EEE319 Evaluation of Algorithms

Prof. Xinheng Wang

xinheng.wang@xjtlu.edu.cn

Office: EE512

Nonlinear programming

- Previous weeks
  - Optimisations: Methods and Applications
- Today
  - Algorithms
  - How to evaluate the performance of an algorithm?

- We have already learned a few optimisation methods.
- Question: How to use the optimisation method to solve practical problems?
- Answer: Algorithms
- Algorithm: An algorithm is a set of mathematical instructions or rules that, especially if given to a computer, will help to calculate an answer to a problem (Defined by Cambridge Dictionary).

Specification of an Algorithm

- Flowchart
- Pseudocode

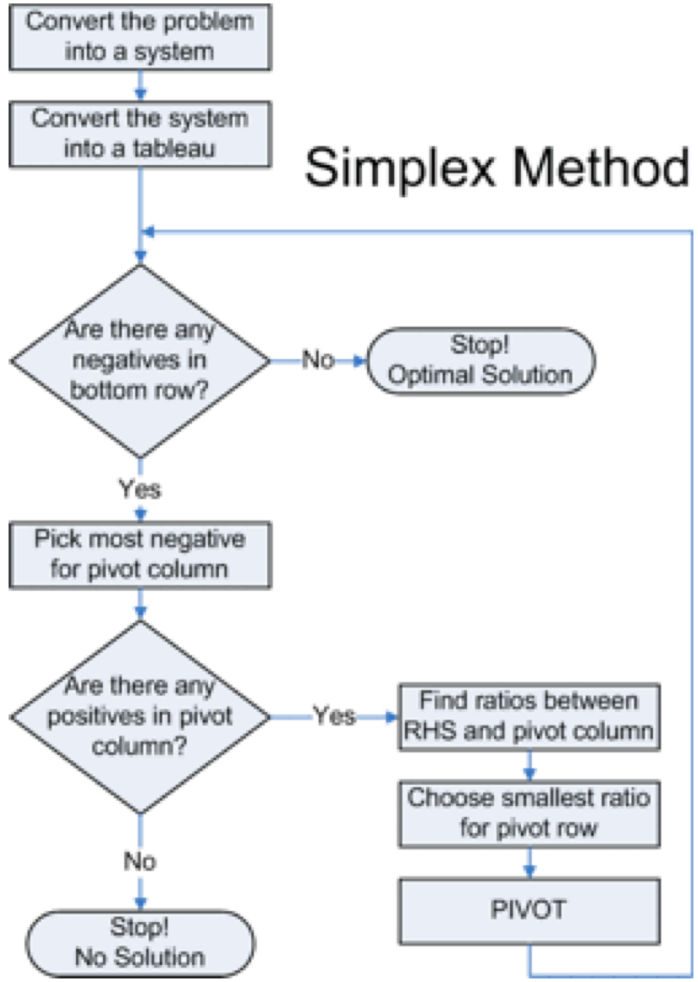
Specification of an Algorithm

- Flowchart
  - A **flowchart** is a diagram that represents a set of **instructions**. Flowcharts normally use standard symbols to represent the different types of instructions. These symbols are used to construct the flowchart and show the step-by-step solution to the problem.

Name	Symbol	Usage
Start or Stop		The beginning and end points in the sequence.
Process		An instruction or a command.
Decision		A decision, either yes or no.
Input or Output		An input is data received by a computer. An output is a signal or data sent from a computer.
Connector		A jump from one point in the sequence to another.
Direction of flow		Connects the symbols. The arrow shows the direction of flow of instructions.

Specification of an Algorithm

- Flowchart
  - Example: Flowchart of Simplex Method

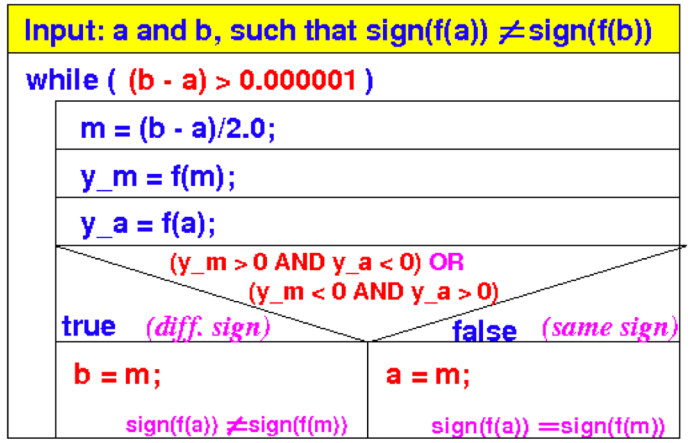


Specification of an Algorithm

- Pseudocode
  - **pseudocode** is an informal high-level description of the operating principle of an algorithm.
  - The purpose of using pseudocode is that it is easier for people to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm.
  - No standard for pseudocode syntax exists, as a program in pseudocode is not an executable program.

Specification of an Algorithm

- Pseudocode
  - **Some notations of pseudocode**
    - **INPUT** – indicates a user will be inputting something
    - **OUTPUT** – indicates that an output will appear on the screen
    - **WHILE** – a **loop** (**iteration** that has a **condition** at the beginning)
    - **FOR** – a counting loop (iteration)
    - **IF – THEN – ELSE** – a decision (**selection**) in which a choice is made



Pseudocode for bisection algorithm

Evaluation of an algorithm

- Characteristics evaluation – design phase
  - Complexity: What are its space and time complexities?
  - Convergence: Can the algorithm converge?
  - Robustness: Is algorithm robust against outliers?

- Performance evaluation - after implementation
  - Accuracy: How accurate is the algorithm?
  - Desirability: Is algorithm producing desired results?
  - Efficiency: How fast does the algorithm runs?
  - Convergence: Does the algorithm converge or fluctuate wildly?
  - Robustness: Is algorithm robust against outliers?

Evaluation of Algorithms

- Convergence
  - Definition by understanding: "Convergence" means that the algorithm is "**able to reliably find an answer**".
  - Definition specific to iterative algorithms: An iterative algorithm is said to converge when, as the iterations proceed, the output gets closer and closer to some specific value. More precisely, no matter how small an error range you choose, if you continue long enough the function will eventually stay within that error range around the final value.
  - Mathematical definition: It essentially means that "eventually" a sequence of elements get closer and closer to a single value. We call this single value the "limit".

Evaluation of Algorithms

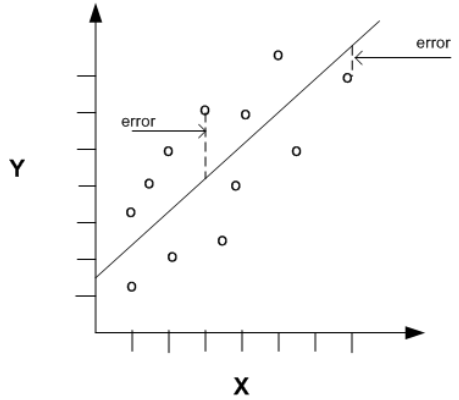
- Robustness
  - Robustness characterises the effectiveness of an algorithm.
  - Understanding: Robustness is the ability for the algorithm to provide useful information and consistent convergence.
  - Formal definition: In computer science, **robustness** is the ability of a computer system to cope with errors during execution and cope with erroneous input. (Wikipedia)
  - Robust algorithms: There exists algorithms that tolerate errors in the input or during the computation. In that case, the computation eventually converges to the correct output.

Evaluation of Algorithms

- Accuracy analysis
  - Probabilistic measures
    - RMSE: Root Mean Squared Error
    - MAE: Mean Absolute Error
  - Confusion matrix

Probabilistic

- Recall regression algorithms, one of the performance evaluation metrics is RMSE
- RMSE: Root Mean Squared Error



$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

where y_j is the actual value and $\hat{y}_j$ is predicted value.

- RMSE in a meaningful way

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(Predicted_j - Actual_j)^2}$$

Probabilistic

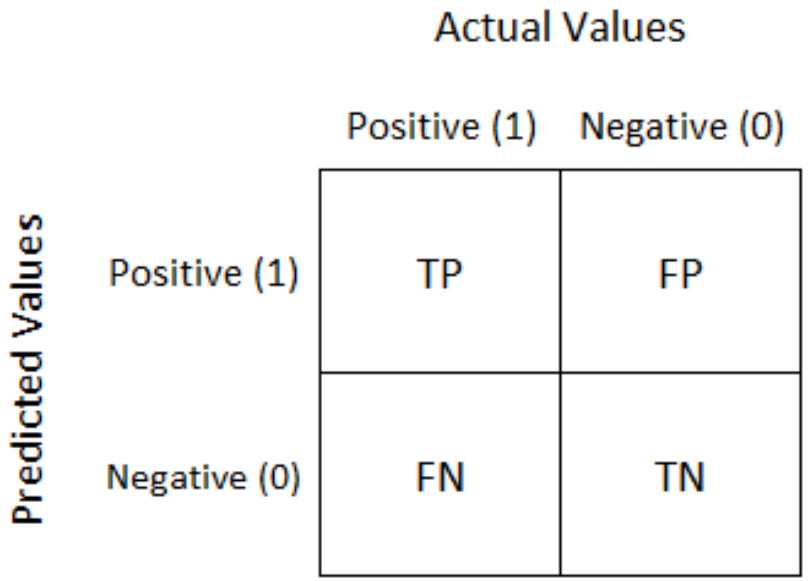
- MAE: Mean Absolute Error

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

where y_j is the actual value and $\hat{y}_j$ is predicted value.

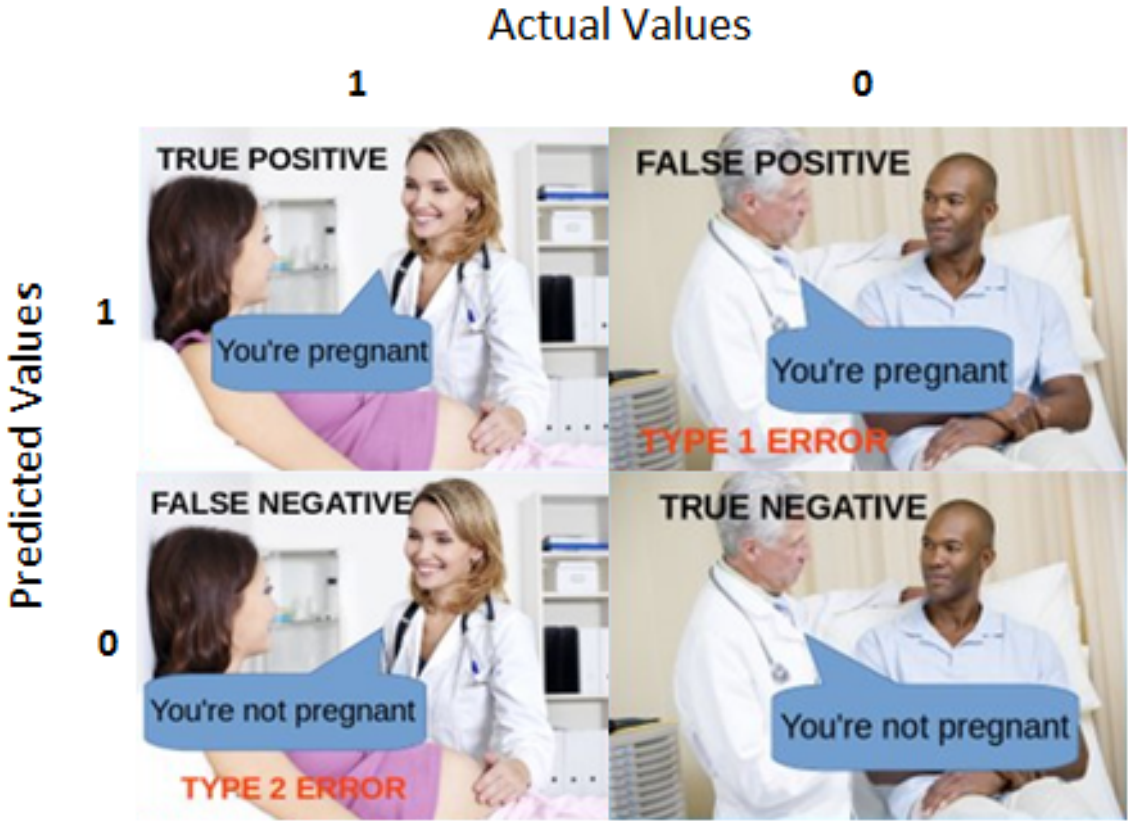
Confusion matrix

- What is a confusion matrix?
- Four combinations of predicted and actual values
- TP, FP, FN, TN



Confusion matrix

- Pregnancy analogy



Confusion matrix

- **True Positive:**
  - Interpretation: You predicted positive and it’s true.
  - You predicted that a woman is pregnant and she actually is.
- **True Negative:**
  - Interpretation: You predicted negative and it’s true.
  - You predicted that a man is not pregnant and he actually is not.

Confusion matrix

- **False Positive: (Type 1 Error)**
  - Interpretation: You predicted positive and it’s false.
  - You predicted that a man is pregnant but he actually is not.
- **False Negative: (Type 2 Error)**
  - Interpretation: You predicted negative and it’s false.
  - You predicted that a woman is not pregnant but she actually is.

Confusion matrix

- **Precision**

$$Precision = \frac{TP}{TP + FP}$$

  - Out of all the classes, how much we predicted correctly. It should be high as possible.

- **Recall**

$$Recall = \frac{TP}{TP + FN}$$

  - Out of all the positive classes, how much we predicted correctly. It should be high as possible.

Confusion matrix

- **F-Measure**

$$F\text{ - measure} = \frac{2*\text{Recall}*\text{Precision}}{\text{Recall} + \text{Precision}}$$

  - It is difficult to compare two models with low precision and high recall or vice versa. So to make them comparable, we use F-Score. F-score helps to measure Recall and Precision at the same time. It uses Harmonic Mean in place of Arithmetic Mean by punishing the extreme values more.

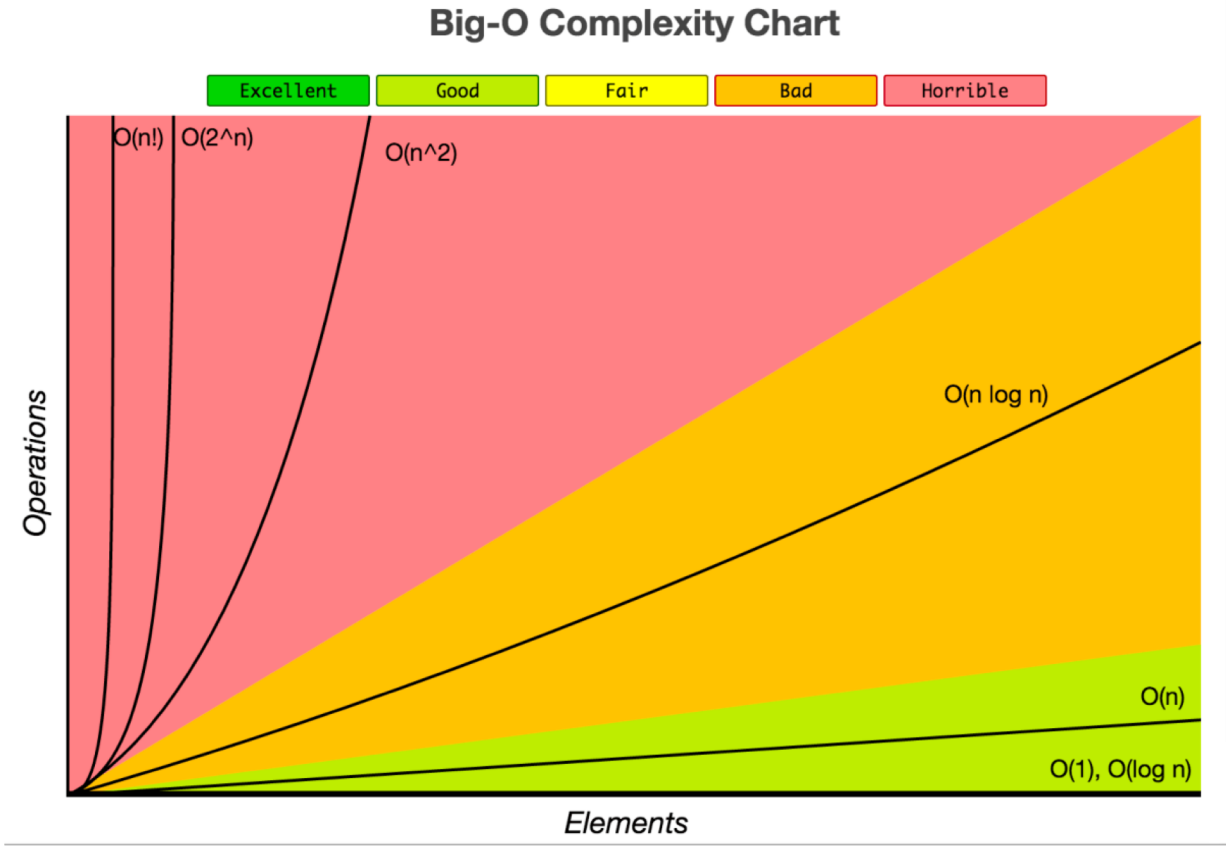
Complexity Analysis

- Time complexity
  - The amount of time taken by an algorithm to run as a function of the length of the input
- Space complexity
  - The amount of memory space taken by an algorithm to run as a function of the length of the input

Execution time ≠ time complexity

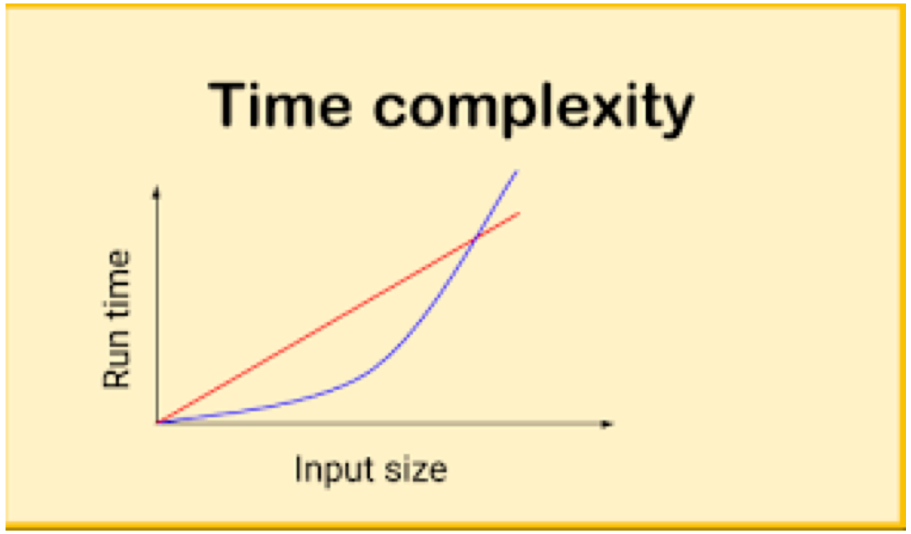
Complexity Analysis – Time Complexity

- Notation of Complexity
  - Big O notation: $O(f)$, where f is the function of the size of the input data, i.e., $O(1)$, $O(\log(N))$, $O(N)$, $O(n^2)$, $O(n * \log n)$, $O(n^3)$, $O(2^n)$, $O(N!)$…
  - The lower the rate, the better
  - How to calculate?
    - Experimental
    - Theoretical



Complexity Analysis - Time Complexity

- Experimental evaluation
  - Write a programme to implement the algorithm
  - Run the programme with inputs with varying size
  - Obtain the running time of each input
  - Plot the results



Complexity Analysis - Time Complexity

- Theoretical evaluation
  - Use pseudocode to describe the algorithm
  - Characterise the running time as a function of the input size, *n*.
  - Take into account of all possible inputs
  - Evaluate the time

Complexity Analysis - Time Complexity

- Notation of Complexity
  - Complexity is to calculate the number of operations that an algorithm takes, in the worst case.
  - Basic operations:
    - one arithmetic operation (e.g., +, *).
    - one assignment
    - one test (e.g., x == 0)
    - one read
    - one write (of a primitive type)

Complexity Analysis - Time Complexity

- Example 1
  - Calculate the sum of a given vector [4, 3, 2, …, 20]
    - *Sum*=0
    - for each *i* in the given vector
      - *Sum* += *i*
    - Return Sum

1 operation

2 ops (Sum+*I; Sum*=) for each *i*; If length of vector is *n*, then 2*n* operations

1 operation

  - Total operations: 2*n*+2.
  - Consider the term influenced the most, 2*n*, and get rid of coefficient, 2, complexity is obtained as *O*(*n*)

Complexity Analysis - Time Complexity

- Example 2

  1 3 … 5
  . .
  4 9 … 6

  - Calculate the sum of a given vector *n***n* array
  - *Sum*=0
    - for each row in the given array
      - for each *i* in row
      - *Sum* += *i*
    - Return Sum

1 operation

2*n* operations for each row, then 2*n**2*n* operations for two loops

1 operation

- Total operations: $4n^2+2$.
- Consider the term influenced the most, $4n^2$, and get rid of coefficient, 4, complexity is obtained as $O(n^2)$

Complexity Analysis - Time Complexity

- Coefficient is not quite important.
- Consider the most important terms.

THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

@XJTLU

Xi'an Jiaotong-Liverpool University
西交利物浦大學