PCA applied to image compression

High Performance Computing for Science and Engineering I

November 3, 2017

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Data set

Gray-scale image with $m \times n$ pixels

Represent the data by the matrix $A \in \mathbb{R}^{m \times n}$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Set of observations:

$$\boldsymbol{x}_k \in \mathbb{R}^m \quad k = 1, 2, \ldots, n$$

Algorithm

We want to maximize (under the constraint $\boldsymbol{u}^T\boldsymbol{u} = 1$):

$$I(\boldsymbol{u}) = \boldsymbol{u}^T C \boldsymbol{u} = \boldsymbol{u}^T \lambda \boldsymbol{u} = \lambda \quad \Rightarrow \quad C\boldsymbol{u} = \lambda \boldsymbol{u}$$

Steps:

1. Normalize data $\bar{A}$ (zero-mean and unit std.)
2. Build covariance matrix $C \in \mathbb{R}^{n \times n}$
3. Solve eigenvalue problem using LAPACK
4. Compress the image by representing the uncompressed data A in a subspace spanned by the basis $\boldsymbol{u}_k, k = 1, 2, \ldots, p$ where $p < n$.

LAPACK

You can solve an eigenvalue problem by using the `dsyev` routine from the LAPACK library.

http://www.netlib.org/lapack/explore-html/d2/d8a/group__double_s_yeigen_ga442c43fca5493590f8f26cf42fed4044.html#ga442c43fca5493590f8f26cf42fed4044

Use *workspace* for best performance:

- You can set `LWORK = -1`
- Call `dsyev` to determine optimal size of the workspace (eigenvalue problem is not solved in this case)
- Allocate the workspace and call `dsyev` a second time passing the pointer to the workspace array (workspaces are used in LAPACK for blocking)

Image compression

Image can be compressed by only taking into account $p < n$ (principal) components.

$$B = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_p] \in \mathbb{R}^{n \times p} \qquad (\lambda_1 > \lambda_2 > \cdots > \lambda_p)$$

Compute the p principal components $P \in \mathbb{R}^{m \times p}$

$$\boxed{P = \bar{A} B}$$

where the matrix $\bar{A}$ is the normalized image data.

Image decompression

To decompress the image (reconstruction), we simply invert the process. We compute

$$PB^T = \bar{A}BB^T = \bar{A}$$

The last step is then to denormalize $\bar{A}$ to get back to an approximation of A (lossy compression).