ECE 506 Optimization Theory

Name: Solutions

Problem 1: 15 /15

Problem 2: 20/20

Problem 3: 10 /10

Problem 4: 30 /30

Problem 5: 25 /25

Total: 100 /100

Good Luck!

Problem 1 (15 points)

1(a)(**5 points**) Consider the cubic function given by

$$f(x_1, x_2) = a(x_1 - 1)^3 + b(x_2 - 2)^3.$$

Give expressions for ∇f and $\nabla^2 f$.

$$\nabla f(x) = \begin{bmatrix} 3a(x_1-1)^2 \\ 3b(x_2-2)^3 \end{bmatrix}$$

$$\nabla^2 f(x) = \begin{bmatrix} 6a(x_1-1) & 0 \\ 0 & 6b(x_2-2) \end{bmatrix}$$

1(b)(**5 points**) When is f stationary?

$\nabla f(x) = 0$ for $x^* = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $a, b \neq 0$.

1(c)(**5 points** At the stationary point, is it possible to gave a strict minimum? If yes, when?

$\nabla^2 f(1,2) = 0 \Rightarrow$ Not possible.

Problem 2 (20 points)
We begin by restating theorem 4.1 from your text. The vector p^* is a global solution of the trust-region problem:

$$\min_{p \in R^n} m(p) = f + g^T p + \frac{1}{2} p^T B p, \quad s.t. \quad ||p|| \leq \Delta,$$

if and only if p^* is feasible and there is a scalar $\lambda \geq 0$ such that the following conditions are satisfied:

1. $(B + \lambda I)p^* = -g$
2. $\lambda(\Delta - ||p^*||) = 0$
3. $(B + \lambda I)$ is positive semidefinite.

2(a)(**5 points**) First, let us consider the unconstrained problem. Give an optimal expression for p^* by solving $\nabla m(p^*) = 0$.

$\nabla m(p^*) = g + Bp^* = 0 \Rightarrow p^* = -B^{-1}g$

(B invertible)

2(b)(**5 points**) Suppose that $||p^*|| < \Delta$ lies inside the feasible region. Show that this can only happen if B is positive semidefinite. Furthermore, how does the optimal p^* compare with the one found in 2(a)?

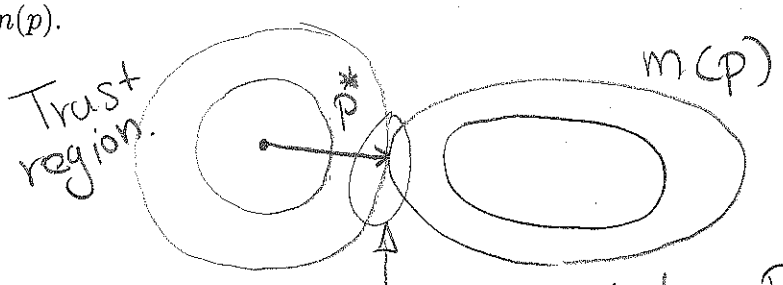
From $\lambda(\Delta - ||p^*||) = 0 \Rightarrow \lambda = 0$. Thus $Bp^* = -g$, which is the same solution as in 2(a).

Since $\lambda = 0$, B is positive semidefinite.

2(c)(**5 points**) Suppose that the solution is on the boundary: $||p^*|| = \Delta$ and $\lambda > 0$. Derive an expression for p^* in terms of $\nabla m(p)$.

From (1) $(B + \lambda I)p^* = -g \Rightarrow p^* = -\frac{(g + Bp^*)}{\lambda} = -\frac{\nabla m(p^*)}{\lambda}$

2(d)(**5 points**) Based on your answer in 2(c), sketch p^*, the trust region, and the contours of $m(p)$.



At this point: $p^* = -\frac{\nabla m(p^*)}{\lambda}$

(also see book)

Problem 3 (10 points).
Suppose that f is twice differentiable and that the Hesian $\nabla^2 f(x)$ is Lipschitz continuous in a neighborhood of the solution x^*, at which the second order sufficient conditions for a strict minimum are satisfied. Consider the Newton algorithm:

$$x_{k+1} = x_k + p_k$$
$$p_k^N = -\nabla^2 f_k^{-1} \nabla f_k.$$

Assuming that the starting point is sufficiently close to x, we can show that $\{x_k\}$ converges to x^* quadratically. You are not asked to prove this. You need to show that $\{||\nabla f_k||\}$ converges quadratically to zero.

Hint: Start from:

$$||\nabla f(x_{k+1})|| = ||\nabla f(x_{k+1}) - 0||,$$

replace 0 by a suitable expression and then use the mean-value theorem for the gradient:

$$\nabla f(x+p) = \nabla f(x) + \int_0^1 \nabla^2 f(x+tp)p\,dt, \quad t \in (0,1).$$

From $p_k^N = -\nabla^2 f_k^{-1}(\nabla f_k) \Rightarrow 0 = p_k^N + \nabla^2 f_k^{-1} \nabla f_k$

Multiply both sides by $\nabla^2 f_k$:

$$0 = \nabla f_k - \nabla^2 f_k \, p_k^N \quad (*)$$

$$\|\nabla f(x_{k+1})\| = \|\nabla f(x_{k+1}) - \nabla f_k - \nabla^2 f_k \, p_k^N\|$$

$$= \left\| \int_0^1 \nabla^2 f(x_k + t p_k^N) p_k^N \, dt - \nabla^2 f_k \, p_k^N \right\|, \quad t \in (0,1)$$

$$\leq \int_0^1 \|\nabla^2 f(x_k + t p_k^N) - \nabla^2 f(x_k)\| \, \|p_k^N\| \, dt$$

$$\leq \tfrac{1}{2} L \|p_k^N\|^2 \quad \text{for Lipschitz constant } L.$$

$$\leq \tfrac{1}{2} L \|\nabla^2 f(x_k)^{-1}\|^2 \|\nabla f_k\|^2 \quad \text{by } p_k^N \text{ definition}$$

$$\leq 2L \|\nabla^2 f(x^*)^{-1}\|^2 \|\nabla f_k\|^2 \quad \text{for } x_k \to x^*.$$

Problem 4 (30 points)

4(a)(4 points) Consider $a^T b$ where a, b are n-dimensional vectors. Give:

- M : the number of multiplications: $M = n$
- A : the number of additions: $A = n-1$
- S : the storage requirements for B-bytes per vector element (only for a, b): $2nB = S$
- m : the number of memory accesses to the elements of a, b: $m = 2n$

4(b) Consider the following algorithm:

```
0.1  q ← ∇f_k                                   n memory accesses
0.2  for i=k-1, k-2, ..., k-m do                m iterations
0.3  |  α_i ← ρ_i s_i^T q;                       1 vector prod + 1 mult + 2 mem. acc.
0.4  |  q ← q − α_i y_i                          n mults + n subs + 3n mem. acc.
0.5  end
0.6  r ← H_k^0 q;
0.7  for i=k-m, k-m+1, ..., k-1 do              m iterations
0.8  |  β ← ρ_i y_i^T r;                        1 vector prod + 1 mult + 3 mem. acc.
0.9  |  r ← r + s_i(α_i − β);                   n subs + n mults + n adds + 3n mem acc.
0.10 end
```

Algorithm 1: L-BFGS two-loop recursion for computing $r = H_k \nabla f_k$.

Give a list of all of the input variables (2 points):

$\{s_i\}$, $\{y_i\}$, ρ_i for $i = k-1, \ldots, k-m$; ∇f_k, H_k^0

What are the storage requirements for representing H_k (4 points)?:

$\{s_i\}$, $\{y_i\}$ are $m \times n \cdot B$

ρ_i is $m \cdot B$, H_k^0 is scalar or diagonal.

Answer $= m \cdot B + 2m \cdot n \cdot B + n \times B$

$= B(2mn + m + n)$

Without considering the cost of computing $r \leftarrow H_k^0 q$, based on your answer in (a), give the following (10 points):

- T_M : the total number of multiplications
- T_A : the total number of additions
- $Tsub$: the total number of subtractions
- $Tmem$: the number of memory accesses

$T_m = m \cdot [\underbrace{(n+1+n)}_{\text{first loop}} + \underbrace{(n+1+3+n)}_{\text{2nd loop}}] = m(4n+5)$

$T_A = m \cdot [\underbrace{(n-1)}_{\text{first loop}} + \underbrace{(n-1+n)}_{\text{2nd loop}}] = m(3n-2)$

$T_{sub} = m[n+n] = 2nm$

$T_{mem} = m[\underbrace{(2n+2+3n)}_{\text{1st loop}} + \underbrace{(2n+3+3n)}_{\text{2nd loop}}]$

$= m(10n+5)$

4(c)(5 points) Assume that we initialize using $H_k^0 = \gamma_k I$, where $\gamma_k = s_{k-1}^T y_{k-1}/(y_{k-1}^T y_{k-1})$. Based on (a), give the number of multiplications and additions needed to compute $r \leftarrow H_k^0 q$.

$(n+n) = 2n$ mults + 1 division for γ_k

Then n multiplies for $H_k^0 q$.

```
1.1  Choose starting point x_0, integer m > 0;
1.2  k ← 0;
1.3  repeat
1.4  |  Choose H_k^0
1.5  |  Compute p_k = −H_k ∇f_k using L-BFGS two-loop recursion algorithm.
1.6  |  Compute x_{k+1} ← x_k + α_k p_k, where α_k is
1.7  |        chosen so as to satisfy the Wolfe conditions;
1.8  |  if k > m then
1.9  |  |  Discard the vector pair {s_{k−m}, y_{k−m}} from storage;
1.10 |  end
1.11 |  Compute and save s_k ← x_{k+1} − x_k, y_k ← ∇f_{k+1} − ∇f_k, ρ_k ← 1/(y_k^T s_k).
1.12 |  k ← k + 1;
1.13 until convergence;
```

Algorithm 2: L-BFGS.

4(d) In the L-BFGS algorithm given above, assume that it takes P function evaluations for f_k and Q function evaluations for ∇f_k to find α_k that satisfies the Wolfe conditions. Please note that $\alpha_k = 1$ is expected to work. For each iteration, give (3 points):

- A_f : the average number of function evaluations
- A_{Df} : the average number of ∇f_k evaluations

$A_f = P$

$A_{Df} = Q+1$ if stored properly

4(e)(2 points) Assume that each gradient component is estimated using

$$\frac{f_i}{\partial x_i}(x) \approx \frac{f(x + \epsilon e_i) - f(x)}{\epsilon},$$ ← already known

where ϵ is chosen as discussed on page 196 of your book (not discussed here). In this case, give A_f accounting for the average number of function evaluations in 4(d).

$A_f = P + (Q+1)n$

↑ 1 for each e_i.

Problem 5 (25 points) Stochastic Optimization

5(a)(4 points) We want to minimize $f(x)$ where x is a vector of **non-negative integers**:

$$(x_1, x_2, \ldots, x_p).$$

Let each integer x_i be represented with b-bits. For p-dimensional vectors, how many possible integer vectors do we have? Give an example for $p = 64$, $b = 8$.

$\underbrace{2^b \times 2^b \times \ldots \times 2^b}_{p \text{ times}} = 2^{bp}$

For $p=64, b=8$, we have: $2^{64\times 8} = 2^{512}$

5(b)(4 points) Define a neighborhood system that changes a **single** variable at a time by adding or subtracting 1. Thus, the following ~~three~~ two vectors will be neighbours of $(x_1, x_2, \ldots, x_p)$:

$$(x_1 + 1, x_2, \ldots, x_p), (x_1 - 1, x_2, \ldots, x_p).$$

Here, for simplicity, we assume wrap-around at the edges (eg: in 8-bits, $255+1$ gets mapped to 0). Give $|N(x)|$, the number of neighboring vectors for x. Will this neighborhood system reach all possible vectors? Explain briefly. Note that it is clear how to expand this approach to arbitrary regions by simply mapping the results from the current approach to any other range.

$|N(x)| = 2p$. Yes, this approach can reach all possible cases.

Eg: To go to y from x, take the steps $y_i - x_i$ for $i = 1, 2, \ldots, p$.

For example, for $y = (2,2)$, $x = (0,2)$, simply do $x + (1,0) + (1,0)$ for $i = 1$.

5(c)(4 points) Recall the Markov-chain transition probability expression:

$$p = \min\left\{1, \frac{\exp\left(\lambda_n f(y)\right)/|N(y)|}{\exp\left(\lambda_n f(x)\right)/|N(x)|}\right\}.$$

Provide a simplified expression for p for this case.

From $|N(x)| = |N(y)|$

$$\Rightarrow p = \min\left\{1, \exp\left[\lambda_n\left(f(y) - f(x)\right)\right]\right\}$$

5(d)(4 points) Provide a simple **for**-loop that can be used to maximize $f(.)$ *without storing* all of the generated values. Your code should terminate after *MaxIterations* and also specify λ_n.

Inputs: x_0, MaxIterations, C

$f_{max} = f(x_0)$

$x_{max} = x_0$, $x = x_0$

for $i = 1, 2, \ldots,$ MaxIterations

Randomly choose y, one of the $2n$-neighbors of x.

$\lambda_i = C\log(1+i)$

Move

$x = \begin{cases} y, & \text{with probability } p \text{ (see above)} \\ x, & \text{else.} \leftarrow \text{Stay} \end{cases}$

If $f(x) > f_{max}$ update f_{max}, x_{max}

end

5(e)(9 points) Suppose that you are asked to provide a more global approach that will be used to generate Q starting integer vectors, based on a 1-D uniform distribution on each component. *Hint:* Generate each $x = (x_1, x_2, \ldots, x_p)$ by generating each component in the sequence: $x_1, x_2, \ldots, x_p$.

Give the pseudo-code for generating the integer vectors **and** how to combine the results from each of the Q-runs to get the global maximum.

for pt = 1, 2, ..., Q

  for i = 1, 2, ..., p

    x_i = floor($(2^b - 1) * U(0,1)$)

  end

  $x = (x_1, x_2, \ldots, x_p)$

  Run code in 5(d)

  Update x_{max}, f_{max}

end

uniform random variable in the (0,1) range