COMP 576 Assignment 0

Sathyavat Eshvar Chandran (sec14)

September 11th 2019

Python Machine Learning Stack (Anaconda)

Output of "conda info"

```
 active environment : base
active env location : /usr/local/anaconda3
        shell level : 1
   user config file : /home/eshvar/.condarc
```

populated config files : conda version : 4.6.14 conda-build version : 3.17.6 python version : 3.6.7.final.0 base environment : /usr/local/anaconda3 (read only) channel URLs : https://repo.anaconda.com/pkgs/main/linux-64 https://repo.anaconda.com/pkgs/main/noarch https://repo.anaconda.com/pkgs/free/linux-64 https://repo.anaconda.com/pkgs/free/noarch https://repo.anaconda.com/pkgs/r/linux-64 https://repo.anaconda.com/pkgs/r/noarch package cache : /usr/local/anaconda3/pkgs /home/eshvar/.conda/pkgs envs directories : /home/eshvar/.conda/envs /usr/local/anaconda3/envs platform : linux-64 user-agent : conda/4.6.14 requests/2.19.1 CPython/3.6.7 Linux/4.15.0-60-generic ubuntu/18.04.1 glibc/2.27 UID:GID : 1000:1000 netrc file : None offline mode : False

Transition from MATLAB to Python

```
In [1]: import numpy as np
        import scipy.linalg
        a = np.arange(25).reshape((5, 5))
        b = np.arange(75, 100).reshape((5, 5))

In [2]: a.ndim

Out[2]: 2

In [3]: a.size
```

```
Out[3]: 25

In [4]: a.shape

Out[4]: (5, 5)

In [5]: n = 2
        a.shape[n-1]

Out[5]: 5

In [6]: np.array([[1.,2.,3.], [4.,5.,6.]])

Out[6]: array([[1., 2., 3.],
               [4., 5., 6.]])

In [7]: p = np.arange(10)
        q = np.arange(10, 20)
        r = np.arange(20, 30)
        s = np.arange(30, 40)
        np.block([[p,q], [r,s]])

Out[7]: array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
                16, 17, 18, 19],
               [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
                36, 37, 38, 39]])

In [8]: a[-1]

Out[8]: array([20, 21, 22, 23, 24])

In [9]: a[1,4]

Out[9]: 9

In [10]: a[1]

Out[10]: array([5, 6, 7, 8, 9])

In [11]: a[:5]

Out[11]: array([[ 0,  1,  2,  3,  4],
                [ 5,  6,  7,  8,  9],
                [10, 11, 12, 13, 14],
                [15, 16, 17, 18, 19],
                [20, 21, 22, 23, 24]])

In [12]: a[-5:]
```

```
Out[12]: array([[ 0,  1,  2,  3,  4],
                [ 5,  6,  7,  8,  9],
                [10, 11, 12, 13, 14],
                [15, 16, 17, 18, 19],
                [20, 21, 22, 23, 24]])

In [13]: a[0:3][:,4:9]

Out[13]: array([[ 4],
                [ 9],
                [14]])

In [14]: a[np.ix_([1,3,4],[0,2])]

Out[14]: array([[ 5,  7],
                [15, 17],
                [20, 22]])

In [15]: a[ 2:21:2,:]

Out[15]: array([[10, 11, 12, 13, 14],
                [20, 21, 22, 23, 24]])

In [16]: a[ ::2,:]

Out[16]: array([[ 0,  1,  2,  3,  4],
                [10, 11, 12, 13, 14],
                [20, 21, 22, 23, 24]])

In [17]: a[ ::-1,:]

Out[17]: array([[20, 21, 22, 23, 24],
                [15, 16, 17, 18, 19],
                [10, 11, 12, 13, 14],
                [ 5,  6,  7,  8,  9],
                [ 0,  1,  2,  3,  4]])

In [18]: a[np.r_[:len(a),0]]

Out[18]: array([[ 0,  1,  2,  3,  4],
                [ 5,  6,  7,  8,  9],
                [10, 11, 12, 13, 14],
                [15, 16, 17, 18, 19],
                [20, 21, 22, 23, 24],
                [ 0,  1,  2,  3,  4]])

In [19]: a.transpose()

Out[19]: array([[ 0,  5, 10, 15, 20],
                [ 1,  6, 11, 16, 21],
                [ 2,  7, 12, 17, 22],
                [ 3,  8, 13, 18, 23],
                [ 4,  9, 14, 19, 24]])
```

```
In [20]: a.conj().T

Out[20]: array([[ 0,  5, 10, 15, 20],
                [ 1,  6, 11, 16, 21],
                [ 2,  7, 12, 17, 22],
                [ 3,  8, 13, 18, 23],
                [ 4,  9, 14, 19, 24]])

In [21]: a @ b

Out[21]: array([[ 900,  910,  920,  930,  940],
                [3025, 3060, 3095, 3130, 3165],
                [5150, 5210, 5270, 5330, 5390],
                [7275, 7360, 7445, 7530, 7615],
                [9400, 9510, 9620, 9730, 9840]])

In [22]: a * b

Out[22]: array([[   0,   76,  154,  234,  316],
                [ 400,  486,  574,  664,  756],
                [ 850,  946, 1044, 1144, 1246],
                [1350, 1456, 1564, 1674, 1786],
                [1900, 2016, 2134, 2254, 2376]])

In [23]: a/b

Out[23]: array([[0.        , 0.01315789, 0.02597403, 0.03846154, 0.05063291],
                [0.0625    , 0.07407407, 0.08536585, 0.09638554, 0.10714286],
                [0.11764706, 0.12790698, 0.13793103, 0.14772727, 0.15730337],
                [0.16666667, 0.17582418, 0.18478261, 0.19354839, 0.20212766],
                [0.21052632, 0.21875   , 0.22680412, 0.23469388, 0.24242424]])

In [24]: a**3

Out[24]: array([[    0,     1,     8,    27,    64],
                [  125,   216,   343,   512,   729],
                [ 1000,  1331,  1728,  2197,  2744],
                [ 3375,  4096,  4913,  5832,  6859],
                [ 8000,  9261, 10648, 12167, 13824]])

In [25]: (a>0.5)

Out[25]: array([[False,  True,  True,  True,  True],
                [ True,  True,  True,  True,  True],
                [ True,  True,  True,  True,  True],
                [ True,  True,  True,  True,  True],
                [ True,  True,  True,  True,  True]])

In [26]: np.nonzero(a>0.5)
```

```
Out[26]: (array([0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4,
                 4, 4]),
          array([1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2,
                 3, 4]))

In [27]: v = np.arange(5)
         a[:,np.nonzero(v>0.5)[0]]

Out[27]: array([[ 1,  2,  3,  4],
                [ 6,  7,  8,  9],
                [11, 12, 13, 14],
                [16, 17, 18, 19],
                [21, 22, 23, 24]])

In [28]: a[:,v.T>0.5]

Out[28]: array([[ 1,  2,  3,  4],
                [ 6,  7,  8,  9],
                [11, 12, 13, 14],
                [16, 17, 18, 19],
                [21, 22, 23, 24]])

In [29]: a[a<0.5]=0
         a

Out[29]: array([[ 0,  1,  2,  3,  4],
                [ 5,  6,  7,  8,  9],
                [10, 11, 12, 13, 14],
                [15, 16, 17, 18, 19],
                [20, 21, 22, 23, 24]])

In [30]: a * (a>0.5)

Out[30]: array([[ 0,  1,  2,  3,  4],
                [ 5,  6,  7,  8,  9],
                [10, 11, 12, 13, 14],
                [15, 16, 17, 18, 19],
                [20, 21, 22, 23, 24]])

In [31]: a[:] = 3

In [32]: y, x = np.arange(10).reshape((5,2)), np.arange(59, 69).reshape((5,2))

In [33]: y = x.copy()
         y

Out[33]: array([[59, 60],
                [61, 62],
                [63, 64],
                [65, 66],
                [67, 68]])
```

```
In [34]: y = x[1,:].copy()
         y

Out[34]: array([61, 62])

In [35]: y = x.flatten()
         y

Out[35]: array([59, 60, 61, 62, 63, 64, 65, 66, 67, 68])

In [36]: np.arange(1.,11.)

Out[36]: array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])

In [37]: np.arange(10.)

Out[37]: array([0., 1., 2., 3., 4., 5., 6., 7., 8., 9.])

In [38]: np.arange(1.,11.)[:, np.newaxis]

Out[38]: array([[ 1.],
                [ 2.],
                [ 3.],
                [ 4.],
                [ 5.],
                [ 6.],
                [ 7.],
                [ 8.],
                [ 9.],
                [10.]])

In [39]: np.zeros((3,4))

Out[39]: array([[0., 0., 0., 0.],
                [0., 0., 0., 0.],
                [0., 0., 0., 0.]])

In [40]: np.zeros((3,4,5))

Out[40]: array([[[0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.]],

                [[0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.]],

                [[0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.]]])
```

```
In [41]: np.ones((3,4))

Out[41]: array([[1., 1., 1., 1.],
                [1., 1., 1., 1.],
                [1., 1., 1., 1.]])

In [42]: np.eye(3)

Out[42]: array([[1., 0., 0.],
                [0., 1., 0.],
                [0., 0., 1.]])

In [43]: m = np.arange(25).reshape((5, 5))
         np.diag(m)

Out[43]: array([ 0,  6, 12, 18, 24])

In [44]: np.diag(m, 0)

Out[44]: array([ 0,  6, 12, 18, 24])

In [45]: np.random.rand(3,4)

Out[45]: array([[0.15107717, 0.78514642, 0.570351  , 0.8477816 ],
                [0.69636378, 0.39097365, 0.32918297, 0.53315161],
                [0.58151268, 0.83845932, 0.88263595, 0.09122831]])

In [46]: np.linspace(1,3,4)

Out[46]: array([1.        , 1.66666667, 2.33333333, 3.        ])

In [47]: np.mgrid[0:9.,0:6.]

Out[47]: array([[[0., 0., 0., 0., 0., 0.],
                 [1., 1., 1., 1., 1., 1.],
                 [2., 2., 2., 2., 2., 2.],
                 [3., 3., 3., 3., 3., 3.],
                 [4., 4., 4., 4., 4., 4.],
                 [5., 5., 5., 5., 5., 5.],
                 [6., 6., 6., 6., 6., 6.],
                 [7., 7., 7., 7., 7., 7.],
                 [8., 8., 8., 8., 8., 8.]],

                [[0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.]]])
```

```
In [48]: np.ogrid[0:9.,0:6.]
```

```
Out[48]: [array([[0.],
                 [1.],
                 [2.],
                 [3.],
                 [4.],
                 [5.],
                 [6.],
                 [7.],
                 [8.]]), array([[0., 1., 2., 3., 4., 5.]])]
```

```
In [49]: np.meshgrid([1,2,4],[2,4,5])
```

```
Out[49]: [array([[1, 2, 4],
                 [1, 2, 4],
                 [1, 2, 4]]), array([[2, 2, 2],
                 [4, 4, 4],
                 [5, 5, 5]])]
```

```
In [50]: m = 5
         n = 4
         np.tile(x, (m, n))
```

```
Out[50]: array([[59, 60, 59, 60, 59, 60, 59, 60],
                [61, 62, 61, 62, 61, 62, 61, 62],
                [63, 64, 63, 64, 63, 64, 63, 64],
                [65, 66, 65, 66, 65, 66, 65, 66],
                [67, 68, 67, 68, 67, 68, 67, 68],
                [59, 60, 59, 60, 59, 60, 59, 60],
                [61, 62, 61, 62, 61, 62, 61, 62],
                [63, 64, 63, 64, 63, 64, 63, 64],
                [65, 66, 65, 66, 65, 66, 65, 66],
                [67, 68, 67, 68, 67, 68, 67, 68],
                [59, 60, 59, 60, 59, 60, 59, 60],
                [61, 62, 61, 62, 61, 62, 61, 62],
                [63, 64, 63, 64, 63, 64, 63, 64],
                [65, 66, 65, 66, 65, 66, 65, 66],
                [67, 68, 67, 68, 67, 68, 67, 68],
                [59, 60, 59, 60, 59, 60, 59, 60],
                [61, 62, 61, 62, 61, 62, 61, 62],
                [63, 64, 63, 64, 63, 64, 63, 64],
                [65, 66, 65, 66, 65, 66, 65, 66],
                [67, 68, 67, 68, 67, 68, 67, 68],
                [59, 60, 59, 60, 59, 60, 59, 60],
                [61, 62, 61, 62, 61, 62, 61, 62],
                [63, 64, 63, 64, 63, 64, 63, 64],
                [65, 66, 65, 66, 65, 66, 65, 66],
                [67, 68, 67, 68, 67, 68, 67, 68]])
```

```
In [51]: p = np.arange(10).reshape((5, 2))
         q = np.arange(10, 20).reshape((5,2))
         np.concatenate((p, q),1)

Out[51]: array([[ 0,  1, 10, 11],
                [ 2,  3, 12, 13],
                [ 4,  5, 14, 15],
                [ 6,  7, 16, 17],
                [ 8,  9, 18, 19]])

In [52]: np.concatenate((p,q))

Out[52]: array([[ 0,  1],
                [ 2,  3],
                [ 4,  5],
                [ 6,  7],
                [ 8,  9],
                [10, 11],
                [12, 13],
                [14, 15],
                [16, 17],
                [18, 19]])

In [53]: p.max()

Out[53]: 9

In [54]: p.max(0)

Out[54]: array([8, 9])

In [55]: p.max(1)

Out[55]: array([1, 3, 5, 7, 9])

In [56]: np.maximum(p, q)

Out[56]: array([[10, 11],
                [12, 13],
                [14, 15],
                [16, 17],
                [18, 19]])

In [57]: np.linalg.norm(p)

Out[57]: 16.881943016134134

In [58]: np.logical_and(p,q)
```

```
Out[58]: array([[False,  True],
                [ True,  True],
                [ True,  True],
                [ True,  True],
                [ True,  True]])

In [59]: np.logical_or(p,q)

Out[59]: array([[ True,  True],
                [ True,  True],
                [ True,  True],
                [ True,  True],
                [ True,  True]])

In [60]: p & q

Out[60]: array([[0, 1],
                [0, 1],
                [4, 5],
                [0, 1],
                [0, 1]])

In [61]: p | q

Out[61]: array([[10, 11],
                [14, 15],
                [14, 15],
                [22, 23],
                [26, 27]])

In [62]: sq = 16*np.eye(3)
         np.linalg.inv(sq)

Out[62]: array([[0.0625, 0.    , 0.    ],
                [0.    , 0.0625, 0.    ],
                [0.    , 0.    , 0.0625]])

In [63]: np.linalg.pinv(sq)

Out[63]: array([[0.0625, 0.    , 0.    ],
                [0.    , 0.0625, 0.    ],
                [0.    , 0.    , 0.0625]])

In [64]: np.linalg.matrix_rank(sq)

Out[64]: 3

In [65]: fl = np.arange(3)
         np.linalg.solve(sq, fl)
```

```
Out[65]: array([0.    , 0.0625, 0.125 ])

In [66]: U, S, Vh = np.linalg.svd(sq)
         V = Vh.T
         V

Out[66]: array([[1., 0., 0.],
                [0., 1., 0.],
                [0., 0., 1.]])

In [67]: np.linalg.cholesky(sq).T

Out[67]: array([[4., 0., 0.],
                [0., 4., 0.],
                [0., 0., 4.]])

In [68]: D,V = np.linalg.eig(sq)
         D

Out[68]: array([16., 16., 16.])

In [69]: V

Out[69]: array([[1., 0., 0.],
                [0., 1., 0.],
                [0., 0., 1.]])

In [70]: D,V = scipy.linalg.eig(a,b)
         D

Out[70]: array([ 1.01131057e-16+2.59174838e-09j,  1.01131057e-16-2.59174838e-09j,
                -1.78123079e-15+0.00000000e+00j, -5.35875994e-16+0.00000000e+00j,
                 1.09135681e-18+0.00000000e+00j])

In [71]: V

Out[71]: array([[-8.48026509e-04-1.35277096e-09j, -8.48026509e-04+1.35277096e-09j,
                 -5.12327641e-01+0.00000000e+00j, -8.60480647e-01+0.00000000e+00j,
                  6.58814331e-01+0.00000000e+00j],
                [ 7.66975099e-01+1.81020068e-11j,  7.66975099e-01-1.81020068e-11j,
                  4.47175965e-01+0.00000000e+00j,  2.46504433e-01+0.00000000e+00j,
                  6.61626047e-03+0.00000000e+00j],
                [-4.86413996e-02-1.14802547e-12j, -4.86413996e-02+1.14802547e-12j,
                 -5.17386229e-01+0.00000000e+00j,  4.02466298e-01+0.00000000e+00j,
                 -5.58275019e-03+0.00000000e+00j],
                [-6.34414422e-01-1.49733338e-11j, -6.34414422e-01+1.49733338e-11j,
                  5.15088444e-01+0.00000000e+00j,  1.90788119e-01+0.00000000e+00j,
                  8.73224981e-02+0.00000000e+00j],
                [-8.30712512e-02-1.96063257e-12j, -8.30712512e-02+1.96063257e-12j,
                  6.74494613e-02+0.00000000e+00j,  2.07217977e-02+0.00000000e+00j,
                 -7.47170340e-01+0.00000000e+00j]])
```

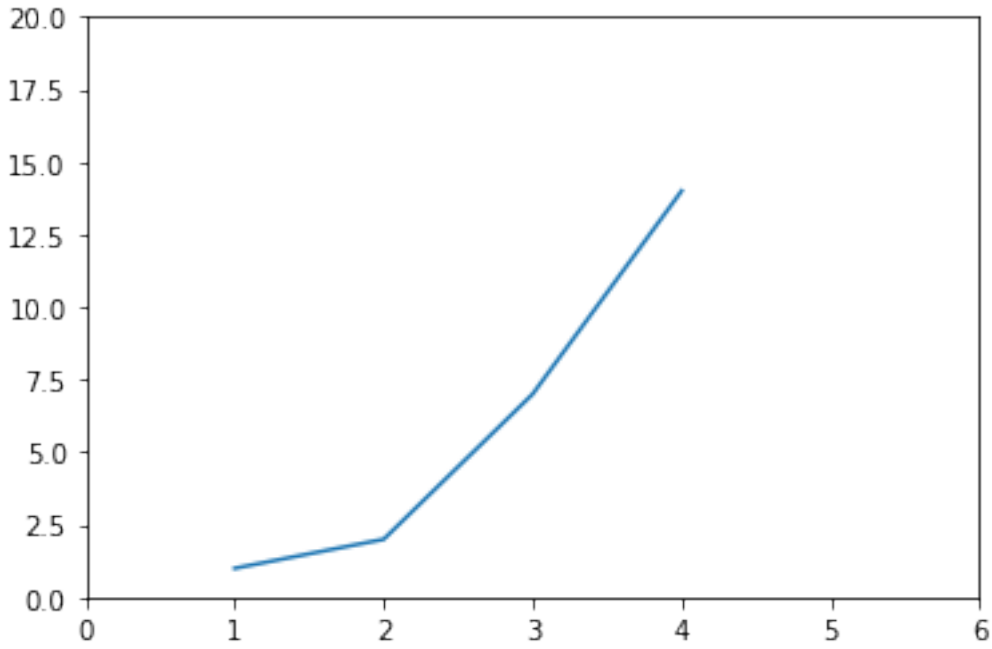
```
In [72]: Q,R = scipy.linalg.qr(sq)
         Q

Out[72]: array([[ 1.,  0.,  0.],
                [-0.,  1.,  0.],
                [-0., -0.,  1.]])

In [73]: R

Out[73]: array([[16.,  0.,  0.],
                [ 0., 16.,  0.],
                [ 0.,  0., 16.]])

In [74]: LU,P=scipy.linalg.lu_factor(sq)
         LU

Out[74]: array([[16.,  0.,  0.],
                [ 0., 16.,  0.],
                [ 0.,  0., 16.]])

In [75]: P

Out[75]: array([0, 1, 2], dtype=int32)

In [76]: scipy.fft(sq)

Out[76]: array([[16. +0.j        , 16. -0.j        , 16. +0.j        ],
                [16. +0.j        , -8.-13.85640646j, -8.+13.85640646j],
                [16. +0.j        , -8.+13.85640646j, -8.-13.85640646j]])

In [77]: scipy.ifft(a)

Out[77]: array([[3.00000000e+00+0.j, 1.33226763e-16+0.j, 1.33226763e-16+0.j,
                 1.33226763e-16+0.j, 1.33226763e-16+0.j],
                [3.00000000e+00+0.j, 1.33226763e-16+0.j, 1.33226763e-16+0.j,
                 1.33226763e-16+0.j, 1.33226763e-16+0.j],
                [3.00000000e+00+0.j, 1.33226763e-16+0.j, 1.33226763e-16+0.j,
                 1.33226763e-16+0.j, 1.33226763e-16+0.j],
                [3.00000000e+00+0.j, 1.33226763e-16+0.j, 1.33226763e-16+0.j,
                 1.33226763e-16+0.j, 1.33226763e-16+0.j],
                [3.00000000e+00+0.j, 1.33226763e-16+0.j, 1.33226763e-16+0.j,
                 1.33226763e-16+0.j, 1.33226763e-16+0.j]])

In [78]: np.sort(sq)

Out[78]: array([[ 0.,  0., 16.],
                [ 0.,  0., 16.],
                [ 0.,  0., 16.]])

In [79]: i = 1
         I = np.argsort(sq[:,i])
         b=sq[I,:]
```

```
In [80]: b

Out[80]: array([[16.,  0.,  0.],
                [ 0.,  0., 16.],
                [ 0., 16.,  0.]])

In [81]: X = np.arange(1000).reshape(50, 20)
         y = np.arange(50)
         np.linalg.lstsq(X,y)

/usr/local/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: FutureWarning: `rcond`
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using
  This is separate from the ipykernel package so we can avoid doing imports until


Out[81]: (array([ 9.28571429e-03,  8.57142857e-03,  7.85714286e-03,  7.14285714e-03,
                  6.42857143e-03,  5.71428571e-03,  5.00000000e-03,  4.28571429e-03,
                  3.57142857e-03,  2.85714286e-03,  2.14285714e-03,  1.42857143e-03,
                  7.14285714e-04,  2.04386817e-17, -7.14285714e-04, -1.42857143e-03,
                 -2.14285714e-03, -2.85714286e-03, -3.57142857e-03, -4.28571429e-03]),
          array([], dtype=float64),
          2,
          array([1.82434969e+04, 9.12242369e+01, 8.69840203e-13, 1.46028724e-13,
                 1.40783146e-14, 3.73747416e-15, 3.24443681e-15, 2.90098324e-15,
                 2.52006962e-15, 2.01629092e-15, 1.75606870e-15, 1.41380078e-15,
                 1.28460017e-15, 9.82697478e-16, 8.42624773e-16, 7.47495524e-16,
                 6.69994431e-16, 4.96122002e-16, 2.98743541e-16, 2.76947030e-16]))

In [82]: np.unique(sq)

Out[82]: array([ 0., 16.])

In [83]: sq.squeeze()

Out[83]: array([[16.,  0.,  0.],
                [ 0., 16.,  0.],
                [ 0.,  0., 16.]])
```

Plotting (MatPlotLib/PyPlot)

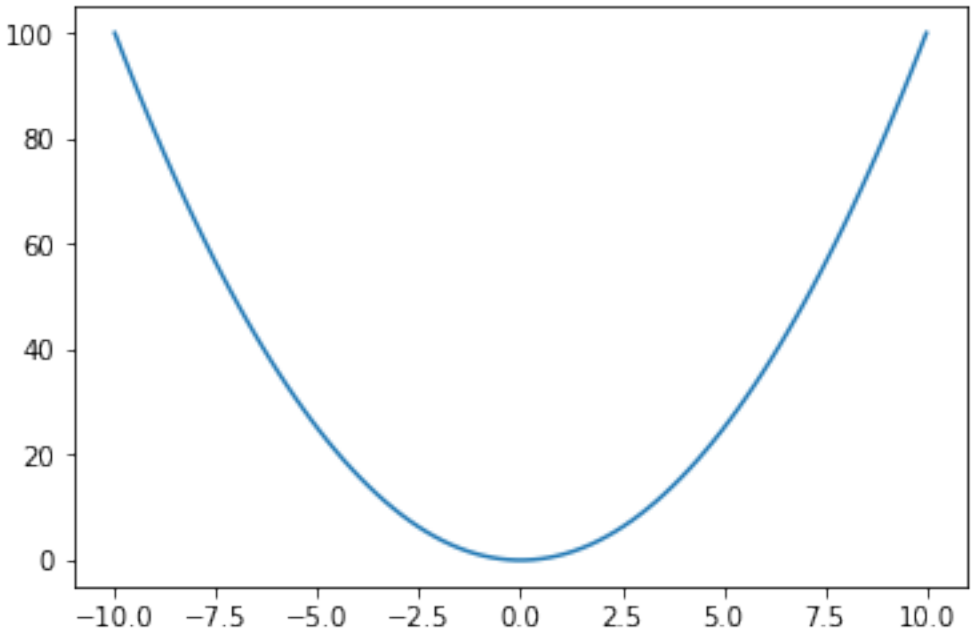
0.1 Provided plotting code

```
In [87]: import matplotlib.pyplot as plt
         plt.plot([1,2,3,4], [1,2,7,14])
         plt.axis([0, 6, 0, 20])
         plt.show()
```



In [85]:
```
## Simple self plot
```

In [86]:
```
x = np.linspace(-10, 10, 100)
y = x**2
plt.plot(x, y)
plt.show()
```



Version Control System (Github)

My github username is eshvarc.

This is my github profile page.
(Full link - https://github.com/eshvarc)

Integrated Development Environment (PyCharm)

My github repository can be found here
(Full link - https://github.com/eshvarc/IntroDeepLearning

Assignment 0 can be found here
(Full link - https://github.com/eshvarc/IntroDeepLearning/tree/master/Assignment0)

Assignment 0 contains the following relevant files:

1. Assignment0.ipynb - The jupyter notebook that corresponds to assignment 0. It contains the conda info stub, all the numpy and matplotlib code and the details of my Version control and IDE.
2. Assignment0.pdf - The PDF version of the above notebook.
3. AssignmentZeroDescription.pdf - The provided PDF file that contains the assignment description and tasks.
4. HelloWorld.py - A simple python script that prints "Hello World!". This was made to test the IDE.