Tensor computation Assignment 2 Solution

Chandan Kumar Singh
SR No -13-19-02-19-52-24-1-24598

April 4, 2025

0.1 Square root of a symmetric positive definite tensor:

Suppose a tensor A of third order :

$$A \in \mathbf{R}^{(n,n,m)}$$

Above Tensor A has m frontal slice , we need to make each frontal slice symmetric first. To make each frontal slice symmetric we can take any random $n * n$ matrix R and find $R * R^T$.
Further to make a tensor positive definite , we need to ensure below :

$$< X, A * X >> 0 \quad \forall \; X \in \mathbf{R}^{(n,1,m)}$$

We making a frontal slice symmetric using $R * R^T$, but if it's not a full rank n because it has zero eigen values , Then it's positive semi definite . If we add $n * I$ any small positive value to lift the smallest eigen value above zero , it will be strict positive definite.

Algorithm 1 Making a symmetric positive definite tensor

```
input size of 3rd order tensor as n,n,p
Initialize a tensor A of size (n * n * p)
Initialize a identity matrix I of size (n * n)
for  i = 1 to p do
    Create a random matrix R of size (n * n)
    A = R * R^T + n * I
end for
return  A
```

Checkout the code: Click here

Square of positive definite tensor using T-Product:

Once we get the symmetric positive definite tensor , we can get eigen decomposition matrix for each frontal slice in **Fourier Transform** :

$$A = V * D * V^T$$

now we can do square root of the diagonal matrix for each frontal slice in **Fourier Transform** :

$$\sqrt{A} = V * \sqrt{D} * V^T$$

Then ifft back to tensor domain . Remember that if we multiply $\sqrt{A} * \sqrt{A}$ we will get the original A . Also $\sqrt{A}$ will also be a symmetric tensor .

Algorithm 2 Square root of symmetric positive definite tensor using T-Product

```
Input a Tensor A of size n,n,p
Set Af = FFT(A, axis = 2)
Initialize V of size n, n, p with all element as 0 and type complex
Initialize D of size n, n, p with all element as 0 and type complex
Initialize Sf of size n, n, p with all element as 0 and type complex
for i = 1 to p do
    eVal,EVec = eign(Af[:,:,i])
    V[:,:,i]=EVec
    D[:,:,i]= diagonal matrix using eVal
    Sf= V[:,:,i] * sqrt(D[:,:,i]) * V[:,:,i].T
end for
Set S = iFFT(Sf, axis = 2 )
return  S
```

Checkout the code: Click here

Square of positive definite tensor using M-Product:

Once we get the symmetric positive definite tensor , we can get eigen decomposition matrix for each frontal slice in **M-Transform domain**:

$$A = V * D * V^T$$

now we can do square root of the diagonal matrix for each frontal slice in **M-Transform domain** :

$$\sqrt{A} = V * \sqrt{D} * V^T$$

Then **inverse M-transform** back to tensor domain . Remember that if we multiply $\sqrt{A} * \sqrt{A}$ we will get the original A . Also $\sqrt{A}$ will also be a symmetric tensor .

Algorithm 3 Square root of symmetric positive definite tensor using M-Product

```
Input a Tensor A of size n,n,p
take a random diagonal matrix M of size n*n
M_inv = inverse(M)
Set Am = mode3Product(M * A)
Initialize V of size n, n, p with all element as 0
Initialize D of size n, n, p with all element as 0
Initialize Sm of size n, n, p with all element as 0
for i = 1 to p do
    eVal,EVec = eign(Am[:,:,i])
    V[:,:,i]=EVec
    D[:,:,i]= diagonal matrix using eVal
    Sm= V[:,:,i] * sqrt(D[:,:,i]) * V[:,:,i].T
end for
Set S = mode3Product(M_inv * Sm) )
return S
```

Checkout the code: Click here

0.2 Hide a color video in another video using QR decomposition

In this we have two video given one is cover video and another one is secret video . We will hide secret video inside cover video and send that updated cover video .Now if someone get this cover video then we show how to get the secret video which we wanted to send .

First we get both videos in tensor say Cover video in tensor C and secret video in tensor S.

$$C, S \in \mathbf{R}^{(144*144*20*3)}$$

Now we do QR-Decomposition of both Tensor :

$$C_q, C_r = QRDecp(C)$$

$$S_q, S_r = QRDecp(S)$$

Now we will create r component of video T as T_r which we will send as cover video hiding our secret video inside .

$$T_r = C_r + 0.5 * S_r$$

we will use q component of this transfer video same as C_q

$$T_q = C_q$$

Get the updated Cover video T which hide secret video inside by multiplying the tensors T_r and T_q

$$T = T_r * T_q$$

We will now have to send this updated Cover video T and q component of secret video S_q . Here is what we send , updated cover video. Using M-Product :Click here Using T-Product :Click here

At the receiver end we get a video T (cover video) and S_q . Let's see how to get secret video back from this video T . First get the Q-R decomposition of this T .

$$T_q, T_r = QRDecp(T)$$

Create the r component to get secret video

$$O_r = (T_r - C_r)/0.5$$

q component will be same as S_q as we also send this .

$$O_q = S_q$$

Now get the O video which is the secret video we want to send :

$$O = O_q * O_r$$

Here is what we get , when we create video back from this Tensor O . Using M-Product :Click here Using T-Product :Click here

Here is the implementation using M-Product

Algorithm 4 QR Decomposition application on video using M-Product

Input a tensor A of (m,n,p,q)
Set Am= Transform A to M Domain
Initialize Q_{prm} as all element 0 and have same size as A
Initialize R_{prm} as all element 0 and have same size as A
for $i = 0$ to p **do**
  for $j = 0$ to q **do**
    $Q_t, R_t = QR(Am[:, :, i, j])$
    $Q_{prm}[:, :, i, j] = Q_t$
    $R_{prm}[:, :, i, j] = R_t$
  end for
end for
Q =Transform back Q_{prm} to T Domain
R =Transform back R_{prm} to T Domain
return Q,R

Checkout the code: Click here

Here is the implementation using T-Product

,

Algorithm 5 QR Decomposition application on video using T-Product

Input a tensor A of (m,n,p,q)
Set $A_{prm} = A$
for $i = [2, 3]$ **do**
  $A_{prm} = FFT(A_{prm}, axis = i)$
end for
Initialize Q_{prm} as all element 0 with complex type and have same size as A
Initialize R_{prm} as all element 0 with complex type and have same size as A
for $i = 0$ to p **do**
  for $j = 0$ to q **do**
    $Q_t, R_t = QR(A_{prm}[:, :, i, j])$
    $Q_{prm}[:, :, i, j] = Q_t$
    $R_{prm}[:, :, i, j] = R_t$
  end for
end for
set $Q = Q_{prm}$
set $R = R_{prm}$
for $i = [3, 2]$ **do**
  $Q = iFFT(Q, axis = i)$
  $R = iFFT(R, axis = i)$
end for
return Q,R

Checkout the code: Click here

0.3 Image Compression using SVD

In this problem a image is given , We first have to get image into tensor A of order 3 as input . Then apply SVD to compress the image . The image given is of size 6436 KB.

$$A \in \mathbf{R}^{(2000,1968,3)}$$

Find SVD of A and you get :

$$U \in \mathbf{R}^{(2000,2000,3)}, D \in \mathbf{R}^{(2000,1968,3)}, V \in \mathbf{R}^{(1968,1968,3)} = SVD(A)$$

Here U is orthogonal and D is diagonal , D size is $(1968, 1968, 3)$. Now to compress the image we need to take a lower size of D like $(1500, 1500, 3)$ and take U and V accordingly . Once get U , D and V , we can multiply them to get the image back . In this case the size of compress image will be 3815 KB almost half of original .

Let's check the implementation using T-Product and M-Product one by one and we can see M-Product is faster.

Algorithm 6 Image Compression :Application of SVD using T-Product

Input a Tensor A of size m,n,p
Set $AFT = FFT(A, axis = 2)$
Initialize UFT of size m, n, p with all element as 0 and type complex
Initialize DFT of size m, n, p with all element as 0 and type complex
Initialize VFT of size m, n, p with all element as 0 and type complex
for $i = 1$ to p **do**
  $U_F, D_F, V_F = svd(AFT[:,:,i])$
  $UFT[:,:,i] = U_F$
  $DFT[:,:,i] = DiagonalofD_F$
  $VFT[:,:,i] = V_F.T$
end for
Set $U = iFFT(UFT, axis = 2\ )$
Set $D = iFFT(DFT, axis = 2\ )$
Set $V = iFFT(VFT, axis = 2\ )$
return U,D,V

Checkout the code: Click here

Algorithm 7 Image Compression :Application of SVD using M-Product

Input a Tensor A of size m,n,p
take a random diagonal matrix M of size p*p
$M_{inv} = inverse(M)$
Set $Am = mode3Product(M * A)$
Initialize Um of size m, n, p with all element as 0
Initialize Dm of size m, n, p with all element as 0
Initialize Vm of size m, n, p with all element as 0
for $i = 1$ to p **do**
  $U_F, D_F, V_F = svd(Am[:,:,i])$
  $Um[:,:,i] = U_F$
  $Dm[:,:,i] = DiagonalofD_F$
  $Vm[:,:,i] = V_F.T$
end for
Set $U = mode3Product(M_{inv} * Um)$
Set $D = mode3Product(M_{inv} * Dm)$
Set $V = mode3Product(M_{inv} * Vm)$
return U,D,V

Checkout the code: Click here

0.4 Color image de-blurring Using Moore-Penrose inverse of a tensor

This problem is application of Moore-Penrose inverse of a tensor . Below is the algo to find inverse of tensor using both T-product and M-product . Let's see how we can use given image and blur it . Suppose A is a tensor created using given image .

$$A \in \mathbf{R}^{(2000,1968,3)}$$

Let's take any Random $X \in \mathbf{R}^{(1968,2000,3)}$ such that :

$$A * X = B$$

This B is our blur image , which will look like this.
To reconstruct original image using this Blur image B . You just need to multiply tensor B (blur image) with Moore-Penrose inverse of tensor X which we have used to blur it .

$$A = B * X^{-1}$$

And we get this original image back :

Algorithm 8 Moore-Penrose inverse using T-Product

```
1: Input a Tensor A of size m,n,p
2: Set AFT = FFT(A, axis = 2)
3: Initialize AIFT of size m, n, p with all element as 0 and type complex
4: for i = 1 to p do
5:     AIFT =pinv(AFT[:,:,i])
6: end for
7: Set AI = iFFT(AIFT, axis = 2 )
8: return AI
```

Checkout the code: Click here

Algorithm 9 Moore-Penrose inverse using using M-Product

```
1: Input a Tensor A of size m,n,p
2: take a random diagonal matrix M of size p*p
3: M_inv = inverse(M)
4: Set Am = mode3Product(M * A)
5: Initialize AIm of size (m,n,p) with all element as 0
6: for i = 1 to p do
7:     AIm = pinv(Am[:, :, i])
8: end for
9: Set AI = mode3Product(M_inv * AIm)
10: return AI
```

Checkout the code: Click here