Chapter 1 Data Mining

Tianlong Zhou

2021\08\08

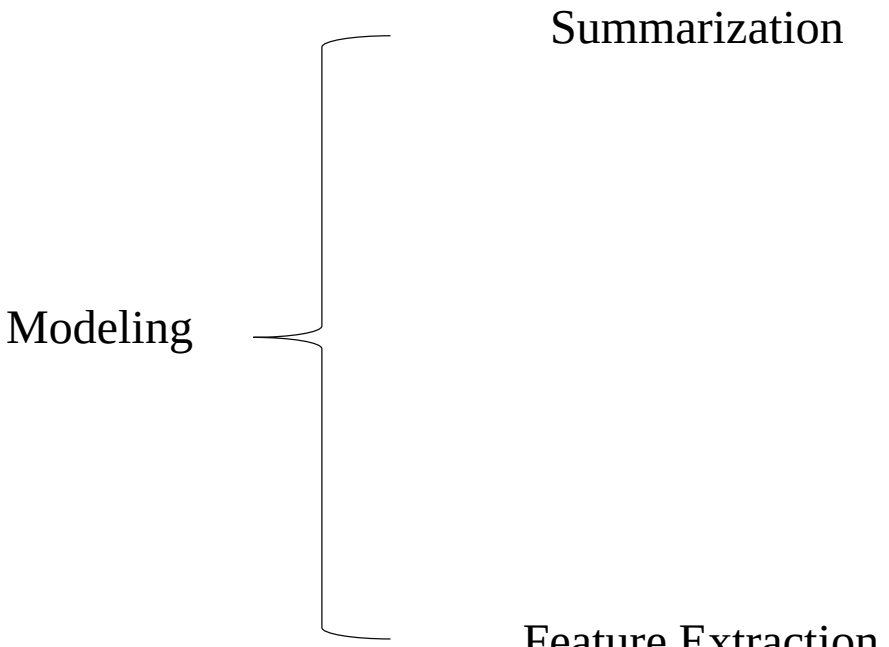
What is Data Mining?

Concept:

Use the most powerful hardware, the most powerful programming systems, and the most efficient algorithms to **solve problems** in science, commerce, healthcare, government, the humanities, and many other fields of human endeavor.

What is Data Mining?

What is Data Mining?



Tools:

1. **Machine learning**
2. **Statitical Modeling**
3. **…**

Modeling:

Use **tools** to **Summarization/Feature Extraction**

Summarization

Concepts:
Summarizing the data succinctly and approximately.（找分布）

Example:
1. **PageRank(Google)**
2. **Clustering(霍乱)**

Feature Extraction

Concepts:
Extracting the most prominent features of the data and ignoring the rest.（获取数据中的某些特性）

Example:
1. **Frequent Itemsets(元素间的相关性，啤酒 and 尿布)**
2. **Similar Items(集合间的相似性，产品推荐)**

Statistical Limits on Data Mining

Content:

A common sort of data-mining problem involves **discovering unusual events hidden within massive amounts of data**. This section is a discussion of the problem, including “Bonferroni’s Principle,” a warning against **overzealous use of data mining.**

Statistical Limits on Data Mining

Bonferroni's Principle:
Without going into the statistical details, we offer an informal version, **Bonferroni's principle**, that helps us avoid treating random occurrences as if they were real.

Bogus:
Suppose you have a certain amount of data, and you look for events of a certain type within that data. You can expect events of this type to occur, **even if the data is completely random**, and **the number of occurrences of these events will grow as the size of the data grows.** These occurrences are "bogus,"

Statistical Limits on Data Mining

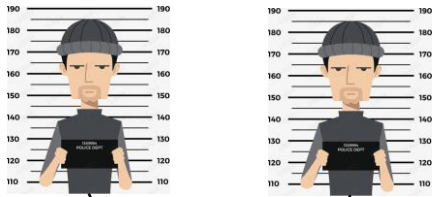
Bonferroni's Principle Process:

1. Calculate the expected number of occurrences of the events you are looking for, on the assumption that data is random.
2. If this number is significantly larger than the number of real instances you hope to find, then you must expect almost anything you find to be bogus.

An Example of Bonferroni's Principle

Suppose:

evil-doers



gather at a hotel, periodically

Condition:

1. There are **one billion** people who might be evil-doers.
2. Everyone **goes to a hotel one day in 100**.
3. A hotel **holds 100 people**. Hence, there are **100,000 hotels** – enough to **hold the 1%** of a billion people who visit a hotel on any given day.
4. We shall examine hotel records for **1000 days.**

Problem:
To **find evil-doers** in this data, we shall look for people who, on **two different days**, were both **at the same hotel.**

An Example of Bonferroni's Principle

Solution:

Step 1: Calculate the expected number of occurrences of the events you are looking for, on the assumption that data is random.

Suppose, however, that **there really are no evil-doers**. That is**, everyone behaves at random**, deciding with **probability 0.01 to visit a hotel** on any given day, and if so, **choosing one of the 10^5 hotels at random.**

The probability of **any two people** both deciding to visit a hotel on any given day is **0.0001.**

The chance that **they will visit the same hotel** is this probability divided by **10^5.**

Thus, the chance that **they will visit the same hotel on one given day is 10^−9**

.

The chance that they will **visit the same hotel on two different given days** is **the square of this number, 10^−18.**

An Example of Bonferroni's Principle

Solution:

Step 1 (cont.): Calculate the expected number of occurrences of the events you are looking for, on the assumption that data is random.

the number of pairs of people is **C(10^9,2) = 5 * 10^17**

The number of pairs of days is **C(1000,2) = 5 * 10^5**

The expected number of events that look like evil-doing:

$$5 \times 10^{17} \times 5 \times 10^{5} \times 10^{-18} = 250,000$$

Things Useful to Know -- TF.IDF

In several applications of data mining, we shall be faced with the problem of **categorizing documents (sequences of words) by their topic.** Typically, topics **are identified by finding the special words** that characterize documents about that topic. However, **until we have made the classification**, it is **not possible to identify these words as characteristic.**

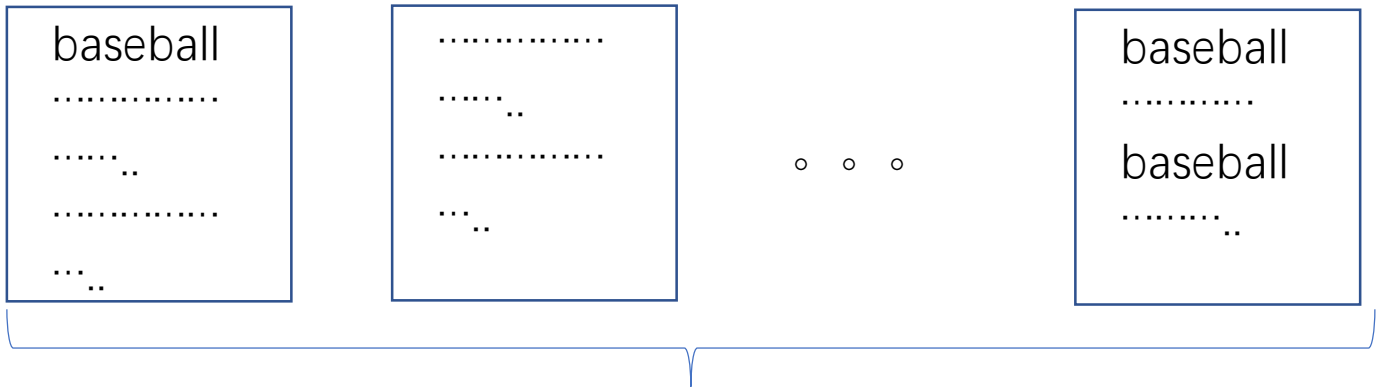
Thus, classification often starts by **looking at documents**, and **finding the significant words in those documents.**

The formal measure of **how concentrated into relatively few documents** are the occurrences of a given word is **called TF.IDF (Term Frequency times Inverse Document Frequency).**

Things Useful to Know -- TF.IDF

Condition:

Suppose we have a collection of N documents.

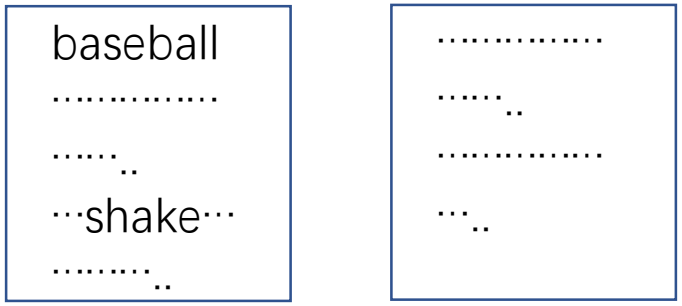


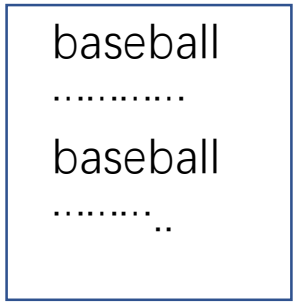
Things Useful to Know -- TF.IDF

Soluion:

Step1: TF Calculation

Define f_{ij} to be the frequency (number of occurrences) of term (word) i in document j..





Formula:

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

Baseball → $f_{11} = 1$ → $TF_{11} = 1/10$

shake → $f_{21} = 1$ → $TF_{21} = 1/10$

$\max_k f_{kj}$ = 10

Baseball → $f_{1n} = 2$ → $TF_{1n} = 2/9$

$\max_k f_{kj}$ = 9

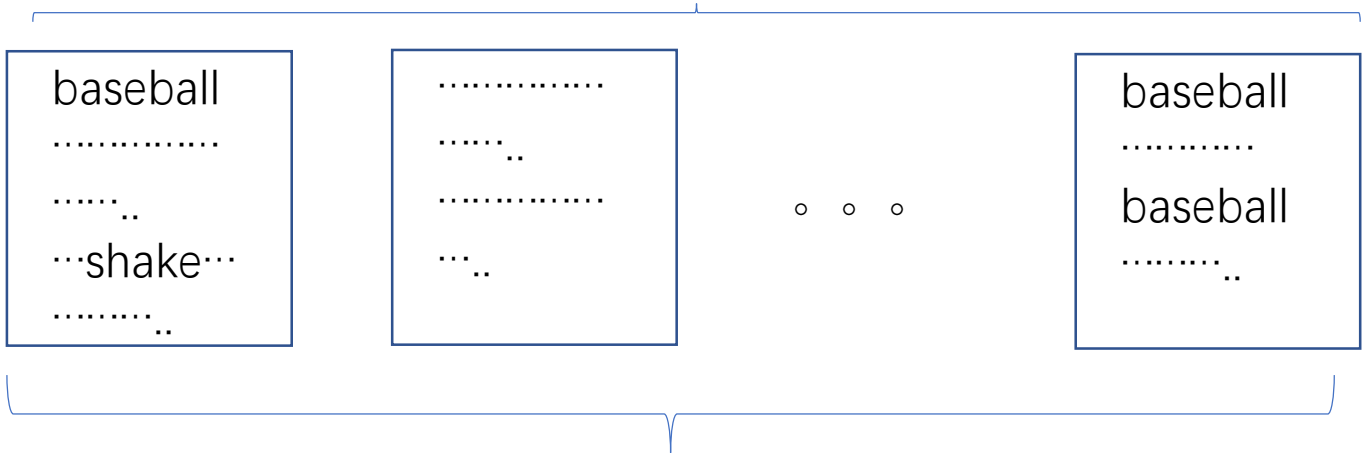
Things Useful to Know -- TF.IDF

Soluion:

Step2: IDF Calculation

Suppose term i appears in n_i of the N documents in the collection.

'baseball' appears in 14 of the 20 docutments → $n_1 = 14$, N = 20



Baseball → $IDF_1 = log_2(\frac{20}{14})$

Formula:

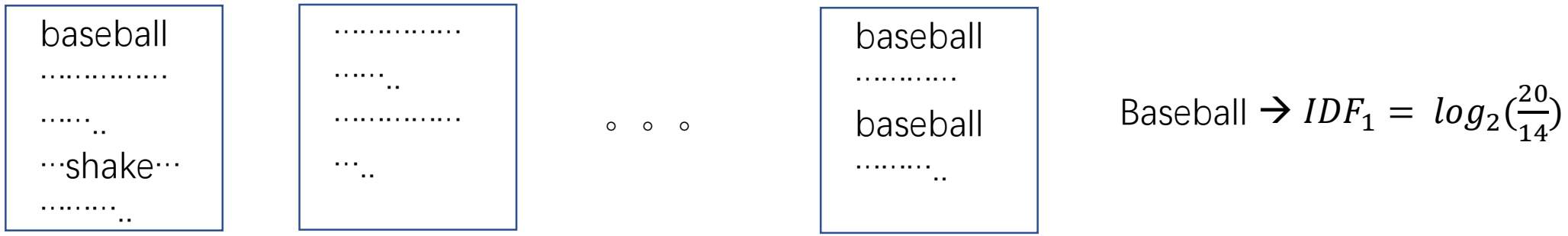
$$IDF_i = \log_2(N/n_i).$$

Things Useful to Know -- TF.IDF

Soluion:

Step2: TF.IDF Calculation

The TF.IDF score for term i in document j is then defined to be $TF_{ij} \times IDF_i$.



Baseball → $IDF_1 = log_2(\frac{20}{14})$

Baseball → $f_{11} = 1$ → $TF_{11} = \frac{1}{10}$

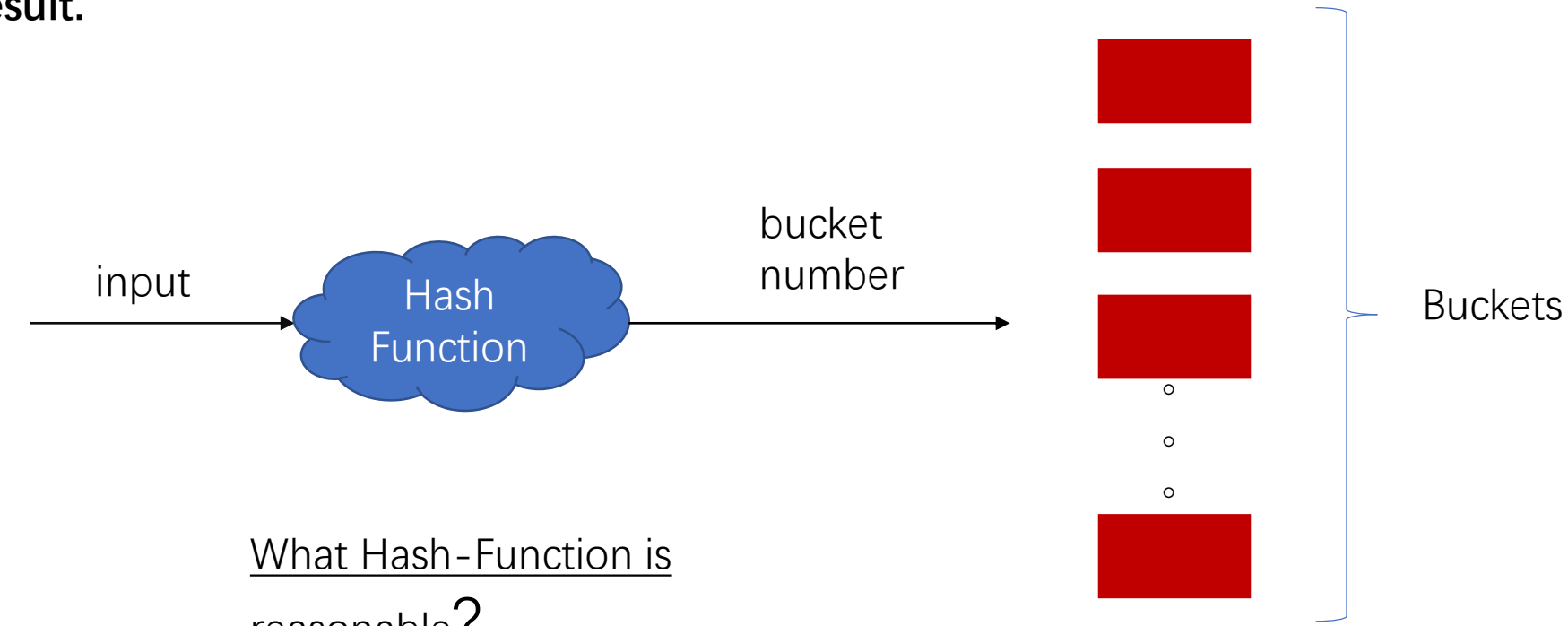
→ Score = $TF_{11} * IDF_1 = \frac{1}{10} * log_2(\frac{20}{14})$

Baseball → $f_{1n} = 2$ → $TF_{1n} = \frac{2}{9}$

→ Score = $TF_{1n} * IDF_1 = \frac{2}{9} * log_2(\frac{20}{14})$

Things Useful to Know -- Hash Function

Concept:
A hash function ***h*** takes a **hash-key value as an argument** and produces **a bucket number as a result.**



What Hash-Function is reasonable?

Things Useful to Know -- Hash Function

Concept:
Hash-Function **h** should **send approximately equal numbers of hash-keys to each of the B buckets.**

Example:

A common and simple hash function is to pick h(x) = x mod B.

That choice works well **if our population of hash-keys is all positive integers.**

However, **suppose our population is the even integers, and B = 10.** Then **only buckets 0, 2, 4, 6, and 8** can be the value of h(x)

if we **picked B = 11**, then we would find **that 1/11th of the even integers get sent to each of the 11 buckets**

Things Useful to Know -- Hash Function

Concept:
Hash-Function **h** should **send approximately equal numbers of hash-keys to each of the B buckets.**

Example:

A common and simple hash function is to pick h(x) = x mod B.

That choice works well **if our population of hash-keys is all positive integers.**

However, **suppose our population is the even integers, and B = 10.** Then **only buckets 0, 2, 4, 6, and 8** can be the value of h(x)

if we **picked B = 11**, then we would find **that 1/11th of the even integers get sent to each of the 11 buckets**

Things Useful to Know -- Hash Function

Application: Index

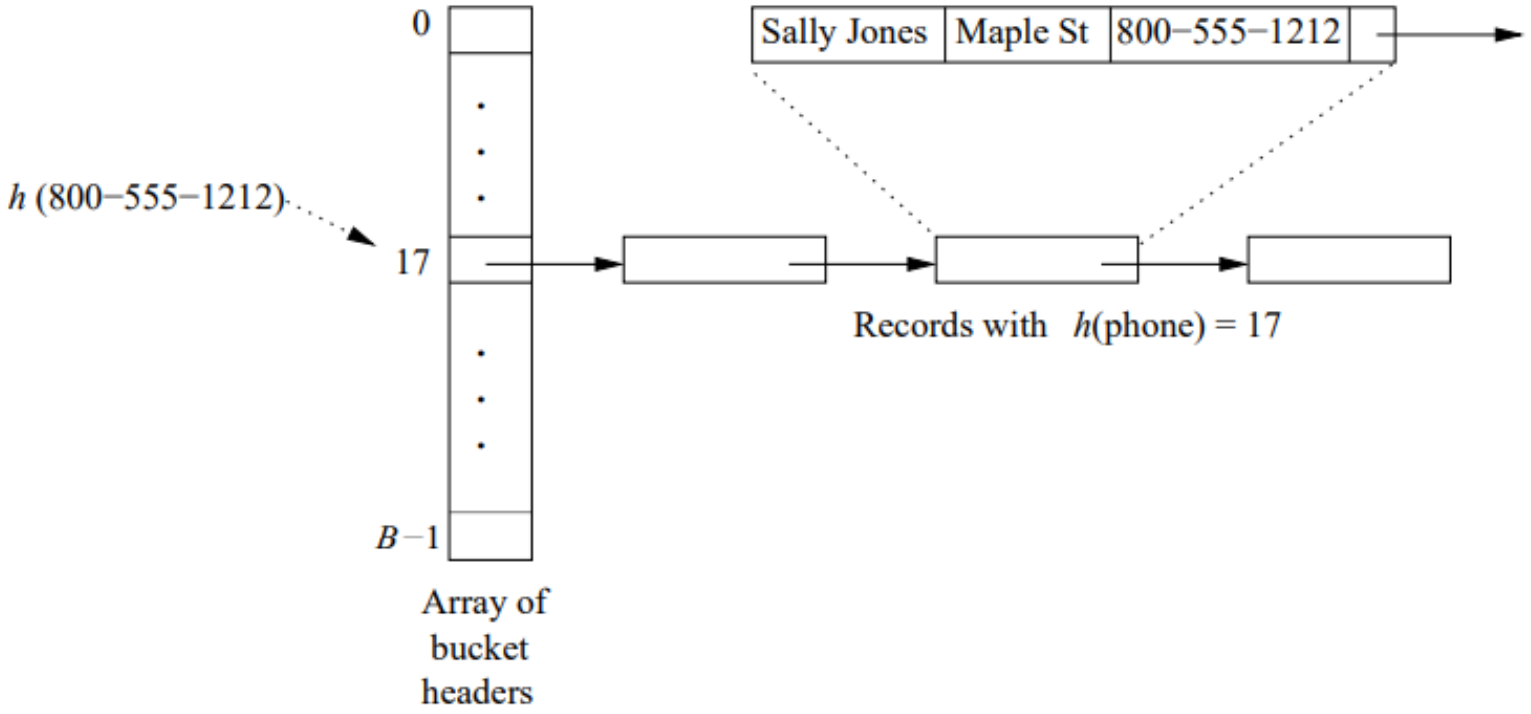


Figure 1.2: A hash table used as an index; phone numbers are hashed to buckets, and the entire record is placed in the bucket whose number is the hash value of the phone

Things Useful to Know -- Secondary Storage

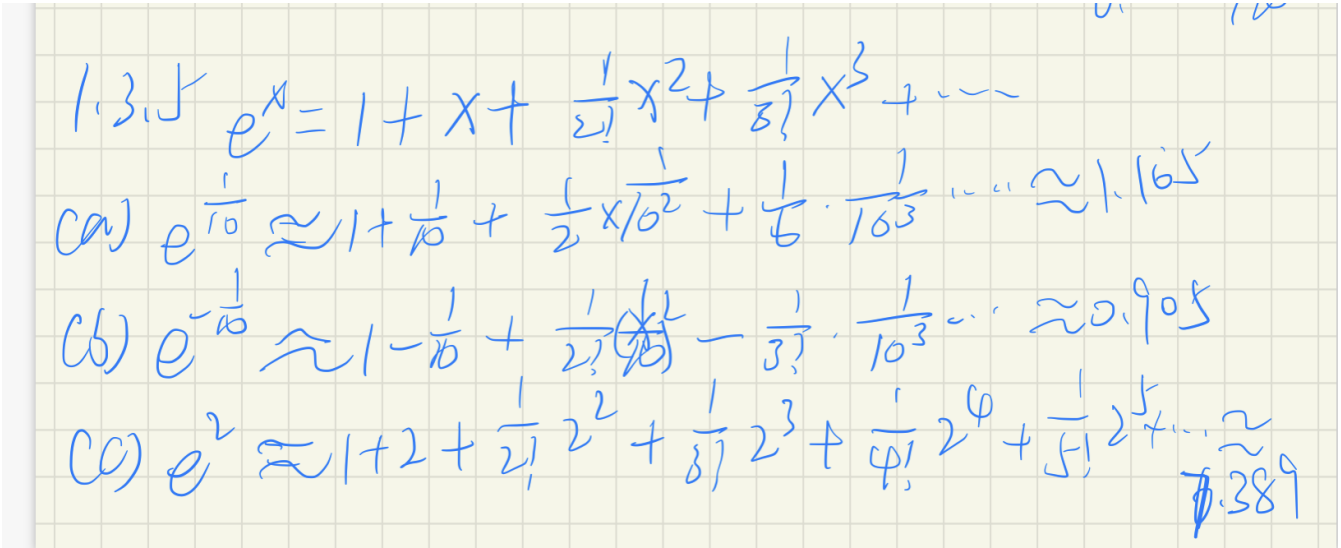
Key-point:
Disks are organized into blocks, which are the minimum units that the operating system uses to move data between main memory and disk. For example, the Windows operating system uses blocks of 64K bytes (i.e., 216 = 65,536 bytes to be exact). It takes approximately ten milliseconds to access (move the disk head to the track of the block and wait for the block to rotate under the head) and read a disk block. **That delay is at least five orders of magnitude (a factor of 105) slower than the time taken to read a word from main memory, so if all we want to do is access a few bytes, there is an overwhelming benefit to having data in main memory.**

Things Useful to Know -- The Base of Natural Logarithms

Key-point:

1. $\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^{n}=e$
2. $\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n}=\frac{1}{e}$
3. $e^{-i\omega t}=\sum_{i=0}^{\infty}\frac{x^{i}}{i!}$

Exercise:



Things Useful to Know -- Power Laws

Concept:

There are many phenomena that relate two variables by a power law, that is, **a linear relationship between the logarithms of the variables.**

i.e.

For two variables x and y, the relationship between them can be:

$$y = cx^a$$

a and c are constants.

Things Useful to Know -- Power Laws

Examples:

1. Node Degrees in the Web Graph.(Degree count)

2. Sales of Products.(Sale count)

3. Sizes of Web Sites.(page count)

4. Zipf ’s Law:(Word count)