Time vs Space

Running time complexity of algorithms

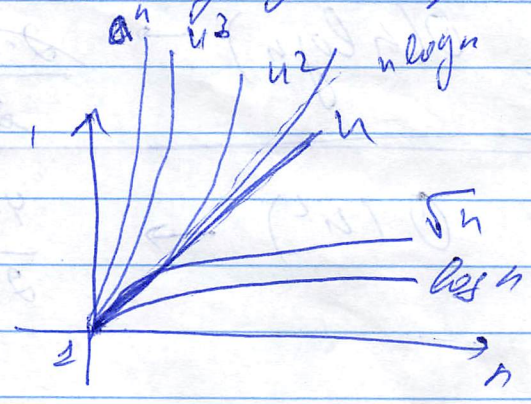
Time complexity — amount of time taken by algorithm to run as a function of size of the input n ← items

Described asymptotically, $n \to \infty$
(how it changes when n grows)

Big "O" notation — asymptotic notation of function growth

$T \sim n$ (unordered search)
$O(n)$

$T \sim O(n^2)$ (naive sort)



$$n + n-1 + n-2 + \dots = \frac{n(n-1)}{2} = \frac{n^2}{2} + \frac{n}{2}$$

Asymptotic constant $\frac{1}{2}$ $n \to \infty$ $O(n^2)$

n (n) $\| n$ $O(n^2)$

n (n) (n) n = $O(n^3)$

Procesor: 2 GHz $\Rightarrow 2 \cdot 10^9$ ops/sec.

Date: 1 GB date = $\frac{10^9 \text{ bytes}}{4 \text{ bytes}} = 2 \cdot 10^8$ numbers (data points)

$O(n) \rightarrow \frac{2 \cdot 10^8}{2 \cdot 10^9} = 0{,}1\ s$

$O(n^2) \rightarrow 4 \cdot 10^{16}$

$O(n \log n) \rightarrow \frac{2 \cdot 10^8 \cdot 8}{2 \cdot 10^9} = 0{,}8\ s$

$O(n^2) \rightarrow \frac{4 \cdot 10^{16}}{2 \cdot 10^9} = 2 \cdot 10^7\ s = \frac{2 \cdot 10^7}{86 \cdot 10^3 \cdot 365} =$

$= \frac{2 \cdot 10^7}{30.000 \cdot 10^3} = \frac{2}{3}$ yr

$O(n \log n) = \frac{2 \cdot 10^8 \cdot 1{,}4 \cdot 10^9}{2 \cdot 10^9} = 10^3\ s = 1400\ s = 23$ min.

$X = n \begin{bmatrix} \ \end{bmatrix}$ (d columns) — d feature

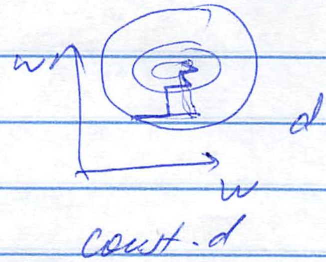
① Linear regression:

$$E = \sum_i (w^T x_i - y_i)^2$$

$$w^* = \min_w E$$

nd — dim, $\sum$ points

optimize



const. d

constant

Model training time complexity — $O(nd^2)$

Model application — $O(d)$

$y = w^T x$

exact solution $w = (X^T X)^{-1} X^T y$

$X^T X \rightarrow$ $d\,[\quad X^T \quad]_n \; [\quad]_d = (\quad)_d \quad O(nd^2)$

$d(\quad)[\,] \sim d^2$

$nd^2 +$

$X^T y$: $[\quad][\,] = [\,]_d \quad nd$

$$O(nd^2 + nd + d^3 + d^2) = O(nd^2)$$

$n \gg d$

(2) Logistic Regression:

$$E = \sum_{i}^{n} \log\left(1 + e^{-y_i(w^T x + w_0)}\right)$$

E $O(nd)$

$w^* = \underset{w}{\text{argmin}}\, E$, at least const-d

Model training — $O(nd^2)$

Model application — $O(d)$ per point.

---

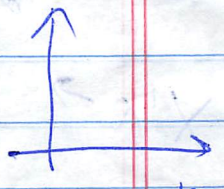
(3) SVM

$$\mathcal{L} = \frac{1}{2}\|w\|^2 + \sum_{i=1}^{n} \lambda_i\left(y_i\left(\underbrace{w_0 + w^T x_i}_{d}\right) - 1\right)$$

d^2

$\mathcal{L} \to O(nd)$

Unknowns $w = \underset{w, \lambda_i}{\text{argmin}}$



Training

Model train = $O(n^2 d)$

Model apply $O(d)$

Kernel SVM

$\mathcal{L} = \alpha_1^* \ldots \alpha_n^* = \underset{\alpha_1 \ldots \alpha_n}{\text{argmax}}\left(-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i x_j + \sum \alpha_i\right)$

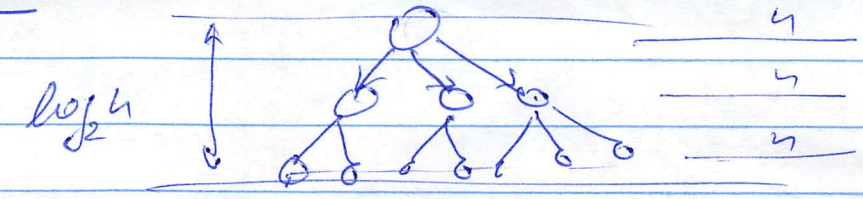
$w = \sum \alpha_i y_i x_i$

Optimization $O(n^2) - O(n^3)$

Model train $= O(n^2)$

Model apply $O(d)$

(4) Decision trees ID3

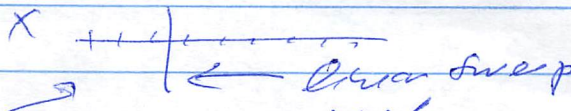


Every step → check among all features - d
compute Information gain - n

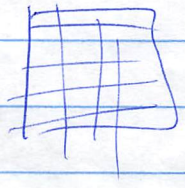
$$\text{Training} = O(n \cdot d \cdot \log n)$$
$$\text{Testing} = O(\log n)$$

Regression trees

n positions
CART



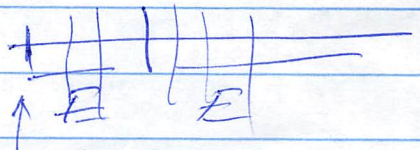
for all variables
for all positions
compute E - n



$d \cdot n \cdot n$ ← for one level.

$O(n^2 d \log n)$

$$\text{Training} = O(n d \log n)$$
$$\text{Testing} = O(\log n)$$



for best split
Compute all n point
constant time updates.
→ going thru all splits
+ E ~ n