MATHEMATICAL MODELING

MATH 390R / MATH 490R

By: Dr. Tuan Pham

April 26, 2025

Introduction

In this course, you will learn various models to solve problems from the real world using calculus tools and some additional tools you will learn in this course such as linear algebra, numerical methods, differential equations, and optimization. The topics of this course include:

- Linear system of equations and Gauss–Jordan elimination
- Linear programming with Simplex method
- Matrix multiplication and PageRank algorithm
- Optimization of single-variable functions
- Multivariable functions – level sets and partial derivatives
- Gradient and directional derivatives
- Gradient Descent/Ascent algorithm
- Multivariable optimization with/without constraints
- Differential equations; separation of variables and integrating factor methods
- Finite difference method for first/second order ODE and heat equation
- Neural network method and applications to differential equations

Linear System of Equations and Gauss-Jordan Elimination

Lecture 2

A zero row of a matrix is a row that only contains zeros.

A nonzero row of a matrix is a row that contains at least one nonzero entry.

The reduced row echelon form of a matrix is a matrix in which:

- All the zero rows are at the bottom of the matrix.
- The first nonentry of each nonzero row is 1. We call this entry the *pivot* 1.
- The pivot 1 of each row is on the right of the pivot 1 of the row above it.
- Each column that contains the pivot 1 has only one nonzero entry, which is the pivot 1 itself.

Examples of reduced row echelon form (RREF).

Practice reducing a matrix into a RREF. See the worksheet.

Augmented matrix of a linear system of equations: row operations transform a linear system into an equivalent linear system. The simplest equivalence of a linear system is the linear system corresponding to the RREF of the augmented matrix.

Lecture 3

The process of transforming a matrix into an RREF is called ***row reduction*** or ***Gauss-Jordan elimination***.

Example 1:

Consider a system of linear equations:

$$\begin{cases} x + 2y + 3z = 5 \\ 2x - y - z = -1 \\ y + z = 1 \end{cases}$$

Associated matrix (augmented matrix)

$$\begin{bmatrix} 1 & 2 & 3 & 5 \\ 2 & -1 & -1 & -1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Observation:

Any elementary row operation on this matrix results in an equivalent system of linear equation. Consequently, the RREF of the above matrix corresponds to an equivalent system of linear equation. This new system is very easy to solve.

$$\begin{bmatrix} 1 & 2 & 3 & 5 \\ 2 & -1 & -1 & -1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \xrightarrow{\text{RREF}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

This new matrix (RREF) corresponds to the system:

$$\begin{cases} x = 0 \\ y = -2 \\ z = 3 \end{cases}$$

Example 2:

Consider the system:

$$\begin{cases} x - y + z = 1 \\ 2x + y = 2 \\ 4x - y + 2z = 3 \end{cases}$$

Associated matrix:

$$\begin{bmatrix} 1 & -1 & 1 & 1 \\ 2 & 1 & 0 & 2 \\ 4 & -1 & 2 & 3 \end{bmatrix} \xrightarrow{\text{on the way to RREF}} \begin{bmatrix} 1 & -1 & 1 & 1 \\ 0 & 3 & -2 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

The new matrix corresponds to a system whose last row is $0x + 0y + 0z = -1$. This system is inconsistent (i.e. having no solutions). So is the original system.

Conclusion:

If you can row-reduce an associated matrix into a matrix that has a row $0\ 0\ 0 \ldots 0\ a$ where $a \neq 0$, then the linear system has no solutions.

Example 3:

Consider a system of linear equations:

$$\begin{cases} x - y + z = 0 \\ 2x + y - z = 1 \\ 4x - y + z = 1 \end{cases}$$

Associated matrix (augmented matrix):

$$\begin{bmatrix} 1 & -1 & 1 & 0 \\ 2 & 1 & -1 & 1 \\ 4 & -1 & 1 & 1 \end{bmatrix} \xrightarrow{\text{RREF}} \begin{bmatrix} 1 & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 1 & -\frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

This matrix corresponds to the system

$$\begin{cases} x = \frac{1}{3} \\ y - z = \frac{1}{3} \end{cases}$$

The third column of the matrix doesn't have a pivot 1. The variable corresponding to this column, which is z, can be chosen as a "free" variable (meaning it can take an arbitrary value). Therefore,

$$z = t, \quad y = t + \frac{1}{3}, \quad x = \frac{1}{3}$$

One can rearrange as $(x, y, z) = \left(\frac{1}{3},\ t + \frac{1}{3},\ t\right)$, where $t \in \mathbb{R}$.

Linear Programming with Simplex Method

Lecture 4

The goal of the next 3 lectures is to help you understand and solve linear programming (also called *linear optimization*) problems. A *linear program* is a method to obtain the best outcome (minimum or maximum) of a linear function under some linear constraints. There are many available linear programs, but we will focus on the *Simplex method*. It is the simplest algorithm and can be performed by hand.

Motivating example:

You work two part-time jobs, as a Math Tutor ($14/hour) and at PCC ($12/hour). Each hour working as a Math Tutor requires 15 min of prep time. Each hour working at PCC requires only 10 minutes of prep time. You are allowed to work at most 20 hours a week. Due to school work, you don't want to spend more than 4 hours a week for preparation. Find the number of working hours for each job per week to maximize your income.

Let x be the number of hours working as Math Tutor, and y be the number of hours working at PCC. Constraints:

$$x, y \geq 0$$

$$x + y \leq 20$$

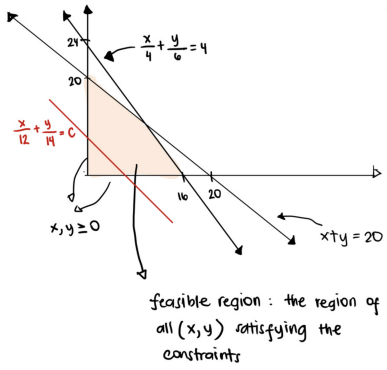
$$\frac{x}{4} + \frac{y}{6} \leq 4$$

Objective function:

$$f(x, y) = 14x + 12y$$

This is the function you want to maximize.

The problem can be solved *graphically*.



Keep in mind that the line $\frac{x}{a} + \frac{y}{b} = 1$ is the line that intersects the x-axis at $(a, 0)$ and the y-axis at $(0, b)$. The line $14x + 12y = c$ is parallel to the line $14x + 12y = 14 \cdot 12$, which is equivalent to:

$$\frac{x}{12} + \frac{y}{14} = 1$$

Feasible region is the set of all pairs (x, y) that satisfy the constraints of the problem. It is a region bounded by lines.

Objective function is the function we want to maximize or minimize. If it is to be minimized, it is often called a *cost function*.

In the previous problem, we need to find a point in the feasible region at which c is maximum. This can occur only at one of the vertices (i.e., corner points) of the feasible region. There are four corner points: $(0, 0), (16, 0), (0, 20), (a, b)$ where (a, b) are the coordinates of the intersection point of the lines:

$$x + y = 20$$
$$\frac{x}{4} + \frac{y}{6} = 4$$

We solve this system of equations using the Gauss-Jordan elimination method:

$$\begin{bmatrix} 1 & 1 & 20 \\ \frac{1}{4} & \frac{1}{6} & 4 \end{bmatrix} \xrightarrow{\text{RREF}} \begin{bmatrix} 1 & 0 & 8 \\ 0 & 1 & 12 \end{bmatrix}$$

Which yields $(a, b) = (8, 12)$. Compare the objective function $f(x, y)$ at the four corner points. The maximum is at $(x, y) = (8, 12)$ with $f(8, 12) = 256$.

A general linear programming problem is of the form of maximizing or minimizing the objective function

$$f(x_1, x_2, \ldots, x_n) = a_1x_1 + a_2x_2 + \cdots + a_nx_n$$

under the constraints

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &\leq (\geq)\ b_1 \\ a_{21}x_1 + \cdots + a_{2n}x_n &\leq (\geq)\ b_2 \\ &\vdots \\ a_{k1}x_1 + \cdots + a_{kn}x_n &\leq (\geq)\ b_k \end{aligned}$$

Comment:

The graphic method is intuitive and simple. However, it is only applicable for a simple problem where there are two, or maybe three, variables. But even with two or three variables, it can get complicated if there are a lot of constraints.

We will discuss an alternative method that can deal (still by hand) with a larger linear optimization problem. It is called the *simplex method*.

$$x, y \geq 0$$

$$x + y \leq 20 \tag{1}$$

$$\frac{x}{4} + \frac{y}{6} \leq 4 \tag{2}$$

Objective function: $f(x, y) = 14x + 12y$

Process:

- Step 1: turn the problem into a *standard form*. This is a form in which:
  - must be a maximization problem
  - all linear constraints must be in a less-than-or-equal-to inequality
  - all variables are non-negative

  The example under consideration is already in standard form.

- Step 2: introduce slack variables to turn all inequality constraints to equality constraints:

$$x, y, s_1, s_2 \geq 0$$

$$x + y + s_1 = 20 \tag{3}$$

$$\frac{x}{4} + \frac{y}{6} + s_2 = 4 \tag{4}$$

  If the right-hand side of any equation is negative, multiply that equation by -1. We want to maximize $z = 14x + 12y$

Observation: Consider the linear system of 3 equations and 5 unknowns:

$$\begin{aligned} x + y + s_1 &= 20 \\ \frac{x}{4} + \frac{y}{6} + s_2 &= 4 \\ -14x - 12y + P &= 0 \end{aligned}$$

This system is associated with the matrix

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 20 \\ \frac{1}{4} & \frac{1}{6} & 0 & 1 & 0 & 4 \\ -14 & -12 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Recall that any elementary row operation transforms this matrix to a matrix that corresponds to an equivalent system of equations. So, what is the matrix corresponding to the simplest linear programming problem? Take a look at the following examples:

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 & 0 & 5 \\ 0 & 2 & 0 & 1 & 1 & 7 \end{bmatrix}$$

The last row reads $2x_2 + s_1 + P = 7$, which implies $P = 7 - 2x_2 - s_1$

Then $\max P = 7$, attained when $x_2 = s_1 = 0$. The first row implies $x_1 = 2$. The solution $(x_1, x_2) = (2, 0)$ is acceptable (i.e., in the feasible region).

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 1 & 0 & 5 \\ 0 & 2 & 0 & 1 & 1 & 7 \end{bmatrix}$$

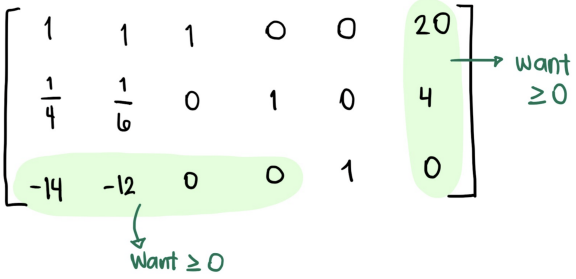
The last row reads $2x_2 + s_1 + P = 7$, which implies $P = 7 - 2x_2 - s_1$

Then $\max P = 7$, attained when $x_2 = s_1 = 0$. The first row implies $x_1 = -2$. However, the solution $(x_1, x_2) = (-2, 0)$ is not acceptable (i.e., outside the feasible region).

Therefore, matrix A is the more favorable situation.

Our goal in the next step (Step 3) is to transform this matrix into a matrix where

- all coefficients on the left of the column of P on the last row are nonnegative,
- all coefficients on the right column and above the last row are nonnegative.



- Step 3:
  - Locate the key column. This is the column containing the largest negative coefficient of the last row.
  - Locate the pivot element on the key column. This is the element $a_j > 0$ such that the quotient b_j/a_j, where b_j is the far-right coefficient on row j, is smallest. This guarantees that the coefficients on the right-hand side are always nonnegative.
  - Use elementary row operations to turn the key column into a column in which that pivot element becomes 1 and the rest become 0s.
  - Repeat the first 3 sub-steps above with the new matrix. Stop when all coefficients on the left of the column of P on the bottom row are nonnegative.

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 20 \\ \frac{1}{4} & \frac{1}{6} & 0 & 1 & 0 & 4 \\ -14 & -12 & 0 & 0 & 1 & 0 \end{bmatrix} \xrightarrow[R_3=R_3+14R_2]{\substack{R_2=4R_2 \\ R_1=R_1-R_2}} \begin{bmatrix} 0 & \frac{1}{3} & 1 & -4 & 0 & 4 \\ 1 & \frac{2}{3} & 0 & 4 & 0 & 16 \\ 0 & \frac{8}{3} & 0 & 56 & 1 & 224 \end{bmatrix}$$

$$\xrightarrow[R_3=R_3-\frac{8}{3}R_1]{\substack{R_1=3R_1 \\ R_2=R_2-\frac{2}{3}R_1}} \begin{bmatrix} 0 & 1 & 3 & -12 & 0 & 12 \\ 1 & 0 & -2 & 12 & 0 & 8 \\ 0 & 0 & 8 & 24 & 1 & 256 \end{bmatrix}$$

This matrix corresponds to the system:

$$y + 3s_1 - 12s_2 = 12$$

$$x - 2s_1 + 12s_2 = 8$$

$$8s_1 + 24s_2 + P = 256$$

Note that the last equation can be rewritten as $P = 256 - 8s_1 - 24s_2$

Obviously, $\max P = 256$, attained when $s_1 = s_2 = 0$.

Example:

Maximize $3x_1 + 2x_2 + 5x_3$ under the constraints:

$$x_1 + 2x_2 + x_3 \geq 43$$

$$3x_1 + 2x_3 \leq 46$$

$$x_1 + 4x_2 \leq 42$$

$$x_1, x_2, x_3 \geq 0$$

Lecture 5

Standard form of a linear programming problem:

- must be a maximization problem,
- all linear constraints must be in a less-than-or-equal-to inequality,
- all variables are non-negative

Introduce slack variables:

- to turn all inequality constraints to equality constraints
- If the right-hand side of any equation is negative, multiply that equation by -1.

Example:

Minimize $C = -2x + y$ subject to

$$x - y \leq 3$$

$$2x + 3y \leq 12$$

$$x + y \geq 2$$

$$x, y \geq 0$$

Convert the above linear programming problem into standard form and introduce slack variables.

Maximize $P = -C = 2x - y$ subject to

$$x - y + s_1 = 3$$

$$2x + 3y + s_2 = 12$$

$$x + y - s_3 = 2$$

$$x, y, s_1, s_2, s_3 \geq 0$$

Perform row operations:

$$\begin{bmatrix} 1 & -1 & 1 & 0 & 0 & 0 & 3 \\ 2 & 3 & 0 & 1 & 0 & 0 & 12 \\ 1 & 1 & 0 & 0 & -1 & 0 & 2 \\ 2 & -1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \xrightarrow[R_2=R_2-2R_3]{R_1=R_1-R_3} \begin{bmatrix} 0 & -2 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 2 & 0 & 8 \\ 1 & 1 & 0 & 0 & -1 & 0 & 2 \\ 4 & 1 & 0 & 0 & -2 & 1 & 4 \end{bmatrix}$$

$$\xrightarrow[R_4=R_4+2R_1]{R_2=R_2-2R_1 \atop R_3=R_3+R_1} \begin{bmatrix} 0 & -2 & 1 & 0 & 1 & 0 & 1 \\ 0 & 5 & -2 & 1 & 0 & 0 & 6 \\ 1 & -1 & 1 & 0 & 0 & 0 & 3 \\ 4 & -3 & 2 & 0 & 0 & 1 & 6 \end{bmatrix} \xrightarrow[R_3=R_3+R_2 R_4=R_4+R_2]{R_2=R_2/5 R_1=R_1+2R_2} \begin{bmatrix} 0 & 0 & \frac{1}{5} & \frac{2}{5} & 1 & 0 & \frac{17}{5} \\ 0 & 1 & -\frac{2}{5} & \frac{1}{5} & 0 & 0 & \frac{6}{5} \\ 1 & 0 & \frac{3}{5} & \frac{1}{5} & 0 & 0 & \frac{21}{5} \\ 4 & -2 & \frac{8}{5} & \frac{1}{5} & 0 & 1 & \frac{36}{5} \end{bmatrix}$$

The last row reads $\frac{8}{5}s_1 + \frac{1}{5}s_2 + P = \frac{36}{5}$. Thus, $P = \frac{36}{5} - \frac{8}{5}s_1 - \frac{1}{5}s_2 \leq \frac{36}{5}$

Then $C = -P \geq -\frac{36}{5}$

The equality occurs when $s_1 = s_2 = 0$. Then: $x = \frac{21}{5}, \quad y = \frac{6}{5}$

Example:

Maximize $P = 3x_1 + 2x_2 + 5x_3$ under the constraints

$$x_1 + 2x_2 + x_3 \geq 43$$
$$3x_1 + 2x_3 \leq 46$$
$$x_1 + 4x_2 \leq 42$$
$$x_1, x_2, x_3 \geq 0$$

Lecture 6

Finish the example last time.

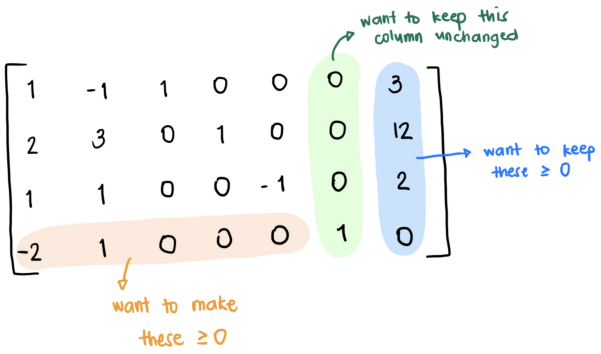
After **Step 1** and **Step 2**, we have turned the linear programming problem into:

Maximize $P = 2x - y$ subject to

$$x - y + s_1 = 3$$
$$2x + 3y + s_2 = 12$$
$$x + y - s_3 = 2$$
$$x, y, s_1, s_2, s_3 \geq 0$$

Step 3: Write and manipulate the associated matrix to turn it into the following form:

$$\begin{bmatrix} 1 & -1 & 1 & 0 & 0 & 3 \\ 2 & 3 & 0 & 1 & 0 & 12 \\ 1 & 1 & 0 & 0 & -1 & 2 \\ -2 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$



Algorithm:

- Identify the key column. This is the column containing the *"most" negative* entry of the last row.
- Identify the pivot element on the key column. This is the element $a_{ij} > 0$ such that the quotient b_j/a_{ij}, where b_j is the far-right coefficient on row j, is smallest. This guarantees that the coefficients on the right-hand side are always nonnegative.
- Use elementary row operations to turn the key column into a column in which that pivot element becomes 1 and the rest become 0s.
- Go back to the first step. Stop when a desirable form is achieved: all coefficients on the left of the column of P on the bottom row are nonnegative.

After a number of row operations, we obtain the following matrix: the worksheet.

$$\begin{bmatrix} 0 & 1 & 0 & -\frac{4}{5} & \frac{2}{5} & 0 & 1 & 0 & \frac{17}{5} \\ 0 & 0 & 1 & -\frac{2}{5} & \frac{1}{5} & 0 & 0 & 1 & \frac{6}{5} \\ 1 & 0 & 0 & \frac{3}{5} & \frac{1}{5} & 0 & 0 & 0 & \frac{21}{5} \\ 0 & 0 & 0 & \frac{8}{5} & \frac{1}{5} & 0 & 0 & 1 & \frac{36}{5} \end{bmatrix}$$

This matrix corresponds to the following system of equations:

$$\frac{4}{5}s_1 + \frac{2}{5}s_2 + s_3 = \frac{17}{5}$$
$$y - \frac{2}{5}s_1 + \frac{1}{5}s_2 = \frac{6}{5}$$
$$x + \frac{3}{5}s_1 + \frac{1}{5}s_2 = \frac{21}{5}$$
$$\frac{8}{5}s_1 + \frac{1}{5}s_2 + P = \frac{36}{5}$$

Note that P (what we want to maximize) only appears in the last equation:

$$P = \frac{36}{5} - \frac{8}{5}s_1 - \frac{1}{5}s_2 \leq \frac{36}{5}$$

$$\max P = \frac{36}{5}$$

The equality occurs when $s_1 = s_2 = 0$, which implies:

$$x = \frac{21}{5}, \quad y = \frac{6}{5}$$

Matrix Multiplication and Page-Rank Algorithm

Lecture 7

The goal of the next three lectures is the PageRank algorithm, which is the original algorithm that Google used to rank websites for their search engine. To get there, you need to know some matrix algebra.

You can add or subtract matrices only if they have the same size. In that case, you add/subtract entry by entry. For example,

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 0 & -2 \\ 2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 5 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} - \begin{bmatrix} 0 & -2 \\ 2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 1 & 5 \end{bmatrix}$$

You can multiply any matrix by a number. For example,

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad 2A = \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}, \quad -A = \begin{bmatrix} -1 & -2 \\ -3 & -4 \end{bmatrix}$$

Multiplying a matrix by a matrix is a little more complicated. First, you need to know how to multiply a row by a column:

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \\ 7 \end{bmatrix} = 1(5) + 2(6) + 3(7) = 38$$

To be able to multiply a row by a column, they must be of the same length. To multiply matrix A and matrix B, you want to be able to multiply each row of A by each column of B. This puts a restriction on the sizes of A and B: the number of columns of A must be equal to the number of rows of B.

Example:

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 3 & 2 \\ 5 & 6 & 4 \end{bmatrix}$$

A is a 2×2 matrix, and B is a 3×2 matrix. AB is not defined, but BA is well-defined.

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} * & * \\ * & * \\ * & * \end{bmatrix}$$

In general, if A is an $m \times n$ matrix and B is an $n \times p$ matrix then AB is an $m \times p$ matrix.

Identity matrix:

For each value of n, there is an identity matrix for each size n, denoted by I_n, which acts like number 1 in the regular multiplication.

$$I_1 = \begin{bmatrix} 1 \end{bmatrix}, \quad I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Check that $AI_2 = I_2A = A, BI_2 = B, I_2, I_3B = B$.

Practice on the worksheet.

A vector is a matrix with only one row or only one column. For example,

$$\begin{bmatrix} 1 & 2 \end{bmatrix} \text{ is a row vector of length 2}$$

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \text{ is a column vector of length 3}$$

A nonzero column vector v is said to be a stationary vector of matrix A if $Av = v$.

Lecture 8

A vector is a matrix with only one row or only one column. A column vector v is said to be a stationary vector of matrix A if $v \neq 0$ and $Av = v$. Vector v is called a probability vector if every entry is nonnegative and the sum of all entries is equal to 1.

Example:

$$A = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \end{bmatrix}$$

Does A have a stationary vector? If so, does it have a probability stationary vector?

$$Av = v = I_2 v$$

$$(A - I_2)v = 0$$

Let $v = \begin{bmatrix} x \\ y \end{bmatrix}$. Then the equation $(A - I_2)v = 0$ is equivalent to a linear system of two equations, whose associated matrix is

$$\begin{bmatrix} 0 & 2 & 0 \\ 3 & 3 & 0 \end{bmatrix} \xrightarrow{\text{RREF}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

The solution is $v = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, which is not acceptable. Matrix A doesn't have a stationary vector.

Example:

$$A = \begin{bmatrix} \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{2}{3} \end{bmatrix}$$

Does A have a stationary vector? If so, does it have a probability stationary vector?

Perron-Frobenius theorem (1907, 1912): Let A be a stochastic matrix (i.e. a matrix whose columns are probability vectors). Suppose that either A or some power A^k has all positive entries. Then A has a unique probability stationary vector.

Lecture 9

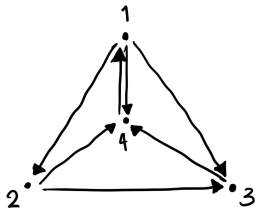
In 1998, while being PhD students at Stanford University, Lawrence Page and Sergey Brin published a paper "The anatomy of a large-scale hypertextual Web search engine" on the journal of *Computer Networks and ISDN systems*. They discovered a profound algorithm to rank web pages. They called it PageRank algorithm, a name that attributes one of the co-authors and also refers to the web *pages*. This algorithm is the basis of the Google search engine.

The overall idea is that web pages are ranked based on their importance. The most important page is ranked first. Each web page will be given a weight between 0 and 1. The sum of the weights of all web pages is equally to 1.

Some natural rules apply:

- Any link from a web page to itself does not count.
- Multiple links from web page A to web page B are only counted as one.

Note that the rank of a web page is not simply determined by the number of inbound links to it, but also by the rank of the pages that link to it.
Consider the following internet:



Page 4 has inbound links from all other pages, so it should be the most important page. In term of inbound links, page 1 and 2 each has 1 link, page 3 has 2 links, page 4 has 3 links. Based on these numbers, you may be haste to say that page 3 should be ranked above page 1. But the page that links to page 1 is page 4, which is the most important page. The two pages that link to page 3 are less important than page 4.

Brin and Page's solution (simple version):

Imagine that you freely navigate from one website to another by clicking any link on the page that you are on. All outbound links have the same chance of being clicked on. *The importance of the web page is defined by the chance that you land on it.* Suppose there are n web pages, labelled from 1 to n. The internet above can be represented by a matrix:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 1 & 0 \end{bmatrix}$$

The entry A_{ij} at row i and column j is the probability of going to page i from page j. Matrix A is called a *transition matrix*. You can notice that A is a stochastic matrix.

Denote by E_i the chance that you are on page i, and F_i the event that you click on a link that takes you to page i. The probability of landing on page i is

$$P(E_i) = P(E_1 \cap F_i) + P(E_2 \cap F_i) + \cdots + P(E_n \cap F_i) = P(F_i \mid E_1)P(E_1) + \cdots + P(F_i \mid E_n)P(E_n)$$

Let $p_i = P(E_i)$. Then $p_i = A_{i1}p_1 + A_{i2}p_2 + \ldots + A_{in}p_n$.
In terms of matrix multiplication, $p = Ap$, where

$$p = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}$$

Therefore, p is a probability stationary vector of A. You can check that A^5 has all positive entries. Therefore, according to Perron-Frobenius theorem, A has a unique probability stationary vector. This vector is called the *PageRank vector* of the internet.

To find this vector, you solve the equation $(A - I_4)p = 0$.

Associated matrix:

$$\begin{bmatrix} -1 & 0 & 0 & 1 & 0 \\ \frac{1}{3} & -1 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & -1 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 1 & -1 & 0 \end{bmatrix} \xrightarrow{\text{RREF}} \begin{bmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & -\frac{1}{3} & 0 \\ 0 & 0 & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

All the stationary vectors are $v = \begin{bmatrix} \frac{1}{3}t \\ \frac{1}{2}t \\ t \end{bmatrix}$. For this vector to be a probability vector, $t = \frac{6}{17}$. Therefore, the only probability stationary vector is $p = \begin{bmatrix} \frac{6}{17} \\ \frac{2}{17} \\ \frac{3}{17} \\ \frac{6}{17} \end{bmatrix}$. This is the PageRank vector of the internet. Consequently, page 1 and page 4 are ranked equally highest. Page 3 is ranked next. Page 2 is ranked lowest.

Brin and Page's solution (general version):

There is a drawback with the simple version mentioned above. In the previous example, A, A_2, A_3, A_4 each has at least one zero entry. It is possible to have a situation where no powers A^k have all positive entries. This is indeed the case if the internet has disconnected clusters.

The consequence is that the probability stationary vector may not be unique. In such a case, it is ambiguous how to order the pages.

Brin and Page's solution is to introduce a damping parameter $d \in [0, 1]$. A typical choice

of d is $d = 0.85$.

The transition matrix is

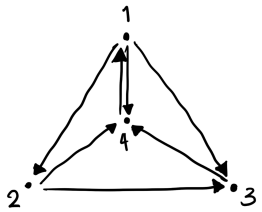
$$A = dA' + (1-d) \cdot \frac{1}{n} \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

where A' is the regular (without damping) transition matrix. As long as $d < 1$, A will have all positive entries.

To see the rationale behind the damping parameter, imagine that you freely navigate from one website to another in the following manner: with probability d, you click on any link on the page that you are on (all outbound links still have the same chance of being clicked on). With probability $1 - d$, you randomly jump to any of the page on the internet. This is a simple and clever solution to "connect" all disconnected cluster.

PageRank Example

Consider the following internet:



Without damping (d=1), the transition matrix is

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 1 & 0 \end{bmatrix}$$

The PageRank vector is $p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$ where p_i is the probability that you arrive on page i. It is a probability stationary vector of A. That is, $p = Ap$. To find this vector, we write the equation as $(A - I_4)p = 0$, which is equivalent to a linear system of equations. The associated matrix is

$$\begin{bmatrix} -1 & 0 & 0 & 1 & 0 \\ \frac{1}{3} & -1 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & -1 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 1 & -1 & 0 \end{bmatrix} \xrightarrow{\text{RREF}} \begin{bmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & -\frac{1}{3} & 0 \\ 0 & 0 & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The solution is $p_4 = t, p_3 = \frac{1}{2}t, p_2 = \frac{1}{3}t, p_1 = t$. For p to be a probability vector, we need $1 = p_1 + p_2 + p_3 + p_4 = \frac{17}{6}t$ which leads to $t = \frac{6}{17}$. Therefore, the PageRank vector of the internet is

$$p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} = \begin{bmatrix} \frac{6}{17} \\ \frac{2}{17} \\ \frac{3}{17} \\ \frac{6}{17} \end{bmatrix}$$

This means, as you browse the internet indefinitely, going from one web page to another by clicking on a link available on the web page, you will have a 6/17 chance of landing on web page 1, 2/17 chance of landing on web page 2, 3/17 chance of landing on web page 3, and 6/17 chance of landing on web page 4. Consequently, page 1 and page 4 are ranked equally highest. Page 3 is ranked next. Page 2 is ranked lowest.

With damping parameter d=$\frac{2}{3}$, the transition matrix is

$$A = d \begin{bmatrix} 0 & 0 & 0 & 1 \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 1 & 0 \end{bmatrix} + (1-d) \cdot \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{3}{4} \\ \frac{11}{36} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \\ \frac{11}{36} & \frac{5}{12} & \frac{1}{12} & \frac{1}{12} \\ \frac{11}{36} & \frac{5}{12} & \frac{3}{4} & \frac{1}{12} \end{bmatrix}$$

To find the PageRank vector $p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$, we write the equation as $(A - I_4)p = 0$, which is equivalent to a linear system of equations. The associated matrix is

$$\begin{bmatrix} \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{3}{4} \\ \frac{11}{36} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \\ \frac{11}{36} & \frac{5}{12} & \frac{1}{12} & \frac{1}{12} \\ \frac{11}{36} & \frac{5}{12} & \frac{3}{4} & \frac{1}{12} \end{bmatrix} \xrightarrow{\text{RREF}} \begin{bmatrix} 1 & 0 & 0 & -\frac{201}{220} \\ 0 & 1 & 0 & -\frac{9}{20} \\ 0 & 0 & 1 & -\frac{3}{5} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

From here, you can solve for the PageRank vector

$$p = \begin{bmatrix} \frac{201}{652} \\ \frac{99}{652} \\ \frac{132}{652} \\ \frac{220}{652} \end{bmatrix}$$

Therefore, with the damping parameter $d = \frac{2}{3}$, web page 4 is ranked highest, page 1 ranked second, page 3 ranked third, and page 2 ranked lowest.

Optimization of Single-Variable Functions

Lecture 10

Earlier, you learned how to solve linear programming problems by hand using the *simplex method*. The goal for the next few weeks is to learn numerical methods for *nonlinear programming problems* (more commonly called nonlinear optimization problems). These methods are, in general, tedious to do by hand, even with a pocket calculator.

A nonlinear optimization problem is a problem of finding min/max value of a nonlinear function f. Most functions from real-life applications have more than one variable (multivariable functions). We will start with optimization problems for functions of a single variable.

Let $f = f(x)$ be a single-variable function. How do you find min/max value of this function? Note that min/max don't always exist. For example, $f(x) = x$ doesn't have min/max on the entire real line, but it has both min and max on the interval $[-2, 3]$. In Calculus I, you learned the **Extreme Value Theorem:**

If f is a continuous on a closed interval $[a, b]$ then both $\min\limits_{[a,b]} f$ and $\max\limits_{[a,b]} f$ exist.

You also learned how to find min/max of a function using **Fermat's theorem**, which says that if f is differentiable then min/max, if occurs at $c \in (a, b)$, must be a critical number of f, i.e. $f'(c) = 0$.

However, the equation $f'(x) = 0$ is not always easy to solve. You will learn how to solve such an equation numerically later. The most natural way to find the min/max of a function f is perhaps by graphing f and locating the position of min/max on the graph. This visual method sounds attractive, but it requires you to be able to evaluate $f(x)$ at a lot of values

of x.

Imagine that you are somewhere on a mountain (on a 2D plane) and you want to reach the bottom. The mountain can be thought as the graph of a function f. You want to reach its minimum. You don't know what this function is, so you can't graph it. However, with proper measuring instrument, you can tell the slope of the mountain at your position. This information turns out to be enough to solve the problem. Let's consider a simple and intuitive algorithm called **gradient descent** to find minimum. The counterpart to find maximum is called *gradient ascent*. However, since maximum can be converted into minimum easily by multiplying the function by -1, we will only consider the problem of finding minimum.

Starting at an *initial guess* x_0 for the location of the minimum. We then update this guess by x_1 , then update it again by x_2 , and so on. Intuitively, $x_{n+1} - x_n$ should be propotional to $-f'(x_n)$.

$$x_{n+1} = x_n - \alpha f'(x_n)$$

where α is called the learning rate.

Example:

Approximate the minimum of $f(x) = x^2$ with $n = 5$, initial guess $x_0 = 1$, and learning rate $\alpha = 0.4, 0.6, 1, 2$.

Example:

Approximate the maximum value of $y = y(x)$ where function y satisfies the differential equation $y' = y^2 - x, y(0) = 0.7$.

Lecture 11

Gradient descent algorithm for finding minimum:

- Guess x_0 (location of minimum)
- Update: $x_{n+1} = x_n - \alpha f'(x_n)$

Gradient ascent algorithm for finding maximum:

- Guess x_0 (location of maximum)
- Update: $x_{n+1} = x_n + \alpha f'(x_n)$

Gradient descent/ascent algorithms are *short-sighted* in nature. They don't see the big picture (the graph) of the function. They only give you a *local* minimum and *local* maximum.

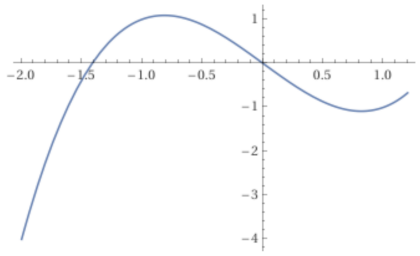
Example:

Let $f(x) = x^3 - 2x$ Find the minimum and maximum value of $f(x)$ on the interval $[-2, 1.2]$ using the Gradient descent/ascent method. Pick an initial guess x_0 and a learning rate α.

$f'(x) = 3x^2 - 2$

Gradient descent: $x_{n+1} = x_n - \alpha(3_n^2 - 2)$

Gradient ascent: $x_{n+1} = x_n + \alpha(3_n^2 - 2)$



If $x_0 = 0$, the Gradient descent method will give you a local minimum at about 0.8.

If $x_0 = 1$, the Gradient ascent method will give you a local maximum at about 1.2.

If x_n falls out of bound, i.e. the interval $[-2, 1.2]$, then stop. If your algorithm stops too early, perhaps that is because α is too large. You can reduce α and try again.

Bisection method:

This is a root-finding algorithm relying on Intermediate Value Theorem.

Motivating example:

I have a natural number between 1 and 100 in mind. I will tell you whether my number is bigger than or smaller than the number you guess. How can you guess it with the least number of guesses? Guess 50 first. Then guess either 25 or 75 based on my answer to your first guess. And so on. This is the key idea of bisection method. Its mathematical basis is the *Intermediate Value Theorem:*

If f is continuous on $[a, b]$ and $f(a)$, $f(b)$ have different signs, then there exists $c \in (a, b)$ such that $f(c) = 0$.

Example:

Find $\sqrt{2}$ using the bisection method. Note that $\sqrt{2}$ is a root of $f(x) = x^2 - 2$. We have $f(1) = -1 < 0$ and $f(2) = 2 > 0$.

Lecture 12

Bisection method:

Bracket the root of $f(x) = 0$ with shorter and shorter intervals. A search interval is an interval $[a, b]$ such that $f(a)f(b) < 0$. You calculate $c = \frac{a+b}{2}$. If $f(c)$ has the same sign as $f(a)$, you narrow the search interval to $[c, b]$. Otherwise, you narrow it to $[a, c]$. After n times of dividing, the search interval is of length $\frac{b-a}{2^n}$. The true solution lies on this interval.

If you pick the midpoint of this interval, called x_*, then

$$|x_{\text{exact}} - x_*| < \frac{b-a}{2^{n+1}}$$

If you want the error to be lesson than some $\epsilon > 0$, you need to pick sufficiently large n such that $\frac{b-a}{2^{n+1}} < \epsilon$

Example:

Find the intersection of the line $y = x$ and the curve $y = e^{-x}$. The x-coordinate of the intersection is a root of $f(x) = x - e^{-x}$. You can check that $f'(x) > 0$, so f has a most one root. You can also check that $f(0) < 0$ and $f(1) > 0$. Thus, f has exactly one root, and it is between 0 and 1. Use the Bisection method to find the root with allowable error 0.01.

Newton-Raphson method (1669, 1699):

Starting at an initial guess x_0 , we approximate f by a linear function

$$f(x) \approx L(x) = f(x_0) + (x - x_0)f'(x_0)$$

And then solve the equation $L(x) = 0$. The result is called x_1

$$x_1 = x_0 - f(x_0)f'(x_0)$$

Then view x_1 as a new guess for root and repeat the process to get $x_2, x_3, \ldots$ In general,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

This sequence, if converges, converges very fast to a solution to a root.

Example:

Do the previous example using Newton-Raphson method with initial guess $x_0 = 0$ with an allowable error of 0.0001. That is, you stop when $|x_{n+1} - x_n| < 0.0001$ and take x_{n+1} as the final answer.

Multivariable Functions – Level Sets and Partial Derivatives

Lecture 13

Goal for the next couple weeks: optimization of multivariable functions.

Examples of multivariable functions:

$$f(x,y) = x + y$$

$$f(x,y) = \sin(y)$$

$$f(x,y) = 2$$

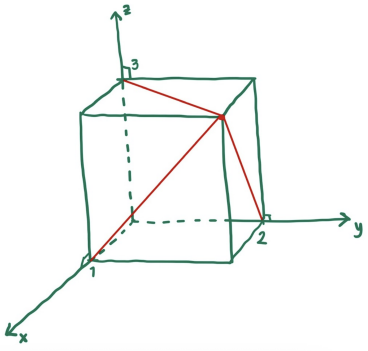
$$f(x,y,z) = yz + x$$

$$f(x,y,z) = xe^y$$

Graph of a function $f(x)$ is the collection of points (x,y) on the plane such that $y = f(x)$. It is a curve.

Graph of a function $f(x,y)$ is the collection of points (x,y,z) in the space such that $z = f(x,y)$. It is a surface.

How do you locate the position of the point $(x,y,z) = (1,2,3)$?



You can graph a function on Python using the package **plotly** or **matplotlib**. The package matplotlib is more commonly used. However, it doesn't allow interaction with the graph such as zooming, dragging, rotating. To graph the function $f(x, y) = x^2 + y^2$ with plotly, use the following code:

```
from plotly.graph_objects import *
from numpy import *
def f(x,y):
    return x**2 + y**2
x = linspace(-5, 5, 50)
y = linspace(-5, 5, 50)
X, Y = meshgrid(x, y)
Z = f(X,Y)
fig1 = Figure(data=[Surface(z=Z, x=X, y=Y)])
fig1.show()
```

The c-level set (or level curve) of f is the collection of points (x, y) on the plane such that $f(x, y) = c$.

For example, consider the function $f(x, y) = x^2 + y^2$. The 0-level set of f is the origin. The 1-level set of f is the unit circle. The 2-level set of f is the circle of radius 2. The (-1)-level set of f is the empty set. The collection of all level sets of f is called the contour map.

You can draw the contour map on Python with the plotly package as follows.

```
fig2 = Figure(data=Contour(z=Z,x=x,y=y))
fig2.update_layout(width=600,height=600)
fig2.show()
```

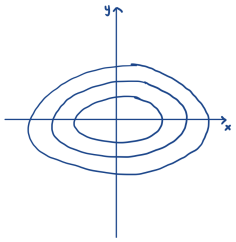
Lecture 14

Contour map of $f(x, y) = x^2 + 4y^2$.

The 0-level set of f is the origin.

The 1-level set of f is the unit circle.

The (-1)-level set of f is the empty set.



Practice drawing contour maps on the worksheet.

Partial derivatives:

Let $f(x, y) = xy + e^{xy^2}$.

The partial derivative of f with respect to x is

$$fx = y + y^2 e^{xy^2} \quad \text{(fix y, differentiate with respect to x)}$$

The partial derivative of f with respect to y is

$$fy = x + 2xyexy2 \quad \text{(fix x, differentiate with respect to y)}$$

Some practice on the worksheet.

Lecture 15

Last time, you learned how to compute partial derivatives. Note that f_x and f_y are functions of x and y themselves.

$f_x(a, b)$ is the rate of change of the function $t \mapsto f(t, b)$ at $t = a$. Geometrically, $f_x(a, b)$ is the slope at (a, b) of the curve obtained by intersecting the graph of f with the plane $x = a$.

In other words, it is the rate of change of f at (a, b) in the direction of the x-axis. Similarly, $f_y(a, b)$ is the rate of change of f at (a, b) in the direction of the y-axis.

Gradient vector is simply the vector consisting of all partial derivatives. It is the equivalence of derivative of a single-variable function.

Example:

$$f(x, y) = x \sin(xy)$$
$$\nabla f(x, y) = (f_x, f_y) = (\sin(xy) + xy \cos(xy),\ x^2 \cos(xy))$$

Example:

$$f(x, y, z) = xy + z^2$$
$$\nabla f(x, y, z) = (f_x, f_y, f_z) = (y, x, 2z)$$

The gradient vector turns out to be all you need to compute the rate of change of f in *any* direction. We need to make precise that "direction" means. Mathematically, it is a vector. A vector is an arrow with a length and a direction.

$u = (u_1, u_2)$ represents an arrow starting at the origin $(0, 0)$ and ending at the point (u_1, u_2). The length of u is computed by the Pythagorean theorem:

$$|u| = \sqrt{u_1^2 + u_2^2}$$

We often need to compare two different directions. In that sense, we are interested in computing the angle between two vectors. It is surprisingly easy to compute (proof is a bit involved, using the Cosine Law of a triangle):

$$\cos\theta = \frac{u \cdot v}{|u||v|}$$

where $u \cdot v$ denotes the *dot product* of two vectors. It is computed as follows:

$$u \cdot v = u_1 v_1 + u_2 v_2$$

where $u = (u_1, u_2)$ and $v = (v_1, v_2)$.

Do a practice problem on the worksheet.

Gradient and Directional Derivatives

Lecture 16

At the point (x, y), the rate of change of f in the direction of *unit* vector u is

$$D_u f(x, y) = \nabla f(x, y) \cdot u$$

The reason why u must be a unit vector is because we are only interested in the direction of u. We normalize its length to be 1.

Question:

For what direction u is the rate of change $D_u f(x, y)$maximum?

$$D_u f(x, y) = \nabla f(x, y) \cdot u = |\nabla f(x, y)||u| cos\theta = |\nabla f(x, y)| cos\theta \leq |\nabla f(x, y)|$$

The equality occurs when $\theta = 0$. That is, when u points to the direction of $\nabla f(x, y)$. In other words, the maximum rate of increase is in the direction

$$u = \frac{\nabla f}{|\nabla f|}$$

and the maximum rate is

$$\max D_f(x, y) = |\nabla f(x, y)|$$

.

The gradient vector is the direction of the *steepest ascent*. The negative of the gradient vector is the direction of *steepest descent*. The direction tangent to the level set is the

direction of zero rate of increase.

Work on some problems on the worksheet.

Find min/max of a multivariable function:

The familiar procedure for optimizing a single-variable function $f(x)$ on the interval $[a, b]$ is, first, to find all critical numbers of f, and then to compare the values of f at these critical numbers and at the endpoints of the interval. You determine the critical numbers because they are potentially places where f attains min/max.

The same procedure holds for multivariable functions. The equivalence of critical numbers are critical points (x, y) at which $\nabla f(x, y) = (0, 0)$. Critical points are potentially places where f attains min/max. However, the "endpoints of the interval" will now have to be understood as the *boundary of the region* of (x, y) where you are to look for min/max. Finding min/max on the boundary is a more complex problem.

Example:

Find min/max of the function $f(x, y) = x^3 + xy^2 - 4x + 2y$ on the disk $x^2 + y^2 \leq 4$.

$$\nabla f(x, y) = (f_x, f_y) = (3x^2 + y^2 - 4, 2xy + 2)$$

$\nabla f(x, y) = (0, 0)$ if and only if $3x^2 + y^2 - 4 = 0$ and $2xy + 2 = 0$. From the second equation, you get $y = -\frac{1}{x}$. Substitute this y into the first equation:

$$3x^2 + 1x^2 - 4 = 0$$

which can be rewritten as $3x^2 + y^2 - 4 = 0$. You can factor the polynomial:

$$(x^2 - 1)(3x^2 - 1) = 0$$

You get $x = \pm 1, \pm\frac{1}{\sqrt{3}}$. Therefore, you get 4 critical points:

$$(x, y) = (1, -1),\ (-1, 1),\ \left(-\frac{1}{\sqrt{3}}, 3\right),\ \left(\frac{1}{\sqrt{3}}, -3\right)$$

On the boundary of the disk $x^2 + y^2 \leq 4$, we have $x^2 + y^2 = 4$. We will need to find min/max of $f(x, y)$ on the boundary, i.e. under the constraint $x^2 + y^2 = 4$.

You have $y^2 = 4 - x^2$. Then

$$f(x, y) = x^3 + x(4 - x^2) - 4x + 2y = 2y$$

On the circle of radius 2,

$$\max 2y = 4 \quad \text{(attained when } x = 0,\ y = 2)$$
$$\min 2y = -4 \quad \text{(attained when } x = 0,\ y = -2)$$

Therefore,

$$\max f = \max\{f(-1, 1),\ f(1, -1),\ f\left(-\frac{1}{\sqrt{3}}, 3\right),\ f\left(\frac{1}{\sqrt{3}}, -3\right),\ f(0, 2)\} = \ldots$$
$$\min f = \min\{f(-1, 1),\ f(1, -1),\ f\left(-\frac{1}{\sqrt{3}}, 3\right),\ f\left(\frac{1}{\sqrt{3}}, -3\right),\ f(0, -2)\} = \ldots$$

Gradient Descent/Ascent Algorithm

Lecture 17

Goal for today: Gradient Descent/Ascent method for finding min/max of multivariable function f(x,y).

Key observation:

At a point (a, b),

- function f increases at the fastest in the direction of $\nabla f(a, b)$
- function f decreases at the fastest in the direction of $-\nabla f(a, b)$
- function is *level* in the direction orthogonal to $\nabla f(a, b)$

Gradient Ascent method for finding local maximum:

Start with an initial guess (x_0, y_0).

Then follow the direction of fastest rate of increase, which is $\nabla f(x_0, y_0)$:

$$(x_1, y_1) = (x_0, y_0) + \alpha \nabla f(x_0, y_0)$$

Here $\alpha > 0$ is called the learning rate. Then follow the direction of fastest rate of increase, which is $\nabla f(x_1, y_1)$:

$$(x_2, y_2) = (x_1, y_1) + \alpha \nabla f(x_1, y_1)$$

In general,

$$(x_{n+1}, y_{n+1}) = (x_n, y_n) + \alpha \nabla f(x_n, y_n)$$

This equation is in fact a couple of equations:

$$x_{n+1} = x_n + \alpha f_x(x_n, y_n)$$

$$y_{n+1} = y_n + \alpha f_y(x_n, y_n)$$

Gradient Descent method for finding local minimum:

You follow the direction of fastest decrease instead, which is $-\nabla f$. The same recursive formula as above, except that the plus sign is replaced by the minus sign.

$$x_{n+1} = x_n - \alpha f_x(x_n, y_n)$$

$$y_{n+1} = y_n - \alpha f_y(x_n, y_n)$$

Practice on the worksheet.

Lecture 18

Find min/max of a multivariable function: The familiar procedure for optimizing a single-variable function $f(x)$ on the interval $[a, b]$ is, first, to find all critical numbers of f, and then to compare the values of f at these critical numbers and at the endpoints of the interval. You determine the critical numbers because they are potentially places where f attains min/max.

The same procedure holds for multivariable functions. The equivalence of critical numbers are critical points (x, y) at which $\nabla f(x, y) = (0, 0)$. Critical points are potentially places where f attains min/max. However, the "endpoints of the interval" will now have to be understood as the *boundary of the region* of (x, y) where you are to look for min/max. Finding min/max on the boundary is a more complex problem.

Example: Find min/max of the function $f(x,y) = x^3 + xy^2 - 4x + 2y$ on the disk $x^2 + y^2 \leq 4$.

$$\nabla f(x,y) = (f_x, f_y) = (3x^2 + y^2 - 4, 2xy + 2)$$

$\nabla f(x,y) = (0,0)$ if and only if $3x^2 + y^2 - 4 = 0$ and $2xy + 2 = 0$. From the second equation, you get $y = -\frac{1}{x}$. Substitute this y into the first equation:

$$3x^2 + 1x^2 - 4 = 0$$

which can be rewritten as $3x^2 + y^2 - 4 = 0$. You can factor the polynomial:

$$(x^2 - 1)(3x^2 - 1) = 0$$

You get $x = \pm 1, \pm\frac{1}{\sqrt{3}}$. Therefore, you get 4 critical points:

$$(x,y) = (1,-1),\ (-1,1),\ \left(-\frac{1}{\sqrt{3}}, 3\right),\ \left(\frac{1}{\sqrt{3}}, -3\right)$$

On the boundary of the disk $x^2 + y^2 \leq 4$, we have $x^2 + y^2 = 4$. We will need to find min/max of $f(x,y)$ on the boundary, i.e. under the constraint $x^2 + y^2 = 4$.

You have $y^2 = 4 - x^2$. Then

$$f(x,y) = x^3 + x(4 - x^2) - 4x + 2y = 2y$$

On the circle of radius 2,

$$\max 2y = 4 \quad \text{(attained when } x = 0,\ y = 2)$$

$$\min 2y = -4 \quad \text{(attained when } x = 0,\ y = -2)$$

Therefore,

$$\max f = \max\{f(-1,1),\ f(1,-1),\ f\left(-\frac{1}{\sqrt{3}},3\right),\ f\left(\frac{1}{\sqrt{3}},-3\right),\ f(0,2)\} = \dots$$
$$\min f = \min\{f(-1,1),\ f(1,-1),\ f\left(-\frac{1}{\sqrt{3}},3\right),\ f\left(\frac{1}{\sqrt{3}},-3\right),\ f(0,-2)\} = \dots$$

Example: Find min/max of the function $f(x,y) = -x + xy + y^2$ on the triangle with vertices at $(-3,0), (-2,2), (0,0)$.

Lecture 19

Work on the following problem on the worksheet:

Find min/max of the function $f(x,y) = -x + xy + y^2$ on the triangle with vertices at $(-3,0), (-2,2), (0,0)$.

Multivariable Optimization with/without Constraints

Lecture 20

Theorem:

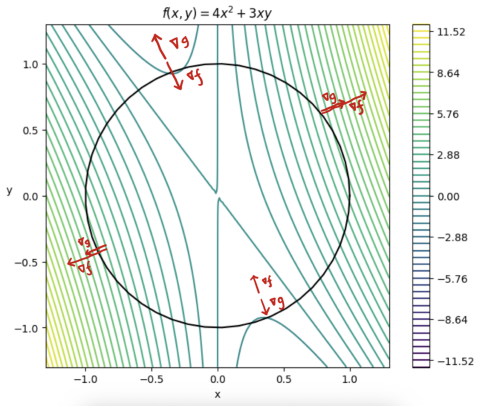
If f is continuous on a closed and bounded region D then it has minimum and maximum values on D.

Finding min/max of a function $f(x, y)$ on the boundary of a region D is equivalent to finding min/max of f under a *constraint*. This constraint is of equality type.

Example:

Find min/max of $f(x, y) = 4x^2 + 3xy$ on the boundary of the unit circle.

The boundary unit circle is closed and bounded, so f admits a minimum value and a maximum value there. The problem is equivalent to finding min/max of f under the constraint $x^2 + y^2 = 1$. Last time, we consider an example where the function is significantly reduced under such a constraint. Unfortunately, this scenario is not always the case, such as in the present example. You can write $y = \pm\sqrt{1 - x^2}$ and plug it in the formula of f and then reduce the problem to one-variable optimization. This is quite inconvenient, especially because you have to consider both possible formula for y.



Let $f(x, y) = 4x^3 + 3xy$ and $g(x, y) = x^2 + y^2$. We want to minimize/maximize f under the constraint $g(x, y) = 0$. The min/max of f is attained where the level curves of f just touches the 0-level curve of g. At this touching point, the level curve of f is tangent to the level curve of g. Therefore, the gradient of f (which is perpendicular to the level curve of f) must be parallel to the gradient of g which is perpendicular to the 0-level curve of g):

$$\nabla f = \lambda \nabla g$$

where λ is a number called Lagrange multiplier. This equation together with the equation $g = 0$ constitutes a system of 3 equations and 3 unknowns x, y, λ.

$$f_x = 8x + 3y, \quad g_x = 2x$$

$$f_y = 3x, \quad g_y = 2y$$

The system to solve for x, y, λ is

$$8x + 3y = \lambda 2x$$

$$3x = \lambda 2y$$

$$x^2 + y^2 = 1$$

From here, you can get $\lambda = -\frac{1}{2}$ and $\lambda = \frac{9}{2}$. Then you substitute it back to the equation to solve for x, y. You will get 4 points (x, y).

Then, you compare the values of f at these 4 points. The largest of them is the maximum of f. The smallest of them is the minimum of f.

Lecture 21

Theorem:

If f is continuous on a closed and bounded region then it has minimum and maximum values on that region.

Recall the procedure to find min/max of a function f on a region:

- Step 1: find all critical points of f inside that region
- Step 2: find min/max of f on the boundary
- Step 3: compare and conclude

Lagrange multiplier method is specialized for Step 2, although it is not the only method that is available.

Work on the example on the worksheet:

Find the point on the ellipse $x^2 + 4y^2 = 1$ that is closest to the point $(2, 3)$.

The objective function is: $f(x, y) = (x - 2)^2 + (y - 3)^2$.

We want to minimize this function subject to the constraint $g(x, y) = 0$ where $g(x, y) = x^2 + 4y^2 - 1$.

Computation of partial derivatives:

$$f_x = 2(x - 2), \quad f_y = 2(y - 3)$$

$$g_x = 2x, \quad g_y = 8y$$

The system to solve for x, y, λ:

$$2(x-2) = 2\lambda x$$

$$2(y-3) = 8\lambda y$$

$$x^2 + 4y^2 = 1$$

From the first equation, we get

$$x = \frac{2}{1-\lambda}.$$

From the second equation, we get

$$y = \frac{3}{1-4\lambda}.$$

Substituting these into the last equation, we get

$$\frac{4}{(1-\lambda)^2} + \frac{36}{(1-4\lambda)^2} = 1$$

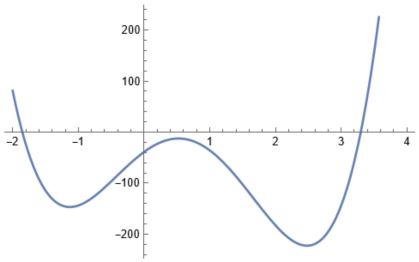
which is equivalent to

$$4(1-4\lambda)^2 + 36(1-\lambda)^2 = (1-\lambda)^2(1-4\lambda)^2$$

which is equivalent to

$$16\lambda^4 - 40\lambda^3 - 67\lambda^2 + 94\lambda - 39 = 0.$$

You can solve this equation numerically using Newton's method.



$$f(\lambda) = 16\lambda^4 - 40\lambda^3 - 67\lambda^2 + 94\lambda - 39$$

$$\lambda_{n+1} = \lambda_n - \frac{f(\lambda_n)}{f'(\lambda_n)}$$

Note: you can use Newton's method to find roots of the function

$$g(\lambda) = \frac{4}{(1-\lambda)^2} + \frac{36}{(1-4\lambda)^2} - 1$$

Differential equation: Separation of Variables and Integrating Factor Methods

Lecture 22

A differential equation is an equation involving a function, its variable(s), and one or more of its derivatives.

Example:

$$y' + y = x$$

$$y'' = yy'$$

$$u_x = x^2 uy$$

A natural phenomenon is usually modeled as a differential equation. A model is a simplification of a natural phenomenon in order for us to describe it in terms of equations.

Example:

1. Population growth: $y(t)$ = population at time t.

   You can imagine that a bigger population grows faster than a small population. It is reasonable to assume that the rate of change of population is proportional to the population itself. This argument produces a differential equation $y'(t) = \alpha y(t)$. Usually, it is written as $y' = \alpha y$. The solution is $y = ce^{\alpha t}$.

   This model is called Malthus' model (1798). It describes well the population for a short time. But it has a fatal implication for long time: that the population grows

exponentially. Another model is proposed by Verhulst (1838) called logistic model: $y' = \alpha y\left(\frac{1-y}{M}\right)$. Here, M is called the sustainable population. There are many more population models.

2. Heat transfer: $u_t = cu_{xx}$ where $u = u(x,t)$ is the temperature at position x at time t.

3. Wave propagation: $u_{tt} = cu_{xx}$

Lecture 23

Some basic analytic methods to solve an ordinary differential equation (ODE):

1. Separation of variables: y$' = f(x)g(y)$

   Example: $y' = x^2y$, $y(1) = 0$

2. Integrating factor:$y' + p(x)y = q(x)$

   Example: $y' + y = x$, $y(0) = 1$

Euler's method is a numerical method for ODE $y' = f(x,y)$. A more general method based on the same idea is the *finite difference method*.

Example: $y' = x + y^2, y(0) = -1$

Finite Difference Method for First/Second Order ODE and Heat Equation

Lecture 24

Euler's method is a numerical method for ODE $y' = f(x, y)$. A more general method based on the same idea is the finite difference method.

Example: $y' = xy, y(0) = 1$

Python code:

```
from numpy import *
from matplotlib.pyplot import *
# solve y'=xy with initial condition y(0)=y0
y0 = 1
# with step size h
h = 0.2
# the number of steps
N = 10
# Array x = [x0,x1,...,xN]
x = linspace(0,N*h,N+1)
# Array y = [y0,y1,...,yN]
y = zeros(N+1)
y[0] = y0
for i in range(N):
        y[i+1] = y[i] + h*x[i]*y[i]
print(x)
print(y)
plot(x,y)
```

```
show()
```

Lecture 25

Euler's method for the initial value problem $y' = xy, y(0) = y0$

$$y_n = y(x_n)$$

The differential equation is approximated by

$$y_{n+1} = y_n + hx_n y_n$$

Python code:

```
from numpy import *
from matplotlib.pyplot import *
# solve y'=xy with initial condition y(0)=y0
y0 = 1
# with step size h
h = 0.2
# the number of steps
N = 10
# Array x = [x0,x1,...,xN]
x = linspace(0,N*h,N+1)
# Array y = [y0,y1,...,yN]
y = zeros(N+1)
y[0] = y0
for i in range(N):
        y[i+1] = y[i] + h*x[i]*y[i]
print(x)
print(y)
```

```
plot(x,y)
show()
```

This equation has an exact solution, which you can find by hand using the separation of variables method or the integrating factor method: $y = y_0 e^{\frac{x^2}{2}}$. You can draw the exact solution together with the approximate solution to compare them. Before the line show() in the code, add the following line:

```
plot(x, y0*e**(x**2/2))
```

Euler's method is an example of a more general method for numerically solving differential equations called *finite difference method*. The idea is that you approximate the derivative by the difference quotient.

Example:

$$\begin{aligned} y'(x) &\approx \frac{y(x+h) - y(x)}{h} \\ y''(x) &\approx \frac{y'(x+h) - y'(x)}{h} \\ &\approx \frac{\dfrac{y(x+h) - y(x)}{h} - \dfrac{y(x) - y(x-h)}{h}}{h} \\ &= \frac{y(x+h) - 2y(x) + y(x-h)}{h^2} \end{aligned}$$

Example:

$$y'' + 2xyy' + y^2 = 0, \quad y(-1) = 1, \quad y'(-1) = 1$$

$$y(x_n) = y_n$$

The differential equation is approximated by

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} + 2x_n y_n \frac{y_n - y_{n-1}}{h} + y_n^2 = 0$$

Therefore,

$$y_{n+1} = 2y_n - y_{n-1} - 2hx_n y_n(y_n - y_{n-1}) - h^2 y_n^2$$

This initial value problem has an exact solution $y = \frac{2}{1+x^2}$.

Python code:

```
from numpy import *
from matplotlib.pyplot import *
# solve y''+2xyy'+y^2=0 with initial condition y(x0)=a
    and y'(x0)=b
x0 = -1
a = 1
b = 1
# with step size h
h = 0.01
# the number of steps
N = 200
# Array x = [x0,x1,...,xN]
x = linspace(x0,x0+N*h,N+1)
# Array y = [y0,y1,...,yN]
y = zeros(N+1)
y[0] = a
y[1] = a+b*h
for n in range(1,N):
        y[n+1] = 2*y[n] - y[n-1] - 2*h*x[n]*y[n]*(y[n]-
            y[n-1])-h**2*y[n]**2
plot(x,y)
plot(x,2/(1+x**2))
show()
```

Lecture 26

Euler's method is an example of a more general method for numerically solving differential equations called *finite difference method*. The idea is that you approximate the derivative by the difference quotient.

Example:

$$\begin{aligned}
y'(x) &\approx \frac{y(x+h)-y(x)}{h} \\
y'(x) &\approx \frac{y(x)-y(x-h)}{h} \\
y''(x) &\approx \frac{y'(x+h)-y'(x)}{h} \\
&\approx \frac{\dfrac{y(x+h)-y(x)}{h} - \dfrac{y(x)-y(x-h)}{h}}{h} \\
&= \frac{y(x+h)-2y(x)+y(x-h)}{h^2}
\end{aligned}$$

To see how good these approximations are when h gets smaller and smaller, we use the **Taylor expansion:**

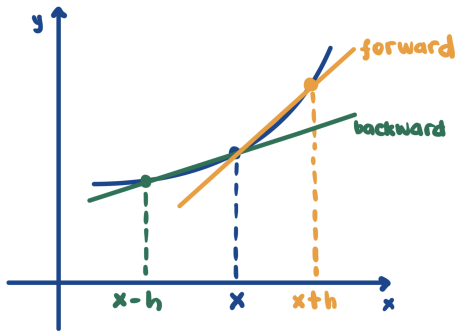
$$y(x+h) = y(x) + y'(x)h + \frac{y''(x)}{2!}h^2 + \frac{y'''(x)}{3!}h^3 + O(h^4)$$

By replacing h with $-h$, we see that

$$\begin{aligned}
y'(x) &= \frac{y(x+h)-y(x)}{h} + O(h) \\
y'(x) &= \frac{y(x)-y(x-h)}{h} + O(h) \\
y'(x) &= \frac{y(x+h)-y(x-h)}{2h} + O(h^2) \qquad y''(x) = \frac{y(x+h)-2y(x)+y(x-h)}{h^2} + O(h^2)
\end{aligned}$$

The $O(h)$ notation means "of order h", which means a quantity whose absolute value is less than or equal to ch for some constant c.

The first equation is called *forward difference approximation* for derivative. The second equation is called *backward difference approximation* for derivative. The third equation is called *centered difference approximation* for derivative. The fourth equation is called *centered difference approximation* for second derivative.



Example:

$y'' + xy' + y = 1, y(-1) = 2, y'(-1) = 1$

$$y(xn) = yn$$

Let us use the centered difference approximation for y'':

$$y''(x_n) \approx \frac{y_{n+1} - 2y_n + y_{n-1}}{h^2}$$

This approximation is of order h^2 as h gets small.

If you choose the approximation $y'(x_n) \approx \frac{y_n - y_{n-1}}{h}$, then the differential equation is approximated by

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} + x_n \frac{y_n - y_{n-1}}{h} + y_n = 1$$

This yields

$$y_{n+1} = \left(2 - h^2 - hx_n\right) y_n + (hx_n - 1)\, y_{n-1} + h^2 \qquad (1)$$

Because the approximation of the derivative is of order h while the approximation of the second derivative is of order h^2, the combined approximation is of order h (the worse of the two approximations).

On the other hand, if you choose the approximation $y'(x_n) \approx \dfrac{y_{n+1} - y_{n-1}}{2h}$, which is of order h^2, then the combined approximation is also of order h^2:

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} + x_n \frac{y_{n+1} - y_{n-1}}{2h} + y_n = 1$$

This yields

$$y_{n+1} = \frac{(2 - h^2)y_n + \left(\dfrac{hx_n}{2} - 1\right) y_{n-1} + h^2}{1 + \dfrac{hx_n}{2}} \qquad (2)$$

To compute y_{n+1}, you need to know y_n and y_{n-1}. You know $y_0 = 2$. How to find y_1?

Once you have y_1, you will be able to find y_2, y_3, y_4, ... using the above recursive formula ((1) or (2)), depending on how you approximate the first derivative.

If you choose the forward difference approximation for y', then y_1 is easy to find:

$$y'(x_0) \approx \frac{y_1 - y_0}{h}$$

$$y_1 \approx y_0 + hy'(x_0) = 2 + h$$

If you choose the centered difference approximation for y', then

$$y'(x_0) \approx \frac{y_1 - y_{-1}}{2h}$$

which yields $y_1 = y_{-1} + 2h$. Applying (2) for $n = 0$, you get y_1 in terms of y_0 (known)

and y_{-1}. From there, you can solve for y_1 (as well as y_{-1}).

Exercise:

With step size $h = 0.1$, approximate $y(-0.7)$ using the recursive formula (1). Compare it with the true solution $y(x) = 1 + e^{(1-x^2)/2}$.

Python code: (using recursive formula (1))

```
from numpy import *
from matplotlib.pyplot import *
# solve y''+xy'+y=1 with initial condition y(x0)=a and
    y'(x0)=b
x0 = -1
a = 2
b = 1
# with step size h
h = 0.1
# the number of steps
N = 20
# Array x = [x0,x1,...,xN]
x = linspace(x0,x0+N*h,N+1)
# Array y = [y0,y1,...,yN]
y = zeros(N+1)
y[0] = a
y[1] = a+b*h
for n in range(1,N):
        y[n+1] = (2-h**2-h*x[n])*y[n] + (h*x[n]-1)*y[n
            -1] + h**2
plot(x,y)
plot(x,1+e**((1-x**2)/2))
show()
```

Lecture 27

Heat equation on a rod: $u_t = cu_{xx}$

This is a partial differential equation (PDE) rather than an ordinary differential equation (ODE) because the unknown u is a function of more than one variable. Here, $u = u(x,t)$ is the temperature at position x on the rod at time t, and c is the *thermal diffusivity coefficient*.

Suppose the rod is the interval $[0,1]$.

Initial condition: $u(x,0) = u_0(x)$

Boundary conditions: $u(0,t) = f_1(t)$ and $u(1,t) = f_2(t)$

The initial condition and boundary conditions are typically controllable or measurable. So, u_0, f_1, f_2 can be considered as something given rather than something to be found.

Discretize the rod: $0 = x_0 < x_1 < \ldots < x_n = 1$ with *spatial step size* $h = x_i - x_{i-1}$

Discretize the time: $0 < t_1 < t_2 < \ldots$ with *time step size* $\tau = t_j - t_{j-1}$

The partial differential evaluated at (x_i, t_j):

$$u_t(x_i,t_j) = c\,u_{xx}(x_i,t_j) \qquad (1)$$

Let $u_{i,j} = u(x_i,t_j)$. Approximate the derivatives using finite difference method:

$$u_t(x_i,t_j) \approx \frac{u(x_i,t_{j+1}) - u(x_i,t_j)}{\tau} = \frac{u_{i,j+1} - u_{i,j}}{\tau}$$
$$u_{xx}(x_i,t_j) \approx \frac{u(x_{i+1},t_j) - 2u(x_i,t_j) + u(x_{i-1},t_j)}{h^2} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

Equation (1) becomes:

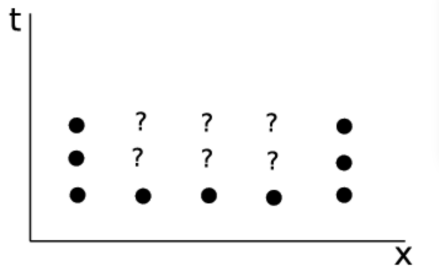
$$\frac{u_{i,j+1} - u_{i,j}}{\tau} = \frac{c}{h^2}\left(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}\right)$$

which is equivalent to

$$u_{i,j+1} = u_{i,j} + a\left(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}\right)$$

where $a = \dfrac{c\tau}{h^2}$. The number a is called the *Courant number* of the recursive formula. For the recursive formula to give reasonable results, it is required that $a < \dfrac{1}{2}$.

Example:



With $c = 0.5$, $u_0(x) = \sin(\pi x)$, $f_1(t) = f_2(t) = 0$, find the temperature at position $x = 0,\ 0.25,\ 0.5,\ 0.75,\ 1$ at time $t = 0.1$. Use spatial step size $h = 0.25$ and time step size $\tau = 0.05$.

Neural Network Method and Applications to Differential Equations

Lecture 28

Artificial intelligence is a big field that includes machine learning, a computational method that mimic human learning process. Machine learning includes neural networks, a computational method that mimic the structure of the brain. Neural networks includes deep learning, which is a multi-layer neural network.

A neural network consists of inputs, one or more hidden layers, activated hidden layers, and outputs. The below figure is a simple neural network with input x, one hidden layer Z, activated hidden layer $\sigma(Z)$, and output y.

$$x \to Z \to \sigma(Z) \to y$$

Our brain perceives things through many layers of neurons. The very first layer (xßZ) is when the input first gets into the brain. Z is a vector $Z = (z_1, z_2, \ldots, z_H)$, where z_i is what the i'th neuron perceives about x. The dependence of Z on x is perhaps simple. So, we assume that Z depends linearly on x.

$$z_i = a_i x + b_i$$

Then the data z_i will be processed by neuron i. We model this process by applying a nonlinear function σ, called an activation function, to z_i. So, each z_i becomes $\sigma(z_i)$.

$$\sigma(Z) = (\sigma(z_1), \sigma(z_2), \ldots, \sigma(z_H))$$

Then the data processed by neurons $1, 2, \ldots, H$ will be synthesized together to an output, our single perception of x. The synchronization of neurons will be assumed to be linear. So,

$$y = c_0 + c_1\sigma(z_1) + c_2\sigma(z_2) + \ldots + c_H\sigma(z_H)$$

The free coefficients $b_1, b_2, \ldots, b_H, c_0$ are called *biases*. The coefficients of the "variables", namely $a_1, a_2, \ldots, a_H, c_1, c_2, \ldots, c_H$ are called weights. The total number of weights and biases is therefore $3H + 1$.

Lecture 29

Let us consider an application of neuron networks for solving the differential equation $u' = x$ (without a specified initial condition). We know that the correct solution is $u(x) = \dfrac{x^2}{2} + C$.

Consider the simplest neuron network with one layer and one node:

$$x \to z \to \sigma(z) \to y = \tilde{u}(x)$$

Here, $z = ax + b$ and $y = c\sigma(z) + d = c\sigma(ax + b) + d$. The activation function can be any nonlinear function, such as $\sigma(x) = \sin x$.

We wish, by choosing suitable coefficients a, b, c, d and activation function σ, that $\tilde{u}$ satisfy $\tilde{u}' = x$ for all values of x. With a fixed choice of function σ, it seems almost impossible to choose a, b, c, d such that $\tilde{u}' = x$ for all x. The difficulty is the high demand "for all".

We will relax the requirement "for all" to for certain prescribed values of x, say $x_1, x_2, \ldots, x_n$, called grid points. That is, to require

$$\tilde{u}(x_i) - x_i = 0 \quad \text{for all } i = 1, 2, \ldots, n.$$

These are n equations, while we have 4 unknown coefficients to work with. The new

requirement demands that $n \leq 4$, which lays quite a severe restriction on n. We further relax this requirement by only requiring the quantity

$$\phi = \sum_{i=1}^{n} (\tilde{u}(x_i) - x_i)^2$$

to be as small as possible. ϕ is called an error function because it measures the error. It is also called a cost function because it is to be minimized. You can notice that ϕ only depends on a, b, c, d.

$$\phi(a, b, c, d) = \sum_{i=1}^{n} (c\sigma(ax_i + b) + d - x_i)^2$$

Therefore, the problem is to find 4 numbers a, b, c, d that minimize the function $\phi(a, b, c, d)$. You already knew how to do this numerically – by using Gradient Descent method:

Process:

- Initial guess: a_0, b_0, c_0, d_0
- Update:$(a_{n+1}, b_{n+1}, c_{n+1}, d_{n+1}) = (a_n, b_n, c_n, d_n) - \alpha\nabla\phi(a_n, b_n, c_n, d_n)$ where $\alpha > 0$ is the learning rate.

After a number of steps, we stop and take the value of a, b, c, d to be a_n, b_n, c_n, d_n. These values give us a function $\tilde{u}(x) = c\sigma(ax_i + b) + d$, which is an approximation of the true function u.

Since u is not unique ($u = \frac{x^2}{2} + C$), how do we know which of these possibilities of u is the one that $\tilde{u}$ approximates? Recall that $\tilde{u}$ approximates u in the sense that $\tilde{u} \approx x$.

Thus, $\tilde{u}$ may not approximate any functions of the form $\frac{x^2}{2} + C$.

Python code:

```
from autograd.numpy import*
from autograd import elementwise_grad as diff
```

```
from matplotlib.pyplot import*

x_grid = np.linspace(0.0,0.5,9)
a0,b0,c0,d0 = 1.,1.,1.,1.
alpha = 0.1
num_iter = 1000

def sigma(x):
        return sin(x)
def z(x,a,b,c,d): return a*x + b
def u(x,a,b,c,d): return c*sigma(z(x,a,b,c,d)) + d
def du(x,a,b,c,d): return diff(u,0)(x,a,b,c,d)
def phi(a,b,c,d):
        return sum([(du(t,a,b,c,d) - t)**2 for t in
   x_grid])

def gradd(phi,a0,b0,c0,d0,alpha,num_iter):
        a,b,c,d = a0,b0,c0,d0
        dphi = diff(phi,(0,1,2,3))
        for i in range(num_iter):
                gradient = array(dphi(a,b,c,d))
                a,b,c,d = array([a,b,c,d]) - alpha*
   gradient
        return a,b,c,d

a,b,c,d = gradd(phi,a0,b0,c0,d0,alpha,num_iter)

print('[a,b,c,d] =',array([a,b,c,d]))
print("Initial error: ",phi(a0,b0,c0,d0))
print("Final error: ",phi(a,b,c,d))
x = linspace(0,0.5,20)
```

```
du_appr = array([du(t,a,b,c,d) for t in x])
du_exact = x
print("Max absolute difference: ",max(abs(du_appr-
    du_exact)))
plot(x,du_appr,label='approximate sol')
plot(x,du_exact,label='exact sol')
legend()
show()
```