JSS30, Summer School, COM5: Machine learning in inverse and ill-posed problems

Larisa Beilina*

Department of Mathematical Sciences, Chalmers University of Technology and Gothenburg University, SE-42196 Gothenburg, Sweden

https://www.jyu.fi/en/research/

QR and SVD. Solution of rank-deficient problems. Principal Component Analysis (PCA).
Lecture 4

QR Decomposition

THEOREM QR decomposition. *Let A be m-by-n with $m \geq n$. Suppose that A has full column rank. Then there exist a unique m-by-n orthogonal matrix $Q(Q^T Q = I_n)$ and a unique n-by-n upper triangular matrix R with positive diagonals $r_{ii} > 0$ such that $A = QR$.*

Proof. Can be two proofs of this theorem: using the Gram-Schmidt orthogonalization process and using the Hauseholder reflections. The first proof: this theorem is a restatement of the Gram-Schmidt orthogonalization process [P. Halmos. Finite Dimensional Vector Spaces. Van Nostrand, New York, 1958]. If we apply Gram-Schmidt to the columns a_i of $A = [a_1, a_2, \ldots, a_n]$ from left to right, we get a sequence of **orthonormal vectors** (if they are orthogonal and unit vectors) q_1 through q_n spanning the same space: these orthogonal vectors are the columns of Q. Gram-Schmidt also computes coefficients $r_{ji} = q_j^T a_i$ expressing each column a_i as a linear combination of q_1 through q_i: $a_i = \sum_{j=1}^{i} r_{ji} q_j$. The r_{ji} are just the entries of R.

ALGORITHM *The classical Gram-Schmidt (CGS) and modified Gram-Schmidt (MGS) Algorithms for factoring* $A = QR$:

for $i = 1$ *to* n */* compute* i*th columns of* Q *and* R **/*

$q_i = a_i$

for $j = 1$ *to* $i - 1$ */* subtract component in* q_j *direction from* a_i **/*

$$\begin{cases} r_{ji} = q_j^T a_i & CGS \\ r_{ji} = q_j^T q_i & MGS \end{cases}$$

$q_i = q_i - r_{ji} q_j$

end for

$r_{ii} = \|q_i\|_2$

if $r_{ii} = 0$ */** a_i *is linearly dependent on* $a_1, \ldots, a_{i-1}$ **/*

quit

end if

$q_i = q_i / rii$

end for

If A has full column rank, r_{ii} will not be zero.

Notes:

- Unfortunately, CGS is numerically unstable in floating point arithmetic when the columns of A are nearly linearly dependent.
- MGS is more stable and will be used in algorithms later in this course but may still result in Q being far from orthogonal ($\|Q^T Q - I\|$ being far larger than ε) when A is ill-conditioned
- Literature on this subject:

  Å. Björck. Solution of Equations volume 1 of Handbook of Numerical Analysis, chapter Least Squares Methods. Elsevier/North Holland, Amsterdam, 1987.

  Å. Björck. Least squares methods. Mathematics Department Report, Linkoping University, 1991.

  Å. Björck. Numerical Methods for Least Squares Problems. SIAM, Philadelphia, PA, 1996.

  N. J. Higham. Accuracy and Stability of Numerical Algorithms. SIAM, Philadelphia, PA, 1996.

We will derive the formula for the x that minimizes $\|Ax - b\|_2$ using the decomposition $A = QR$ in three slightly different ways. First, we can always choose $m - n$ more **orthonormal vectors** $\tilde{Q}$ so that $[Q, \tilde{Q}]$ is a square orthogonal matrix and thus $\tilde{Q}^T Q = 0$ (for example, we can choose any $m - n$ more independent vectors $\tilde{X}$ that we want and then apply QR Algorithm to the n-by-n nonsingular matrix $[Q, \tilde{X}]$). Then

$$\begin{aligned}
\|Ax - b\|_2^2 &= \|[Q, \tilde{Q}]^T (Ax - b)\|_2^2 \\
&= \left\| \begin{bmatrix} Q^T \\ \tilde{Q}^T \end{bmatrix} (QRx - b) \right\|_2^2 \\
&= \left\| \begin{bmatrix} I^{n\times n} \\ O^{(m-n)\times n} \end{bmatrix} Rx - \begin{bmatrix} Q^T b \\ \tilde{Q}^T b \end{bmatrix} \right\|_2^2 \\
&= \left\| \begin{bmatrix} Rx - Q^T b \\ -\tilde{Q}^T b \end{bmatrix} \right\|_2^2 \\
&= \left\| Rx - Q^T b \right\|_2^2 + \|\tilde{Q}^T b\|_2^2 \geq \|\tilde{Q}^T b\|_2^2.
\end{aligned}$$

We can solve $Rx - Q^T b = 0$ for x, since A and R have the same rank, n, and so R is nonsingular. Then $x = R^{-1} Q^T b$, and the minimum value of $\|Ax - b\|_2$ is $\|\tilde{Q}^T b\|_2$.

Here is a second, slightly different derivation that does not use the matrix $\tilde{Q}$. Rewrite $Ax - b$ as

$$\begin{aligned} Ax - b &= QRx - b = QRx - (QQ^T + I - QQ^T)b \\ &= Q(Rx - Q^T b) - (I - QQ^T)b. \end{aligned}$$

Note that the vectors $Q(Rx - Q^T b)$ and $(I - QQ^T)b$ are orthogonal, because $(Q(Rx - Q^T b))^T((I - QQ^T)b) = (Rx - Q^T b)^T[Q^T(I - QQ^T)]b = (Rx - Q^T b)^T[0]b = 0$. Therefore, by the Pythagorean theorem,

$$\begin{aligned} \|Ax - b\|_2^2 &= \|Q(Rx - Q^T b)\|_2^2 + \|(I - QQ^T)b\|_2^2 \\ &= \|Rx - Q^T b\|_2^2 + \|(I - QQ^T)b\|_2^2. \end{aligned}$$

where we have used $\|Qy\|_2^2 = \|y\|_2^2$. This sum of squares is minimized when the first term is zero, i.e., $x = R^{-1}Q^T b$.

Finally, here is a third derivation that starts from the normal equations solution:

$$\begin{aligned} x &= (A^T A)^{-1} A^T b \\ &= (R^T Q^T Q R)^{-1} R^T Q^T b = (R^T R)^{-1} R^T Q^T b \\ &= R^{-1} R^{-T} R^T Q^T b = R^{-1} Q^T b. \end{aligned}$$

Singular values

The singular values, or s-numbers of a compact operator $T : X \to Y$ acting between Hilbert spaces X and Y, are the square roots of the eigenvalues of the nonnegative self-adjoint operator $T^*T : X \to X$ (where T^* denotes the adjoint of T).

$$\sigma(T) = \sqrt{\lambda(T^*T)}.$$

The singular values are nonnegative real numbers, usually listed in decreasing order $(s_1(T), s_2(T), ...)$. If T is self-adjoint, then the largest singular value $s1(T)$ is equal to the operator norm of T.
In the case of a normal matrix A (or $A^*A = AA^*$, when A is real then $A^TA = AA^T$), the spectral theorem can be applied to obtain unitary diagonalization of A as $A = U\Lambda U^*$. Therefore, $\sqrt{A^*A} = U|\Lambda|U^*$ and so the singular values are simply the absolute values of the eigenvalues.

Singular Value Decomposition

THEOREM 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 *Let $A = U\Sigma V^T$ be the SVD of the m-by-n matrix A, where $m \geq n$. (There are analogous results for $m < n$.)*

- 1. Suppose that A is symmetric, with eigenvalues λ_i and orthonormal eigenvectors u_i. i.e., $A = U\Lambda U^T$ is an eigendecomposition of A, with $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$, and $U = [u_1, \ldots, u_n]$, and $UU^T = I$. Then an SVD of A is $A = U\Sigma V^T$, where $\sigma_i = |\lambda_i|$ and $\upsilon_i = sign(\lambda_i)u_i$, where $sign(0) = 1$.
- 2. The eigenvalues of the symmetric matrix A^TA are σ_i^2. The right singular vectors υ_i are corresponding orthonormal eigenvectors.
- 3. The eigenvalues of the symmetric matrix AA^T are σ_i^2 and $m - n$ zeroes. The left singular vectors u_i are corresponding orthonormal eigenvectors for the eigenvalues σ_i^2. One can take any $m - n$ other orthogonal vectors as eigenvectors for the eigenvalue 0.

- 4. Let $H = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$, where A is square and $A = U\Sigma V^T$ is the SVD of A. Let $\Sigma = diag(\sigma_1, \ldots, \sigma_n)$, $U = [u_1, \ldots, u_n]$, and $V = [v_1, \ldots, v_n]$. Then the 2n eigenvalues of H are $\pm\sigma_i$, with corresponding unit eigenvectors $\frac{1}{\sqrt{2}} \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$.
- 5. If A has full rank, the solution of $\min_x \|Ax - b\|_2$ is $x = V\Sigma^{-1}U^T b$.
- 6. $\|A\|_2 = \sigma_1$. If A is square and nonsingular, then $\|A^{-1}\|_2^{-1} = \sigma_n$ and $\|A\|_2 \cdot \|A^{-1}\|_2 = \frac{\sigma_1}{\sigma_n}$.
- 7. Write $V = [v_1, v_2, \ldots, v_n]$ and $U = [u_1, u_2, \ldots, u_n]$, so $A = U\Sigma V^T = \sum_{i=1}^{n} \sigma_i u_i v_i^T$ (a sum of rank-1 matrices). Then a matrix of rank $k < n$ closest to A (measured with $\|\cdot\|_2$) is $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ and $\|A - A_k\|_2 = \sigma_{k+1}$. We may also write $A_k = U\Sigma_k V^T$ where $\Sigma_k = diag(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$.

Proof.

1. **Suppose that A is symmetric, with eigenvalues λ_i and orthonormal eigenvectors u_i. In other words $A = U\Lambda U^T$ is an eigendecomposition of A, with $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$, and $U = [u_1, \ldots, u_n]$, and $UU^T = I$. Then an SVD of A is $A = U\Sigma V^T$, where $\sigma_i = |\lambda_i|$ and $v_i = sign(\lambda_i)u_i$, where $sign(0) = 1$.**

This is true by the definition of the SVD.

2. **The eigenvalues of the symmetric matrix A^TA are σ_i^2. The right singular vectors υ_i are corresponding orthonormal eigenvectors.**
$A^TA = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T$. This is an eigendecomposition of A^TA, with the columns of V the eigenvectors and the diagonal entries of Σ^2 the eigenvalues.

3. **The eigenvalues of the symmetric matrix AA^T are σ_i^2 and $m - n$ zeroes. The left singular vectors u_i are corresponding orthonormal eigenvectors for the eigenvalues σ_i^2. One can take any $m - n$ other orthogonal vectors as eigenvectors for the eigenvalue 0.**
Choose an m-by-$(m - n)$ matrix $\tilde{U}$ so that $[U, \tilde{U}]$ is square and orthogonal. Then write

$$AA^T = U\Sigma V^T V\Sigma U^T = U\Sigma^2 U^T = \left[U, \tilde{U}\right] \cdot \begin{bmatrix} \Sigma^2 & 0 \\ 0 & 0 \end{bmatrix} \cdot \left[U, \tilde{U}\right]^T.$$

This is an eigendecomposition of AA^T.

4. **Let** $H = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$, **where** A **is square and** $A = U\Sigma V^T$ **is the SVD of** A**. Let** $\Sigma = diag(\sigma_1, \ldots, \sigma_n)$, $U = [u_1, \ldots, u_n]$, **and** $V = [v_1, \ldots, v_n]$**. Then the 2n eigenvalues of** H **are** $\pm\sigma_i$**, with corresponding unit eigenvectors** $\frac{1}{\sqrt{2}} \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$.

We substitute $A = U\Sigma V^T$ into H to get: $H = \begin{bmatrix} 0 & V\Sigma U^T \\ U\Sigma V^T & 0 \end{bmatrix}$

Choose orthogonal matrix G such that

$$G = \frac{1}{\sqrt{2}} \begin{bmatrix} V & V \\ U & -U \end{bmatrix}$$

It is orthogonal since $I = GG^T = \frac{1}{2} \begin{bmatrix} VV^T + VV^T & 0 \\ 0 & UU^T + UU^T \end{bmatrix}$

Then we observe that

$$G \begin{bmatrix} \Sigma & 0 \\ 0 & \Sigma \end{bmatrix} G^T = \begin{bmatrix} 0 & V\Sigma U^T \\ U\Sigma V^T & 0 \end{bmatrix} = H$$

Then using the spectral theorem we can conclude that the $2n$ eigenvalues of H are $\pm\sigma_i$, with corresponding eigenvectors $\frac{1}{\sqrt{2}} \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$.

5. **If A has full rank, the solution of $\min_x \|Ax - b\|_2$ is $x = V\Sigma^{-1}U^T b$.**

$\|Ax - b\|_2^2 = \|U\Sigma V^T x - b\|_2^2$. Since A has full rank, so does Σ, and thus Σ is invertible. Now let $[U, \tilde{U}]$ be square and orthogonal as above so

$$\begin{aligned}
\|U\Sigma V^T x - b\|_2^2 &= \left\| \begin{bmatrix} U^T \\ \tilde{U}^T \end{bmatrix} (U\Sigma V^T x - b) \right\|_2^2 \\
&= \left\| \begin{bmatrix} \Sigma V^T x - U^T b \\ -\tilde{U}^T b \end{bmatrix} \right\|_2^2 \\
&= \|\Sigma V^T x - U^T b\|_2^2 + \|\tilde{U}^T b\|_2^2.
\end{aligned}$$

This is minimized by making the first term zero, i.e., $x = V\Sigma^{-1}U^T b$.

6. $\|A\|_2 = \sigma_1$. **If A is square and nonsingular, then $\|A^{-1}\|_2^{-1} = \sigma_n$ and $\|A\|_2 \cdot \|A^{-1}\|_2 = \frac{\sigma_1}{\sigma_n}$.**
It is clear from its definition that the two-norm of a diagonal matrix is the largest absolute entry on its diagonal. Thus, by property of the norm, $\|A\|_2 = \|U^T A V\|_2 = \|U^T U \Sigma V^T V\|_2 = \|\Sigma\|_2 = \sigma_1$ and $\|A^{-1}\|_2 = \|V^T A^{-1} U\|_2 = \|\Sigma^{-1}\|_2 = \sigma_n^{-1}$.
Remark: $\|A^{-1}\|_2 = \|V^T A^{-1} U\|_2 = \|V^T (U \Sigma V^T)^{-1} U\|_2 = \|\Sigma^{-1}\|_2 = \sigma_n^{-1}$.

7. Write $V = [v_1, v_2, \ldots, v_n]$ and $U = [u_1, u_2, \ldots, u_n]$, so $A = U\Sigma V^T = \sum_{i=1}^{n} \sigma_i u_i v_i^T$ (a sum of rank-1 matrices). Then a matrix of rank $k < n$ closest to A (measured with $\|\cdot\|_2$) is $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ and $\|A - A_k\|_2 = \sigma_{k+1}$. We may also write $A_k = U\Sigma_k V^T$ where $\Sigma_k = diag(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$.

7. Write $V = [v_1, v_2, \ldots, v_n]$ and $U = [u_1, u_2, \ldots, u_n]$, so $A = U\Sigma V^T = \sum_{i=1}^{n} \sigma_i u_i v_i^T$ (a sum of rank-1 matrices). Then a matrix of rank $k < n$ closest to A (measured with $\|\cdot\|_2$) is $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ and $\|A - A_k\|_2 = \sigma_{k+1}$. We may also write $A_k = U\Sigma_k V^T$ where $\Sigma_k = diag(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$.
A_k has rank k by construction and

$$\|A - A_k\|_2 = \left\| \sum_{i=1}^{n} \sigma_i u_i v_i^T - \sum_{i=1}^{k} \sigma_i u_i v_i^T \right\|$$

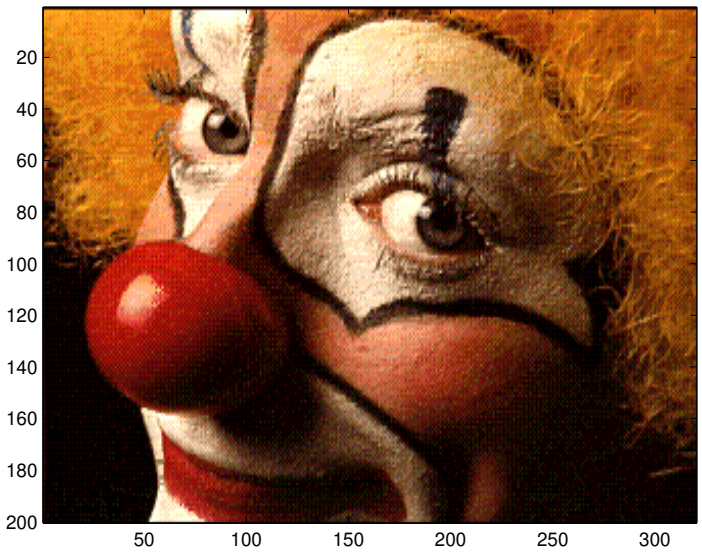
$$= \left\| \sum_{i=k+1}^{n} \sigma_i u_i v_i^T \right\| = \left\| U \begin{bmatrix} 0 & & & \\ & \sigma_{k+1} & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} V^T \right\|_2 = \sigma_{k+1}.$$

It remains to show that there is no closer rank k matrix to A. Let B be any rank k matrix, so its null space has dimension $n-k$. The space spanned by $\{v_1, ..., v_{k+1}\}$ has dimension $k+1$. Since the sum of their dimensions is $(n-k)+(k+1) > n$, these two spaces must overlap. Let h be a unit vector in their intersection. Then

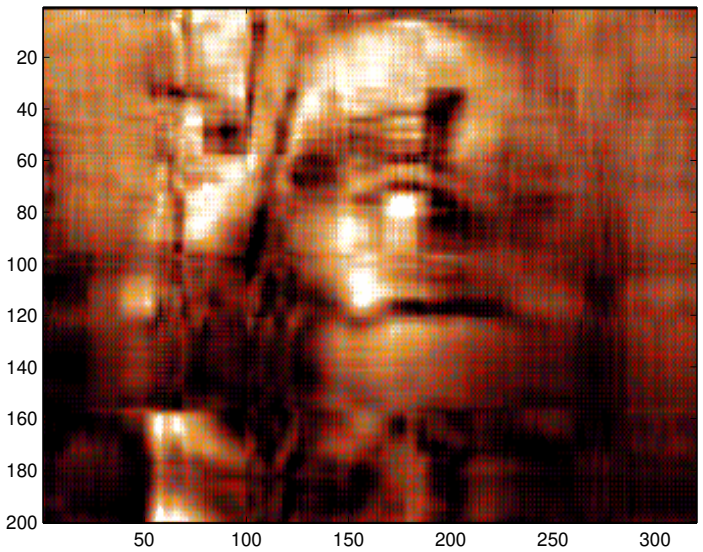
$$\begin{aligned} \|A-B\|_2^2 \geq \left\|(A-B)h\right\|_2^2 = \|Ah\|_2^2 = \left\|U\Sigma V^T h\right\|_2^2 \\ = \left\|\Sigma(V^T h)\right\|_2^2 \geq \sigma_{k+1}^2 \left\|V^T h\right\|_2^2 = \sigma_{k+1}^2. \end{aligned}$$

□

Example of application of linear systems: image compression using SVD



a) Original image



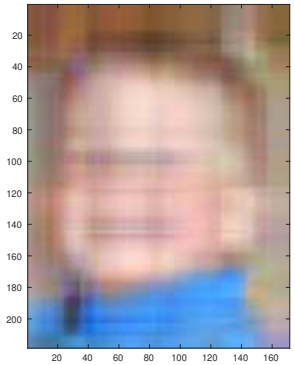
b) Rank k=20 approximation

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

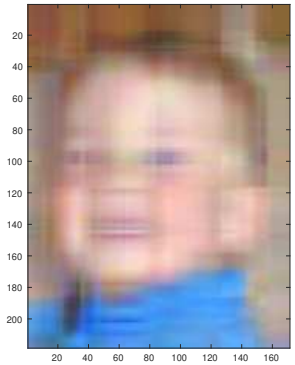
See path for other pictures:
/matlab-2012b/toolbox/matlab/demos
load clown.mat;
Size(X) = $m \times n = 320 \times 200$ pixels.
[U,S,V] = svd(X);
colormap(map);
k=20;
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
Now: size(U)= $m \times k$, size(V)= $n \times k$.

Image compression using SVD in Matlab

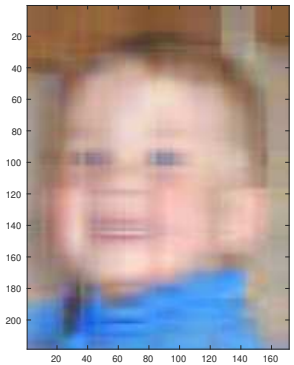
a) Original image



b) Rank k=4 approximation



b) Rank k=5 approximation



c) Rank k=6 approximation

d) Rank k=10 approximation

d) Rank k=15 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. You can also try to use following matlab code for your own pictures:

```
A = imread('Child.jpg'); // Real size of A: size(A) ans= 218 171 3
DDA=im2double(A); //convert from 'uint8' fromat to double format
figure(1); image(DDA);
//size of DDA will be (1:m,1:n,1:3)
[U1,S1,V1] = svd(DDA(:,:,1)); // we perform SVD for every 3 entries of DDA
[U2,S2,V2] = svd(DDA(:,:,2));
[U3,S3,V3] = svd(DDA(:,:,3));
k=15; //number of approximations: this number you can change
svd1 = U1(:,1:k)*S1(1:k,1:k)*V1(:,1:k)'; //compute new approximated matrices svd1, svd2, svd3
svd2 = U2(:,1:k)*S2(1:k,1:k)*V2(:,1:k)';
svd3 = U3(:,1:k)*S3(1:k,1:k)*V3(:,1:k)';
DDAnew = zeros(size(DDA));
DDAnew(:,:,1) = svd1; DDAnew(:,:,2) = svd2; DDAnew(:,:,3) = svd3;

figure(2); image(DDAnew);
```

Matrix norm. Induced norm

If vector norms on K_m and K_n are given (K is field of real or complex numbers), then one defines the corresponding induced norm or operator norm on the space of m-by-n matrices as the following maxima:

$$\begin{aligned} \|A\| &= \max\{\|Ax\| : x \in K^n \text{ with } \|x\| = 1\} \\ &= \max\left\{\frac{\|Ax\|}{\|x\|} : x \in K^n \text{ with } x \neq 0\right\}. \end{aligned}$$

If $m = n$ and one uses the same norm on the domain and the range, then the induced operator norm is a sub-multiplicative matrix norm.

Condition number of the square matrix

Condition number of the square matrix in any induced norm

$$k(A) = cond(A) = \|A\| \cdot \|A^{-1}\|$$

Example

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; A^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; A^TA - \lambda I = \begin{bmatrix} 1-\lambda & 0 \\ 0 & 1-\lambda \end{bmatrix} = 0;$$

$\lambda_1 = 1,\ \lambda_2 = 1;\ \|A\|_2 = \max \sqrt{\lambda(A^TA)} = \max(1,1) = 1.$
In this example,

$$A^{-1} = A; \|A^{-1}\|_2 = 1; k(A) = cond(A) = \|A\| \cdot \|A^{-1}\| = 1.$$

Perturbation Theory for the Least Squares Problem

When A is not square, we define its condition number with respect to the 2-norm to be

$$k_2(A) \equiv \sigma_{max}(A)/\sigma_{min}(A)$$

This reduces to the usual condition number when A is square. The next theorem justifies this definition.

THEOREM *Suppose that A is m-by-n with $m \geq n$ and has full rank. Suppose that x minimizes $\|Ax - b\|_2$. Let $r = Ax - b$ be the residual. Let $\tilde{x}$ minimize $\|(A + \delta A)\tilde{x} - (b + \delta b)\|_2$. Assume* $\epsilon \equiv \max(\frac{\|\delta A\|_2}{\|b\|_2}, \frac{\|\delta b\|_2}{\|b\|_2}) < \frac{1}{k_2(A)} = \frac{\sigma_{min}(A)}{\sigma_{max}(A)}$. *Then*

$$\frac{\|\tilde{x} - x\|}{\|x\|} \leq \epsilon \cdot \left\{ \frac{2 \cdot k_2(A)}{\cos\theta} + \tan\theta \cdot k_2^2(A) \right\} + O(\epsilon^2) \equiv \epsilon \cdot k_{LS} + O(\epsilon^2),$$

where $\sin\theta = \frac{\|r\|_2}{\|b\|_2}$. *In other words, θ is the angle between the vectors b and Ax and measures whether the residual norm $\|r\|_2$ is large (near $\|b\|$) or small (near 0). k_{LS} is the condition number for the least squares problem.*

Sketch of Proof. Expand $\tilde{x} = ((A + \delta A)^T(A + \delta A))^{-1}(A + \delta A)^T(b + \delta b)$ in powers of δA and δb. Then remove all non-linear terms, leave the linear terms for δA and δb. □

Rank-deficient Least Squares Problems

Proposition

Let A be m by n with $m \geq n$ and rank $A = r < n$. Then there is an $n - r$ dimensional set of vectors that minimize $\|Ax - b\|_2$.

Proof

Let $Az = 0$. Then of x minimizes $\|Ax - b\|_2$ then $x + z$ also minimizes $\|A(x + z) - b\|_2$.

This means that the least-squares solution is not unique.

Moore-Penrose pseudoinverse for a full rank A

Definition

Suppose that A is m by n with $m > n$ and has full rank with $A = QR = U\Sigma V^T$ being a QR and SVD decompositions of A, respectively. Then

$$A^+ \equiv (A^T A)^{-1} A^T = R^{-1} Q^T = V \Sigma^{-1} U^T$$

is called the Moore-Penrose pseudoinverse of A. If $m < n$ then $A^+ \equiv A^T (A A^T)^{-1}$.

The pseudoinverse of A allows write solution of the full-rank overdetermined least squares problem as $x = A^+ b$. If A is square and a full rank then this formula reduces to $x = A^{-1} b$. The A^+ is computed as *pinv(A)* in Matlab.

$$A^+ \equiv (A^T A)^{-1} A^T = ((QR)^T QR)^{-1} (QR)^T = (R^T Q^T QR)^{-1} (QR)^T$$
$$= (R^T R)^{-1} R^T Q^T = R^{-1} Q^T;$$

$$A^+ \equiv (A^T A)^{-1} A^T = ((U\Sigma V^T)^T U\Sigma V^T)^{-1} \cdot (U\Sigma V^T)^T$$
$$= (V\Sigma U^T U\Sigma V^T)^{-1} V\Sigma U^T = (V\Sigma^2 V^T)^{-1} V\Sigma U^T = V\Sigma^{-1} U^T$$

Moore-Penrose pseudoinverse for rank-deficient A

Definition

Suppose that A is m by n with $m > n$ and is rank-deficient with rank $r < n$. Let $A = U\Sigma V^T = U_1 \Sigma_1 V_1^T$ being a SVD decompositions of A such that

$$\text{A} = [\text{U}_1, U_2] \left[\begin{array}{c|c} \Sigma_1 & 0 \\ \hline 0 & 0 \end{array}\right] [V_1, V_2]^T = U_1 \Sigma_1 V_1^T$$

Here, $size(\Sigma_1) = r \times r$ and is nonsingular, U_1 and V_1 have r columns. Then

$$A^+ \equiv V_1 \Sigma_1^{-1} U_1^T$$

is called the Moore-Penrose pseudoinverse for rank-deficient A. The solution of the least-squares problem is always $x = A^+ b$, when A is rank-deficient then x has minimum norm.

The next proposition states that if A is nearly rank deficient then the solution x of $Ax = b$ will be ill-conditioned and very large.

Proposition

Let $\sigma_{min} > 0$ is the smallest singular value of the nearly rank deficient A. Then

- 1. If x minimizes $\|Ax - b\|_2$, then $\|x\|_2 \geq \frac{|u_n^T b|}{\sigma_{min}}$ where u_n is the last column of U in SVD decomposition of $A = U\Sigma V^T$.
- 2. Changing b to $b + \delta b$ can change x to $x + \delta x$ where $\|\delta x\|_2$ can be estimated as $\frac{\|\delta b\|_2}{\sigma_{min}}$, or the solution is very ill-conditioned.

Proof

1: We have that for the case of full-rank matrix A the solution of $Ax = b$ is given by $x = (U\Sigma V^T)^{-1} b = V\Sigma^{-1} U^T b$. The matrix $A^+ = V\Sigma^{-1} U^T$ is Moore-Penrose pseodoinverse of A. Thus, we can write also this solution as $x = V\Sigma^{-1} U^T b = A^+ b$.

Then taking norms from both sides of above expression we have:

$$\|x\|_2 = \|\Sigma^{-1}U^T b\|_2 \geq |(\Sigma^{-1}U^T b)_n| = \frac{|u_n^T b|}{\sigma_{min}}, \tag{1}$$

where $|(\Sigma^{-1}U^T b)_n|$ is the n-th column of this product.
2. We apply now (1) for $\|x + \delta x\|$ instead of $\|x\|$ to get:

$$\begin{aligned} \|x + \delta x\|_2 &= \|\Sigma^{-1}U^T(b + \delta b)\|_2 \geq |(\Sigma^{-1}U^T(b + \delta b))_n| \\ &= \frac{|u_n^T(b + \delta b)|}{\sigma_{min}} = \frac{|u_n^T b + u_n^T \delta b|}{\sigma_{min}}. \end{aligned} \tag{2}$$

We observe that $\frac{|u_n^T b|}{\sigma_{min}} + \frac{|u_n^T \delta b|}{\sigma_{min}} \leq \|x + \delta x\|_2 \leq \|x\|_2 + \|\delta x\|_2$.
Choosing δb parallel to u_n and applying again (1) for estimation of $\|x\|_2$ we have

$$\|\delta x\|_2 \geq \frac{\|\delta b\|_2}{\sigma_{min}}. \tag{3}$$

In the next proposition we prove that the minimum norm solution x is unique and may be well-conditioned if the smallest nonzero singular value is not too small.

Proposition

When A is exactly singular, then x that minimize $\|Ax - b\|_2$ can be characterized as follows. Let $A = U\Sigma V^T$ have rank $r < n$. Write svd of A as

$$\text{A} = [\text{U}_1, U_2] \left[\begin{array}{c|c} \Sigma_1 & 0 \\ \hline 0 & 0 \end{array}\right] [V_1, V_2]^T = U_1 \Sigma_1 V_1^T$$

Here, $size(\Sigma_1) = r \times r$ and is nonsingular, U_1 and V_1 have r columns. Let $\sigma = \sigma_{min}(\Sigma_1)$. *Then*

- 1. All solutions x can be written as $x = V_1 \Sigma_1^{-1} U_1^T + V_2 z$
- 2. The solution x has minimal norm $\|x\|_2$ when $z = 0$. Then $x = V_1 \Sigma_1^{-1} U_1^T$ and $\|x\|_2 \leq \frac{\|b\|_2}{\sigma}$.
- 3. Changing b to $b + \delta b$ can change x as $\frac{\|\delta b\|_2}{\sigma}$.

Proof

We choose the matrix $\tilde{U}$ such that $[U, \tilde{U}] = [U_1, U_2, \tilde{U}]$ be an $m \times m$ orthogonal matrix. Then

$$\begin{aligned}
\|Ax - b\|_2^2 &= \|[U_1, U_2, \tilde{U}]^T (Ax - b)\|_2^2 \\
&= \left\| \begin{bmatrix} U_1^T \\ U_2^T \\ \tilde{U}^T \end{bmatrix} (U_1 \Sigma_1 V_1^T x - b) \right\|_2^2 \\
&= \|[I^{r\times r}, O^{m\times(n-r)}, 0^{m\times m-n}]^T (\Sigma_1 V_1^T x - [U_1, U_2, \tilde{U}]^T \cdot b)\|_2^2 \\
&= \|[\Sigma_1 V_1^T x - U_1^T b; -U_2^T b; -\tilde{U}^T b]^T\|_2^2 \\
&= \|\Sigma_1 V_1^T x - U_1^T b\|_2^2 + \|U_2^T b\|_2^2 + \|\tilde{U}^T b\|_2^2
\end{aligned}$$

1. Then $\|Ax - b\|_2$ is minimized when $\Sigma_1 V_1^T x - U_1^T b = 0$. We can also write that the vector $x = (\Sigma_1 V_1^T)^{-1} U_1^T b + V_2 z$ or $x = V_1 \Sigma_1^{-1} U_1^T b + V_2 z$ is also solution of this minimization problem, because $V_1^T V_2 z = 0$ since columns of V_1 and V_2 are orthogonal.

2. Since columns of V_1 and V_2 are orthogonal, then by Pythagorean theorem we have that $\|x\|_2^2 = \|V_1\Sigma_1^{-1}U_1^T b\|^2 + \|V_2 z\|^2$ which is minimized for $z = 0$.
3. Changing b to δb in the expression above we have:

$$\|V_1\Sigma_1^{-1}U_1^T\delta b\|_2 \leq \|V_1\Sigma_1^{-1}U_1^T\|_2 \cdot \|\delta b\|_2 = \|\Sigma_1^{-1}\|_2 \cdot \|\delta b\|_2 = \frac{\|\delta b\|_2}{\sigma}, \qquad (4)$$

where σ is smallest nonzero singular value of A. In this proof we used properties of the norm: $\|QAZ\|_2 = \|A\|_2$ if Q, Z are orthogonal.

How to solve rank-deficient least squares problems using QR decomposition with pivoting

QR decomposition with pivoting is cheaper but can be less accurate than SVD technique for solution of rank-deficient least squares problems.
If A has a rank $r < n$ with independent r columns QR decomposition can look like that

$$A = QR = Q \cdot \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

(5)

with nonzingular R_{11} is of the size $r \times r$ and R_{12} is of the size $r \times (n - r)$.
We can try to get

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \\ 0 & 0 \end{bmatrix}, \tag{6}$$

where elements of R_{22} are very small and are of the order $\varepsilon \|A\|_2$.

If we set $R_{22} = 0$ and choose $[Q, \tilde{Q}]$ which is square and orthogonal then we will minimize

$$\begin{aligned} \|Ax - b\|_2^2 &= \left\| \begin{bmatrix} Q^T \\ \tilde{Q}^T \end{bmatrix} (Ax - b) \right\|_2^2 \\ &= \left\| \begin{bmatrix} Q^T \\ \tilde{Q}^T \end{bmatrix} (QRx - b) \right\|_2^2 \\ &= \left\| \begin{bmatrix} \mathrm{Rx} - Q^T b \\ -\tilde{Q}^T \mathrm{b} \end{bmatrix} \right\|_2^2 \\ &= \|Rx - Q^T b\|_2^2 + \|\tilde{Q}^T b\|_2^2. \end{aligned} \tag{7}$$

Here we again used properties of the norm: $\|QAZ\|_2 = \|A\|_2$ if Q, Z are orthogonal.

Let us now decompose $Q = [Q_1, Q_2]$ with $x = [x_1, x_2]^T$ and

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} \tag{8}$$

such that equation (7) becomes

$$\begin{aligned} \|Ax - b\|_2^2 &= \left\| \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} Q_1^T b \\ Q_2^T b \end{bmatrix} \right\|_2^2 + \|\tilde{Q}^T b\|_2^2 \\ &= \|R_{11}x_1 + R_{12}x_2 - Q_1^T b\|_2^2 + \|Q_2^T b\|_2^2 + \|\tilde{Q}^T b\|_2^2. \end{aligned} \tag{9}$$

We take now derivative with respect to x to get $(\|Ax - b\|_2^2)'_x = 0$. We see that minimum is achieved when

$$x = \begin{bmatrix} R_{11}^{-1}(Q_1^T b - R_{12}x_2) \\ x_2 \end{bmatrix} \tag{10}$$

for any vector x_2. If R_{11} is well-conditioned and $R_{11}^{-1}R_{12}$ is small than the choice $x_2 = 0$ will be good one.

The described method is not reliable for all rank-deficient least squares problems. This is because R can be nearly rank deficient for the case when no R_{22} is small. In this case can help QR decomposition with column pivoting: we factorize $AP = QR$ with permutation matrix P. To compute this permutation we do as follows:
1. In all columns from 1 to n at step i we select from the unfinished decomposition of part A in columns i to n and rows i to m the column with largest norm and exchange it with i-th column.
2. Then compute usual Householder transformation to zero out column i in entries $i + 1$ to m.
Recent research is devoted to more advanced algorithms called rank-revealing QR algorithms which detects rank more faster and more efficient.
C. Bischof, Incremental condition estimation, *SIAM J.Matrix Anal.Appl.*, 11:312-322, 1990.
T.Chan, Rank revealing QR factorizations, *Linear Algebra Applications*, 88/89:67-82, 1987.

Outline

- SVD for image compression
- Principal component analysis (PCA) formulation (mean, covariance matrix, computation of eigenvalues and eigenvectors of covariance matrix)
- Principal component analysis (PCA) to find patterns, for data compression and for image 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 PCA for image compression and object rotation.

Literature

The proposed books in machine learning:

- Christopher M. Bishop, *Pattern recognition and machine learning*, Springer, 2009.
- Ian Goodfellow, Yoshua Bengio and Aaron Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`
- Miroslav Kurbat, *An Introduction to Machine Learning*, Springer, 2017.

PCA formulation

Let $\{x_n\}, n = 1, ..., N$ is a data set of observations, where x_n is a variable of the dimension d. The goal in PCA is to project the data onto a space which has dimension $M < d$ maximizing the variance of the projected data. Let $M = 1$ and thus, we will consider projection onto one-dimensional space. Let the vector u_1 is the direction of this space such that it is a unit vector and $u_1^T u_1 = 1$. Then every point x_n is projected onto a scalar value $u_1^T x_n$.

PCA formulation

The sample set mean $\bar{x}$ is defined as

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n,$$

and the variance of the projected data $u_1^T \bar{x}$ is

$$\frac{1}{N}\sum_{n=1}^{N}(u_1^T x_n - u_1^T \bar{x})^2 = u_1^T S u_1,$$

where S is the data covariance matrix

$$S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T.$$

PCA formulation

The next step is maximize the projected variance $u_1^T S u_1$ with respect to u_1 using constrained minimization with the term $u_1^T u_1 = 1$. To enforce this constraint, we use Lagrange multiplier λ to minimize

$$L(u_1) = u_1^T S u_1 + \lambda_1(1 - u_1^T u_1)$$

Now we minimize $L'(u_1)(\bar{u}_1) = 0$ to get

$$0 = L'(u_1)(\bar{u}_1) = (Su_1 - \lambda_1 u_1)(\bar{u}_1),$$

what means that

$$Su_1 = \lambda_1 u_1 \tag{11}$$

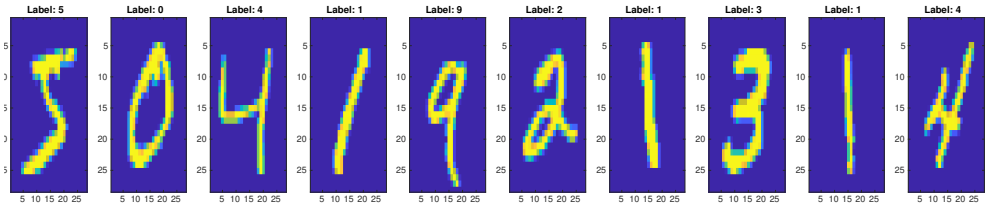
PCA formulation

From the equation (11) follows that (λ_1, u_1) is an eigenpair of S. We observe that

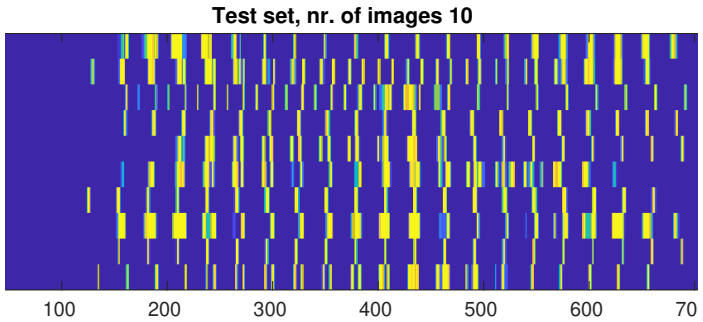
$$u_1^T S u_1 = \lambda_1$$

what means that variance is maximum when we set u_1 to the eigenvector for the largest eigenvalue λ_1 which is called the first principal component. One can obtain all other components in the same way. If we consider M-dimensional projection space then the optimal linear projection for which the variance of the projected data will have maximum, will consists of M eigenvectors $u_1, ..., u_M$ of the covariance matrix S corresponding to the M largest eigenvalues $\lambda_1, ..., \lambda_M$.

PCA formulation



a) Dataset in *mnist_test_10.csv*



$$\dim(X) = 10 \times 784$$

Let now analyze PCA in a common case. Let X be a matrix (for example, it can be training set data matrix of the size $m \times n$, where m is the number of samples and n is the variable in the matrix X. An example of the variable can be some image. Assume, that the data X has been centered such that it has zero mean for each column. Then the covariance matrix can be computed for $N = n - 1$ as

$$C = \frac{1}{N} X X^T \tag{12}$$

such that C is symmetric and dim(C) = $n \times n$.

PCA formulation

The eigenvalue problem for C is:

$$Cu = \lambda u. \tag{13}$$

We observe that

$$x = \sum_{i=1}^{n} \hat{x}_i u_i \tag{14}$$

and approximated vector x can be computed as

$$x \approx \sum_{i=1}^{k} \hat{x}_i u_i \tag{15}$$

Eigenvalues of C can be found using SVD of $X = U\Sigma V^T$. Taking now $X = X^T$ we can write the covariance matrix as

$$C = \frac{1}{N} X^T X = \frac{1}{N} (U\Sigma V^T)^T U\Sigma V^T = \frac{1}{N} V\Sigma U^T U\Sigma V^T = \frac{1}{N} V\Sigma^2 V^T. \tag{16}$$

The covariance matrix is

$$C = \frac{1}{N} X^T X = \frac{1}{N} V \Sigma^2 V^T. \tag{17}$$

Using above equation we observe that

$$\begin{aligned} X^T X &= V \Sigma^2 V^T, \\ (X^T X) V &= V \Sigma^2 V^T V = V \Sigma^2. \end{aligned} \tag{18}$$

We observe also that $XV = (U \Sigma V^T) V = U \Sigma$. Thus from the above equation we see that the right singular vectors V of $X = U \Sigma V^T$ are equivalent to the eigenvectors of $X^T X$. But the singular values σ of X are equal to $\sqrt{\lambda}$ of $X^T X$ (since $\sigma^2 = \lambda$).

We define the score matrix T as

$$T = XV = (U\Sigma V^T)V = U\Sigma. \tag{19}$$

We observe that every column of T is given by one of the left singular vectors of X multiplied by the corresponding singular value.
Next, a truncated $n \times k$ score matrix T_k can be obtained by considering only the first k largest singular values and their singular vectors:

$$T_k = U_k\Sigma_k = XV_k. \tag{20}$$

Here, $T_k = U_k\Sigma_k = XV_k$ because of (19). To reduce the dimensionality of the data to k, we select the k first columns of V. Then the matrix XV_k is of the size $m \times k$ with k principal components.
In order to perform digit recognition, we minimize the following functional for every image i, j in the train and test sets, respectively:

$$\min \|T_k(test(i)) - T_k(train(j))\|_2^2 \tag{21}$$

Here, $T_k(test)$ and $T_k(train)$ are the truncated score matrices for the test and train sets, respectively.

PCA: some notions of statistics

Variance is measure of the spread of data in dataspace which is defined as

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$$

where mean $\bar{X}$ is defined as

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}.$$

Standard deviation is defined as

$$s = \sqrt{var(X)} = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}}$$

PCA: some notions of statistics

Let

X	$(X - \bar{X})$	$(X - \bar{X})^2$
1	-2	4
3	0	0
0	-3	9
5	2	4
6	3	9
Total		26
Divided by $(n-1)$		6.5
Deviation, s		2.5495

Table: *Calculation of variance* $var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$. *Here, mean* $\bar{X} = 15/5 = 3$.

PCA: some notions of statistics

Covariance is measure of the spread of data in dataspace which is defined as

$$cov(X, Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$

Let

X	Y	$(X - \bar{X})$	$(Y - \bar{Y})$	$(X - \bar{X})(Y - \bar{Y})$
1	9	-2	5.4	-10.8
3	1	0	-2.6	0
0	3	-3	-0.6	1.8
5	5	2	1.4	2.8
6	0	3	-3.6	-10.8
Total				-17
Divided by $(n-1)$				-4.25

Table: *Calculation of covariance. Here,* $\bar{X} = 3$, $\bar{Y} = 3.6$.

PCA: some notions of statistics

Below is definition of the covariance matrix for a set of data which have dimension n:

$$C = \begin{bmatrix} cov(Dim_1, Dim_1) & ... & cov(Dim_1, Dim_n) \\ cov(Dim_2, Dim_1) & ... & cov(DIm_2, Dim_n) \\ ... & ... & ... \\ cov(Dim_n, Dim_1) & ... & cov(Dim_n, Dim_n) \end{bmatrix}$$

Here, $dim(C) = n \times n$.
Covariance is always measured between 2 dimensions. For three-dimensional data sets x,y,z one can compute covariance matrix

$$C = \begin{bmatrix} cov(x, x) & cov(x, y) & cov(x, z) \\ cov(y, x) & cov(y, y) & cov(y, z) \\ cov(z, x) & cov(z, y) & cov(z, z) \end{bmatrix}$$

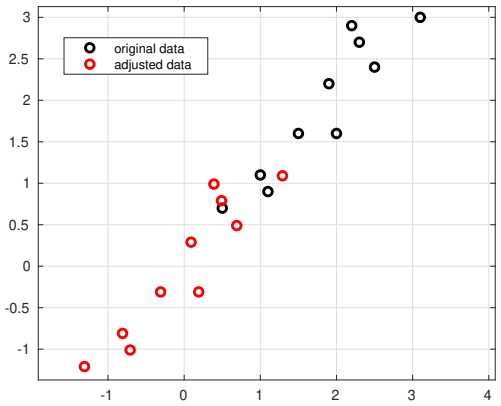
PCA

- PCA can be thought as a way to find patterns in data in order to highlight similarity or difference of data. Thus, it is a powerful tool to analyze data.
- The main advantage of PCA is as soon as patterns in data is found, one can compress the data by reducing number of dimensions without much loss of information. This technique is used in image compression.

PCA: example

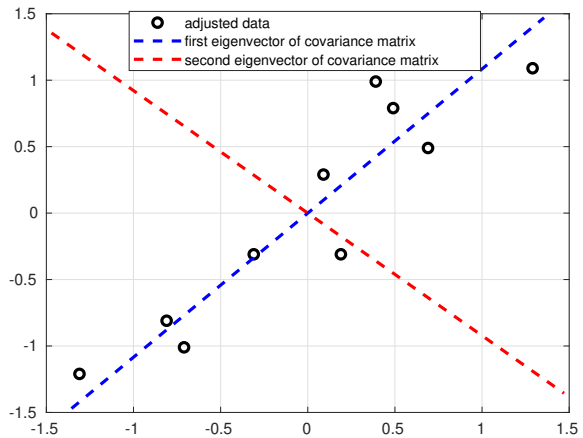
Let us analyze how works PCA on two datasets using Matlab program ExamplePCA.m.

```
  %first create our 2D data
x=[2.5,0.5,2.2,1.9,3.1,2.3,2,1,1.5,1.1];
y=[2.4,0.7,2.9,2.2,3.0,2.7,1.6,1.1,1.6,0.9];
%  compute mean for x
[ax,bx] =size(x);
mean_x= sum(x)/bx;
%  compute mean for y
[ay,by] =size(y);
mean_y = sum(y)/by;
%  compute  adjusted data
adjust_x = x - mean_x;
adjust_y = y - mean_y;
plot(x,y,'o','LineWidth',2,'MarkerEdgeColor','k');  grid('on'); hold on;
 plot(adjust_x,adjust_y,'o','LineWidth',2,'MarkerEdgeColor','r');
legend('original data', 'adjusted data');
```



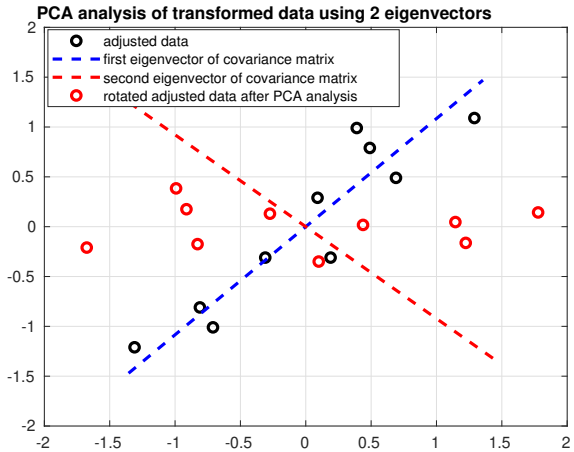
PCA: example

```
  %compute covariance matrix
  C = cov(adjust_x,adjust_y);
% compute eigenvalues and eigenvectors ov the covariance matrix
[eigvec, eigval] = eig(C);
[U,S,V] = svd(C);
% plotting using svd of the covariance matrix ********************************
% first eigenvector via svd: this is the principle component of the data set
% since it's corresponds to the largest eigenvalue.
% We can call it also as feature vector.
u1x=[2*U(1,1),-2*U(1,1)];
u1y=[2*U(2,1),-2*U(2,1)];
plot(u1x,u1y,'-- b', 'LineWidth',2);
% plotting the second eigenvalue vector
u2x=[2*U(2,1),-2*U(2,1)];
u2y=[2*U(2,2),-2*U(2,2)];
plot(u2x,u2y,'-- r', 'LineWidth',2);
```



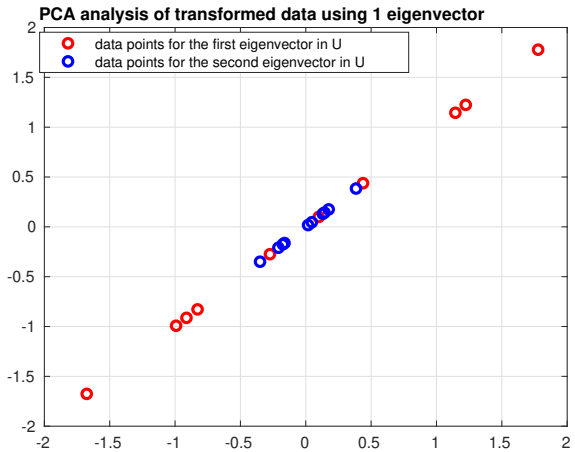
PCA: example

```
  % Now form final rotated data
Data =[adjust_x; adjust_y];
PCA = U*Data;
plot(PCA(1,:),PCA(2,:),'o','LineWidth',2, 'MarkerEdgeColor','r');
axis([-2,2,-2,2])
legend('adjusted data','first eigenvector of covariance matrix','second eigenvector of covariance matrix','
title('PCA analysis of transformed data using 2 eigenvectors')
```



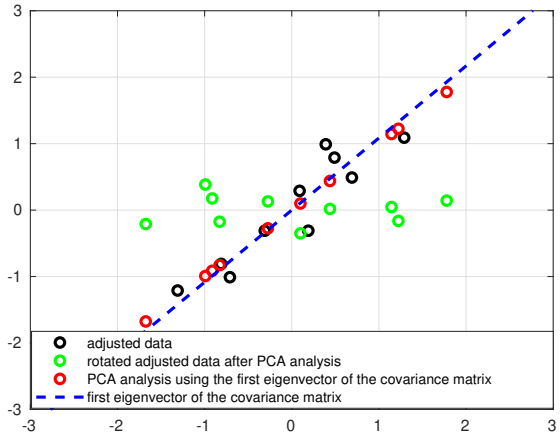
PCA: example

```
% now we plot out vector with the largest eigenvalue
PCAnew1 = U(1,:)*Data;
plot(PCAnew1,PCAnew1,'o','LineWidth',2, 'MarkerEdgeColor','r');
hold on
PCAnew2 = U(2,:)*Data;
plot(PCAnew2,PCAnew2,'o','LineWidth',2, 'MarkerEdgeColor','b');
  grid('on');
legend('data points for the first eigenvector in U','data points for the second eigenvector in U')
title('PCA analysis of transformed data using 1 eigenvector ')
```



PCA: example

```
    plot(adjust_x,adjust_y,'o','LineWidth',2,'MarkerEdgeColor','k');
     grid('on');
hold on
plot(PCA(1,:),PCA(2,:),'o','LineWidth',2, 'MarkerEdgeColor','g');
plot(PCAnew1,PCAnew1,'o','LineWidth',2, 'MarkerEdgeColor','r');
u1x=[5*U(1,1),-5*U(1,1)];
u1y=[5*U(2,1),-5*U(2,1)];
plot(u1x,u1y,'-- b', 'LineWidth',2);
axis([-3,3,-3,3])
legend('adjusted data','rotated adjusted data after PCA analysis',...
'PCA analysis using the first eigenvector of the covariance matrix',...
'first eigenvector of the covariance matrix');
```



Project “PCA for image recognition”

Use PCA to find patterns (recognize handwritten numbers) in MNIST Dataset of Handwitten Digits which can be downloaded from the course homepage.

- Use Matlab programs

  `loadmnist_matlab.m`
  `import_mnist.m`

  on the course homepage to download MNIST Datasets

  `mnist_test_10.csv`
  `mnist_train.csv`

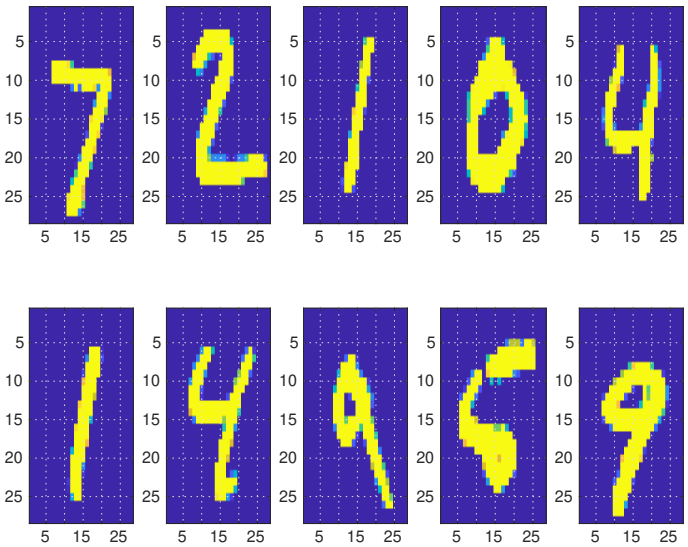
- Perform PCA analysis for the following problem: given an image from the dataset

  `mnist_test_10.csv`

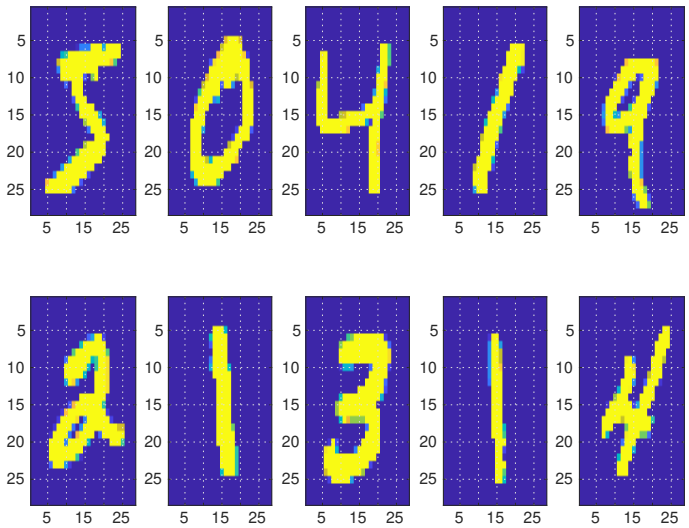
  check if there exists the same or similar image in the train dataset

  `mnist_train.csv`

MNIST datasets

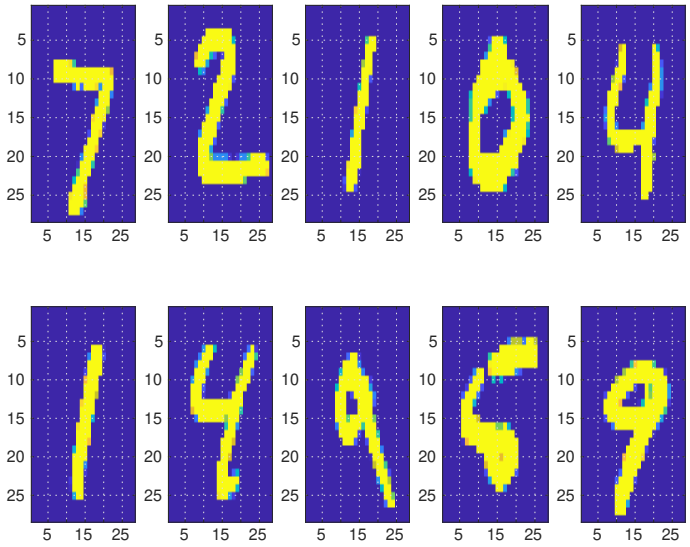


a) Dataset in *mnist_test_10.csv*

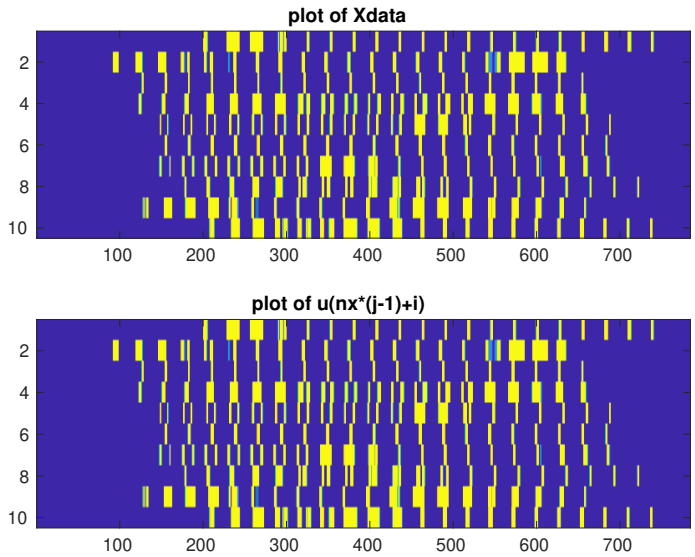


b) Images from *mnist_train.csv*

PCA to find patterns for MNIST dataset



a) Dataset in *mnist_test_*10.*csv*



b) Matrix of images

Dataset

`mnist_test_10.csv`

contains 10 images presented in Figure a). This is a test dataset which should be used to recognise handwritten digits in the dataset

`mnist_train.csv`

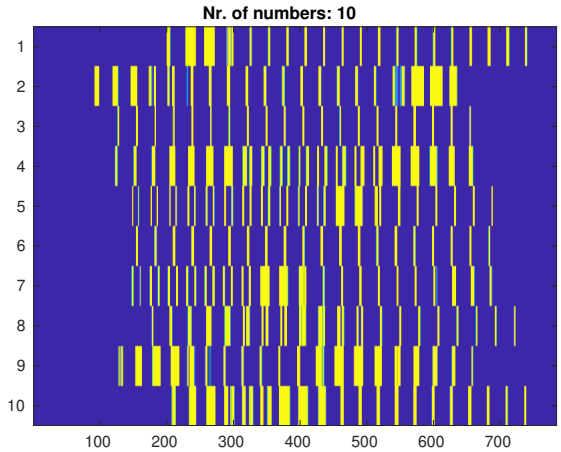
PCA to find patterns for MNIST dataset

Top image of Figure b) represents matrix array Xdata from the program plotmnist.m of all 10 images presented in Figure a). Bottom image of Figure b) is the same array which is read in the another array as

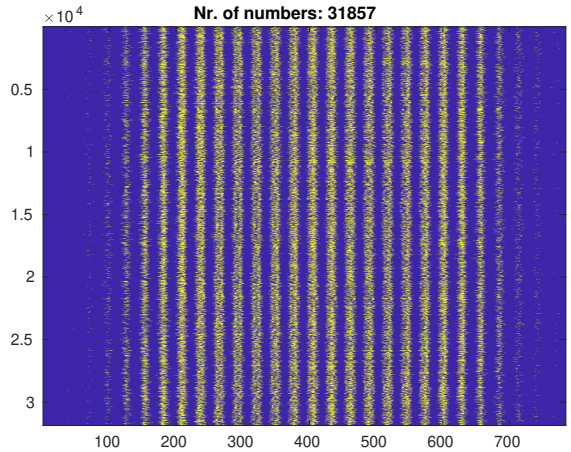
```
[nx,ny] = size(Xdata)
u =   zeros(nx,ny);

for j=1:ny
for i=1:nx
u(nx*(j-1)+i) = Xdata(nx*(j-1)+i);
end
end
```

Every row of this array is a vector of the size $nx \times ny$, where nx is the number of nodes in x direction and ny is the number of nodes in y direction of every image presented in Figure a). In our case, $nx = 28$, $ny = 28$, and this $28 \cdot 28 = 784$ is size of every row of Figure b) for 10 rows. We have 10 rows and 784 columns for matrix of test images.

PCA to find patterns for MNIST dataset



a) Matrix of test images



b) Matrix of train images

We need to perform PCA analysis of matrix of train images and test images. After PCA we will have original data on terms of eigenvectors and eigenvalues of the covariance matrix.
The next step is to measure difference between the new image and the original image, but not along the original axes, but along the new axes which are obtained in PCA analysis.
It was shown [1] that these new axes gives better information for the case of image recognition since the PCA analysis gives the original image in terms of the differences and similarities between data.

[1] J. ZHANG, Y. YAN, M. LADES, Face Recognition: Eigenface, Elastic Matching, and Neural Nets, *Proc. IEEE,* 1997.