

1 A Asymptotics and Computational Cost

We introduce Big-O, little-o and asymptotic notation and see how they can be used to describe computational cost.

1. Asymptotics as $n \rightarrow \infty$
2. Asymptotics as $x \rightarrow x_0$
3. Computational cost

1.1 1. Asymptotics as $n \rightarrow \infty$

Big-O, little-o, and "asymptotic to" are used to describe behaviour of functions at infinity.

Definition (Big-O)

$$f(n) = O(\phi(n)) \quad (\text{as } n \rightarrow \infty)$$

means

$$\left| \frac{f(n)}{\phi(n)} \right|$$

is bounded for sufficiently large n . That is, there exist constants C and N_0 such that, for all $n \geq N_0$, $\left| \frac{f(n)}{\phi(n)} \right| \leq C$.

Definition (little-O)

$$f(n) = o(\phi(n)) \quad (\text{as } n \rightarrow \infty)$$

means

$$\lim_{n \rightarrow \infty} \frac{f(n)}{\phi(n)} = 0.$$

Definition (asymptotic to)

$$f(n) \sim \phi(n) \quad (\text{as } n \rightarrow \infty)$$

means

$$\lim_{n \rightarrow \infty} \frac{f(n)}{\phi(n)} = 1.$$

Examples

$$\frac{\cos n}{n^2 - 1} = O(n^{-2})$$

as

$$\left| \frac{\frac{\cos n}{n^2-1}}{n^{-2}} \right| \leq \left| \frac{n^2}{n^2-1} \right| \leq 2$$

for $n \geq N_0 = 2$.

$$\log n = o(n)$$

as

$$\lim_{n \rightarrow \infty} \frac{\log n}{n} = 0.$$

$$n^2 + 1 \sim n^2$$

as

$$\frac{n^2 + 1}{n^2} \rightarrow 1.$$

Note we sometimes write $f(O(\phi(n)))$ for a function of the form $f(g(n))$ such that $g(n) = O(\phi(n))$.

1.1.1 Rules

We have some simple algebraic rules:

Proposition (Big-O rules)

$$O(\phi(n))O(\psi(n)) = O(\phi(n)\psi(n)) \quad (\text{as } n \rightarrow \infty)$$

$$O(\phi(n)) + O(\psi(n)) = O(|\phi(n)| + |\psi(n)|) \quad (\text{as } n \rightarrow \infty).$$

1.2 2. Asymptotics as $x \rightarrow x_0$

We also have Big-O, little-o and "asymptotic to" at a point:

Definition (Big-O)

$$f(x) = O(\phi(x)) \quad (\text{as } x \rightarrow x_0)$$

means

$$\frac{|f(x)|}{|\phi(x)|}$$

is bounded in a neighbourhood of x_0 . That is, there exist constants C and r such that, for all $0 \leq |x - x_0| \leq r$, $|\frac{f(x)}{\phi(x)}| \leq C$.

Definition (little-O)

$$f(x) = o(\phi(x)) \quad (\text{as } x \rightarrow x_0)$$

means

$$\lim_{x \rightarrow x_0} \frac{f(x)}{\phi(x)} = 0.$$

Definition (asymptotic to)

$$f(x) \sim \phi(x) \quad (\text{as } x \rightarrow x_0)$$

means

$$\lim_{x \rightarrow x_0} \frac{f(x)}{\phi(x)} = 1.$$

Example

$$\exp x = 1 + x + O(x^2) \quad \text{as } x \rightarrow 0$$

Since

$$\exp x = 1 + x + \frac{\exp t}{2} x^2$$

for some $t \in [0, x]$ and

$$\left| \frac{\frac{\exp t}{2} x^2}{x^2} \right| \leq \frac{3}{2}$$

provided $x \leq 1$.

1.3 3. Computational cost

We will use Big-O notation to describe the computational cost of algorithms. Consider the following simple sum

$$\sum_{k=1}^n x_k^2$$

which we might implement as:

```
function sumsq(x)
    n = length(x)
    ret = 0.0
    for k = 1:n
        ret = ret + x[k]^2
    end
    ret
end

n = 100
x = randn(n)
sumsq(x)
```

118.10555092168023

Each step of this algorithm consists of one memory look-up ($\mathbf{z} = \mathbf{x}[\mathbf{k}]$), one multiplication ($\mathbf{w} = \mathbf{z} * \mathbf{z}$) and one addition ($\mathbf{ret} = \mathbf{ret} + \mathbf{w}$). We will ignore the memory look-up in the following discussion. The number of CPU operations per step is therefore 2 (the addition and multiplication). Thus the total number of CPU operations is $2n$. But the constant 2 here is misleading: we didn't count the memory look-up, thus it is more sensible to just talk about the asymptotic complexity, that is, the *computational cost* is $O(n)$.

Now consider a double sum like:

$$\sum_{k=1}^n \sum_{j=1}^k x_j^2$$

which we might implement as:

```
function sumsq2(x)
    n = length(x)
    ret = 0.0
    for k = 1:n
        for j = 1:k
            ret = ret + x[j]^2
        end
    end
    ret
end
```

```
n = 100
x = randn(n)
sumsq2(x)
```

4730.625569045884

Now the inner loop is $O(1)$ operations (we don't try to count the precise number), which we do k times for $O(k)$ operations as $k \rightarrow \infty$. The outer loop therefore takes

$$\sum_{k=1}^n O(k) = O\left(\sum_{k=1}^n k\right) = O\left(\frac{n(n+1)}{2}\right) = O(n^2)$$

operations.