hw4

September 21, 2018

1 Week 4 Homework

Minimize each of the following functions using an appropriate routine from **scipy.optimize**. Provide a printout of your code, along with the solution.

```
In [1]: from scipy.optimize import minimize
        from scipy.optimize import minimize_scalar
```

1.1 Question 1

$f(x_1, x_2, x_3) = |x_1^2 3x_3 + 4| + (x_2 3)^2$ starting from the point $(0, 0, 0)$.

This is a multivarable scaler function therefore we will use the **minimize** function.

```
In [2]: def f(x):
            return abs(x[0]**2 - 3.*x[2] + 4.) + (x[1] - 3.)**2

        min_val = minimize(f, x0=[0,0,0], method='Nelder-Mead')
        print(min_val)

 final_simplex: (array([[-0.45756728,  3.00002551,  1.4031226 ],
       [-0.45750694,  2.99995972,  1.4031042 ],
       [-0.45749527,  3.00000023,  1.40310064],
       [-0.45748499,  2.99997449,  1.40309751]]), array([1.66963175e-09, 5.40377641e-09, 6.69364
           fun: 1.6696317548007262e-09
       message: 'Optimization terminated successfully.'
          nfev: 413
           nit: 227
        status: 0
       success: True
             x: array([-0.45756728,  3.00002551,  1.4031226 ])
```

The output indicates an optimum at $x_1 = 0, x_2 = 3, x_3 = \frac{4}{3}$. Here, the objective function has a value of **approximiately 0**.

1.2 Question 2

$f(x) = (x1)(x)(x + 3) + x^4$ over the interval $[10, 10]$.

This is a univariate scalar function therefore we'll use the **minimize_scalar** function.

```
In [3]: def f(x):
            return (x - 1.)*x*(x + 3.) + x**4

        min_val = minimize_scalar(f, bounds=(-10.,10.), method="Golden")
        print(min_val)

    fun: -0.8156731036447341
   nfev: 45
    nit: 40
success: True
      x: 0.47441768985981336
```

The output indicates an optimum at x = **0.4744** with objective value **-.81567**.

1.3 Question 3

Let $A = \begin{bmatrix} 2 & 3 \\ 4 & 1 \end{bmatrix}$ and $b = \begin{bmatrix} 1 \\ 9 \end{bmatrix}$. Then $f(x) = ||Axb||$ where $x \in \mathbf{R}^2$. Start from $(0, 0)$.

We can first treat this like a least-squares problem and use **nnls** to solve.

```
In [4]: from scipy.optimize import nnls
        A = [[2, -3], [4,1]]
        b = [1, 9]

        min_val = nnls(A, b)

        print(min_val)

(array([2., 1.]), 0.0)
```

The output indicates an optimal solution at $[2, 1]$ with value 0.

Alternatively, we can treat this as a minimization of the length of a vector with 2 components y_1 and y_2 with each being compsed of 2 variables x_1 and x_2. Expanding $f(x) = ||Ax - b||$ yields an objective function of $\sqrt{y_1^2 + y_2^2}$ with $y_1 = 2x_1 - 3x_2 - 1$ and $y_2 = 4x_1 + x_2 - 9$

```
In [5]: from math import sqrt
        def f(x):
            return sqrt((2.*x[0] - 3.*x[1] - 1.)**2 + (4.*x[0] + x[1] - 9.)**2)

        min_val = minimize(f, x0=[0,0], method='Nelder-Mead')
        print(min_val)
```

```
 final_simplex: (array([[2.00000833, 1.00002649],
       [1.99997718, 1.00000875],
       [2.00000648, 0.99996275]]), array([8.67449863e-05, 1.09457457e-04, 1.25229896e-04]))
           fun: 8.674498631302434e-05
       message: 'Optimization terminated successfully.'
          nfev: 129
           nit: 68
        status: 0
       success: True
             x: array([2.00000833, 1.00002649])
```

The output indicates an optimum at $x_1 = 2$ and $x_2 = 1$ with value 0. This is the same value achieved using the least squares solver.