mysvd

November 21, 2021

[104]:
```
import numpy as np
from numpy.linalg import norm

from random import normalvariate
from math import sqrt
```

[105]:
```
def randomUnitVector(n): #defining a random unit vector 'x'
    unnormalized = [normalvariate(0, 1) for _ in range(n)]
    Norm = sqrt(sum(x * x for x in unnormalized))
    return [x / Norm for x in unnormalized]

def svd_1d(A, epsilon=1e-10): #finding a singular vector
    ''' The one-dimensional SVD '''

    m, n = A.shape
    x = randomUnitVector(n)
    prevV = None
    currentV = x
    B = np.dot(A.T, A)

    iterations = 0
    while True:
        iterations += 1
        prevV = currentV
        currentV = np.dot(B, prevV)
        currentV = currentV / norm(currentV)

        if abs(np.dot(currentV, prevV)) > 1 - epsilon:
            print("converged in {} iterations".format(iterations))
            return currentV
```

[106]:
```
if __name__ == "__main__":
    movieRatings = np.array([
        [2, 5, 3],
        [1, 2, 1],
        [4, 1, 1],
        [3, 5, 2],
```

```
        [5, 3, 1],
        [4, 5, 5],
        [2, 4, 2],
        [2, 2, 5],
    ], dtype='float64')
```

[107]:
```
def mysvd(A, epsilon=1e-10):
    m, n = A.shape
    svdSoFar = []

    for i in range(n):       #doing iteration till 'n'
        matrix_1D = A.copy()

        for singularValue, u, v in svdSoFar[:i]:
            matrix_1D -= singularValue * np.outer(u, v)

        v = svd_1d(matrix_1D, epsilon=epsilon)  # next singular vector
        u_unnormalized = np.dot(A, v)
        sigma = norm(u_unnormalized)  # next singular value
        u = u_unnormalized / sigma

        svdSoFar.append((sigma, u, v))

    # transform it into matrices of the right shape
    singularValues, us, vs = [np.array(x) for x in zip(*svdSoFar)]

    return singularValues, us.T, vs
```

[108]:
```
#Example 1:
if __name__ == "__main__":
    A = np.array([
        [2, 5, 3],
        [1, 2, 1],
        [4, 1, 1],
        [3, 5, 2],
        [5, 3, 1],
        [4, 5, 5],
        [2, 4, 2],
        [2, 2, 5],
    ], dtype='float64')

    a, b, c = mysvd(A)
    print("The diagonal elements of sigma matrix are " + str(a))
    print("U = " + str(b))
    print("V = " + str(c))
```

```
converged in 6 iterations
converged in 30 iterations
converged in 2 iterations
The diagonal elements of sigma matrix are [15.09626916  4.30056855  3.40701739]
U = [[ 0.39458528 -0.23922996  0.35446498]
 [ 0.15830232 -0.03054664  0.15299835]
 [ 0.22155197  0.52085541 -0.39336182]
 [ 0.39692634  0.08649674  0.41052676]
 [ 0.34630252  0.64128745 -0.07384417]
 [ 0.53347451 -0.19169128 -0.19948869]
 [ 0.31660465 -0.06109329  0.3059967 ]
 [ 0.32840227 -0.45971334 -0.62353641]]
V = [[ 0.54184789  0.67070996  0.50650668]
 [ 0.75151584 -0.11679481 -0.64929416]
 [-0.37633071  0.73246646 -0.56733418]]
```

[109]:
```python
#checking with svd function from numpy
from numpy.linalg import svd

A = [
    [2, 5, 3],
    [1, 2, 1],
    [4, 1, 1],
    [3, 5, 2],
    [5, 3, 1],
    [4, 5, 5],
    [2, 4, 2],
    [2, 2, 5],
]

U, singularValues, V = svd(A)
print("Singular values = "+ str(singularValues))
print("U = " + str(U))
print("V = " + str(V))
```

```
Singular values = [15.09626916  4.30056855  3.40701739]
U = [[-0.39458526  0.23923575 -0.35445911 -0.38062172 -0.29836818 -0.49464816
  -0.30703202 -0.29763321]
 [-0.15830232  0.03054913 -0.15299759 -0.45334816  0.31122898  0.23892035
  -0.37313346  0.67223457]
 [-0.22155201 -0.52086121  0.39334917 -0.14974792 -0.65963979  0.00488292
  -0.00783684  0.25934607]
 [-0.39692635 -0.08649009 -0.41052882  0.74387448 -0.10629499  0.01372565
  -0.17959298  0.26333462]
 [-0.34630257 -0.64128825  0.07382859 -0.04494155  0.58000668 -0.25806239
   0.00211823 -0.24154726]
 [-0.53347449  0.19168874  0.19949342 -0.03942604  0.00424495  0.68715732
  -0.06957561 -0.40033035]
```

```
 [-0.31660464  0.06109826 -0.30599517 -0.19611823 -0.01334272  0.01446975
   0.85185852  0.19463493]
 [-0.32840223  0.45970413  0.62354764  0.1783041   0.17631186 -0.39879476
   0.06065902  0.25771578]]
V = [[-0.54184808 -0.67070995 -0.50650649]
 [-0.75152295  0.11680911  0.64928336]
 [ 0.37631623 -0.73246419  0.56734672]]
```

[110]:

```
#Example 2:
if __name__ == "__main__":
    A = np.array([
        [3, 2],
        [2, 3],
        [2, -2],
    ], dtype='float64')

    a, b, c = mysvd(A)
    print("The diagonal elements of sigma matrix (singular values) are " +␣
 ↪str(a))
    print("U = " + str(b))
    print("V = " + str(c))
```

```
converged in 12 iterations
converged in 2 iterations
The diagonal elements of sigma matrix (singular values) are [5. 3.]
U = [[-7.07105901e-01 -2.35709591e-01]
 [-7.07107661e-01  2.35694929e-01]
 [ 3.51892412e-06 -9.42809042e-01]]
V = [[-0.70710238 -0.70711118]
 [-0.70711118  0.70710238]]
```

[111]:

```
#checking with svd function from numpy
from numpy.linalg import svd

A = [
    [3, 2],
    [2, 3],
    [2, -2],
]

U, singularValues, V = svd(A)
print("Singular values = "+ str(singularValues))
print("U = " + str(U))
print("V = " + str(V))
```

```
Singular values = [5. 3.]
U = [[-7.07106781e-01  2.35702260e-01 -6.66666667e-01]
 [-7.07106781e-01 -2.35702260e-01  6.66666667e-01]
 [-1.66533454e-16  9.42809042e-01  3.33333333e-01]]
```

```
V = [[-0.70710678 -0.70710678]
 [ 0.70710678 -0.70710678]]
```

[112]:

```
import numpy as np
from scipy.linalg import svd


A = np.array([[2, 3, 2], [3,3,-2]])
print("A: ",X)
# performing SVD
U, singular, V_t = svd(X)

print("U: ",U)
print("Singular array: ",s)
print("V^T: ",V_t)


#Calculate Pseudo inverse

singular_inv = 1.0 / singular # inverse of singular matrix is just the
 ↪reciprocal of each element
s_inv = np.zeros(A.shape)   # creating m x n matrix of zeroes and inserting
 ↪singular values in it
s_inv[0][0]= singular_inv[0]
s_inv[1][1] =singular_inv[1]
# calculate pseudoinverse
M = np.dot(np.dot(V_t.T,s_inv.T),U.T)
print("Pseudo-inverse matrix: ",M)
```

```
A:  [[ 3  3  2]
 [ 2  3 -2]]
U:  [[ 0.7815437 -0.6238505]
 [ 0.6238505  0.7815437]]
Singular array:  [[-0.11372532  0.70002955]
 [-0.26712717  0.38298285]
 [-0.44066355  0.36835929]
 [-0.53380995 -0.4504762 ]
 [-0.6607361  -0.15705217]]
V^T:  [[ 0.64749817  0.7599438   0.05684667]
 [-0.10759258  0.16501062 -0.9804057 ]
 [-0.75443354  0.62869461  0.18860838]]
Pseudo-inverse matrix:  [[ 0.11462451  0.04347826]
 [ 0.07114625  0.13043478]
 [ 0.22134387 -0.26086957]]
```

[ ]: