CS5321 Numerical Optimization Homework 2

Annabella Putri Dirgo - 109006238

1. (20%) Check out the TRUST-REGION NEWTON- LANCZOS METHOD in Section 7.1 in the . What kind of problem it wants to solve? and how the Lanczos method solves it.

   In application , The fundamental computing step in trust region method is to identify an approximation minimizer of some model of the genuine objective function inside a "trust" zone where an appropriate corrective norm is within a certain constraint. By simply picking the l_2-norm may not be adequate when the problem is large and the eigenvalues of the Hessian of the model widely spread. Thus, we want to use Lanczos method to solve the problem when the boundary is encountered, while maintaining the efficiencies of the conjugate-gradient method so long as the iterates lie interior.

   Based on the book, the problem that we want to do is to solve the linear system $B_k p = -\nabla f_k$. It is a generalization of the CG method where an nx_j matrix Q_j with orthogonal columns that span the kylor subspace is generated. ${Q_j}^\top B Q_j = T_j$ has a tridiagonal T_j which we can use to find an approximate solution of the trust-region problem in range of Q_j. We first solve

   $$\min_{w \in \mathbb{R}'} f_k + e_1^T Q_j \left(\nabla f_k\right) e_1^T w + \frac{1}{2} w^T T_j w \quad \text{subject to } \|w\| \leq \Delta_k,$$

   when $e_1 = (1, 0, 0, \cdots, 0)^\top define approximate sol \rightarrow P_k = Q_j \omega$; since T_j is tridiagonal, we can factor $T_j + \lambda I$ to solve.

2. (20%) Check out section 8.2 in the deep learning textbook. Give a summary about the major challenges in neural network optimization.

   In general, the optimization issue for training neural networks is non-convex. Some of the difficulties encountered are listed below:

   (a) Ill-conditioning of the Hessian Matrix

   Ill-conditioning is stated to occur when the first term surpasses the second term, as the cost increases.

   Assuming local convexity, another way of looking at ill-conditioned Hessian is by considering its eigenvalues. Condition number of the Hessian is high if the largest positive eigenvalue of the local Hessian is too large, or when the lowest eigenvalue of the Hessian is very close to zero. It can be shown that the optimal step size is less than the inverse of the local smoothness constant of the function (smoothness, is informally proportional to the largest eigenvalue), so for the former case, if the largest eigenvalue of the Hessian is very large, the step-size

required needs to be extremely small (leading to slow convergence). On the other hand, if the least eigenvalue is extremely small, then we again encounter slow convergence due to lack of enough convexity.

If local convexity is not assumed, it gets difficult to say just via first order optimization methods if we are converging to a saddle point or to a local minima.

(b) Local minima

Due to the model identifiability problem, nearly any Deep Learning (DL) model is bound to have an unusually large number of local minima (LM).

When a sufficiently enough training set can rule out all but one value of the model parameters, the model is said to be identifiable. In the case of neural networks, similar models may be obtained simply changing the positions of the neurons. As a result, they are unidentifiable.

However, all of the local minima created by this have the same cost function value, therefore there is no concern. However, if local minima with large costs are prevalent, as demonstrated above, it becomes a major problem. Low gradients can occur at several sites other than local minima. Nowadays, it is typical to strive for a low but not zero cost value.

(c) Plateaus, saddle points and other flat flat regions

A saddle point acts as both a local minima for some neighbors and a local maxima for the others. Thus, Hessian at SP has both positive and negative eigenvalues.

For many classes of random functions, saddle points become more common at high dimensions with the ratio of number of SPs to LMs growing exponentially with n for an n-dimensional space. Many random functions have an amazing property that near points with low cost, the Hessian tends to take up mostly positive eigenvalues. SGD empirically tends to rapidly avoid encountering a high-cost saddle point.

For SGD to rapidly avoid saddle points, it needs to have sufficiently high stochastic variance along all directions. Langevin dynamics stochastic gradient, is a set of SGD based algorithms that play around with SGD with added white noise to get off local minimas and saddles with high probability under mind conditions. There also might be wide, flat regions of constant value, thereby having a zero gradient. These can be problematic if the cost is high in these regions.

(d) Cliffs and exploding gradients

Neural Networks (NNs) might sometimes have extremely steep regions resembling cliffs due to the repeated multiplication of weights.

Suppose we use a 3-layer (input-hidden-output) neural network with all the activation functions as linear. We choose the same number of input, hidden and output neurons, thus, using the same weight ‘W‘ for each layer. The output layer ‘y = W*h‘ where ‘h = W*x‘ represents the hidden layer, finally giving ‘y = W*W‘ x. So, deep

neural networks involve multiplication of a large number of parameters leading to sharp non-linearities in the parameter space. These non-linearities give rise to high gradients in some places. At the edge of such a cliff, an update step might throw the parameters extremely far.

(e) Long-term dependencies

This problem is encountered when the NN becomes sufficiently deep. For example, if the same weight matrix W is used in each layer, after t steps, we'd get W *W * W ... (t times). Using the eigendecomposition of W:

(f) Inexact gradients

Most optimization algorithms use a noisy/biased estimate of the gradient in cases where the estimate is based on sampling, or in cases where the true gradient is intractable for e.g. in the case of training a Restricted Boltzmann Machine (RBM), an approximation of the gradient is used. For RBM, the contrastive divergence algorithm gives a technique for approximating the gradient of its intractable log-likelihood.

(g) Theoretical limits of optimization

Neural Networks might not end up at any critical point at all and such critical points might not even necessarily exist. A lot of the problems might be avoided if there exists a space connected reasonably directly to the solution by a path that local descent can follow and if we are able to initialize learning within that well-behaved space. Thus, choosing good initial points should be studied.

3. (10%) Let J be an $m \times n$ matrix, $m \geq n$. Show that J has full column rank if and only if $J^T J$ is positive definite.

Since the fact that $J^T J$ is positive definite implies that it is nonsingular, we only need to prove two claims. One of them is that the fact that J has full column rank implies $J^T J$ is positive definite; the other one is that the fact that $J^T J$ is nonsingular implies J has full column rank. (1) If J has full column rank, $Jx \neq 0, \forall x \in \mathbb{R}^n, x \neq 0$. So $x^T J^T J x > 0, \forall x \in \mathbb{R}^n, x \neq 0$, i.e, $J^T J$ is positive definite.

4. (30%) Simplex method (the algorithm is shown in Figure 2): Consider the following linear program:

$$\begin{array}{ll} \max_{x_1,x_2} & z = 8x_1 + 5x_2 \\ \text{s.t.} & 2x_1 + x_2 \leq 1000 \\ & 3x_1 + 4x_2 \leq 2400 \\ & x_1 + x_2 \leq 700 \\ & x_1 - x_2 \leq 350 \\ & x_1, x_2 \geq 0 \end{array}$$

(a) Transform it to the standard form.

$$2x_1 + x_2 \leq 1000 \tag{1}$$

$$3x_1 + 4x_2 \leq 2400 \tag{2}$$

$$x_1 + x_2 \leq 700 \tag{3}$$

$$x_1 - x_2 \leq 350 \tag{4}$$

First, we introduce to a new variable $x_3 \geq 0, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0$.
Rewrite (1) as: $1000 - 2x_1 - x_2 \geq 0$. Now let $x_3 = 1000 - 2x_1 - x_2$, and we have $x_3 \geq 0$. Thus, $2x_1 + x_2 + x_3 = 1000$
Rewrite (2) as: $2400 - 3x_1 - 4x_2 \geq 0$. Now let $x_4 = 2400 - 3x_1 - 4x_2$, and we have $x_4 \geq 0$. Thus, $3x_1 + 4x_2 + x_4 = 2400$
Rewrite (3) as: $700 - x_1 - x_2 \geq 0$. Now let $x_5 = 700 - x_1 - x_2$, and we have $x_5 \geq 0$. Thus, $x_1 + x_2 + x_5 = 700$
Rewrite (4) as: $x_1 - x_2 - 350 \geq 0$. Now let $x_6 = x_1 - x_2 - 350$, and we have $x_6 \geq 0$. Thus, $x_1 - x_2 - x_6 = 350$
Thus, it is transformed into the standard form:

$$\begin{array}{ll} \min_{x_1,x_2} & z = -8x_1 - 5x_2 \\ \text{s.t.} & 2x_1 + x_2 + x_3 = 1000 \\ & 3x_1 + 4x_2 + x_4 = 2400 \\ & x_1 + x_2 + x_5 = 700 \\ & x_1 - x_2 + x_6 = 350 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{array}$$

(b) Suppose the initial guess is $(0, 0)$. Use the simplex method to solve this problem. In each iteration, show

- Basic variables and non-basic variables, and their values.
- Pricing vector.
- Search direction.
- Ratio test result.

Iteration 1

			x_1	x_2	x_3	x_4	x_5	x_6	
Basis	c_B	P	8	5	0	0	0	0	RHS
x_3	0	1000	2	1	1	0	0	0	500
x_4	0	2400	3	4	0	1	0	0	800
x_5	0	700	1	1	0	0	1	0	700
x_6	0	350	1	−1	0	0	0	1	350
max		0	−8	−5	0	0	0		

Iteration 2

			x_1	x_2	x_3	x_4	x_5	x_6	
Basis	c_B	P	8	5	0	0	0	0	RHS
x_3	0	300	0	3	1	0	0	−2	100
x_4	0	1350	0	7	0	1	0	−3	192.86
x_5	0	350	0	2	0	0	1	−1	175
x_1	8	350	1	−1	0	0	0	1	−350
max		2800	0	−13	0	0	0	8	

Iteration 3

			x_1	x_2	x_3	x_4	x_5	x_6	
Basis	c_B	P	8	5	0	0	0	0	RHS
x_2	5	100	0	1	0.33	0	0	−0.67	−150
x_4	0	650	0	0	−2.33	1	0	1.67	390
x_5	0	150	0	0	−0.67	0	1	0.33	450
x_1	8	450	1	0	0.33	0	0	0.33	1350
max		4100	0	0	4.33	0	0	−0.67	

Iteration 4

			x_1	x_2	x_3	x_4	x_5	x_6	
Basis	c_B	P	8	5	0	0	0	0	RHS
x_2	5	360	0	1	−0.6	0.4	0	0	
x_6	0	390	0	0	−1.4	0.6	0	1	
x_5	0	20	0	0	−0.2	−0.2	1	0	
x_1	8	320	1	0	0.8	−0.2	0	0	
max		4360	0	0	3.4	0.4	0	0	

5. (20%) Farkas lemma: Let A be an $m \times n$ matrix and $\vec{b}$ be an m vector. Prove that exact one of the following two statements is true:

   (a) There exists a $\vec{x} \in \mathbb{R}^n$ such that $A\vec{x} = \vec{b}$ and $\vec{x} \geq 0$.

   (b) There exists a $\vec{y} \in \mathbb{R}^m$ such that $A^T\vec{y} \geq 0$ and $\vec{b}^T\vec{y} < 0$.

   (Hint: prove if (a) is true, then (b) cannot be true, and vice versa.)

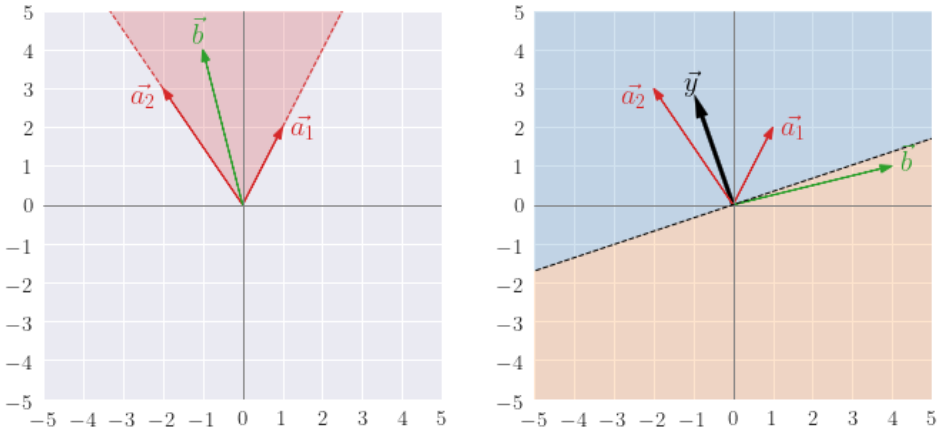
   Proof: First we show that we can't have both (a) and (b). Note that $y^TAx = y^T(Ax) = y^Tb < 0$ since by (a), $Ax = b$ and by (b) $y^Tb < 0$. But also $y^TAx = \left(y^TA\right)x = \left(A^Ty\right)^T x \geq 0$ since by (b) $A^Ty \geq 0$ and by $(a)x \geq 0$.

   Now we must show that if (a) doesn't hold, then (b) does. To do this, let $v_1, v_2, \ldots, v_n$ be the columns of A. Define

$$Q = \text{cone}\,(v_1, \ldots, v_n) \equiv \left\{ s \in \Re^m : s = \sum_{i=1}^{n} \lambda_i v_i, \lambda_i \geq 0, \forall i \right\}.$$

   This is a conic combination of the columns of A, which differs from a convex combination since we don't require that $\sum_{i=1}^n \lambda_i = 1$. Then $Ax = \sum_{i=1}^n x_i v_i$, there exists an x such that $Ax = b$ and $x \geq 0$ if and only if $b \in Q$.

   So if (a) does not hold then $b \notin Q$. We show that condition (b) must hold. We know that Q is nonempty (since $0 \in Q$), closed, and convex, so we can apply the separating hyperplane theorem. The theorem implies that there exists $\alpha \in \Re^m, \alpha \neq 0$, and β such that $\alpha^Tb > \beta$ and $\alpha^Ts < \beta$ for all $s \in Q$. Since $0 \in Q$, we know that $\beta > 0$. Note also that $\lambda v_i \in Q$ for all $\lambda > 0$. Then since $\alpha^Ts < \beta$ for all $s \in Q$, we have $\alpha^T(\lambda v_i) \in Q$ for all $\lambda > 0$, so that $\alpha^Tv_i < \beta/\lambda$ for all $\lambda > 0$. Since $\beta > 0$, as $\lambda \to \infty$, we have that $\alpha^Tv_i \leq 0$. Thus by setting $y = -\alpha$, we obtain $y^Tb < 0$ and $y^Tv_i \geq 0$ for all i. Since the v_i are the columns of A, we get that $A^Ty \geq 0$. Thus condition (b) holds.



case (a) case (b)

Figure 1: Two cases of Farkas Lemma.

Farkas's Lemma (1902) plays an important role in the proof of the KKT condition. The most critical part in the proof of the KKT condition is to show that the Lagrange multiplier $\vec{\lambda}^* \geq 0$ for inequality constraints. We can say if the LICQ condition is satisfied at $\vec{x}^*$, then any feasible direction $\vec{u}$ at $\vec{x}^*$ must have the following properties:

1. $\vec{u}^T \nabla f(\vec{x}^*) \geq 0$ since $\vec{x}^*$ is a local minimizer. (Otherwise, we find a feasible descent direction that decreases f.)
2. $\vec{u}^T \nabla c_i(\vec{x}^*) = 0$ for equality constraints, $c_i = 0$.
3. $\vec{u}^T \nabla c_i(\vec{x}^*) \geq 0$ for inequality constraints, $c_i \geq 0$.

Here is how Farkas Lemma enters the theme. Let $\vec{b}$ be $\nabla f(\vec{x}^*)$, $\vec{y}$ be $\vec{u}$ (any feasible direction at $\vec{x}^*$), the columns of A be $\nabla c_i(\vec{x}^*)$). Since no such $\vec{u}$ exists, according to the properties of $\vec{y}$, statement (a) must hold. The vector $\vec{x}$ in (a) corresponds to $\vec{\lambda}^*$, which just gives us the desired result of the KKT condition.

(1) Given a basic feasible point $\vec{x}_0$ and the corresponding index set $\mathcal{B}_0$ and $\mathcal{N}_0$.

(2) For $k = 0, 1, \ldots$

(3) Let $B_k = A(:, \mathcal{B}_k), N_k = A(:, \mathcal{N}_k)$, $\vec{x}_B = \vec{x}_k(\mathcal{B}_k), \vec{x}_N = \vec{x}_k(\mathcal{N}_k)$, and $\vec{c}_B = \vec{c}_k(\mathcal{B}_k), \vec{c}_N = \vec{c}_k(\mathcal{N}_k)$.

(4) Compute $\vec{s}_k = \vec{c}_N - N_k^T B_k^{-1} \vec{c}_B$ (pricing)

(5) If $\vec{s}_k \geq 0$, return the solution $\vec{x}_k$. (found optimal solution)

(6) Select $q_k \in \mathcal{N}_k$ such that $\vec{s}_k(i_q) < 0$, where i_q is the index of q_k in $\mathcal{N}_k$

(7) Compute $\vec{d}_k = B_k^{-1} A_k(:, q_k)$. (search direction)

(8) If $\vec{d}_k \leq 0$, return `unbounded`. (unbounded case)

(9) Compute $[\gamma_k, i_p] = \min_{i, \vec{d}_k(i) > 0} \dfrac{\vec{x}_B(i)}{\vec{d}_k(i)}$ (ratio test)
(The first return value is the minimum ratio;
the second return value is the index of the minimum ratio.)

(10) $$x_{k+1} \begin{pmatrix} \mathcal{B} \\ \mathcal{N} \end{pmatrix} = \begin{pmatrix} \vec{x}_B \\ \vec{x}_N \end{pmatrix} + \gamma_k \begin{pmatrix} -\vec{d}_k \\ \vec{e}_{i_q} \end{pmatrix}$$
($\vec{e}_{i_q} = (0, \ldots, 1, \ldots, 0)^T$ is a unit vector with i_qth element 1.)

(11) Let the i_pth element in $\mathcal{B}$ be p_k. (pivoting)
$\mathcal{B}_{k+1} = (\mathcal{B}_k - \{p_k\}) \cup \{q_k\}$, $\mathcal{N}_{k+1} = (\mathcal{N}_k - \{q_k\}) \cup \{p_k\}$

Figure 2: The simplex method for solving (minimization) linear programming