# Linear_algebra Operations

December 3, 2021

## 1 Linear algebra

[1]:
```
import numpy as np
```

[2]:
```
np.__version__
```

[2]:
```
'1.20.1'
```

### 1.1 Matrix and vector products

Q1. Predict the results of the following code.

[8]:
```
x = [1,2]
y = [[4, 1],[2,2]]
print(np.dot(x, y))
print(np.matmul(x,y))
print(np.inner(x, y))
```

```
[8 5]
[8 5]
[6 6]
```

Q2. Predict the results of the following code.

[14]:
```
x = np.array([[1, 4], [5, 6]])
y = np.array([[4, 1], [2, 2]])
print(np.vdot(x, y))
print(np.vdot(y, x))
print(np.dot(x.flatten(), y.flatten()))
print((x*y).sum())
```

```
30
30
30
30
```

### 1.2 Decompositions

Q5. Get the lower-trianglular `L` in the Cholesky decomposition of x and verify it.

[17]:
```
x = np.array([[4, 12, -16], [12, 37, -43], [-16, -43, 98]], dtype=np.int32)
L = np.linalg.cholesky(x)
print(L)
```

```
[[ 2.  0.  0.]
 [ 6.  1.  0.]
 [-8.  5.  3.]]
```

Q6. Compute the qr factorization of x and verify it.

[18]:
```
x = np.array([[12, -51, 4], [6, 167, -68], [-4, 24, -41]], dtype=np.float32)
q, r = np.linalg.qr(x)
print("q=\n", q, "\nr=\n", r)
```

```
q=
 [[-0.85714287  0.3942857   0.33142856]
 [-0.42857143 -0.9028571  -0.03428571]
 [ 0.2857143  -0.17142858  0.94285715]]
r=
 [[ -14.  -21.   14.]
 [   0. -175.   70.]
 [   0.    0.  -35.]]
```

Q7. Factor x by Singular Value Decomposition and verify it.

[21]:
```
x = np.array([[1, 0, 0, 0, 2], [0, 0, 3, 0, 0], [0, 0, 0, 0, 0], [0, 2, 0, 0,␣
 ↪0]], dtype=np.float32)
U, s, V = np.linalg.svd(x, full_matrices=False)
print("U=\n", U, "\ns=\n", s, "\nV=\n", V)
```

```
U=
 [[ 0.  1.  0.  0.]
 [ 1.  0.  0.  0.]
 [ 0.  0.  0. -1.]
 [ 0.  0.  1.  0.]]
s=
 [3.       2.236068 2.       0.      ]
V=
 [[-0.         0.         1.         0.         0.       ]
 [ 0.4472136  0.         0.         0.         0.8944272]
 [-0.         1.         0.         0.         0.       ]
 [ 0.         0.         0.         1.         0.       ]]
```

### 1.3 Matrix eigenvalues

Q8. Compute the eigenvalues and right eigenvectors of x. (Name them eigenvals and eigenvecs, respectively)

[25]:
```
x = np.diag((1, 2, 3))
eigenvals,eigenvecs = np.linalg.eig(x)
print("eigenvalues are\n", eigenvals)
print("eigenvectors are\n", eigenvecs)
```

```
eigenvalues are
 [1. 2. 3.]
eigenvectors are
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

### 1.4 Norms and other numbers

Q10. Calculate the Frobenius norm of x

[31]:
```
x = np.arange(1, 10).reshape((3, 3))
print(np.linalg.norm(x,'fro'))
```

```
16.881943016134134
```

Q11. Calculate the determinant of x.

[32]:
```
x = np.arange(1, 5).reshape((2, 2))
out1 = np.linalg.det(x)
#out2 = x[0, 0] * x[1, 1] - x[0, 1] * x[1, 0]
print(out1)
```

```
-2.0000000000000004
```

Q12. Calculate the rank of x.

[33]:
```
x = np.eye(4)
out1 = np.linalg.matrix_rank(x)
print(out1)
```

```
4
```

Q13. Compute the sign and natural logarithm of the determinant of x.

[34]:
```
x = np.arange(1, 5).reshape((2, 2))
sign, logdet = np.linalg.slogdet(x)
det = np.linalg.det(x)
print(sign, logdet)
```

```
-1.0 0.6931471805599455
```

Q14. Return the sum along the diagonal of x.

[35]:
```
x = np.eye(4)
out1 = np.trace(x)
```

```
#out2 = x.diagonal().sum()
print(out1)
```

```
4.0
```

### 1.5 Inverting matrices

Q15. Compute the inverse of x.

[36]:
```
x = np.array([[1., 2.], [3., 4.]])
out1 = np.linalg.inv(x)
print(out1)
```

```
[[-2.   1. ]
 [ 1.5 -0.5]]
```

[ ]:
```

```