Principal component analysis for image processing and object orientation

- Principal component analysis (PCA) is a machine learning technique which is widely used for data compression in image processing (data visualization) or in the determination of object orientation.
- PCA problem is closely related to the numerical linear algebra (NLA) problem of finding eigenvalues and eigenvectors for the covariance matrix.
- We will study application of different methods for solution of nonsymmetric eigenvalue problems (inverse iteration with shift, orthogonal iteration, QR iteration and QR-iteration with shift) for image compression and object rotation.

- The goal of this project is numerical studies of different methods for solution of nonsymmetric eigenvalue problems applied for PCA in image compression and object rotation using following numerical linear algebra (NLA) algorithms: inverse iteration with shift, orthogonal iteration, QR iteration and QR-iteration with shift.
- Note that all described above NLA algorithms can be found in the book L. Beilina, E. Karchevskii, M. Karchevskii, *Numerical Linear Algebra: Theory and Applications, Springer, 2017*, and all matlab-programs for solution of nonsymmetric eigenvalue problems (inverse iteration with shift, orthogonal iteration, QR iteration and QR-iteration with shift) are free for download from the GitHub link

  `https://github.com/springer-math/Numerical_Linear_Algebra_Theory_and_Applications`

  In the project we will study application of programs from this link for PCA in image compression and object rotation.

Project exercises

- Discover convergence of following algorithms for solution of nonsymmetric eigenvalue problems applied for PCA in image compression and object rotation:

  inverse iteration with shift, orthogonal iteration, QR iteration and QR-iteration with shift.

- Compare performance of all algorithms with respect to applicability, reliability, accuracy, and efficiency.

- Programs written in Matlab should demonstrate performance of every algorithm on the concreteexample of image compression or orientation.

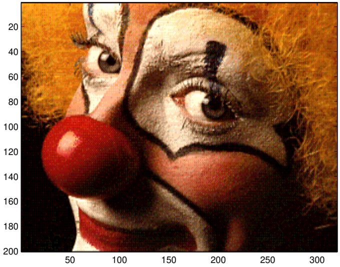
Example of application of linear systems: image compression using SVD

Definition SVD *Let A be an arbitrary m-by-n matrix with $m \geq n$. Then we can write $A = U\Sigma V^T$, where U is m-by-n and satisfies $U^T U = I$, V is n-by-n and satisfies $V^T V = I$, and $\Sigma = diag(\sigma_1, \ldots, \sigma_n)$, where $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. The columns $u_1, \ldots, u_n$ of U are called left singular vectors. The columns $v_1, \ldots, v_n$ of V are called right singular vectors. The σ_i are called singular values. (If $m < n$, the SVD is defined by considering A^T.)*

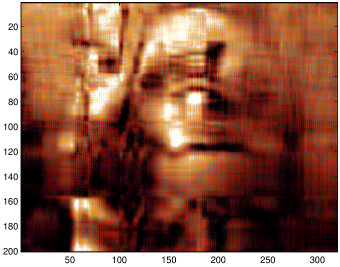
Theorem

Write $V = [\upsilon_1, \upsilon_2, \ldots, \upsilon_n]$ and $U = [u_1, u_2, \ldots, u_n]$, so $A = U\Sigma V^T = \sum_{i=1}^n \sigma_i u_i \upsilon_i^T$ (a sum of rank-1 matrices). Then a matrix of rank $k < n$ closest to A (measured with $||\cdot||_2$ is $A_k = \sum_{i=1}^k \sigma_i u_i \upsilon_i^T$ and $||A - A_k||_2 = \sigma_{k+1}$. We may also write $A_k = U\Sigma_k V^T$ where $\Sigma_k = diag(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$.

Example of application of linear systems: image compression using SVD



a) Original image



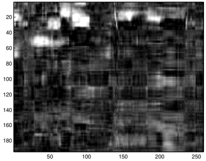
b) Rank k=20 approximation

Example of application of linear systems: image compression using SVD in Matlab

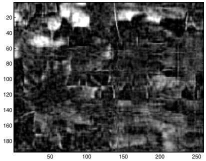
See path for other pictures:
/matlab-2012b/toolbox/matlab/demos
load clown.mat;
Size(X) $= m \times n = 320 \times 200$ pixels.
[U,S,V] = svd(X);
colormap(map);
k=20;
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
Now: size(U)$= m \times k$, size(V)$= n \times k$.

Example of application of linear systems: image compression using SVD in Matlab

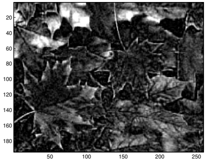
a) Original image



b) Rank k=10 approximation



c) Rank k=20 approximation



d) Rank k=50 approximation

Example of application of linear systems: image compression using SVD for arbitrary image

To get image on the previous slide, I took picture in jpg-format and loaded it in matlab like that:

A = imread('autumn.jpg');

You can not simply apply SVD to A: svd(A) Undefined function 'svd' for input arguments of type 'uint8'.

Apply type "double" to A: DA = double(A), and then perform

```
[U,S,V] = svd(DA);
colormap('gray');
k=20;
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
```

Now: size(U)= $m \times k$, size(V)= $n \times k$.

Project literature

The proposed project books:

- AI: Miroslav Kurbat, *An Introduction to Machine Learning*, Springer, 2017.
- AI: Christopher M. Bishop, *Pattern recognition and machine learning*, Springer, 2009.
- NLA problems:

  L. Beilina, E. Karchevskii, M. Karchevskii, *Numerical Linear Algebra: Theory and Applications, Springer, 2017*. Book is available at Cremona.

  Matlab and C++ programs for examples in this book are available for download from the course homepage: go to the link of the book and click to

  `GitHub Page with MATLAB Source Codes`

  on the bottom of this page.