Review of Big-O Notation

- The notation “\( g(n) = O(f(n)) \)” means:
  - “for all large \( n \), \( g(n) \) less than \( c \cdot f(n) \) for some constant \( c > 0 \)”.

- Examples:

  \[ 20n + 50 = O(n). \]

  \[ 5n^2 + 34n + 3 = O(n^2) \]

  \[ 10 = O(1). \]

  \[ 10 \cdot \log(n) + n = O(n). \]

  \[ n \cdot \log(n) + 20n = O(n \log n). \]

  \[ 2^n + 1000 \cdot n^{10} = O(2^n). \]
Review of Big-O Notation

- “Runtime of algorithm is $O(n)$” means that:
  - “In worst case, algorithm requires $O(n)$ operations.”
  - Typically, ‘$n$’ will measure size of input.

- Why is this important?
  - We can only apply $O(2^n)$ algorithms to tiny datasets.
  - We can apply $O(n^2)$ algorithms to medium-sized dataset.
  - We can apply an $O(nd)$ algorithm if one of ‘$n$’ or ‘$d$’ is medium-sized.
  - We can apply $O(n)$ algorithms to huge datasets.
Review of Big-O Notation

- Examples of algorithm runtimes:
  - Finding the maximum in a list of ‘n’ numbers: O(n).
    - We do a constant number of operations for each of the ‘n’ numbers.
  - Finding item number ‘i’ in a list of ‘n’ numbers: O(1).
    - Just return that number.
  - Printing each element times each other element: O(n^2):
    - A constant number of operations for each combination.
  - Finding an element in a sorted list: O(log n).
  - Sorting a list of N numbers: O(n log n).
    - Well-known result, see algorithms class.
  - Classifying an example in depth-m decision tree: O(m).
    - You do a constant number of operations at each depth.
    - N.B., no dependence on dimension ‘d’.
Review of Big-O Notation

- Examples of algorithm runtimes depending on two parameters:
  - Finding the maximum of an ‘n’ by ‘d’ matrix: $O(nd)$
  - Fitting decision stump with ‘n’ objects and ‘d’ features:
    - $O(n^2 d)$ if score costs $O(n)$.
    - $O(nd \log n)$ if all scores cost $O(n \log n)$.
  - Fitting a decision tree of depth ‘m’:
    - Naïve analysis: $2^{m-1}$ trees, so $O(2^{m-1} \ n \ d \ \log(n))$.
    - But each object appears once at each depth: $O(m \ n \ d \ \log n)$.
  - Finding optimal decision tree:
    - There are $O(C(n)\cdot n!)$ tree structures, where $C(n)$ is huge.
    - Then you also have to find the thresholds for each structure.
    - You are never going to do this.