02610 Optimization and Data Fitting

Exercises for Week 3

1 Exact solver for trust-region subproblem

Consider

$$f(\boldsymbol{x}) = f(x_1, x_2) = 10(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

1. Compute the gradient and Hessian of $f(\boldsymbol{x})$ by hand.

2. In Python, at $\boldsymbol{x} = [0, -1]^T$ draw the contour lines of the quadratic model

$$m(\boldsymbol{p}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T \boldsymbol{p} + \frac{1}{2} \boldsymbol{p}^T \nabla^2 f(\boldsymbol{x}) \boldsymbol{p}$$

   with respect to $\boldsymbol{p}$.

3. (By hand) At $\boldsymbol{x} = [0, -1]^T$, consider the spectral decomposition of $\nabla^2 f(\boldsymbol{x}) = Q \Lambda Q^T$ with an orthogonal matrix $Q = [q_1, q_2]$ and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2)$, where $\lambda_1 \leq \lambda_2$ are the eigenvalues of $\nabla^2 f(\boldsymbol{x})$. What are Q and Λ?

4. At $\boldsymbol{x} = [0, -1]^T$, referring to Chapter 4.3 or the slides, implement the exact solver of

$$\min_{\boldsymbol{p} \in \mathbb{R}^2} m(\boldsymbol{p}) \quad \text{s. t. } \|\boldsymbol{p}\|_2 \leq \Delta.$$

   Do we need consider the hard case?

   To implement the exact solver, you can use the following template, which can be also found in DTU Learn:

```
p_star =         # TODO - compute the full step -B\g.
norm_p_star = numpy.linalg.norm(p_star)

min_p = []   # It's a vector to save all p.

for delta in np.arange(0.1, 2.1, 0.1):
    if      # TODO  -- First case
        p =   # To be completed
        min_p.append(p)
    else:         # TODO  -- Second case
        # ================================================
        # TODO -- Easy case
        #  We use Newton's method to find the root of phi2

        lambda = 20  # The starting point for Newton
                     # It need sufficiently large to ensure
                     # B+\lambda I positive definite.
```

```
R = scipy.linalg.cho_factor(# To be completed #)
pl = -scipy.linalg.cho_solve(R, df)
norm_pl = numpy.linalg.norm(pl)
phi2 = 1/delta-1/norm_pl

# Loop in Newton's method
while phi2 > 1e-8:
    # TODO -- Newton iterations


p = pl
min_p.append(p)

# ================================================
# TODO -- Hard case
# But do we have a hard case in this test problem?
# If not, you can ignore this part.
```

5. Execute your exact solver to find the solutions $\boldsymbol{p}$ as the trust region radius varies from $\Delta = 0.1$ to $\Delta = 2$. Plot all the solutions of $\boldsymbol{p}$ on the contour plot.

6. Repeat this at $\boldsymbol{x} = [0, 0.5]^T$.

2 Implementation of trust-region methods (in Python)

In this exercise, we will write a Python function `TR_Dogleg` that implements the trust-region method with dogleg, then apply it to solve the minimization problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^2} f(\boldsymbol{x}) = f(x_1, x_2) = \frac{1}{2}x_1^2 + 5x_2^2.$$

1. Compute the gradient and Hessian of $f(\boldsymbol{x})$ by hand. What is the minimizer $\boldsymbol{x}^*$?

2. Download the Python function as `TR_Dogleg` from DTU Learn. The algorithm is implemented according to Algorithm 4.1 in the textbook page 69. Here we set the initial $\Delta_0 = 1$ and the upper bound $\hat{\Delta} = 10$. The solution $\boldsymbol{p}_k$ of the subproblem can be obtained by calling another function `dogleg`, which will be implemented in the next step.

3. Now, we will implement the dogleg method to solve the subproblem. We choose B_k to be the exact Hessian. The dogleg method can be found in the textbook page 73 or in the slides, and you can use the following template, which can be found in DTU Learn:

```
import numpy as np

def dogleg(df, d2f, delta):
'''
Inputs:
df:      the gradient (column vector)
d2f:    the Hessian (matrix)
delta: the trust-region radius

Output:
p:       the approximated minimizer to the subproblem through dogleg.
'''


    # Calculate pu and pb
    pu =  # TODO
    norm_pu = np.linalg.norm(pu)
    pb =  # TODO
    norm_pb = np.linalg.norm(pb)

    # Implement conditionals to determine p
    if norm_pu >= delta:
        p = # TODO
    elif norm_pb <= delta:
        p = # TODO
    else:
        # TODO
        # For intermediate value of delta, solve a quadratic equation on tau.
        # You need use quadratic root formula to find both roots, tau1 and tau2.

        if tau1 <= 2 and tau1 >= 1:
            p = # TODO
        elif tau2 <= 2 and tau2 >= 1:
            p = # TODO

    return p
```

4. Test your implementation with the starting points $\boldsymbol{x}_0 = [10, 10]$ and $\eta = 0.2$. Plot $f(\boldsymbol{x}_k)$, $e_k = \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2$ and $\|\nabla f(x_k)\|_2$ as functions of the iteration number.

5. Experiment with the update rule for the trust region by changing the constants η and the bound for ρ in Algorithm 4.1. What is your observation and conclusion?

3 Double-dogleg method (by hand)

When B is positive definite, the *double-dogleg method* constructs a path with three line segments from the origin to the full step. The four points that define the path are

- the origin;
- the unconstrained Cauchy step $\boldsymbol{p}^C = -(\boldsymbol{g}^T\boldsymbol{g})/(\boldsymbol{g}^T B\boldsymbol{g})\boldsymbol{g}$;
- a fraction of the full step $\bar{\gamma}\boldsymbol{p}^B = -\bar{\gamma}B^{-1}\boldsymbol{g}$, for some $\bar{\gamma} \in (\gamma, 1]$, where $\gamma = \|\boldsymbol{g}\|_2^4/[(\boldsymbol{g}^T B\boldsymbol{g})(\boldsymbol{g}^T B^{-1}\boldsymbol{g})] \leq 1$; and
- the full step $\boldsymbol{p}^B = -B^{-1}\boldsymbol{g}$.

Show that $\|\boldsymbol{p}\|_2$ increases monotonically along this path. (Hint: You would need use the Cauchy-Schwarz inequality, i.e, $|\boldsymbol{u}^T\boldsymbol{v}| \leq \|\boldsymbol{u}\|_2\|\boldsymbol{v}\|_2$.)