Machine Learning I: Supervised Methods

B. Keith Jenkins

Announcements

- Homework 1 is due Friday
- Homework 2 will be posted Friday
- Discussion 3 session Friday
  - 11:00 - 11:50 AM, OHE 122
- Python Instruction session Friday
  - 12:00 - 12:50 PM, OHE 120
- To do well in this class: keep up with lectures, discussion sessions, reading, homeworks

Reading

- Reminder (for next week):
  - Bishop 5.2.4, Gradient descent
  - Note: Bishop's E is our J

Today's lecture

- Computational complexity
  - Example
  - Big-O, big-omega, big-theta
- Fundamental assumptions in supervised ML
- Augmented notation and space

Lecture 6 ←

Computational Complexity

Why?

- Some datasets are very large (even huge)
- Number of features (or input variables) can vary by a lot
  - e.g., 10^1 to 10^8
- Sometimes algorithms are run many times
  - Model selection
  - cross validation
- Sometimes computation hardware is limited
- Faster hardware is more expensive.

→ Useful to have an idea of how runtime (and memory requirements) depend on N, D, etc.

Example

How many scalar adds (A), multiplies (M), and divisions (D) do the following take, using the most common approach (algorithm)?

Compute the class sample-mean vectors for C classes

Let N_c = # of data points in each class $(N_1 = N_2 = \cdots = N_c)$

D = # of features

$$\underline{\mu}_k = \frac{1}{N_c} \sum_{i=1}^{N_c} \underline{x}_i^{(k)} = F_c \sum_{i=1}^{N_c} \underline{x}_i^{(k)}, \quad F_c = \frac{1}{N_c}$$

Adds: $(N_c - 1) D C$

Mult's: $D C$

Div's: 1

Total compute time or complexity $= (N_c - 1) D C T_A + D C T_M + 1 \cdot T_D = T$

T_P = Complexity or computation time of one P operation ($P = A, M,$ or D)

Time and Space Complexity

→ Specify the algorithm and type of machine
→ Then evaluate time and space complexity.

Time complexity

The number of operations, or total computation time, required to run the algorithm.

Operations counted: multiply, add (subtract), divide, compare, logical test ($A = B$?), etc.

Operations not counted: memory read, memory write, data input, data output.

(Auxiliary) Space Complexity

The number of memory storage locations required to run the algorithm.

Memory counted: any memory required within the computation of the algorithm

Memory not counted: memory used for inputs and outputs while they aren't being used for the computation itself.

Example

1. Assume a serial machine that runs the algorithm below.

# memory elements	Algorithm α	Time complexity
2	$F_c = \frac{1}{N_c}$	T_D
1	For each $k = 1, \dots, C$	$\times C$
2	For each $j = 1, \dots, D$	$\times D$
1	Initialize $x' = 0$	
1	For each $i = 1, \dots, N_c$	$\times N_c$
1	Load $x_{ij}^{(k)}$	
0	$x' = x' + x_{ij}^{(k)}$ # $\underline{x}_{sum}^{(k)} = \sum_{i=1}^{N_c} \underline{x}_i^{(k)}$	T_A
1	$\mu_j^{(k)} = F_c \cdot x'$	T_M
0	Output $\mu_j^{(k)}$	

$\Rightarrow$ space complexity is $4 + 5 = 9$

$\Rightarrow$ Time complexity is

$$\tau = T_D + C\left[D\left(N_c T_A + T_M\right)\right]$$

$$\tau = CDN_cT_A + CDT_M + T_D \quad (1)$$

Let $T_D = T_A = T_M = 1$

Then $\tau = CDN_c + CD + 1$

2. Assume a parallel machine
that can compute D add's in parallel, D mult's in parallel

We can reduce the time complexity by using <u>Algorithm B</u>:

2 $F_c = \frac{1}{N_c}$

For each $k = 1, \cdots, C$

D Initialize $\underline{x}^{(k)}_{sum} = \underline{0}$

For each $i = 1, \cdots, N_c$

Load $\underline{x}^{(k)}_i$

$\underline{x}^{(k)}_{sum} = \underline{x}^{(k)}_i + \underline{x}^{(k)}_{sum}$

} D adds in parallel 1

$\underline{\mu}^{(k)} = F_c \cdot \underline{x}^{(k)}_{sum}$ } D multiplies in parallel

Output $\underline{\mu}^{(k)}$

1 $\Rightarrow$ time complexity:

1 $\Rightarrow$ space complexity:

We can simplify the analysis by looking at <u>asymptotic complexity</u>

Serial machine running Algorithm $\mathcal{A}$

→ e.g., dependence on N_c, D as they get large. Let C = constant.

→ $\tau = C D N_c T_A + C D T_M + T_D$ (from (1) above)

not relevant (constant)

→ $\tau \sim N_c D \underbrace{C T_A}_{} + D \underbrace{C T_M}_{} \sim N_c D + D = (N_c + 1) D$

$\sim N_c D$

Const. of N_c, D

$\Rightarrow \tau = \mathcal{O}(N_c D)$

big-oh notation.

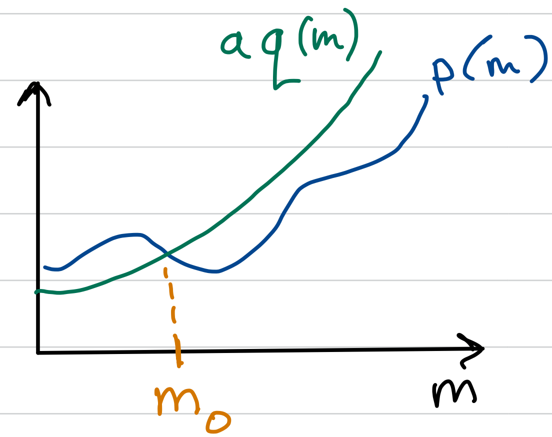
→ Let's be more precise about what $\mathcal{O}(\cdot)$ means

Definitions – big-oh, big-omega, big-theta

Let p and q be functions of m.

- $p(m) = \mathcal{O}(q(m))$ if there exists positive constants a and m_0 such that $0 \le p(m) \le a\,q(m) \quad \forall\, m \ge m_0$.

  $\Rightarrow$ asymptotic upper bound.



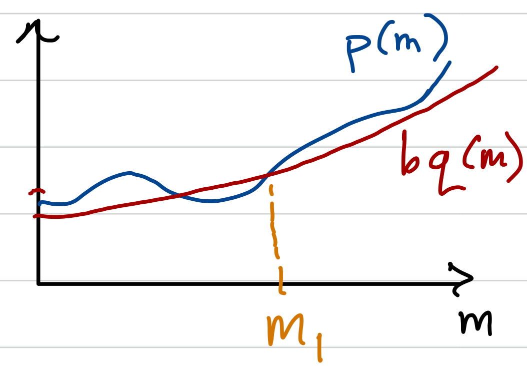
Ex: $T = 5 - 3m + 2m^2 + 1.5m^3 \leftarrow p(m)$

✓ $T = \mathcal{O}(\underbrace{m^3}_{q(m)})$

✓ $T \stackrel{?}{=} \mathcal{O}(m^4)$ ✗ $T \stackrel{?}{=} \mathcal{O}(m^2)$

- $p(m) = \Omega(q(m))$ if there exists positive constants b and m_1 such that $0 \le b\,q(m) \le p(m) \quad \forall\, m \ge m_1$

  $\Rightarrow$ asymptotic lower bound

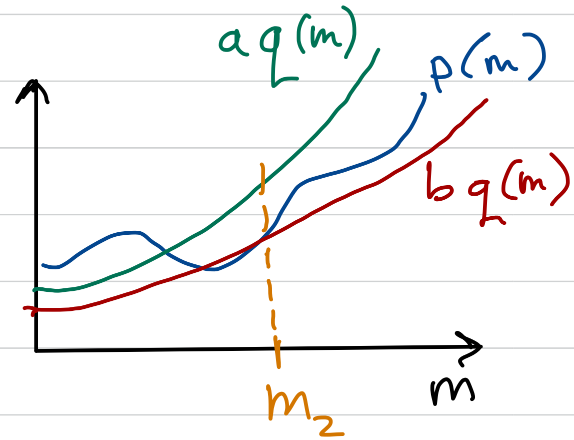


Ex: $T = 5 - 3m + 2m^2 + 1.5m^3 \leftarrow p(m)$

✓ $T \stackrel{?}{=} \Omega(m^3)$, also $T \stackrel{?}{=} \Omega(m^2)$ ✓

- $p(m) = \Theta(q(m))$ if there exists positive constants a, b, and m_2 such that $0 \le b\,q(m) \le p(m) \le a\,q(m) \quad \forall\ m \ge m_2$

$\Rightarrow$ asymptotic tight bound



Ex: $\Upsilon = 5 - 3m + 2m^2 + 1.5m^3 \quad \leftarrow p(m)$

is $\Upsilon = \Theta(m^3)$? ✓

is $\Upsilon = \Theta(m^2)$? ✗

<u>**Comment:**</u> if $\Upsilon = O(q(m))$ and $\Upsilon = \Omega(q(m))$, (same $q(m)$)

then $\Upsilon = \Theta(q(m))$.

Revisit earlier example

Compute the class sample-mean vectors for C classes.

$$\tau = N_c C D T_A + C D T_M + T_D$$

→ Let D, N_c, C be (potentially large) variables.
⇒ T_A, T_M, T_D are constants of D, N_c, C.

$$\tau = \mathcal{O}(N_c C D + C D)$$
$$= \mathcal{O}(C D (N_c + 1))$$
$$= \mathcal{O}(C D N_c)$$