Nonmonotone Trust-Region Method for unconstrained optimization

Exploratory Project

Hardik Sharma

22124018

Methods of Unconstrained Optimization:

Line Search Method:

1. Purpose of Line Search:

Line Search optimizes the step size along a specified direction to efficiently find the minimum or maximum of a function.

2. Update Step Equation:

The update step is determined by minimizing a one-dimensional function along the search direction:

Trust Region Method:

1. Local Model Approximation:
   - The method employs a local model (typically a quadratic approximation) to approximate the behavior of the objective function within a trust region.
   - The trust region is a region around the current iterate where the model is expected to be a good approximation of the true function.
2. Iteration Process:
   - The algorithm iteratively updates the iterate within the trust region by solving a subproblem that balances the improvement in the model and the adherence to the trust region constraint.
   - The trust region is adjusted based on the performance of the model; if the model accurately represents the function, the trust region can be expanded, and if not, it may be contracted.

ASSUMPTIONS FOR TRUST REGION ALGORITHM:

Consider the following unconstrained optimization problem

$$\min f(x), \quad \text{subject to } x \in \mathbf{R}^n,$$

1. **Twice Differentiability:** The objective function must be twice continuously differentiable, ensuring well-defined and continuous gradients and Hessians for the quadratic approximation.
2. **Bounded Below:** The objective function should be bounded below, indicating the existence of a minimum value towards which the optimization can converge.
3. **Lipschitz Continuity of the Gradient (Common but not Always Explicit):** The gradient of the objective function should satisfy Lipschitz continuity, meaning the rate of change of the gradient is bounded, enhancing the stability and convergence of the method.

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

Here in this trust region the model function can be trusted to approximate the objective function. Hence it is called the Trust Region Method.

Reduction ratio:

In the trust region algorithm, the reduction ratio is a key metric used to assess the effectiveness of each iteration and to adjust the size of the trust region for future iterations. It is defined as the ratio of the actual reduction in the objective function to the predicted reduction based on the model used within the trust region.

Mathematically, the reduction ratio, often denoted as ρ, is calculated as:

$$\rho = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}$$

Here:

- $f(x_k)$ is the value of the objective function at the current iterate x_k.
- $f(x_k + p_k)$ is the value of the objective function at the proposed new point.
- $m_k(p_k)$ is the value of the model function (usually a quadratic approximation) at the step p_k.
- $m_k(0)$ is the value of the model function at the current iterate.

The reduction ratio is used to decide whether to accept the proposed step and how to update the trust region radius for the next iteration:

- If ρ is close to 1, it indicates that the model accurately predicted the reduction, and the step is likely to be accepted, possibly with an increased trust region radius.
- If ρ is significantly less than 1, it suggests that the model's prediction was overly optimistic, and the step may be rejected or accepted with a reduced trust region radius.

This mechanism ensures that the trust region size adapts dynamically, reflecting the accuracy of the model's predictions and guiding the algorithm towards convergence.

TRUST REGION ALGORITHM:

1. **Initialization:**
   - Choose an initial guess for the solution, x_0.
   - Choose an initial trust region radius, Δ_0, which defines the size of the region within which the model is expected to be accurate.
2. **Iteration:**
   - **Model Construction:**
     - Evaluate the objective function $f(x_k)$ and its gradient g_k at the current iterate x_k.
     - Optionally, approximate the Hessian matrix H_k or compute an approximation to it.
     - Construct a quadratic model $m_k(p) = f(x_k) + g_k^T p + \frac{1}{2} p^T H_k p$.

- **Update the Solution:**
  - Update the solution based on the step p_k if the step is accepted. Otherwise, keep the current solution.

$$x_{k+1} = \begin{cases} x_k + p_k & \text{if } \rho_k > \eta_1 \\ x_k & \text{otherwise} \end{cases}$$

  where η_1 is a chosen threshold (typically close to zero, like 10^{-4}).
- **Update the Trust Region:**
  - Adjust the trust region radius for the next iteration based on the success of the step.

$$\Delta_{k+1} = \begin{cases} \gamma_1 \Delta_k & \text{if } \rho_k < \eta_2 \\ \gamma_2 \Delta_k & \text{if } \rho_k > \eta_3 \text{ and } \|p_k\| = \Delta_k \\ \Delta_k & \text{otherwise} \end{cases}$$

  where η_2, η_3, γ_1, and γ_2 are parameters chosen to control the trust region update.

- **Solving the Trust Region Subproblem:**
  - Solve the trust region subproblem to find a step p_k that minimizes the model $m_k(p)$ within the trust region:

    $\min_p m_k(p) \text{ subject to } \|p\| \leq \Delta_k$

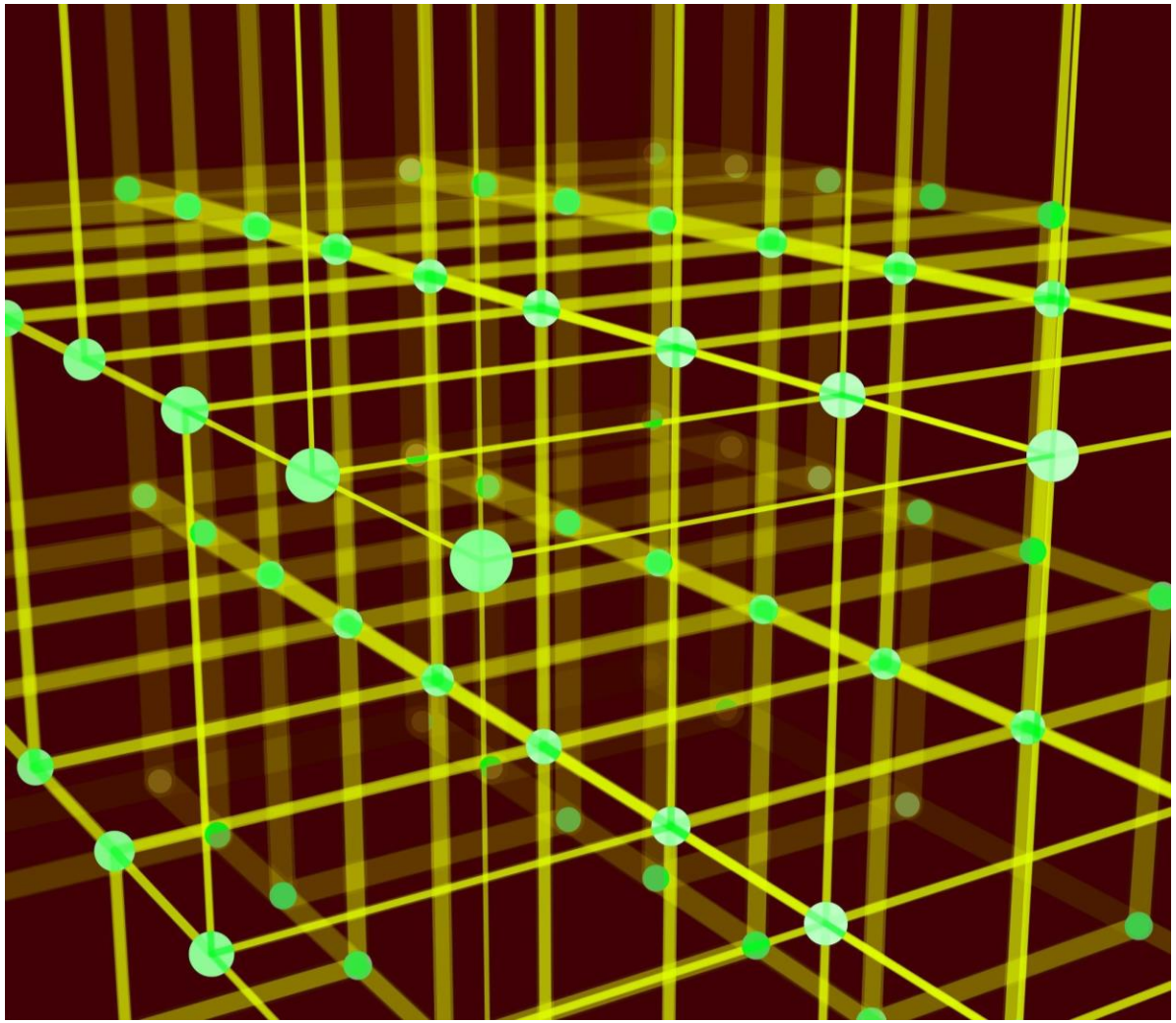
    There are various methods to solve this subproblem, such as the dogleg method, conjugate gradient method, or the more general trust region Newton method.
- **Evaluate the Actual Reduction:**
  - Compute the actual reduction in the objective function:

    $\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}$

    where $m_k(0)$ is the predicted reduction based on the model without actually taking the step.

- **Convergence Check:**
  - Check for convergence based on the change in the objective function or the norm of the gradient being below certain tolerances.
- **Iteration Update:**
  - Increment the iteration counter and return to step 2 if convergence criteria are not met.

3. **Termination:**
   - The algorithm terminates when a stopping criterion is satisfied, such as reaching a maximum number of iterations, achieving a sufficiently small change in the objective function, or the norm of the gradient falling below a specified threshold.



VALIDITY IF THE ALGORITHM:

THE TRUST REGION ALGORITHM is based on the fact there exist a solution of the model function that minimizes it.

The proof of the obve comes from Wierstas Extremum Value Theorem ,
which states if a continuous function is studied in a closed bound region then there exist a minimising factor.

This was our assumption 2.

1. Cauchy Point

The Cauchy point is the point along the steepest descent direction that minimizes a quadratic model of the objective function within the trust region. It is a conservative choice that guarantees a decrease in the function value. The formula for the Cauchy point is:

$$p_C = -\alpha \nabla f(x)$$

where α is the step size that minimizes the quadratic model along the steepest descent direction, $\nabla f(x)$ is the gradient of the function at the current point x.

2. Quasi-Newton Point

The quasi-Newton point is derived using a quasi-Newton approximation to the Hessian of the objective function. It typically involves solving the trust region subproblem using the BFGS or other similar approximations. The formula depends on the specific quasi-Newton method used, but generally, it involves solving:

$$B_k p = -\nabla f(x)$$

where B_k is the quasi-Newton approximation of the Hessian.

3. Dogleg Point

The dogleg method is a two-phase strategy that first moves in the steepest descent direction and then along a curve towards the Newton or quasi-Newton point. The dogleg path is a piecewise linear path consisting of two segments: the first segment is along the steepest descent direction, and the second segment is along the direction from the Cauchy point to the quasi-Newton point.

4. Double Dogleg Point

The double dogleg method extends the dogleg method by adding a third segment. This method is useful when the trust region is large. It includes a step towards an intermediate point between the Cauchy and the quasi-Newton points before proceeding to the quasi-Newton point.

PROBLEM TO BE SOLVED!!!!!

Find the minimum value of the two variable function

F(x,y) = x^4 + y^4 - y^2 – x^2*y

Given

Initial point = (1 2)

Delta max = 1

Initial delta = 0.6

Eta1 = ¼

Eta2= ¾

Shrinking ratio = 0.5

Expanding ratio = 2

Epsilon = 10^-4

TRUST REGION METHOD

Given function

$$f(x_1, x_2) = x_1^4 + x_2^4 - x_2^2 - x_1^2 x_2$$

Given parameters

$x^0 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ $\quad \bar{\Delta} = 1$ $\quad \Delta_0 = 3/5$

$\eta_1 = 1/4$ $\quad \eta_2 = 3/4$ $\quad r_1 = 1/2$ $\quad r_2 = 2$

$\varepsilon = 10^{-1}$

$$g(x_1, x_2) = \begin{vmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{vmatrix} = \begin{vmatrix} 4x_1^3 - 2x_1 x_2 \\ 4x_2^3 - 2x_2 - x_1^2 \end{vmatrix}$$

$$H(x_1, x_2) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 12x_1^2 - 2x_2 & -2x_1 \\ -2x_1 & 12x_2^2 - 2 \end{bmatrix}$$

$g^0 = \begin{bmatrix} 0 \\ 27 \end{bmatrix}$ $\quad \|g^0\| = 27 > \varepsilon$

$H^0 = \begin{bmatrix} 8 & -2 \\ -2 & 46 \end{bmatrix} = B^0$ $\quad$ [Symmetric]

$$\alpha_0^c = \min \left| \frac{\Delta_0}{\|g^0\|}, \frac{\|g^0\|^2}{(g^0)^T B^0 g^0} \right|$$

$$= \min \left| \frac{3}{5 \cdot 27}, \frac{27 \cdot 27}{27 \cdot 27 \cdot 46} \right|$$

$$\alpha_0^c = \min \left\{ \frac{1}{45}, \frac{1}{46} \right\}$$

$$\boxed{\alpha_0^c = \frac{1}{46}}$$

Cauchy point

$$P_k^c = -\alpha_k^c g^k$$

$$P_c^0 = -\alpha_0^c g^0 = -\begin{pmatrix} 0 \\ 0.587 \end{pmatrix}$$

Newton point

$$p_N^0 = -(B^k)^{-1} g^k = -(B^0)^{-1} g^0$$

$$= -\begin{bmatrix} 0.148 \\ 0.593 \end{bmatrix}$$

$$\|p_c^0\| = 0.587 < \Delta_0 < 0.611 = \|p_N^0\|$$

Now for Dogleg Step

we know $p_D^0 = p_c^0 + \tau(p_N^0 - p_c^0)$

$\tau \in [0,1]$

we have

$$\|p_c^0 + \tau(p_N^0 - p_c^0)\| = \Delta_0$$

which gives

$$22\tau^2 + 7\tau - 16 = 0$$

$$\Rightarrow \boxed{\tau = 0.708}$$

$$p_D^0 = p_c^0 + 0.708(p_N^0 - p_c^0)$$

$$= \begin{bmatrix} -0.104 \\ -0.591 \end{bmatrix}$$

$$\rho_0 = \frac{\Delta R}{PR} = \frac{f(x^0) - f(x^0 + p_D^0)}{m_0(0) - m_0(p_D^0)} = \frac{9.504}{7.923} > \eta_2$$

$$m_0(p) = f_0 + g^{0T} p_D^0 + \frac{1}{2} p_D^{0T} B^0 p^T$$

$$= 11 + 27 p_2 + 23 p_2^2$$

as $\rho_0 > \eta_2$ Thus it is a good approx.

$$x' = x^0 + p_D^0 = \begin{bmatrix} 0.896 \\ 1.409 \end{bmatrix}$$

$$\Delta_1 = \min\{\bar{\Delta}, 2\Delta_0\} = 1.$$

now check $\|g^1\|$.

Repeat all the steps till I get

$\|g\| < \epsilon$ which is the stopping condition

else max iterations are reached.

Prove that there are only a finite no of failed iterations between two successful iterations:

We are given the fact that mk(pk)<= mk(pkc) and ||gk||>epsilon and

Delta(k)<epsilon/2*B

NON MONOTONE TRUST REGION ALGORITHM

The Nonmonotone Trust Region (NMTR) method is an optimization technique used for solving unconstrained optimization problems. Unlike traditional trust region methods that require a monotonic decrease in the objective function at each iteration, NMTR allows for occasional increases in the objective function value. This approach helps in escaping local minima and enhances the algorithm's ability to find global optima, especially in complex, non-convex optimization landscapes. NMTR methods typically involve adaptive strategies for managing the trust region radius and step acceptance criteria, balancing historical and current information to navigate the optimization space more effectively.

NEED FOR NMTR N :

1. **Escaping Local Minima:** NMTR methods allow occasional increases in the objective function, which helps in escaping local minima. This is especially advantageous in non-convex optimization problems where local minima are prevalent and can hinder finding the global optimum.
2. **Improved Convergence in Complex Landscapes:** Due to their nonmonotonic nature, NMTR methods can better navigate complex optimization landscapes. They are less likely to be trapped by stringent descent requirements of monotonic methods, leading to improved convergence properties.
3. **Robustness to Noisy Functions:** NMTR methods are more effective in scenarios where the objective function is noisy or discontinuous. Their flexibility in accepting non-descending steps makes them more robust against erratic function behaviors.
4. **Adaptive Strategy:** The adaptive nature of the NMTR methods, particularly in adjusting the balance between historical and current information, allows for a dynamic approach that can be tailored to the specific characteristics of the problem, enhancing the overall efficiency and effectiveness of the optimization process.

Algorithm NMTR-N

Step 0: Initialization

1. **Initial Setup**:
   - Choose an initial point $x_0 \in \mathbb{R}^n$.
   - Initialize a symmetric matrix $B_0 \in \mathbb{R}^{n \times n}$.
   - Set an initial trust-region radius $\delta_0 > 0$.
   - Define constants $0 < \mu_1 \leq \mu_2 < 1, 0 < \gamma_1 \leq \gamma_2 < 1, 0 \leq \eta_{\min} \leq \eta_{\max} < 1$, and $N \geq 0$.
   - Compute the initial function value $f(x_0)$.
   - Set iteration counter $k = 0$.

Step 1: Gradient Computation

- Compute the gradient $g(x_k)$ at the current point x_k.
- **Stop Condition**: If $g(x_k)$ is less than or equal to a predefined stopping threshold, stop the algorithm.

Step 2: Subproblem Solution

- Solve the subproblem to find a trial step d_k such that $\|d_k\| \leq \delta_k$.

Step 3: Nonmonotone Strategy and Step Acceptance

- Compute the values $m(k)$, $fl(k)$, and R_k.

$$fl(k) = \max\{f(x_{k-i}) \mid i \in \{0, 1, ..., m\}\}$$

$$R_k = \eta_k \cdot fl(k) + (1 - \eta_k) \cdot f_k$$

- Define ρ_k as:

$$\rho_k = \frac{R_k - f(x_k + d_k)}{m_k(0) - m_k(d_k)}$$

- **Step Acceptance:**
  - If $\rho_k \geq \mu_1$, update the next point as $x_{k+1} = x_k + d_k$.

Step 4: Trust-Region Radius Update

- Update the trust-region radius δ_{k+1} based on ρ_k as follows:
  - If $\rho_k \geq \mu_2$, set δ_{k+1} to be in the range $[\delta_k, \infty)$.
  - If $\mu_1 \leq \rho_k < \mu_2$, set δ_{k+1} to be in the range $[\gamma_2 \delta_k, \gamma_2 \delta_k]$.
  - If $\rho_k < \mu_1$, set δ_{k+1} to be in the range $[\gamma_1 \delta_k, \gamma_2 \delta_k)$.

Step 5: Matrix Update and Iteration

- Update the matrix B_{k+1} using a quasi-Newton formula.
- Increment the iteration counter k by 1.
- Return to Step 1 for the next iteration.

IMPLEMENTATION OF THE ALGORITHMS:

NORMAL TRUST REGION METHOD:

```
def ntr_algorithm(obj_func, start_point, max_iterations, initial_delta, eta, max_delta):
    x = start_point
    delta = initial_delta
    function_values = []

    for i in range(max_iterations):
        grad = opt.approx_fprime(x, obj_func, epsilon=1e-8)
        hess = nd.Hessian(obj_func)(x)
        p = trust_region_subproblem(x, grad, hess, delta)

        actual_reduction = obj_func(x) - obj_func(x + p)
        predicted_reduction = -grad.dot(p) - 0.5 * p.dot(hess).dot(p)

        if abs(predicted_reduction) > 1e-8:
            rho = actual_reduction / predicted_reduction
        else:
            rho = 0

        if rho < 0.25:
            delta *= 0.25
        elif rho > 0.75 and np.linalg.norm(p) == delta:
            delta = min(2.0 * delta, max_delta)

        if rho > eta:
            x = x + p

        function_values.append(obj_func(x))

    return x, function_values
```

NON MONOTONE TRUST REGION METHOD

```python
def nmtr_algorithm(obj_func, start_point, max_iterations, initial_delta, eta, history_length, eta_0, max_delta):
    x = start_point
    delta = initial_delta
    f_history = [obj_func(x)]
    function_values = []
    eta_history = [eta_0]

    for k in range(1, max_iterations + 1):
        grad = opt.approx_fprime(x, obj_func, epsilon=1e-8)
        hess = nd.Hessian(obj_func)(x)
        p = trust_region_subproblem(x, grad, hess, delta)

        eta = update_eta(k, eta_history, eta_0)
        eta_history.append(eta)

        max_f = max(f_history[-min(k, history_length):])
        actual_reduction = max_f - obj_func(x + p)
        predicted_reduction = -grad.dot(p) - 0.5 * p.dot(hess).dot(p)

        if abs(predicted_reduction) > 1e-8:
            rho = actual_reduction / predicted_reduction
        else:
            rho = 0

        if rho < 0.25:
            delta *= 0.25

        elif rho > 0.75 and np.linalg.norm(p) == delta:
            delta = min(2.0 * delta, max_delta)

        if rho > eta:
            x = x + p
            f_history.append(obj_func(x))

        function_values.append(obj_func(x))

    return x, function_values
```

```python
def nmtr_n1_algorithm(obj_func, start_point, max_iterations, initial_delta, eta, history_length, max_delta):
    # NMTR-N1 uses a specific eta_0 value
    eta_0_n1 = 0.85
    return nmtr_algorithm(obj_func, start_point, max_iterations, initial_delta, eta, history_length, eta_0_n1, max_delta)

def nmtr_n2_algorithm(obj_func, start_point, max_iterations, initial_delta, eta, history_length, max_delta):
    # NMTR-N2 uses a different eta_0 value
    eta_0_n2 = 0.2
    return nmtr_algorithm(obj_func, start_point, max_iterations, initial_delta, eta, history_length, eta_0_n2, max_delta)
```

LINK FOR THE GOOGLE COLAB NOTEBOOK:

https://colab.research.google.com/drive/1opiD_YVlFHz2uQZ-JfCCGuUQskfKtEFX#scrollTo=j7pGuWuujQcY

REFERENCES

1. YOUTUBE LECTURE ON TRUST REGION ALGORITHM BY DR. DEBDAS GOSH

   https://www.youtube.com/watch?v=Z-T0n9cdwcw&t=3044s&ab_channel=IITKANPUR-NPTEL

2. RESEARCH PAPER

   An efficient nonmonotone trust-region method for unconstrained optimization

   By Masoud Ahookhosh · Keyvan Amini

   Received: 8 September 2010 / Accepted: 24 August 2011 /

Published online: 6 September 2011

Thank You