LECTURE NOTES

of

Numerical Optimisation Workshop

held at

Deen Dayal Upadhyay College

University of Delhi

This document contains my lecture notes of the workshop held at *Deen Dayal Upadhyay College, University of Delhi* on *Numerical Optimisation and its Future Perspectives* over the course of 5 days beginning from 12 March, 2024. The most recent version is always available at `https://github.com/zplus11/NA-workshop-DDUC.git`. Contributions to the notes can be freely made there.

Naman Taggar
`namantaggar.11@gmail.com`

Contents

Day 1 Introduction

Dr. Rashmi
Ms. Aparna
Mr. Amlendu Kumar

Optimising means the optimisation of certain things (functions): for example in business, to maximise the production or minimise the cost of production. We proceed with an example.

Example. If there be a ship which is to be loaded with stocks of N items (number of items being x_i), and weight of i^{th} item be w_i, volume v_i, and value p_i for $i = 1, 2, \dots, N$. Cargo volume and weight are respectively W and V. Then, the mathematical formulation can be thought of, as maximising

$$\sum_{i=1}^{N} p_i x_i$$

subject to the constraints

$$\sum_{i=1}^{N} w_i x_i \leq W, \quad \sum_{i=1}^{N} v_i x_i \leq V,$$

and each x_i being necessarily non negative. $\diamond$

In each optimisation problems, such objectives are presented along with the related constraints. Sometimes the constraints may be such that the feasible region between them is unfortunately empty. This case is called non-feasible case. Sometimes, the objective function line may coincide with some side of the feasible region; or in other event the objective line may as well never reach the feasible region at all. In the former case, there are infinite solutions, and in latter none.

§ Graphical Solutions Graphical solutions help interpret and solve the systems with ease. However, they have their own (fair) share of drawbacks as well. Namely, that graphical methods are rendered useless in the case of functions in more variables than 2 variables. Then, the system can not be graphically represented. Then, other methods need be devised to relax the dependence on graphical methods.

§ Simplex Method Consider a particular problem.

Question. Maximise $z = 4x_1 + 3x_2$ subject to the constraints $x_1 + x_2 \leq 8, 2x_1 + x_2 \leq 10$ and $x_1, x_2 \geq 0$.

Solution. Here, we will change it into a standard form, such as to maximise

$$z = 4x_1 + 3x_2 + 0x_3 + 0x_4$$

subject to the constraints $x_1 + x_2 + x_3 = 8$, $2x_1 + x_2 + x_4 = 10$ and $x_1, x_2, x_3, x_4 \geq 0$ by making some substitutions. In general, this looks like maximising $z = c^T x$ subject to $Ax = b$ and $x \geq 0$. $\triangle$

We make some assumptions, namely that $b \geq 0$, and that $\operatorname{Rank} A = m (< n)$. Now, the constraint $Ax = b$ can be changed into $Bx_B = b$, which is called the basic solution.

§ **Transportation and Assignment Problems** Transporation problems are a special kind of LPP's to transport a commodity from a group of nodes, called sources to another group of nodes, called destinations such that the total cost of transporting the commodity is minimized. We can mathematically denote it as minimising

$$Z = \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} c_{ij}$$

subject to the constraints that $\sum_{j=1}^{n} x_{ij} = a_i$ and $\sum_{j=1}^{m} x_{ij} = b_j$, where $x_{ij} \geq 0$ for all i and j, a_i is the quantity of commodity at origin i, b_j is the quantity of commondity demanded at destination j, c_{ij} is the cost of transportation from i to j, and x_{ij} is the quantity transported from i to j.

Refer the presentation shared: `resources\workshop day 1 session 2.pdf`, for a detailed example.

Day 2 Numerical Optimisation

Dr. Brij Mohan

Numerical Optimisation is a field of applied Mathematics that focuses on finding the best solution to a problem from a set of possible solutions. It has a goal to minimise or maximise (or both simultaneously!) an objective function by adjusting the values of input variables within a specified range.

Below we list the steps of optimisation.

1. Problem formulation
2. Function definition
3. Constraints definition
4. Feasible region definition
5. *Initialisation*
6. *Iteration*
7. *Convergence*

Now, we will deal with an example.

Example (Gradient Descent Method)**.** This is an iterative optimisation technique that is used for finding the minimum of a function. The first step here is to find the initial parameter — the initial numerical guess for the index. Then, compute the gradient and update your iterations. Let the objective function here be

$$f(x) = ax^2 + bx + c.$$

Then, the gradient here will be $\frac{\mathrm{d}f}{\mathrm{d}x} = 2ax + b$, and the updating rule then will be

$$x_{\text{new}} = x_{\text{old}} - \text{Learning rate} \times \left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x_{\text{old}}.}$$

Let's now consider $a = 1, b = -6, c = 9$ and the initial guess be 2. Then, we can work with the following script of Mathematica software:

Listing 2.1: Mathematica code for Gradient Descent Method

```
f[x_] := x^2 - 6x + 9;
grad[x_] := f'[x];
xold = 2;
learningRate = 0.1;
iterations = 10;
```

```
For[i = 1, i <= iterations , i++,
    xnew = xold - learningRate * grad[xold];
    xold = xnew;
]
Print["Minimum value ", f[xnew], " achieved at x = ", xnew, "."]
```

The output of the above script will be as follows.

```
Minimum value 0.0115292 achieved at x = 2.89263.
```

Increasing the learning rate will result in a faster convergence of $f(x)$ to zero. We note here that in optimisation, we are finding the value of x such that the function is minimum. However, if we do the reverse process, *then* it will be a root finding problem. $\diamond$

§ **Root Finding Techniques** There are numerous techniques to approximate roots for continuous polynomials. One being Bisection Method. In this method, we first find (or guess) a closed interval such that the values of the function at the extreme points of this interval. If the interval is $[a, b]$, then we shall necessarily have $f(a)f(b) < 0$. Here, the initialisation is to set a mid point of the interval so that this interval divides the original interval into two closed intervals. We now apply the same idea to both these intervals and determine which interval contains the root *now*[1].

Algorithm 1 Algorithm for Bisection Method: n iterations

```
$i \leftarrow 0$
while $i < n$ do
    $c \leftarrow \frac{a+b}{2}$
    if sign $f(b) =$ sign $f(c)$ then
        $b \leftarrow c$
    else
        $a \leftarrow c$
    end if
end while
```

§ **Secant Method** Another method is the Secant Method. We saw that the changes being made in Bisection Method was linear. We remember the rate of convergence and order of convergence here. Walk 5 meteres, and run 5 meters: the time and steps taken to do such travel changes in each case. The degree of optimality is the order of convergence, while the ratios between errors in each iteration is rate.

In this method, we initialise the interval, and the iterative function for indices is given by

$$p_{n+1} = p_n - f(p_n)\frac{p_n - p_{n-1}}{f(p_n) - f(p_{n-1})}.$$

Convergence holds here, and this can be seen algebraically by paying attention to $f_n - p$ at n^{th} iteration.

§ **Newton's Method** Newton's Method is similar. He changed the iterative formula to

$$p_{n+1} = p_n - \frac{f(p_n)}{f'(p_n)}.$$

[1]It is interesting to note that both intervals may as well contain a root too!

The efficiency here, however, is different. This method comes with a convergence rate of 2 — highest in all, *yet*. The following Mathematica Code will approximate zeroes of a polynomial.

Listing 2.2: Mathematica Script for Newton's Root Finding Method

```
newtonMethod[expr_, var_, l_List, n_ : 10, eps_ : 0.0000005] :=
    Module[
    {f, fv, xc,  init, deriv},
    Catch[
        init = (l[[1]] + l[[2]])/2;
        f[fv_] := expr /. var -> fv;
        deriv = f'[var];
        For[i = 1, i <= n, i++,
        xc = init - f[init]/(deriv /. var -> init);
        If[Abs[f[xc]] <= eps, Throw[{i, N[xc, 10], N[f[xc], 10]}]];
        init = xc
        ];
        Throw[{n, N[xc, 10], N[f[xc], 10]}]
    ]
]
```

Iterative Techniques In this section, we will discuss about some iterative techniques to solve numerical optimisation problems surrounding matrices.

§ Gauss-Jacobi Method We will talk about the Jacobi Method first of all. We initialise the matrix

$$X^{(0)} = \begin{bmatrix} x_1^{(0)} & x_2^{(0)} & \dots & x_n^{(0)} \end{bmatrix}$$

and the iteration is then

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1;\, j\neq i}^{n} a_{ij} x_j^{(k)} \right].$$

Convergence holds as $\|X^{(k+1)} - X^{(k)}\| < \epsilon$.

Listing 2.3: Gauss-Jacobi Method in Mathematica

```
gJMethod[A_List, maxiters_ : 10] := Module[
    {n, tuple},
    n = Dimensions[A][[1]];
    Catch[
        If[Dimensions[A][[1]] + 1 == Dimensions[A][[2]], Nothing[],
             Throw["Invalid matrix supplied."]];
        tuple = Table[0, {i, n}];
        For[i = 1, i <= maxiters, i++,
            For[ni = 1, ni <= n, ni++,
                tuple[[ni]] = N[1/A[[ni]][[ni]] (A[[ni, n + 1]] -
                    Sum[A[[ni, r]] tuple[[r]], {r, Complement[Table
                    [1, n], {ni}]}]), 10];
            ];
        ];
        Throw[{i - 1, tuple}]
    ]
]
```

§ Gauss-Seidal Method Then, the Gauss-Seidal Method comes, which is an instance of Gauus-Jacobi however with relative weighting. The iteration here is

$$x_i^{(k+1)} = (1-w)x_i^{(k)} + \frac{w}{a_{ii}} \left[b_i - \sum_{j=1;\, j\neq i}^{n} a_{ij} x_j^{(k)} \right]$$

where w is the implied weight.

Day 3 Logistic Regression In ML

Prof. Amber
Prof. Prashant

Assume we have a set of data point $\{x_i \in \mathbb{R}^d\}$, with each x_i having exactly one label, say $y_i = 1$ or $y_i = 0$. Then,

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$$

are the data points we have, each y_i belonging to $\{0, 1\}$. Now, if we have another data point, say x, can we extract some information from the existing data set to identify whether x belongs to category (label) zero, or one? We here introduce a function

$$p(y=1|x) = \frac{1}{1+e^{-\theta^{\mathrm{T}}x}}; \quad p(y=0|x) = 1 - p(y=1|x) = \frac{e^{-\theta^{\mathrm{T}}x}}{1+e^{-\theta^{\mathrm{T}}x}}.$$

What we want here is to determine θ — a parameter, such that it correctly predicts the categories of the known dataset.

Let

$$C(i) = -\left[y_i \log \pi(i) + (1-y_i)\log\big(1-\pi(i)\big)\right],$$

where $\pi(i) = \frac{1}{1+e^{-\theta^{\mathrm{T}}x}}$, be the cost of predicting x_i correctly. Let us say that $x_i \in y_i = 1$ and θ is such that $\pi(i) = 1$. Then here, the cost of predicting also comes out to be zero. This function is called *cost-entropy based function.* Now, let

$$C(\theta) = -\sum_{i=1}^{N}\left[y_i \log \pi(i) + (1-y_i)\log\big(1-\pi(i)\big)\right]$$

be the cost prediction function for all N data-points, withstanding the same definition of π as before.

Our Problem We now wish to find $\min_{\theta\in\mathbb{R}^d} C(\theta)$.

Example. Let $f(x) = x^2 + 2x + 1$. To minimise this, we find the critical points and try to find the extrema points. Now, consider that

$$\nabla_\theta C(\theta) = \begin{bmatrix} \frac{\mathrm{d}C(\theta)}{\mathrm{d}\theta_1} & \frac{\mathrm{d}C(\theta)}{\mathrm{d}\theta_2} & \cdots & \frac{\mathrm{d}C(\theta)}{\mathrm{d}\theta_d} \end{bmatrix}.$$

And, ∇_θ^2 will belong to $\mathbb{R}^{d\times d}$ and the $(i,j)^{\text{th}}$ component will be nothing but $\frac{\partial^2 C(\theta)}{\partial\theta_i \partial\theta_j}$, with eigen values being non-negative. ◇

§ Iterative Descent Idea Minimising the above friend is a tiresome process. Therefore comes iterative methods to ease the process.

$$\theta_{k+1} = \theta_k + \alpha_k d_k$$

where the θ_k is step size, d_k the descent direction, and α_k a positive number. Here, we start with an initial condition, say θ_0 and progress such as

$$\theta_0 \to \theta_1 \to \theta_2 \to \ldots \text{as } k \to \infty.$$

If α_k is such that, at every iteration, $C(\theta_{k+1}) \leq C(\theta_k)$, then eventually as $k \to \infty$, we find a [local] minimum.

We assume that d_k is the gradient of C at θ_k, that is, $d_k = -\nabla C(\theta_k) \in \mathbb{R}^d$. Consider

$$\begin{aligned} C(\theta_{k+1}) &= C\big(\theta_k - \alpha_k \nabla C(\theta_k)\big) \\ &= C(\theta_k) - \alpha_k \left(\nabla C(\theta_k)\right)^{\mathrm{T}} \left(\nabla C(\theta_k)\right) + O(\alpha_k). \end{aligned}$$

We will make an assumption here that $\alpha_k \to 0$, and under this assumption, we overrule this term from the above expression, replacing equality with approximity. Note that

$$\alpha_k \left(\nabla C(\theta_k)\right)^{\mathrm{T}} \left(\nabla C(\theta_k)\right)$$

will be a non-negative quantity, and so we can guarantarily derive that $C(\theta_{k+1}) \leq C(\theta_k)$ if we use this algorithm with sufficiently small α_k. Essentially, we are decreasing the value of cost at every iteration of the gradient descent, and eventually, we will reach the desired minimum.

As for the value of α, one of the ways to find it is hit and trial. But there are more prudent ways to get that value. $\alpha_k = \frac{1}{k}$ is another way that is dependent on k, rather than a hit and trial method. As k becomes large, α_k is going to zero. Or, we can assume that the gradient of the cost function Lipschitz, that is

$$\|\nabla C(\theta) - \nabla C(\phi)\| \leq L\|\theta - \phi\|,$$

then choosing α in $\left(0, \frac{2}{L}\right)$ will guarantee the reach to minimum.

predictions — future guesses for something that we do not know using current data-sets.

§ Linear Regression Linear Regression is one of the simplest prediction techniques. We here deal with an example.

Example. The state government wishes to build a lot of buildings. They wish to decide the price tag for each building. They may refer the data-sets for previous instances and look for factors including number of rooms, or vicinity of train stations, availability of lifts, and others.

The problem here is to find (predict) the price of a house/apartment/flat given the past data. This past data includes

1. the set of factors influencing price of the house: causality between a building having a lift and the settlement of the price, for example; and
2. a label (or, a price) corresponding to each data point.

We hypothesis here that all of these factors affect the price in a linear (though unknown) fashion. Let us assume that this relation is given by

$$\hat{y} = \theta_0 + \sum_{i=1}^{n} \theta_i \hat{x}_i.$$

Now, the error will be

$$\sqrt{\sum_{j=1}^{m} \left[\hat{y}^j - \left(\theta_0 + \sum_{i=1}^{n} \theta_i \hat{x}_i^j\right)\right]^2}.$$

Optimisation Problem We need to minimise here the above error term. That is, the θ_i's are needed to be found, that are bound to no constraints. Define

$$Y = \begin{bmatrix} \hat{y}_1 & \hat{y}_2 & \dots & \hat{y}_m \end{bmatrix},$$

and X be the $(m+1) \times n$ matrix. Then, the error function will be given by

$$\min_{\theta} (Y - X\theta)^{\mathrm{T}} (Y - X\theta).$$

We find now the derivative and set it zero to obtain

$$-X^{\mathrm{T}}Y + 2X^{\mathrm{T}}X\theta - (Y^{\mathrm{T}}X)^{\mathrm{T}} = 0$$

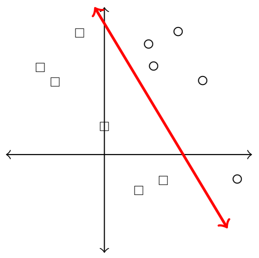
implying further than $\theta = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}Y$. This requires heavy computations, hence we resort to Gradient Descent Method. Begin with $\vec{\theta}^0$ and update $\vec{\theta}^1 = \vec{\theta}^0 + \alpha(-\nabla f)$, continuing until we can sustainably stop. This stopping criterion could be, namely that $|\theta^{k+1} - \theta^k|$ shall be very small. $\diamond$

Day 4 Machine Learning

Prof. Suresh Chandra

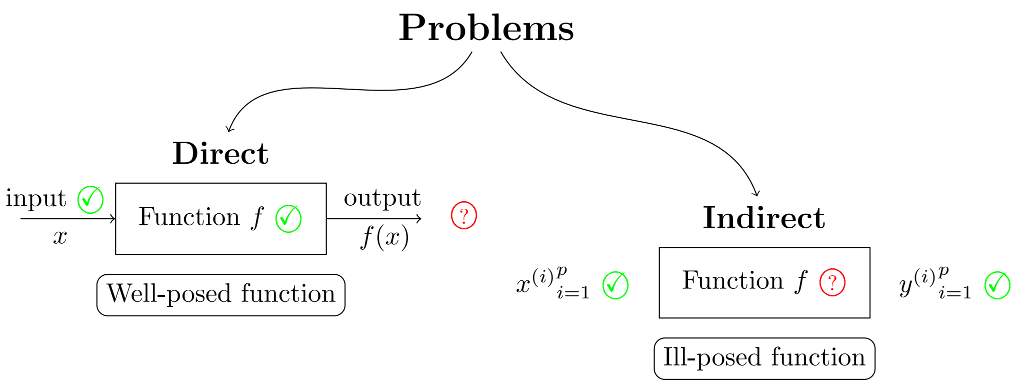
Machine Learning is the development of algorithms trained by data-sets. To understand this, we proceed with an example.

Example. Consider loan giving facility in a bank. The bank has certain datasets including, say k points of the form (x_i, y_i) where x_i is in $\mathbb{R}^n$ containing information about the respective loan request. y_i be the label — 1 if the loan was given, -1 if not; then this data-set can be used to derive whether another point, say x, be labelled 1 or -1.

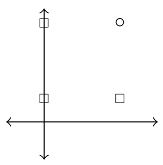


It can be thought of as shown in the figure above. The points to the left of the red line (squares) be the positive ones, and those to the right (circles) be the negative ones. Then, the red line in figure is called the separator. In this case, this separator is linear. Then, similarly, the labels of newer points can be determined by their positioning relative to the red line. The fitting of this red line is done by using already existing data-sets. ◇

§ Well or Ill



Consider the example shown below.



Then, here the sets A and B will be

$$A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \text{and,} \quad B = \begin{bmatrix} 1 & 1 \end{bmatrix}.$$

§ Linear Separability Convex set is a set containing all of the points in the given data-set, while convex hull is the *smallest* such set. Two sets are said to be linearly separable if their *convex hulls* are disjoint. Existence of a plane which strictly separates the sets is guaranteed by the Strict Separation Theorem. However, linear separability holds if and only if this plane is linear. If there does not exist such a plane (and in that case a curve exists), then the separability is non-linear. In our problem, if $w^{\mathrm{T}}x = b$ is the plane, then the points having label $+1$ will satisfy $w^{\mathrm{T}} > b$ and those having label -1 will do $w^{\mathrm{T}} < b$. More formally, the problem is to solve

$$\begin{cases} Aw > be, \\ Bw < be, \\ w \in \mathbb{R}^n, b \in \mathbb{R}, \end{cases}$$

which can be equivalently expressed as

$$\begin{cases} Aw > eb \leq e, & \forall\ y_1 = 1, \\ Bw + be \leq e, & \forall\ y_i = -1, \\ w \in \mathbb{R}^n, b \in \mathbb{R}, \end{cases}$$

§ Measuring Error of Misclassification

$$\boxed{\frac{1}{m}\left\|(-Aw + eb + e)_+\right\|_1 + \frac{1}{k}\left\|(Bw - eb + e)_+\right\|_1.}$$

Our aim here is to minimise the above average infeasibilities. Here, $(X_+)_i = \max(X_i, 0)$ and $\|X\|_1$ is the norm $\sum_{i=1}^{l} |S + i|$.

Question. Let

$$S = \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad \text{and,} \quad B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Solve.

Solution. The LPP formulation for this problem is to minimise $y = \frac{1}{3}(z_1 + z_2 + z_3)$ subject to the constraints

$$\begin{aligned} w_1 + w_2 - b + y &\geq 1, \\ b + z_1 &\geq 1, \\ -w_2 + b + z_2 &\geq 1, \\ -w_2 + b + z_3 &\geq 1, \\ y, z_1, z_2, z_3 &\geq 0, \end{aligned}$$

with w_1, w_2, b being unrestricted. The solution to this LPP is $(w_1, w_2, b) = (2, 2, 3)$, hence the equation of the separating plane is $x_1 + x_2 = 1.5$. △

Example (XOR Problem: Linearly Inseparable)**.** Consider

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \text{and,} \quad B = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}.$$

Then, we will have $\frac{eA}{2} = \frac{eB}{2}$. Therefore, $w = 0, b = 0, y = e, z = e$ is a solution. ◇

Lecture Notes. *12 March – 16 March, 2024.*
Numerical Optimisation And Its Future Perspectives.
Deen Dayal Upadhyay College, University of Delhi.