# Continuation of Examples

Note Title

## Example 2: Face Image Database

- Provided by Prof. L. Sirovich (Mt Sinai S. M.)
- Consists of 143 faces of male caucasian students (and some faculty) at Brown Univ., without glasses, mustache, beard
- Each face is a gray-scale image with $128 \times 128$ pixels
- Horizontal dilation was applied so that the pupils are placed on two fixed points.

Note that each image of size $128 \times 128$ pixels can be represented as a matrix of 128 rows and 128 columns or a vector of length $128 \times 128 = 16384$.

In fact, in MATLAB, if

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,128} \\ \vdots & \vdots & & \vdots \\ x_{128,1} & x_{128,2} & \cdots & x_{128,128} \end{bmatrix} \in \mathbb{R}^{128 \times 128}$$

then $X(:)$ represents a vector of length 16384 as follows:

$$X(:) \leftrightarrow \begin{bmatrix} x_{1,1} \\ \vdots \\ x_{128,1} \\ x_{1,2} \\ \vdots \\ x_{128,2} \\ \vdots \\ x_{1,128} \\ \vdots \\ x_{128,128} \end{bmatrix}$$

By default, a vector is a column vector

# Example 3: Term-Document Matrices for Search

This example is from Berry & Browne: "Understanding Search Engines", 2005.

Consider a set of documents consisting of book titles, and a set of words (or terms) as follows:

| *Terms* | | *Documents* | |
|---|---|---|---|
| T1: | Bab(y,ies,y's) | D1: | Infant & Toddler First Aid |
| T2: | Child(ren's) | D2: | Babies & Children's Room (For Your Home) |
| T3: | Guide | D3: | Child Safety at Home |
| T4: | Health | D4: | Your Baby's Health and Safety: From Infant to Toddler |
| T5: | Home | | |
| T6: | Infant | D5: | Baby Proofing Basics |
| T7: | Proofing | D6: | Your Guide to Easy Rust Proofing |
| T8: | Safety | D7: | Beanie Babies Collector's Guide |
| T9: | Toddler | | |

Then, consider the following term-document matrix of size 9×7.

The 9 × 7 term-by-document matrix before normalization, where the element $\hat{a}_{ij}$ is the number of times term $i$ appears in document title $j$:

Terms

$$\hat{A} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

← rather sparse!

documents

Now suppose we want to retrieve books on "Child Proofing"
T2 T7

Then this can be reprented by a query vector:

$$q = [0\,1\,0\,0\,0\,0\,0\,1\,0\,0]^T$$

Try to match $q$ with each column vector (i.e., a document).
⇒ No exact hit, but the close ones are D2, D3, D5, D6

Generalizing this to:
{ terms = the English dictionary
documents = the entire webpages
the term-document matrix becomes huge!!

est. in 2004

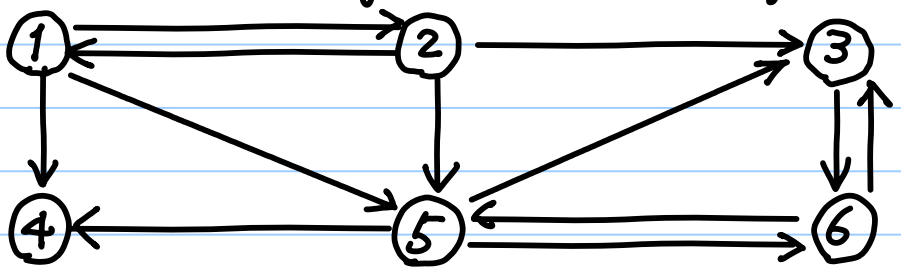
$$300{,}000 \times 4{,}000{,}000{,}000$$

or even larger now!

How to deal with such a huge matrix for search?
⇒ We'll learn how later in this course.

# Example 4: Link Graph Matrix of webpages

Consider a very small set of webpages




This graph means that

① has outlinks to ②, ④, ⑤ (pointers)

① has an inlink from ②

② has ---

A link graph matrix $P = (P_{ij})$ is defined by

$$P_{ij} = \begin{cases} \text{nonzero} & \text{if } \exists\ j \to i \\ 0 & \text{otherwise} \end{cases}$$

The nonzero const. is determined by the normalization

$$\sum_{i=1}^{n} P_{ij} = 1 \quad \text{for } 1 \le j \le n$$

So, in this small example, we have

$$P = \begin{bmatrix} 0 & 1/3 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 1/3 & 1/2 \\ 1/3 & 0 & 0 & 0 & 1/3 & 0 \\ 1/3 & 1/3 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 1 & 0 & 1/3 & 0 \end{bmatrix} \quad //$$

One possible measure of importance of a webpage (i)
= how many other important pages have outlinks to (i)

- This can be solved by the eigenvalue decomposition of $P$
- In reality, $n >$ billions!
- Depending on the literature, $P_n^T$ is denoted as "$P$", i.e., $\sum_{j=1}^{n} P_{ij} = 1, \quad 1 \le i \le n$

⇨ We'll discuss link graph matrices in detail later in this course.

# Floating Point Computations

## ✯ Floating Point Numbers

On a digital computer, one can only use a finite number of bits to represent a real number, e.g., $\sqrt{2}$, $\pi$, $e$, etc.

Hence, the floating point representation was invented and standardized.
$\Rightarrow$ IEEE 754 Standard.

The idealized floating point system is a discrete subset $\mathbb{F} \subset \mathbb{R}$

$$\mathbb{F} := \{0\} \cup \{x \in \mathbb{R} \mid x = \pm 1.d_1 d_2 \cdots d_t \times \beta^e,$$
$$d_j \in \{0, 1, \cdots, \beta-1\},\ 1 \le j \le t$$
$$t \ge 1,\ t \in \mathbb{N},$$
$$\beta \ge 2,\ \beta \in \mathbb{N},$$
$$e \in \mathbb{Z}\ \}$$

$d_1 d_2 \cdots d_t$ = mantissa (or fraction)

$\beta$ = base (or radix), typically $\beta = 2$.

$e$ = exponent, $t$ = precision.

$1.d_1 d_2 \cdots d_t \times \beta^e$ in base $\beta$

$$= \left(\frac{1}{\beta^0} + \frac{d_1}{\beta^1} + \frac{d_2}{\beta^2} + \cdots + \frac{d_t}{\beta^t}\right) \times \beta^e \text{ in base 10}$$

In C++, they are float, double, long double.

**Basic f.p. types for a real number:**

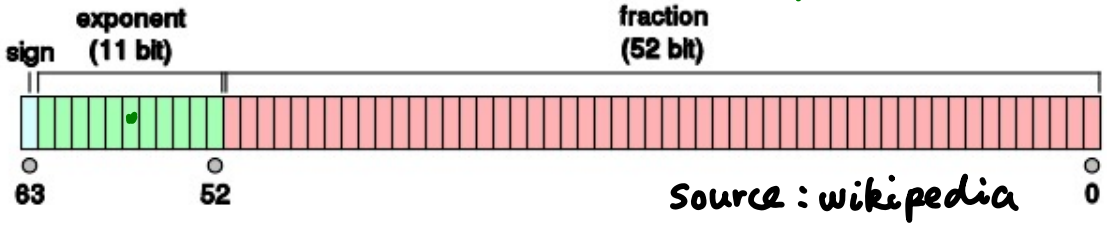
Single precision f.p. number = 32 bits

Double " " = 64 bits

Quadraple " " = 128 bits

MATLAB Default!

⇒ 4, 8, 16 bytes




**How about t ?**

In IEEE 754 Standard,

$$t = \begin{cases} 24 & \text{for single precision} \\ 53 & \text{double} \quad " \\ 113 & \text{quadraple} \quad " \end{cases}$$

$\mu := \frac{1}{2}\beta^{1-t}$ is called the machine epsilon or the unit round-off of the floating point system, often denoted by $\varepsilon_{mach}$.

Single Precision $\mu = 2^{-24} \approx 5.96 \times 10^{-8}$

Double Precision $\mu = 2^{-53} \approx 1.11 \times 10^{-16}$

Quadraple Precision $\mu = 2^{-113} \approx 9.63 \times 10^{-35}$

## How about e ?

In IEEE 754 Standard,

- Single precision : $-126 \leq e \leq 127$
- double " : $-1022 \leq e \leq 1023$
- quadraple " : $-16382 \leq e \leq 16383$

Hence the range of representable positive numbers are roughly:

- single prec. : $10^{-38} \sim 10^{+38}$
- double " : $10^{-308} \sim 10^{+308}$
- quadraple " : $10^{-4932} \sim 10^{+4932}$

Now, we have:

(*) $\forall x \in \mathbb{R}, \exists x' \in \mathbb{F}$ s.t. $|x - x'| \leq \mu |x|$

Let's define the following function:

$fl : \mathbb{R} \to \mathbb{F}$

a fcn giving the closest f.p. approximation to a given real num. = rounded equivalent in $\mathbb{F}$.

So, the statement (*) is equivalent to

$\forall x \in \mathbb{R}$, $\exists \varepsilon \in \mathbb{R}$ with $|\varepsilon| \leq \mu$ s.t.

$$fl(x) = x(1+\varepsilon)$$

Why is this so?

☺ $|x - fl(x)| \le \mu |x|$

$\Leftrightarrow \quad -\mu|x| \le x - fl(x) \le \mu|x|$

$\Leftrightarrow \quad x - \mu|x| \le fl(x) \le x + \mu|x|$

$\Leftrightarrow \quad fl(x) = x + \varepsilon x$ for

$\exists\, \varepsilon$ with $|\varepsilon| \le \mu$ //

In essence, on any computer, for any real number $x$, our best hope is $\left|\frac{x - fl(x)}{x}\right| \le \mu = \varepsilon_{mach}$

$\approx 10^{-16}$ in MATLAB

= relative error

## Floating Point Arithmetic

Basic arithmetic operations

in $\mathbb{R} \Rightarrow +, -, \times, \div$

in $\mathbb{F} \Rightarrow \oplus, \ominus, \otimes, \oslash$

Ideal goal of computer arithmetic:

$\forall x, y \in \mathbb{F}$, $x \circledcirc y = fl(x \circ y)$

$\circ = +, -, \times,$ or $\div$ ; $\circledcirc$: operations in $\mathbb{F}$; $\circ$: operations in $\mathbb{R}$

From the above discussion, we have

(⋆) $\quad x \circledcirc y = (x \circ y)(1 + \varepsilon), \ \exists\, |\varepsilon| < \mu$

If a computer truncates the intermediate result $x \circ y$, then in (☆) $\mu$ should be replaced by $2\mu$.

Nick Trefethen (Oxford) calls this "The Fundamental Axiom of Floating Point Arithmetic."

## ☆ Floating Point Operations (FLOPS)

FLOPS = flops = flop/s
= f.p. operations/second
= a measure of a computer's performance

flop = a unit of one arithmetic operation on a computer

Ex. The statement in a code

```
y = y + a * x ;
```

How many flops does this require?

⇒ 2 flops

as of 11/14/2011, it's K computer, RIKEN, Japan.

Note: The world fastest computer can perform ≥ 10 Peta FLOPS = $10^{16}$ FLOPS!