ST509 Computational Statistics

Lecture 1: Computer Arithmetic

Seung Jun Shin

Department of Statistics
Korea University

E-mail: sjshin@korea.ac.kr

Computer Arithmetic I

- Computer cannot do arithmetic.
- Merely recognize two status: on (1) % off (0)
- Base for arithmetic is 2 or power of 2 such as 16.
- Any positive number z can be written as a base-B number using the set of digits $\{0, 1, \cdots, B-1\}$.

$$z = a_k B^k + \cdots + a_2 B^2 + a_1 B^1 + a_0 + a_{-1} B^{-1} + a_{-2} B^{-2} + \cdots$$

$$\Rightarrow \quad z = (\underbrace{a_k \ \cdots \ a_2 \ a_1 \ a_0}_{\text{integer}} \ \underbrace{\color{red}{.}}_{\color{red}{\text{radix}}} \ \underbrace{a_{-1} \ a_{-2} \cdots}_{\text{fractional part}})_B$$

Computer Arithmetic II

- "Fixed Point" and "Floating Point" refer to the position of the radix point.
- Fixed point numbers are analogs to integers.
- Floating point representation takes the form of:
  - S: a sign
  - E: integer exponent
  - F: the fraction

  written as

$$(S, E, F)$$

Computer Arithmetic III

- FPR can use finite number of digits only, and may have trouble to represent infinite expansions of numbers:

$$(.3750000\ldots) \text{ or } (.3749999\ldots)$$

- Rounding vs Chopping: 6.02257×10^{23} can be represented as for $d = 4$

$$(+, 24, .6023) \quad \text{vs} \quad (+, 24, .6022)$$

Computer Arithmetic IV

- FPR is not unique: 5 can be represented by

$$(+, 1, .5000) \text{ or } (+, 2, .0500) \text{ or } (+, 4, .0005)$$

- Normalization is required.

Fixed Point Arithmetic

- Arithmetic of integers is simple.
- Three representation of integers:
  - Signed Integer: First digit for sign. (non-unique zero / symmetric rage)
  - One's complement: Complement each bit to change the sign (non-unique zero / symmetric rage)
  - Two's complement: Complement each bit and add one to change the sign (Unique zero / asymmetric range)
- Fixed is fast, but has a limited range. (32bit for R)

Floating Point Representations I

- The value represented by (S, E, F) is given by

$$(-1)^S \times \mathrm{Base}^{\mathrm{E-excess}} \times .\mathrm{F}_{\mathrm{base}}$$

- For a base of 2, we have

$$(-1)^S \times 2^{\mathrm{E-excess}} \times 1.\mathrm{F}$$

  when normalized.

Floating Point Representations II

- For 32bit (a.k.a single precision):
  - 1 bit for S, 8 bits for E, and 23 bits for F
  - E ranges from 0 to 255 and Excess $= 2^7 - 1 = 127$, so that the rue exponent rage is -126 to 127.
  - $(S, 255, 0)$ represents $+\infty$ for $S = 0$ and $-\infty$ for $S = 1$.
  - $(S, 255, F)$ for any F other than 0 represents NaN.
  - $(S, 0, F)$ represents denormalized value:

$$(-1)^S \times 2^{-126} \times 0.\mathrm{F}$$

Floating Point Representations III

Figure: IEEE binary floating point representation (single precision)

Number	Conversion	Sign, Exponent	Fraction	Z Format
1	$1.0_{two} \times 2^{0}$	0 011 1111 1 000 0000	0000 0000 0000 0000	3F800000
1/16	$1.0_{two} \times 2^{-4}$	0 011 1101 1 000 0000	0000 0000 0000 0000	3D800000
0	$0.0_{two} \times 2^{-127}$	0 000 0000 0 000 0000	0000 0000 0000 0000	00000000
-15	$-1.111_{two} \times 2^{3}$	1 100 0001 0 111 0000	0000 0000 0000 0000	C1700000
$1.2E-38$	$1.0_{two} \times 2^{-126}$	0 000 0000 1 000 0000	0000 0000 0000 0000	00800000
$1.4E-45$	$1.0_{two} \times 2^{-149}$	0 000 0000 0 000 0000	0000 0000 0000 0001	00000001
$3.4E+38$	$(2-2^{-23}) \times 2^{127}$	0 111 1111 0 111 1111	1111 1111 1111 1111	7FEFFFFF
$+\infty$		0 111 1111 1 111 1111	1111 1111 1111 1111	7FFFFFFF

Half precision	16-bit
Single precision	32-bit
Double precision	64-bit
Quadruple precision	128-bit

Floating Point Representations IV

- 64bit (double precision): 1 bit for S, 11 bits for E, and 52 bits for F, with Excess $= 2^{10} - 1 = 1023$.
- Un the analysis of algorithm, "*flops*" – floating point operations is used to measure the work.
- *flop* consists of a floating point multiply (or divide) and the usually accompanying addition, fetch, and store.

Living with Floating Point Inaccuracies I

- Some numbers, say z, cannot be written exactly on a computed and a floating point approximation of it fl(z) is used.
- Relative error:

$$|\mathrm{fl}(z) - z|/|z|, \quad \text{for } z \neq 0$$

Living with Floating Point Inaccuracies II

- Machine unit U satisfies

$$|\mathrm{fl}(z) - z| \leq U|z| \qquad \text{for all } z$$

where

$$U = \begin{cases} 0.5B^{1-d} & \text{for rounding} \\ B^{1-d} & \text{for chopping} \end{cases}$$

Living with Floating Point Inaccuracies III

- Floating point arithmetic does not obey the laws of algebra.
- Assuming $B = 10$ and $d = 4$, let

$$
\begin{aligned}
a &= \quad 4 \quad &&= (+, 1, .4000)\\
b &= \quad 5003 \quad &&= (+, 4, .5003)\\
c &= \quad 5000 \quad &&= (+, 4, .5000)
\end{aligned}
$$

  then

$$(a + b) + c \neq a + (b + c)$$

  and

$$(2b - 2c) - 2a \neq 2b - (2c + 2a)$$

- The latter is known as (catastrophic) Cancellation.

Living with Floating Point Inaccuracies IV

- Range: about $\pm 10^{\pm 300}$
- Overflow $\rightarrow$ crash / underflow $\rightarrow$ 0
- Exact number: integer to 2^{52} or divided by power of 2 (ex i/1024).
- Accuracy: $\mathrm{fl}(x+y)$ is not usually the same as $x+y$. Testing `x == y` is risky. Then how?
- Cancellation:

$$\begin{aligned}\mathrm{fl}(\pi) - \mathrm{fl}(22/7) &= .314159 \times 10^{1} - .314286 \times 10^{1} \\ &= -.127?? \times 10^{-3}\end{aligned}$$

Living with Floating Point Inaccuracies V

- For $(y_1, y_2, y_3, y_4, y_5) = (356, 357, 358, 359, 360)$,

$$\sum_{i=1}^{5} (y_i - \bar{y})^2 = \sum_{i=1}^{5} y_i^2 - 5\bar{y}^2 < 0$$

  when $d = 4$ and $B = 10$.

- Improvement can be made by

$$\sum_{i=1}^{5} (y_i - \bar{y})^2 = \sum_{y=2}^{5} (y_i - y_1)^2 + 5(y_1 - \bar{y})^2.$$

Living with Floating Point Inaccuracies VI

- Logistic Distribution, $1 - F(t) = 1 - (1 + e^{-t})^{-1}$.
- Computing $1 - F(6) = .002472623$:
  1. $(1 + e^{-6}) = (+, 1, .1002)$ gives $(1 + e^{-6})^{-1} = (+, 0, .9980)$;
     then $(+, 1, .0000) - (+, 0, 9980) = (+, -2, .2000)$.
  2. $1 - F(6) = e^{-6}/(1 + e^{-6})$;
     then $(+, -2, .2479)/(+, 1, .1002) = (+, -2, .2474)$

Living with Floating Point Inaccuracies VII

- Binomial probability:

$$f(k \mid n, p) = \binom{n}{k} p^k (1-p)^{n-k}$$

- Overflow: $n = 180$, $k = 90$, and. $p = 0.01$.
- Underflow: $n = 150$, $k = 141$, and. $p = 0.005$.

Usage

```
dbinom(x, size, prob, log = FALSE)
pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rbinom(n, size, prob)
```

Figure: `dbinom` function in `R`

Conditioned Problems and Stable Algorithm I

- Numerical Algorithm:

$$\texttt{output} = f(\texttt{input})$$

- Condition of a problem:

$$\frac{|\ f(\text{input} + \delta) - \text{output}\ |}{\text{output}} = \text{condition}\, \frac{|\delta|}{\text{input}}$$

- The condition number C is approximated by

$$C = \left| \frac{x f'(x)}{f(x)} \right|$$

Conditioned Problems and Stable Algorithm II

- Finding a smaller root of

$$z^2 - x_1 z + x_2 = 0$$

  where $x_1, x_2 > 0$ with $x_2 \approx 0$.

- Let z_1 the larger and z_2 be smaller one, then

$$z_2(x_1, x_2) = \left(x_1 - \sqrt{x_1^2 - 4x_2}\right)/2$$

  Condition number C with x_1 fixed is

$$C = \left|\frac{z_1}{z_1 - z_2}\right| \approx 1$$

  when z_1 is large and z_2 is very small.

- Finding a smaller root $z_2 (= 0.0047533)$ with $d = 4$.

$$az^2 + bz + c = z^2 - 8.42z + 0.04 = 0.$$

  - $b^2 - 4ac = 70.74$, $\sqrt{70.74} = 8.411$, $(8.420 - 8.411) = 0.0045$

Conditioned Problems and Stable Algorithm III

- Inverse form:

$$a + bu + cu^2 = 1 - 8.42u + 0.04u^2 = 0$$

  where $u = 1/z$.

  - The larger root u_1 is
$$\frac{2c}{b + \sqrt{b^2 - 4ac}}$$
  - $2 \times 0.004000/(8.420 + 8.411) = 0.4753$.