CS5321 Numerical Optimization Homework 2

Due Dec 2

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.

   LANCZOS method 要解決的問題：

   當 Hessian 矩陣 $(\nabla^2 f_k)$ 包含負的特徵值時，搜尋方向應會沿著矩陣中最負的特徵值對應的特徵向量，這個特點可以讓演算法快速地逃離那些不是最小值的駐點 (Stationary Point)，雖然這個方法在大部分情況都很有效，但在矩陣很大時，$\nabla^2 f_k$ 過於難計算。

   LANCZOS method 如何解決此問題：

   為了計算以 CG 方法過於難算的 $B_k = -\nabla f_k$，而使用 LANCZOS method，根據上課講義對 trust-region 問題的方程式

   $$\underset{\vec{p}}{min} \quad m_k(\vec{p}) = f_k + \vec{g_k}^{\mathrm{T}}\vec{p} + \frac{1}{2}\vec{p}^{\mathrm{T}} H_k \vec{p}, \quad s.t. \quad \|\vec{p}\| \le \Delta_k$$

   使用 LANCZOS method 時可以產生一個矩陣 Q_j，再以

   $$Q_j^{\mathrm{T}} B Q_j = T_j$$

   產生對稱三對角矩陣 (tridiagonal)T_j，將 trust-region 問題的方程式改為

   $$\underset{\omega \in R^j}{min} \quad f_k + e_1^{\mathrm{T}} Q_j (\nabla f_k) e_1^{\mathrm{T}} \omega + \frac{1}{2}\omega^{\mathrm{T}} T_j \omega, \quad s.t. \quad \|\omega\| \le \Delta_k$$

   再將 trust-region 問題的近似解定義為 $p_k = Q_j \omega.$，就能利用 T_j 的三對角結構在 Q_j 的範圍內尋找 trust-region 問題的近似解，而不需計算複雜的 $-\nabla f_k$。

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

   ## 2.1 Ill-Conditioning

   Ill-Conditioning 指的是當問題的條件數過多時，只要在輸入值存在一點微小的誤差，就會導致輸出產生劇烈的變動，從而變得比較難以優化，或者說需要更多迭代次數來達到同樣精度，就像課本的 Figure 8.1 所示，儘管神經網路成功訓練，但其梯度明顯增加且學習變得非常緩慢。

2.2 Local Minima

Local Minima 就是在函數中局域的最小值，而不一定是 Global minimum，但在 convex function 中，Local Minima 就是 Global minimum。事實上，幾乎所有的神經網路模型都有非常多的 Local Minima，越複雜且層數越多的模型，就有越多的 Local Minima，而在高維空間中，要確定 Local Minima 可能是非常困難的問題。

2.3 Plateaus, Saddle Points and Other Flat Regions

Plateaus 是指在 local maximum 以外的一段平坦的段落。

鞍點 (Saddle Points) 則是除了 local maximum 和 local minimum 以外的另一種梯度為零的點，像馬鞍的造型一樣，鞍點的周圍同時會存在大於鞍點和小於鞍點的值。在鞍點上，Hessian matrix 同時具有正和負的特徵值，對於一旁較高的點，鞍點可以作為 local minimum，而對於另一旁邊較低的點，鞍點可以作為 local maximum，也就是說當最佳化問題搜尋到的 local minimum 是一個鞍點時，就容易造成最佳化的運算出問題。

Other Flat Regions 則是指像 Plateaus 一樣，是存在一段恆定值的段落，其梯度和 Hessian 都是零，一樣會影響最佳化的運算。

2.4 Cliffs and Exploding Gradients

Cliffs 是指在一個函數中，遇到兩點間有極大高度差距的狀況，此差距會導致在計算梯度和步長時出錯。在神經網路中，這通常發生在多個大權重相乘時，當這種像懸崖一樣極其陡峭的結構發生時，一個普通的梯度更新就可能會導致移動一個非常遠的步長，讓其直接跳過整個 Cliff，而若要找的值剛好就在這 Cliff 中，就無法找到。

而簡單的解決方法就是當梯度下降算法提出一個非常大的步長時，就直接減小步長即可，讓其不要走超過這個 Cliff。

2.5 Long-Term Dependencies

Long-Term Dependencies 是指，當一個神經網路具有很深的層數，也就是在每一個時間刻度都在使用一樣的操作來建立結構時，會發生的問題，像課本公式 (8.11) 舉例的：

$$W^t = (V diag(\lambda) V^{-1})^t = V diag(\lambda)^t V^{-1}$$

可以看到，假設一個 computational graph 在每個時間刻度都是乘上 W 這個權重做更新，在經過 t 步後，等於乘以了 W^t，而 W^t 又可以分解成上式的形式，也就是說式子中的 λ 會對 W^t 有很大的影響，當 $\lambda > 1$ 時，W^t 會爆炸 (explode)；當 $\lambda < 1$ 時，W^t 又會消失 (vanish)，所以這又被稱為 vanishing and exploding gradient problem，這些爆炸或消失的梯度可能導致學習不穩定。

而這種情況經常出現在在每個時間刻度都使用相同 W 的循環神經網路 (RNN) 中，但前饋神經網路 (feedforward networks) 就不會。

2.6 Inexact Gradients

在大多數的最佳化算法中，都需假設可以獲得準確的梯度和 Hessian matrix，但通常在實作時，對這些數據都只能有近似的估計值，而對於幾乎所有的深度學習演算法都是依賴這些近似的估計值。

2.7 Poor Correspondence between Local and Global Structure

現今有很多研究方向都是在探討如何為複雜的全域結構找到適當的起始點，眾所周知，起始點對於最佳化問題非常重要。梯度下降和基本上所有對訓練神經網路有效的學習算法都是基於小的局部移動，只能近似地計算目標函數在這個小的局部中的資訊，像是梯度和步長，但有時候這些局部資訊不一定有用，例如走到 Flat Region。所以當一個起始點就是選擇在直接可以通往解的點的位置時，就可以避免很多問題。

2.8 Theoretical Limits of Optimization

一些理論结果表示，現存為神經網路設計的任何優化算法的性能可能都是有限的，因為有些理論結果只適用在輸出值是離散的情況，但通常神經網路的輸出值通常是連續的，在應用那些理論時可能會出現問題，但無法為這些問題歸類以尋找解法。

另外，在使用神經網路訓練模型時，通常不求找到準確的最小值，而只尋求適當地減小值來獲得良好的泛化誤差 (generalization error)，就像本次 programming 作業找到誤差小於 tolerance 的值即可作為解一樣。

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.

先證明 J has full column rank $\Rightarrow J^T J$ is positive definite

if J is full column rank,

$$J\vec{x} \neq 0, \quad \forall \vec{x} \neq \vec{0}$$

同乘 $(J\vec{x})^T = \vec{x}^T J^T$

$$(J\vec{x})^T J\vec{x} > 0$$
$$\Rightarrow \vec{x}^T (J^T J)\vec{x} > 0$$

所以 $J^T J$ is positive definite.

再證明 $J^T J$ is positive definite $\Rightarrow J$ has full column rank

if J is positive definite,

$$\vec{x}^T (J^T J)\vec{x} = (J\vec{x})^T J\vec{x} > 0, \quad \forall \vec{x} \neq \vec{0}$$
$$\Rightarrow J\vec{x} \neq 0, \quad \forall \vec{x} \neq \vec{0}$$

所以 J has full column rank.

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.

A:

Let $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \vec{c} = \begin{bmatrix} 8 \\ 5 \end{bmatrix}, \quad A = \begin{bmatrix} 2 & 1 \\ 3 & 4 \\ 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 1000 \\ 2400 \\ 700 \\ 350 \end{bmatrix}$

The problem can be written as

$$\begin{array}{ll} \underset{\vec{x}}{max} & z = \vec{c}^{\mathrm{T}}\vec{x} \\ \text{s.t.} & A\vec{x} \leq \vec{b} \\ & \vec{x} \geq 0 \end{array}$$

Converting to the standard form, change inequality constraints to equality constraints

$$\begin{array}{l} 2x_1 + x_2 + x_3 = 1000 \\ 3x_1 + 4x_2 + x_4 = 2400 \\ x_1 + x_2 + x_5 = 700 \\ x_1 - x_2 + x_6 = 350 \end{array}$$

x_3, x_4, x_5, x_6 are slack variables

As a result,

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}, \quad \vec{c} = \begin{bmatrix} 8 \\ 5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad A = \begin{bmatrix} 2 & 1 & 1 & 0 & 0 & 0 \\ 3 & 4 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 1000 \\ 2400 \\ 700 \\ 350 \end{bmatrix}$$

for problem $\underset{\vec{x}}{max} \quad \vec{c}^{\mathrm{T}}\vec{x}$, can change to $-\underset{\vec{x}}{min} \quad -\vec{c}^{\mathrm{T}}\vec{x}$,

so the standard form is:

$$-\underset{\vec{x}}{min} \quad z = \begin{bmatrix} -8 \\ -5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix},$$

$$\text{s.t.} \begin{bmatrix} 2 & 1 & 1 & 0 & 0 & 0 \\ 3 & 4 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 1000 \\ 2400 \\ 700 \\ 350 \end{bmatrix}, \vec{x} \geq 0$$

(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.

A:

First iteration, $k = 0$:

$\mathcal{B}_0 = \{3, 4, 5, 6\}, \quad \mathcal{N}_0\{1, 2\}$

$$B_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad N_0 = \begin{bmatrix} 2 & 1 \\ 3 & 4 \\ 1 & 1 \\ 1 & -1 \end{bmatrix},$$

$$\vec{x_B}(\text{Basic variables}) = \begin{bmatrix} x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = B_0^{-1}\vec{b} = \begin{bmatrix} 1000 \\ 2400 \\ 700 \\ 350 \end{bmatrix},$$

$$\vec{x_N}(\text{non-basic variables}) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$\vec{c_B} = \vec{c}(\mathcal{B}_0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \vec{c_N} = \vec{c}(\mathcal{N}_0) = \begin{bmatrix} -8 \\ -5 \end{bmatrix},$$

$$\vec{s_0}(\text{Price vector}) = \vec{c_N} - ({B_0}^{-1}N_0)^{\mathrm{T}}\vec{c_B} = \vec{c_N} - {N_0}^{\mathrm{T}}({B_0}^{-1})^{\mathrm{T}}\vec{c_B}$$

$$= \begin{bmatrix} -8 \\ -5 \end{bmatrix} - \begin{bmatrix} 2 & 1 \\ 3 & 4 \\ 1 & 1 \\ 1 & -1 \end{bmatrix}^{\mathrm{T}} \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \right)^{\mathrm{T}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -8 \\ -5 \end{bmatrix},$$

Figure 3 所附之演算法第 (4) 行的 ${B_k}^{-1}$ 缺少一個轉置

$q_0 = 1, \quad i_q = 1$

$$\vec{s_0} < 0, \quad \vec{d_0}(\text{search direction}) = {B_0}^{-1}N(:, q_0) = {B_0}^{-1} \begin{bmatrix} 2 \\ 3 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 1 \end{bmatrix},$$

$$[\gamma_0, \quad i_p](\text{Ratio test}) = \min_{i, \vec{d_0}(i)>0} \frac{\vec{x_B}(i)}{\vec{d_0}(i)} = \min_{i, \vec{d_0}(i)>0} \begin{bmatrix} 500 \\ 800 \\ 700 \\ 350 \end{bmatrix}$$

$$\Rightarrow \quad \gamma_0 = 350, \quad i_p = 6, \quad p_0 = 4$$

$$\text{so,} \quad \vec{x_1}\begin{pmatrix} \mathcal{B}_0 \\ \mathcal{N}_0 \end{pmatrix} = \begin{pmatrix} \vec{x_B} \\ \vec{x_N} \end{pmatrix} + \gamma_0 \begin{pmatrix} -\vec{d_0} \\ \vec{e_{i_q}} \end{pmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1000 \\ 2400 \\ 700 \\ 350 \end{bmatrix} + 350 \begin{bmatrix} 1 \\ 0 \\ -2 \\ -3 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 350 \\ 0 \\ 300 \\ 1350 \\ 350 \\ 0 \end{bmatrix}$$

Second iteration, $k = 1$:

$$\mathcal{B}_1 = \{3, 4, 5, 1\}, \quad \mathcal{N}_1\{6, 2\}$$

$$B_1 = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad N_1 = \begin{bmatrix} 0 & 1 \\ 0 & 4 \\ 0 & 1 \\ 1 & -1 \end{bmatrix},$$

$$\vec{x_B} = \begin{bmatrix} x_3 \\ x_4 \\ x_5 \\ x_1 \end{bmatrix} = \vec{x_1}\,(\mathcal{B}_1) = \begin{bmatrix} 300 \\ 1350 \\ 350 \\ 350 \end{bmatrix} = B_1^{-1}\vec{b},$$

$$\vec{x_N} = \begin{bmatrix} x_6 \\ x_2 \end{bmatrix} = \vec{x_1}\,(\mathcal{N}_1) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$\vec{c_B} = \vec{c}(\mathcal{B}_1) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -8 \end{bmatrix}, \quad \vec{c_N} = \vec{c}(\mathcal{N}_1) = \begin{bmatrix} 0 \\ -5 \end{bmatrix},$$

$$\vec{s_1} = \vec{c_N} - {N_1}^{\mathrm{T}}({B_1}^{-1})^{\mathrm{T}}\vec{c_B}$$

$$= \begin{bmatrix} 0 \\ -5 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 0 & 4 \\ 0 & 1 \\ 1 & -1 \end{bmatrix}^{\mathrm{T}} \left(\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \right)^{\mathrm{T}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ -8 \end{bmatrix} = \begin{bmatrix} 8 \\ -13 \end{bmatrix},$$

$$q_1 = 2, \quad i_q = 2$$

$$\vec{s_1} < 0, \quad \vec{d_1} = {B_1}^{-1}N(:, q_1) = {B_1}^{-1} \begin{bmatrix} 1 \\ 4 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 3 \\ 7 \\ 2 \\ -1 \end{bmatrix},$$

$$[\gamma_1, \quad i_p] = \min_{i, \vec{d_1}(i)>0} \frac{\vec{x_B}(i)}{\vec{d_1}(i)} = \min_{i, \vec{d_1}(i)>0} \begin{bmatrix} 100 \\ \frac{1350}{7} \\ 175 \\ -350 \end{bmatrix}$$

$$\Rightarrow \quad \gamma_1 = 100, \quad i_p = 3, \quad p_1 = 1$$

so, $\vec{x_2}\left(\begin{array}{c} \mathcal{B}_1 \\ \mathcal{N}_1 \end{array}\right) = \left(\begin{array}{c} \vec{x_B} \\ \vec{x_N} \end{array}\right) + \gamma_1 \left(\begin{array}{c} -\vec{d_1} \\ \vec{e_{i_q}} \end{array}\right)$

$$= \begin{bmatrix} 350 \\ 0 \\ 300 \\ 1350 \\ 350 \\ 0 \end{bmatrix} + 100 \begin{bmatrix} 1 \\ 1 \\ -3 \\ -7 \\ -2 \\ 0 \end{bmatrix} = \begin{bmatrix} 450 \\ 100 \\ 0 \\ 650 \\ 150 \\ 0 \end{bmatrix}$$

Third iteration, $k = 2$:

$\mathcal{B}_2 = \{2, 4, 5, 1\}, \quad \mathcal{N}_2\{6, 3\}$

$$B_2 = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 4 & 1 & 0 & 3 \\ 1 & 0 & 1 & 1 \\ -1 & 0 & 0 & 1 \end{bmatrix}, \quad N_2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix},$$

$$\vec{x_B} = \begin{bmatrix} x_2 \\ x_4 \\ x_5 \\ x_1 \end{bmatrix} = \vec{x_2}\,(\mathcal{B}_2) = \begin{bmatrix} 100 \\ 650 \\ 150 \\ 450 \end{bmatrix} = B_2^{-1}\vec{b},$$

$$\vec{x_N} = \begin{bmatrix} x_6 \\ x_3 \end{bmatrix} = \vec{x_2}\,(\mathcal{N}_2) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$\vec{c_B} = \vec{c}(\mathcal{B}_2) = \begin{bmatrix} -5 \\ 0 \\ 0 \\ -8 \end{bmatrix}, \quad \vec{c_N} = \vec{c}(\mathcal{N}_2) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$\vec{s_2} = \vec{c_N} - {N_2}^{\mathrm{T}}({B_2}^{-1})^{\mathrm{T}}\vec{c_B}$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}^{\mathrm{T}} \left(\begin{bmatrix} 1 & 0 & 0 & 2 \\ 4 & 1 & 0 & 3 \\ 1 & 0 & 1 & 1 \\ -1 & 0 & 0 & 1 \end{bmatrix}^{-1} \right)^{\mathrm{T}} \begin{bmatrix} -5 \\ 0 \\ 0 \\ -8 \end{bmatrix} = \begin{bmatrix} \frac{-2}{3} \\ \frac{13}{3} \end{bmatrix},$$

$q_2 = 1, \quad i_q = 6$

$$\vec{s_2} < 0, \quad \vec{d_2} = {B_2}^{-1}N(:, q_2) = {B_2}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{-2}{3} \\ \frac{5}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix},$$

Figure 3 所附之演算法第 (7) 行將 $N(:, q_2)$ 寫作 A_k,

而講義 p15 寫的是 $\vec{n_{i_q}}$, the i_{th} column of N, 較不易與 A 搞混

$$[\gamma_2, \quad i_p] = \min_{i, \vec{d_2}(i)>0} \frac{\vec{x_B}(i)}{\vec{d_2}(i)} = \min_{i, \vec{d_2}(i)>0} \begin{bmatrix} -150 \\ 390 \\ 450 \\ 1350 \end{bmatrix}$$

$$\Rightarrow \quad \gamma_2 = 390, \quad i_p = 4, \quad p_2 = 2$$

$$\text{so,} \quad \vec{x_3}\begin{pmatrix} \mathcal{B}_2 \\ \mathcal{N}_2 \end{pmatrix} = \begin{pmatrix} \vec{x_B} \\ \vec{x_N} \end{pmatrix} + \gamma_2 \begin{pmatrix} -\vec{d_2} \\ \vec{e_{i_q}} \end{pmatrix}$$

$$= \begin{bmatrix} 450 \\ 100 \\ 0 \\ 650 \\ 150 \\ 0 \end{bmatrix} + 390 \begin{bmatrix} \frac{-1}{3} \\ \frac{2}{3} \\ 0 \\ \frac{-5}{3} \\ \frac{-1}{3} \\ 1 \end{bmatrix} = \begin{bmatrix} 320 \\ 360 \\ 0 \\ 0 \\ 20 \\ 390 \end{bmatrix}$$

Fourth iteration, $k = 3$:

$$\mathcal{B}_3 = \{2, 6, 5, 1\}, \quad \mathcal{N}_3\{4, 3\}$$

$$B_3 = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 4 & 0 & 0 & 3 \\ 1 & 0 & 1 & 1 \\ -1 & 1 & 0 & 1 \end{bmatrix}, \quad N_3 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

$$\vec{x_B} = \begin{bmatrix} x_2 \\ x_6 \\ x_5 \\ x_1 \end{bmatrix} = \vec{x_3}(\mathcal{B}_3) = \begin{bmatrix} 360 \\ 390 \\ 20 \\ 320 \end{bmatrix} = B_3^{-1}\vec{b},$$

$$\vec{x_N} = \begin{bmatrix} x_4 \\ x_3 \end{bmatrix} = \vec{x_3}(\mathcal{N}_3) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$\vec{c_B} = \vec{c}(\mathcal{B}_3) = \begin{bmatrix} -5 \\ 0 \\ 0 \\ -8 \end{bmatrix}, \quad \vec{c_N} = \vec{c}(\mathcal{N}_3) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$\vec{s_3} = \vec{c_N} - {N_3}^{\mathrm{T}}({B_3}^{-1})^{\mathrm{T}}\vec{c_B}$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^{\mathrm{T}} \left(\begin{bmatrix} 1 & 0 & 0 & 2 \\ 4 & 0 & 0 & 3 \\ 1 & 0 & 1 & 1 \\ -1 & 1 & 0 & 1 \end{bmatrix}^{-1} \right)^{\mathrm{T}} \begin{bmatrix} -5 \\ 0 \\ 0 \\ -8 \end{bmatrix} = \begin{bmatrix} \frac{2}{5} \\ \frac{17}{5} \end{bmatrix},$$

$\vec{s_3} > 0$, so the answer is $x_3 = \begin{bmatrix} 320 \\ 360 \\ 0 \\ 0 \\ 20 \\ 390 \end{bmatrix}$.

$x_1 = 320, \quad x_2 = 360$，如 Figure 1 所示。

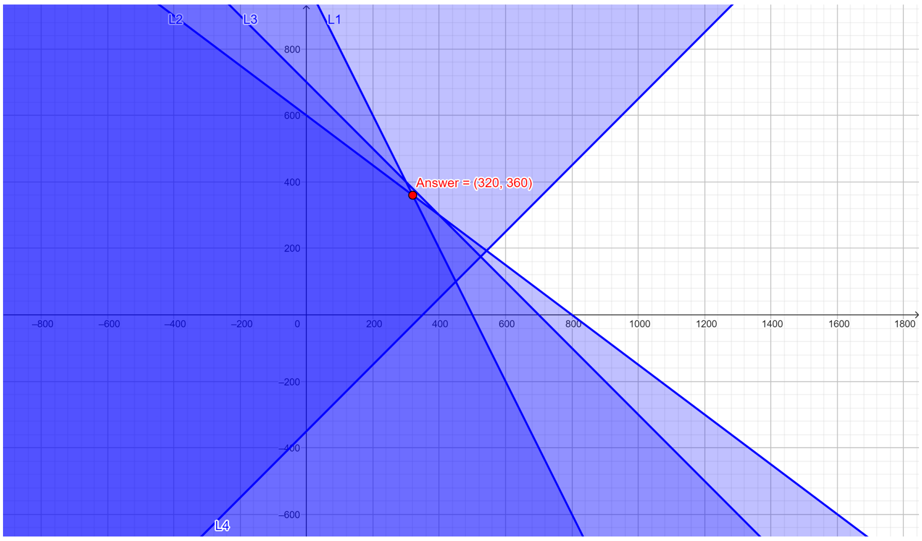

Figure 1: Plot the answer for question 4(b)

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.)

Farkas lemma 解釋：$A \in \mathbb{R}^{m \times n}, \vec{b} \in \mathbb{R}^m$，對於任何的 $\vec{b}$，(a) 和 (b) 論述只會擇一成立。

對於 (a)，可以將 $A\vec{x} = \vec{b}$ 的 A 矩陣拆成 n 個 a 分量

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & \dots & a_n \end{bmatrix}$$

也就是像 Figure 2 的 case(a) 一樣，形成一個立體的錐形 (cone)，定義其為 C，

$$C = cone(a_1, a_2, a_3, ..., a_n)$$

對於 $\vec{b}$ 只會存在兩種互斥情況，(a) $\vec{b}$ 在錐形內，和 (b) $\vec{b}$ 在錐形外，也就是 Farkas lemma 的兩條論述，

若 (a) 成立，說明 $\vec{b}$ 屬於 A 矩陣的線性組合 (或稱為錐組合)，可以找到一

組非負的 $\{x_1, x_2, x_3, ..., x_n\}$ 使 $\vec{b} = a_1x_1 + a_2x_2 + a_3x_3 + ...a_nx_n = A\vec{x}$，也就是 Figure 2 case(a) 的情況。

反之若 (b) 成立，也就是 $\vec{b}$ 在 C 的錐形外時，一定能夠找到一個過原點的超平面將 $\vec{b}$ 與錐形 C 隔開，像 Figure 2 case(b) 中的分隔虛線，而其法向量就是 $\vec{y}$，此時 $y^Ta_1 \geq 0, \quad y^Ta_2 \geq 0, \quad ..., \quad y^Ta_n \geq 0$，也就是 $A^Ty \geq 0$，且此時 $b^Ty = y^Tb < 0$，如 Figure 2 的 case(b) 所示。

參考自 https://www.zhihu.com/question/279644412

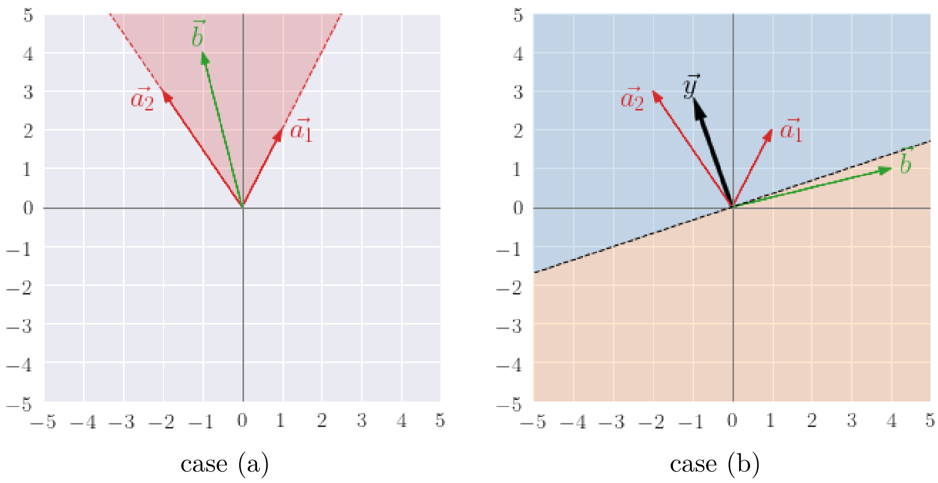

Figure 2: Two cases of Farkas Lemma.

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 $[\gamma_k, i_p] = \min\limits_{i, \vec{d}_k(i) > 0} \dfrac{\vec{x}_B(i)}{\vec{d}_k(i)}$ (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 3: The simplex method for solving (minimization) linear programming