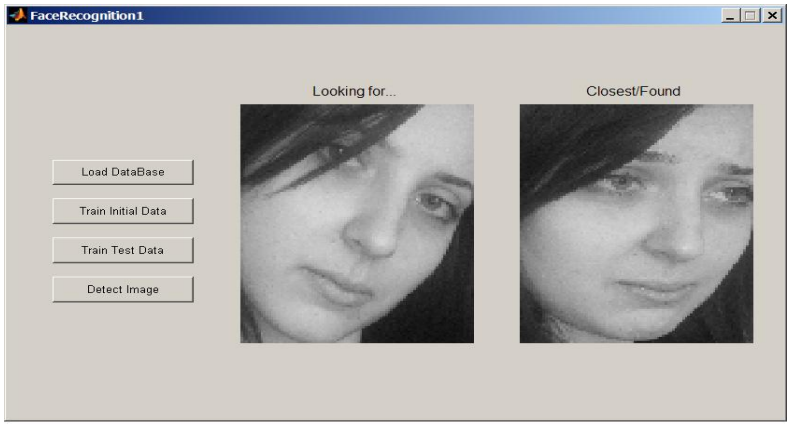
Neural Networks - 2

- Given a specific task to solve, and a class of functions F, *learning* means using a set of observations to find $f^* \in F$ which solves the task in some optimal sense.
- A neural network can be *trained* to produce a correct target vector when presented with corresponding input vector.
- Each traverse through all the training vectors is called an *epoch*.

Examples with NN - 1

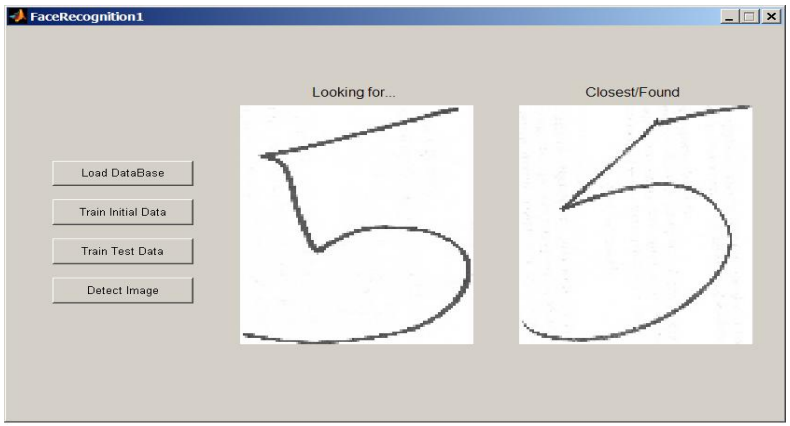
Results for faces (consistent case).

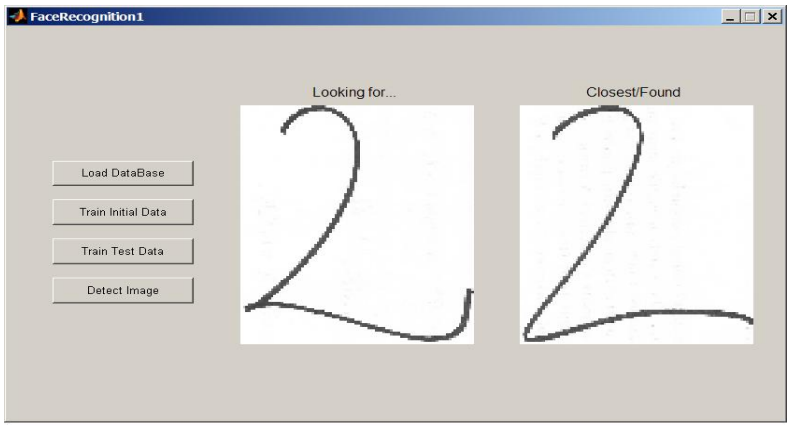




Examples with NN - 2

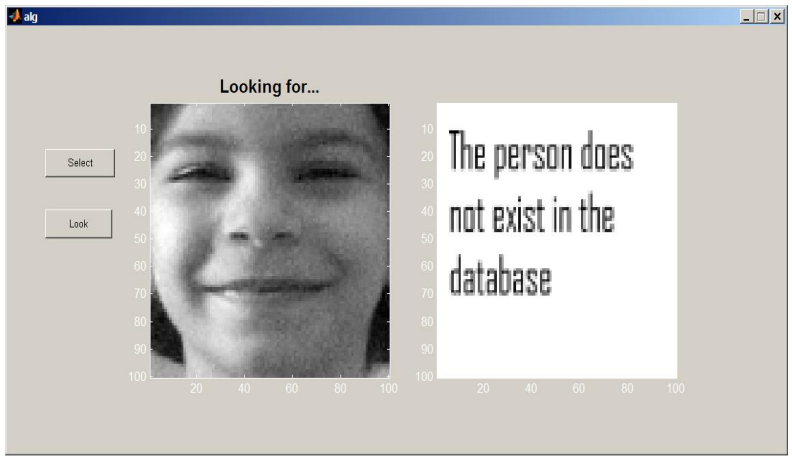
Results for digits (consistent case).





Examples with NN - 3

Results for faces, the inconsistent case.



Conclusions

SVD

- an optimality property; the truncated SVD produces the best rank-k approximation
- speedy and robust computation
- unique factorization; initialization does not affect SVD algorithms orthogonality; resulting basis vectors are orthogonal and allow conceptualization of original data as vectors in space

NMF

- sparsity and nonnegativity; the factorization maintains these properties of the original matrix
- reduction in storage; the factors are sparse, which also results in easier application to new data
- interpretability; the basis vectors naturally correspond to conceptual properties of the data

NN

- NN imitates the human brain
- most important is choosing the architecture of neural network (sequence of transfer functions)
- using artificial neurons in order to train the classifier

References

- Berry M. W., Browne M., Langville A. N., Pauca V. P. and Plemmons R. J., *Algorithms and applications for approximate nonegative matrix factorization*, Computational Statistics and Data Analysis, **52**, 2007, 155-173.
- Bjorck, A., *Numerical Methods For Least Squares Problems*, SIAM, Philadelphia, 1996.
- Boutsidis C. and Gallopoulos E., *SVD based initialization: A head start for nonnegative matrix factorization*, Pattern Recognition archive, **4**, 2008, 1350-1362.
- Elden, L., *Matrix Methods in Data Mining and Pattern Recognition*, SIAM, Philadelphia, 2007.
- Hoyer P. O., *Non-negative Matrix Factorization with Sparsness Constraints*, Journal of Machine Learning Research, **5**, 2004, 1457-1469.
- Golub, G. and Van Loan, C., *Matrix Computation, 3rd Edition*, The John Hopkins University Press, Baltimore 1996.
- Langville A. N., Meyer C. D., Albright R., Cox J. and Duling D., *Algorithms, Initializations, and Convergence for the Nonegative Matrix Factorization*, NCSU Technical Report Math 81706, 2006.

References

- Latha P., Ganesan L. and Annadurai S.,*Face Recognition using Neural Networks*, Signal Processing: An International Journal (SPIJ), **3**, 2009, 153-160.
- Lawrence S., Giles C. L., Tsoi A. C. and Back A. D. *Face Recognition: A Convolutional Neural Network Approach*, IEEE Transactions on Neural Networks, Special Issue on Neural Networks and Pattern Recognition, **8**, 1997, 98113.
- Lee D. D. and Seung H. S., *Algorithms Non-negative Matrix Factorization*, Advances in Neural Information Processing 13 (Proc. NIPS*2000). MIT Press, 2001.
- Turk, M. and Pentland, A., *Eigenfaces for Recognition*, Journal of Cognitive Neuroscience, **1**, 1991, 71-86.
- De Lathauwer L., B. de Moor and J. Vandewalle, *A multilinear singular value decomposition*, SIAM Journal Matrix Anal. Appl, **21**, 2000, 1252-1278.
- Zeb J., Javed M. Y. and Qayyum U., *Low resolution single neural network based face recognition*, International journal of Biomedical Sciences, **2**, 2007, 206-210.

Thank you!