Indian Statistical Institute, Kolkata
Numerical Analysis (BStat–I)

Instructor: Sourav Sen Gupta
Scribe:
Saptarshi Chakraborty(BS1504)
Rohan Hore(BS1519)
Date of Lecture: 05 February 2016

LECTURE
6

More on SVD and Gram-Schmidt Orthogonalization

6.1 Physical Interpretation of SVD

In this section we give the physical interpretation of singular value decomposition(SVD). Before we proceed we need the following theorem.

Theorem 6.1. *The determinant of an orthogonal matrix is either* $+1$ *or* -1

Proof. Since $det(A) = det(A^T)$

$$1 = det(I_n) = det(A^T A) = det(A^T)det(A) = (det(A))^2$$

$$\implies det(A) = +1 or - 1$$

□

Let $A = U\Sigma V^T$ be the svd of $A_{m\times n}$.To interpret svd geometrically we observe matrices U and V^T are orthogonal.Hence they preserve lengths and angles.consider the vector $\vec{x}$ .By multiplying by V^T we find its' components a along the direction of $v_i's$.Where $v_i's$ form the columns of V.Then the matrix Σ expands the component along v_i by a factor of d_i.Then the matrix U throws it into the column space of A preserving the lengths and angles.

6.1.1 relation between $det(A)$ and $det(\Sigma)$

If we assume $A_{n\times n}$ is a square matrix,then

$$\left|det(A)\right| = \left|det(U)\right|\left|det(\Sigma)\right|\left|V^T\right| = \left|det(\Sigma)\right| = \prod_{i=1}^{n} |\sigma_i|$$

Physically a determinant can be interpreted as the volume of the n-cuboid formed by the columns of A.This is equal to the product the diagonal entries of Σ.

6.2 Approximation by low rank matrices

Expanding $A = U\Sigma V^T$ we get,

$$A = \sum_{i=1}^{l} \sigma_i \vec{u_i}\vec{v_i}^T$$

where $l = min\{m, n\}$and $\vec{u_i}$ and $\vec{v_i}$ are the i-th column of U and V respectively.The sum goes to l because the other terms will be zeroed out by Σ.This shows that the matrix A(say of rank r) is the sum of matrices of rank 1 each.In practice if we drop out a few terms having small σ_i,this serves as a very good approximation of A.This is called row rank approximation of the matrix A.

6.2.1 Application of low rank approximation

One application of low rank approximation is to store high quality digital picture in small memory.The amount of memory to store A high.Instead if we store some columns of U,V and some values of σ_i, then we require much less memory.
As a simple illustration we show the picture quality of the following pictures when we take 10,20,30 terms of the sum.(see figure 6.1)

6.3 Gram-Schmidt Process

6.3.1 Regular Gram-Schmidt process

Our first algorithm for QR decomposition is the regular Gram-Schmidt process.Assuming the columns of the matrix $A_{m\times n}$ be linearly independent,we can apply Gram-Schmidt orthogonalization process to orthonormalize the columns of $A_{m\times n}$.To do this we multiply A by a bunch of upper triangular matrices to do the column operations.In this way we get Q.By taking the inverse of the bunch of upper triangular matrices, we get R.The algorithm for regular Gram-Schmidt process is as follows.

```
**function** Gram-Schmidt $(\vec{v_1}, \vec{v_2}, ..., \vec{v_k})$
compute an orthonormal basis $\hat{a}_1, ..., \hat{a}_k$ of the span $\{\vec{v_1}, \vec{v_2}, ..., \vec{v_k}\}$
Assumes $\{\vec{v_1}, \vec{v_2}, ..., \vec{v_k}\}$ are linearly independent. $\hat{a}_1 = \vec{v_1}/\|\vec{v_1}\|$
**for(i in 2:k)**
$\vec{p} = \vec{0}$
for(j in 1:1)
$\vec{p} = \vec{p} + (\vec{v_i}.\hat{a}_j)\hat{a}_j$
$\vec{r} = \vec{v_i} - \vec{p}$
$\hat{a}_i = \vec{r}/\|\vec{r}\|$ return$\{\hat{a}_1, ..., \hat{a}_k\}$
```

6.3.2 modified gram-Schmidt process

Here also,initially we have the column vectors $\vec{v_1}$,$\vec{v_2}$..$\vec{v_n}$ and we go on finding the orthonormal vectors $\hat{q}_1$,$\hat{q}_2$...$\hat{q}_n$ but here once we get one of the orthonormal vectors(say $\hat{q}_i$) we will immediately project $\hat{q}_i$ out of the remaining column vectors $\vec{v_{i+1}}$,$\vec{v_{1+2}}$ and so on.

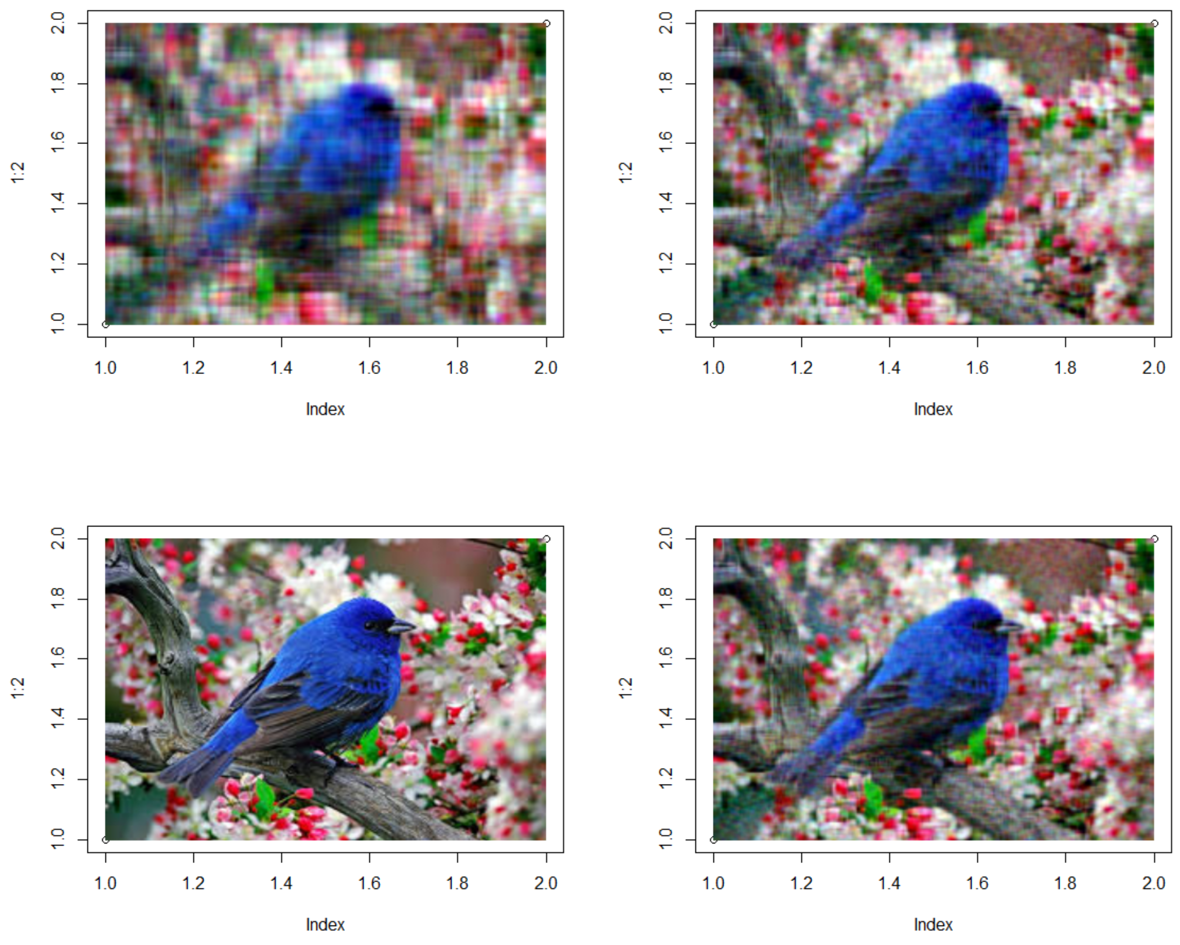

Figure 6.1: (Clockwise)picture having 10,20,30 principal components respectively and the original picture

6.3.2.1 Algorithm For Modified Gram-Schmidt Process

Thus the algorithm for performing the modified gram Schmidt process can be understood by this algorithm written using R language:

```
for(i in 1:k)$\vec{v_i}$= rnorm(n,mean=0,sd=1)
computes an orthonormal basis $\hat{q}_1, .....\hat{q}_k$ for the input $\vec{v_1}, .....\vec{v_k}$
Assumes $\vec{v_1}, .....\vec{v_k}$ linearly independent
for(i in 1:k)$\hat{q}_i = \vec{v_i}/norm(\vec{v_i}, type = "F")$
for(j in i+1:k)$\vec{v_j} = \vec{v_j} - (\langle \vec{v_j}.\hat{q}_i \rangle) * \hat{q}_i$
return( $\hat{q}_1, ....\hat{q}_k$)
```

6.3.3 comparison between the Gram-Schmidt Process

The Gram-Schmidt algorithm sometimes is not good .For instance, due to rounding and other issues,sometimes $\hat{q}_i$'s are not completely orthogonal after the projection step. Our projection formula for finding $\vec{p}$ within the algorithm, however, only works when the $\hat{q}_i$'s are orthogonal. For this reason, in the presence of rounding, the projection $\vec{p}$ of $\vec{p_i}$ becomes less accurate.

One way to solve this issue is the "modified Gram-Schmidt" algorithm which has similar running time but makes a subtle change in the way projections are computed. Rather than computing the projection $\vec{p}$ in each iteration 'i' onto span $[\hat{q}_i; : : : ; \hat{q}_{i-1}]$, as soon as $\hat{q}_i$ is computed it is projected out of $[\vec{p_{i+1}}; : : : ; \vec{p_k}]$ subsequently. we never have to consider $\hat{q}_i$ again. This way, even if the basis is not completely orthogonal due to rounding, the projection step is valid since it only projects onto one $\hat{q}_i$ at a time. In the absence of rounding, modified Gram-Schmidt and regular Gram-Schmidt generate identical output.But,in terms of machine computation "MGS" is better.

6.3.4 Drawbacks of Gram-Schmidt Process

if any two of the input vectors i.e. $\vec{v_i}$ and $\vec{v_j}$ of the input vectors are very close,then that will turn the Gram-Schmidt process unstable.
Suppose,we provide two vectors v_1=(1,1) and $v_2 = (1+\epsilon, 1)$ where $\epsilon << 1$.Then one reasonable basis of span[v_1,v_2] can be $[(0,1)^t$,$(1,0)^t]$ But,if we apply Gram-Schmidt process we obtain
$\hat{q_1} = \vec{v_1}/\|\vec{v_i}\| = (1/\sqrt{2}) * (1,1)^t$
$\vec{p} = ((2+\epsilon)/2) * (1,1)^t$
$\vec{r}$=$\vec{v_2}$-$\vec{p} = (1/2) * (\epsilon, -\epsilon)^t$
$\|\vec{r}\|$=$(1/\sqrt{2}) * \epsilon$
So,computing $\hat{q_2}$ =$(1/\sqrt{2})$*$(1,-1)^t$ needs division by a very small number of o(ϵ)which leads us to unstable numerical operation.
Thus,in these cases we should avoid applying Gram-Schmidt process.

6.4 QR decomposition

We can do QR decomposition by Gram-Schmidt Process.Consider the columns of The given matrix(Say,$A_(mxn)$)as the vectors v_1,v_2.....v_n as given set of vectors.We will apply Gram-Schmidt process to the set of vectors to get q_1,q_2....q_n and then R will be found.Now,A may not be full column rank matrices.In that case,we will consider only the linearly independent columns and then will extend it.
For a full column rank matrix the R matrix can be written as:

$$\begin{bmatrix} \langle x_1.x_2\rangle & \langle x_2.q_1\rangle & ... & \langle x_m.q_1\rangle \\ 0 & \langle x_2.q_2\rangle & ... & \langle x_m.q_2\rangle \\ 0 & 0 & .. & \langle x_m.q_{n-1}\rangle \\ 0 & 0 & 0 & \langle x_m.q_n\rangle \end{bmatrix}$$

6.4.1 QR decomposition by other ways

The general process of QR decomposition by Gram-Schmidt can be pictorised as:
A*U_1*U_2*...*U_{n-1}*U_n=Q
where Q is the ultimate resultant orthogonal matrix.And U are the upper triangular matrices we multiplied with.Thus,the R is evaluated by U_n^{-1},U_{n-1}^{-1}...U_1^{-1} thus here,we are post-multiplying A by upper-triangular matrices to get Q..but let's try to pre multiply A by some matrices to get R.Obviously,the matrices are bound to be orthogonal.Since,Q^T*A=R.thus,We are multiplying A by orthogonal matrices.Now,orthogonal matrices can't do any kind of stretching.Thus,orthogonal matrices can only do reflection and rotation.
Let's try rotation first.

6.4.1.1 QR decomposition by rotation

objective: all the elements below the diagonal of the k^{th} column of the given matrices should be made.
Thus,we want to make all the elements below the first element of column a zero.Say,the first column of A=(a_{11},a_{12},...a_{1n}) should be transformed by this process: A=(a_{11},a_{12},...a_{1n}) –> (a^1_{11},0,.....a^1_{1n})–> (a^2_{11},0,0,...a^2_{1n})–> (a^{n-1}_{11},0,0....0)
Now,since rotation matrices do not perform any kind of stretching,our objective is to do a rotation which performs $\vec{v}$ –> $\|\vec{v}\|$*e_1 where say e_1 is the first vector of canonical basis.In case of R^2 we already know,how to do it.
now,in case of R^2 we know,the rotation matrix turns out to be

$$\begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix}$$

thus we generalised this rotation matrix in case of R^n by the process of Given's rotation matrix.Given defined the matrix $G(i, j, \theta)$ as follows:

$$\begin{bmatrix} 1 & 0 & .. & .. & 0 \\ 0 & 1 & .. & .. & 0 \\ : & .. & [c]_{i,i} & [s]_{i,j} & 0 \\ : & .. & [-s]_{j,i} & [c]_{j,j} & 0 \\ : & .. & .. & .. & 1 \end{bmatrix}$$

where c= $cos(\theta)$ and s=$sin(\theta)$ appears at the i^{th} and j^{th} rows and columns.

The non-zero elements of given matrix $G(i, j, \theta)$is defined as:
g_{kk}=1
g_{ii}=c
g_{jj}=c
g_{ij}=s if $j > i$
g_{ji}=-s if $j < i$
the product $G(i, j, \theta)x$ means a counter clockwise rotation of x by θ radians in the (i,j)planes.
thus,we will multiply A by all $G(i, j, \theta)$ $\forall$ j>i.Thus,by performing all the multiplications,we ultimately get the R matrix.Thus,we are done.

6.4.1.2 QR by reflection

So far,we have only done QR decomposition by pre-multiplying by suitable rotation matrices.Now,since we started our talk with performing QR by any orthonormal matrices,now we should think about whether it is even possible.
The answer is "yes" and the way to do it is called 'householder transformation'.

References

[1] Samuel Conte and Carl de Boor, *Elementary Numerical Analysis – An Algorithmic Approach*, McGraw-Hill Education.

[2] Justin Solomon, *Numerical Algorithms.* Available online on the course website.