Homework 3

Exercises 3.4, 3.5(a-c), and 3.6, 3.7, 3.8, 3.9 from the Convex Optimization Course Notes.

Some of the exercises involve coding. Make sure to upload a jupyter notebook (with explanations) on these problems.

Links:

https://sites.math.duke.edu/courses/mth390/convexOptimization.html#exr-lagrangian

https://sites.math.duke.edu/courses/mth390/convexOptimization.html#exr-lagrangian-multiple-constraints

https://sites.math.duke.edu/courses/mth390/convexOptimization.html#exr-kkt

https://sites.math.duke.edu/courses/mth390/convexOptimization.html#exr-newton

https://sites.math.duke.edu/courses/mth390/convexOptimization.html#exr-newton-equality-const

https://sites.math.duke.edu/courses/mth390/convexOptimization.html#exr-newton-equality-const-implement

> ⓘ **Exercise 3.4**
>
> **a:** Determine the Lagrangian function for the above minimization problem with $f(x, y) = x^2 - 2y^2 - 4$ with $g(x, y) = x^2 + y - 4$.
>
> **b:** Set up the resulting system of equations
>
> **c:** Solve the system

3.4) a) $L(x,y,\lambda) = x^2 - 2y^2 + \lambda(x^2 + y)$

b) $L_x = 2x + 2\lambda x = 0$

$L_y = -4y + \lambda = 0$

$L_\lambda = x^2 + y = 0$

c) $y = -x^2$, $y = \frac{\lambda}{4}$ so $\lambda = 4y = -4x^2$

$2x - 8x^3 = 0$

$\therefore 8x^3 = 2x$

$4x^2 = 1 \rightarrow x^2 = \frac{1}{4} \quad \therefore x = \pm\frac{1}{2}$

$\therefore x = -\frac{1}{2}$ and $y = -1/4$ and $\lambda = -1$

3.5) a) $L(x,y,\lambda_1,\lambda_2) = x^2 - 2y^2 + \lambda_1(x^2 + y^2 - 4) + \lambda_2((x-2)^2 + y^2 - 4)$

b) $L_x = 2x + 2\lambda_1 x + 2\lambda_2 x - 4\lambda_2 = 0$

$L_y = -4y + 2\lambda_1 y + 2\lambda_2 y = 0$

$L_{\lambda_1} = \lambda^2 + y^2 - 4 = 0$

$L_{\lambda_2} = (x-2)^2 + y^2 - 4 = 0$

c) $(x-2)^2 + y^2 - 4 = x^2 - 4x + 4 + y^2 - 4$

$-4 + 4x = 0 \quad \therefore x = 1$

$y^3 - 3 = 0 \quad \therefore \quad y = \pm\sqrt{3}$

$2 + 2\lambda_1 - 2\lambda_2 = 0$

$\lambda_1 - \lambda_2 = -1$ and $\lambda_2 = \lambda_1 + 1$

$\therefore \quad -4(\sqrt{3}) + 2\lambda_1(\sqrt{3}) + 2(\lambda_1 + 1)\sqrt{3}) = 0$

$-4\sqrt{3} + 4\lambda_1\sqrt{3} + 2\sqrt{3} = 0$

$4\lambda_1\sqrt{3} = 2\sqrt{3}$

$\lambda_1 = \frac{1}{2}, \lambda_2 = 3/2$

These values for λ_1, λ_2 work in $2 + 2\lambda_1 - 2\lambda_2 = 0$. This is why we know that $x = 1, y = \sqrt{3}, \lambda_1 = 1/2, \lambda_2 = 3/2$

Exercise 3.6

a: Determine the KKT conditions function for the minimization problem $f(x, y) = x^2 + y^2$ with $g(x, y) = x + y - 1 \leq 0$.

b: Solve the resulting problem. Use the complementary slackness condition to consider both possible cases (i.e. $\lambda = 0$ or $\lambda < 0$) and determine which case contains the minimum.

c: Repeat the above two steps if we "flip" the inequality constraint so that we have the minimization problem $f(x, y) = x^2 + y^2$ with $g(x, y) = -x - y + 1 \leq 0$.

3.6) a) $L(x,y,\lambda) = x^2+y^2+\lambda(x+y-1)$

$$\text{KKT}\begin{cases} L_x = 2x+\lambda=0 \\ L_y = 2y+\lambda=0 \\ x+y-1\geq 0 \\ x\leq 0 \\ \lambda(x+y-1)=0 \end{cases}$$

b) If $\lambda=0$ and $2x=2y$. but $x+y-1\geq 0$, then we know that it is false.

If $\lambda<0$, then $x+y-1=0$ and $y=1-x$ and $2x+\lambda=2-2x+\lambda$ and $4x=2$ so $x=\frac{1}{2}$ and $y=\frac{1}{2}$, which means that this is the minimum!

c) if $\lambda=0$, then $2x=2y=0$ and $-x-y+1\leq 0$ is still false.

if $\lambda<0$, then $-x-y+1=0$ and $y=1-x$ and this is the same result as before!

ⓘ **Exercise 3.7**

a: Rederive Newton's method in 1 demension (i.e. $x \in \mathbb{R}$) in the context of optimization (i.e. find $\min f(x)$ for some smooth and convex function f).

b: Implement (i.e. code) steepest descent and Newton's method for finding the minimum of the function $f(x, y) = x^2 + y^2 + 4y + 2x + 5$. Start at the point $(x, y) = (1, 1)$ and use a stopping criteria of $||\nabla f||_2 < 10^{-6}$. For steepest descent, use backtracking line search to determine α. How many iterations does each method take to converge? What is the final answer? Do you get the same answer? What could explain the speed of Newton's method here?[9]

c: Repeat the previous problem for $f(x, y) = (x - 1)^4 + (y + 2)^4$.

Note: Performing Newton steps requires one to either invert a matrix or solve the system of equations $\nabla^2 f(x^{(k)})u = -\nabla f(x^{(k)})$ for u There are a number of neat algorithms to do this efficiently including LU (but better yet QR) factorizations. We will not go through these methods in this course, but, if you're interested, here is one of my favorite text books (of all books) that happens to focus on Numerical Linear Algebra. For the purposes of this course you can just use simple black-boxed python calls such as `numpy.linalg.solve`. Do this instead of directly computing the inverse (it is better to avoid direct computation!)

a) → Newton's method for optimization in 1D:

goal: minimize a smooth, convex function $f: \mathbb{R} \to \mathbb{R}$

→ first-order optimally condition: the minimizer x^* satisfies $f'(x^*) = 0$

Then, our optimization reduces to finding the root of $f'(x)$.

We know that Newton's method of root finding for a general scalar function $g(x)$ is: $x_{k+1} = x_k - \dfrac{g(x_k)}{g'(x_k)}$

In our case, $f'(x) = g(x)$ so we know: $x_{k+1} = x_k - \dfrac{f'(x_k)}{f''(x_k)}$

b: Implement (i.e. code) steepest descent and Newton's method for finding the minimum of the function $f(x, y) = x^2 + y^2 + 4y + 2x + 5$. Start at the point $(x, y) = (1, 1)$ and use a stopping criteria of $||\nabla f||_2 < 10^{-6}$. For steepest descent, use backtracking line search to determine α. How many iterations does each method take to converge? What is the final answer? Do you get the same answer? What could explain the speed of Newton's method here?[9]

b) exact function:

$$f(x,y) = x^2 + y^2 + 4y + 2x + 5 \quad \text{and} \quad (x_0, y_0) = (1,1)$$

and stopping when $\|\nabla f\|_2 < 10^{-6}$

our code is also accessible @ https://github.com/helloames/durhamfunding

```python
import numpy as np

# define function, gradient, and Hessian
def f(x):
    # x is a 2D numpy array: [x, y]
    return x[0]**2 + x[1]**2 + 4*x[1] + 2*x[0] + 5.0

def grad_f(x):
    # gradient: [2x + 2, 2y + 4]
    return np.array([2*x[0] + 2, 2*x[1] + 4], dtype=float)

def hess_f(x):
    # hessian is constant: 2I
    return np.array([[2.0, 0.0],
                     [0.0, 2.0]])

# backtracking line search
def backtracking_step(x, g, f, grad_norm_sq, alpha=0.3, beta=0.8):
    t = 1.0
    fx = f(x)
    while True:
        x_new = x - t * g
        if f(x_new) <= fx - alpha * t * grad_norm_sq:
            return t
        t *= beta
        if t < 1e-16:  # safeguard against infinite loop
            return t

# steepest descent with backtracking
def steepest_descent(x0, tol=1e-6, max_iters=100000):
    x = x0.copy().astype(float)
    iters = 0
    history = [x.copy()]
    while True:
        g = grad_f(x)
        gn = np.linalg.norm(g)
        if gn < tol or iters >= max_iters:
            break
        grad_norm_sq = np.dot(g, g)
        t = backtracking_step(x, g, f, grad_norm_sq)
        x = x - t * g
        history.append(x.copy())
        iters += 1
    return x, iters, history

# Newton's method
def newton_method(x0, tol=1e-6, max_iters=1000):
    x = x0.copy().astype(float)
    iters = 0
    history = [x.copy()]
    while True:
        g = grad_f(x)
        gn = np.linalg.norm(g)
        if gn < tol or iters >= max_iters:
            break
        H = hess_f(x)
        p = np.linalg.solve(H, g)  # solve H p = g
        x = x - p
        history.append(x.copy())
        iters += 1
    return x, iters, history

# initial point
x0 = np.array([1.0, 1.0])

# run methods
x_sd, iters_sd, hist_sd = steepest_descent(x0, tol=1e-6)
x_newton, iters_newton, hist_newton = newton_method(x0, tol=1e-6)

# print results
print("Steepest descent result:")
print("  Minimizer:", x_sd)
print("  Iterations:", iters_sd)
print("  f(x):", f(x_sd))

print("\nNewton's method result:")
print("  Minimizer:", x_newton)
print("  Iterations:", iters_newton)
print("  f(x):", f(x_newton))
```

[1] ✓ 0.1s

output from code:

```
Steepest descent result:
  Minimizer: [-1.00000013 -2.0000002 ]
  Iterations: 13
  f(x): 5.5067062021407764e-14

Newton's method result:
  Minimizer: [-1. -2.]
  Iterations: 1
  f(x): 0.0
```

final function value

Both methods converge to the same minimizer of $(-1,-2)$ and the reason why Newton is so fast here is because the function is quadratic with a constant Hessian $H = 2I$. Also, for a quadratic objective, the Newton update of $x_{k+1} = x_k - H^{-1}\nabla f(x_k)$ solves the linear system that yields the global minimizer in one step $\therefore$ Newton converged in one iteration.

We got the same answer both ways.

c: Repeat the previous problem for $f(x,y) = (x-1)^4 + (y+2)^4$.

Note: Performing Newton steps requires one to either invert a matrix or solve the system of equations $\nabla^2 f(x^{(k)})u = -\nabla f(x^{(k)})$ for u There are a number of neat algorithms to do this efficiently including LU (but better yet QR) factorizations. We will not go through these methods in this course, but, if you're interested, here is one of my favorite text books (of all books) that happens to focus on Numerical Linear Algebra. For the purposes of this course you can just use simple black-boxed python calls such as `numpy.linalg.solve`. Do this instead of directly computing the inverse (it is better to avoid direct computation!)

→ same code was used as for the previous problem

output from code:

```
Steepest descent result:
  Minimizer: [ 1.          -1.99370055]
  Iterations: 3135
  f(x): 1.574743116617912e-09

Newton's method result:
  Minimizer: [ 1.          -1.99543268]
  Iterations: 16
  f(x): 4.3515546222979796e-10
```

final answers are very similar and close to zero

steepest descent iterations: ~3,135

Newton's method iterations: ~16

Both methods converge to (1, -2) with some margin of tollerance and to a function of zero.

Newton's method is much faster because it uses second-order curvature information while steepest descent struggles in flat regions of the quartic function.

ⓘ **Exercise 3.8**

Show that the above system of equations is equivalent to applying a Newton's step, i.e. for each step we solve

$$\begin{aligned} \min\ & f(x) + \nabla f(x) \cdot u + \frac{1}{2} u \nabla^2 f(x) u \\ \text{subject to} \quad & A(x+u) = b. \end{aligned}$$

where $Ax = b$. *Note: Don't over think this as I'm not asking for anything too complicated: mainly I'm asking you to put this in the context of finding a critical point in terms of ∇_u as was done above and then show that you understand the matrix formulation.*

goal: $\min\ \nabla f(x)^T u + \frac{1}{2} u^T \nabla^2 f(x) u$ st. $A(x+u)=b$

because $Ax=b$, then we know $Au=0$.

$$L(u,\lambda) = g^T u + \frac{1}{2} u^T H u + \lambda^T (Au)$$

we know that a first-order critical point (u,λ) must satisfy the KKT conditions:

$$\partial_u L = g + Hu + A^T\lambda = 0$$

$$\partial_\lambda L = Au = 0$$

as a linear system: $\begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} u \\ \lambda \end{pmatrix} = \begin{pmatrix} -g \\ 0 \end{pmatrix}$

The constrained minimization of the quadratic Taylor Expansion is equivalent to solving the KKT linear system, which is the Newton step under the linear equality constraint $A(x+u)=b$.

ⓘ **Exercise 3.9**

a: Implement Newton's method (i.e. write code) to minimize $f(x,y) = (x-1)^4 + (y+2)^4$ subject to the constraint $g(x,y) = x + y - 1 = 0$. Start at the point $(x,y) = (0,1)$ and use a stopping criteria of $||\nabla f||_2 < 10^{-6}$. How many iterations does it take to converge? What is the final answer?

b: Determine y in terms of x in the constraint and plug it into f to create a new function $\tilde{f}(x)$ which is equivalent to f along the constraint. Minimize $\tilde{f}$ analytically and check your answer from part (a).

a) $\varphi(x) = f(x, 1-x) = (x-1)^4 + (3-x)^4$

because $(1-x)+2 = 3-x$ and we have the constraint affine of $x+y=1$ and $\therefore\ y = 1-x$

$$\varphi'(x) = 4(x-1)^3 - 4(3-x)^3 = 4\left[(x-1)^3 - (3-x)^3\right] = 0$$

$$\therefore (x-1)^3 = (3-x)^3 \rightarrow x-1 = 3-x \quad \therefore 2x = 4 \rightarrow x = 2$$

then, we know that $y = 1-x = -1$ and the constrained minimizer (analytic) is: $(x^*, y^*) = (2,-1)$

when we check the 2nd derivative, we see:

$$\varphi''(x) = 12(x-1)^2 + 12(3-x)^2 > 0$$

so we know for sure that this is indeed a local minimum for the constrained problem.

We can also evaluate ∇f at $(2,-1)$: $\nabla f(2,-1) = \begin{bmatrix} 4(2-1)^3 \\ 4(-1+2)^3 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$

This is collinear with the constraint gradient $\nabla g = [1\ 1]^T$ and indeed $\nabla f = 4\nabla g$ so we know that it holds with multiplier $\lambda = 4$. This verifies the solution.

b: Determine y in terms of x in the constraint and plug it into f to create a new function $\tilde{f}(x)$ which is equivalent to f along the constraint. Minimize $\tilde{f}$ analytically and check your answer from part (a).

b)

saddle system: $\begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix}\begin{pmatrix} u \\ \lambda \end{pmatrix} = \begin{pmatrix} -\nabla f(x,y) \\ 0 \end{pmatrix}$ with $A = (1 \quad 1)$

$$\nabla f(x,y) = \begin{pmatrix} 4(x-1)^3 \\ 4(y-2)^3 \end{pmatrix}, \quad H = \begin{pmatrix} 12(x-1)^2 & 0 \\ 0 & 12(y+2)^2 \end{pmatrix}$$

If we start from feasible point $(0,1)$ which satisfies $0+1=1$, we can compute the first Newton step by hand.

@ $(x,y) = (0,1)$

$\nabla f = [-4, 108]^T$, $H = \text{diag}(12, 108)$

and then we solve $\begin{cases} 12u_1 + \lambda = 4 \\ 108u_2 + \lambda = -108 \\ u_1 + u_2 = 0 \end{cases}$

from this, we get: $u_1 = \frac{14}{15}$ and $u_2 = \frac{-14}{15}$

and $\lambda = -7.2$

new iterate: $(x_1, y_1) = (0,1) + (\frac{14}{15}, -\frac{14}{15}) \approx (.9333, .06777)$

If we repeat the Newton steps, then we get to produce a fast and quadratic convergence to $(2,-1)$. Newton's method with constraints will converge rapidly (with few iterations) to the analytic solution above. The analytic reduction (eliminating y) gave the exact minimizer $(2,-1)$ and matches our answer to a).