WIKIPEDIA

Decision tree model

In computational complexity the **decision tree model** is the model of computation in which an algorithm is considered to be basically a decision tree, i.e., a sequence of *queries* or *tests* that are done adaptively, so the outcome of the previous tests can influence the test is performed next.

Typically, these tests have a small number of outcomes (such as a yes-no question) and can be performed quickly (say, with unit computational cost), so the worst-case time complexity of an algorithm in the decision tree model corresponds to the depth of the corresponding decision tree. This notion of computational complexity of a problem or an algorithm in the decision tree model is called its **decision tree complexity** or **query complexity**.

Decision trees models are instrumental in establishing lower bounds for complexity theory for certain classes of computational problems and algorithms. Several variants of decision tree models have been introduced, depending on the computational model and type of query algorithms are allowed to perform.

For example, a decision tree argument is used to show that a comparison sort of n items must take $n\log(n)$ comparisons. For comparison sorts, a query is a comparison of two items a, b, with two outcomes (assuming no items are equal): either $a < b$ or $a > b$. Comparison sorts can be expressed as a decision tree in this model, since such sorting algorithms only perform these types of queries.

Contents

Comparison trees and lower bounds for sorting

Decision trees are often employed to understand algorithms for sorting and other similar problems; this was first done by Ford and Johnson.[1]

For example, many sorting algorithms are comparison sorts, which means that they only gain information about an input sequence $x_1, x_2, \ldots, x_n$ via local comparisons: testing whether $x_i < x_j$, $x_i = x_j$, or $x_i > x_j$. Assuming that the items to be sorted are all distinct and comparable, this can be rephrased as a yes-or-no question: is $x_i > x_j$?

These algorithms can be modeled as binary decision trees, where the queries are comparisons: an internal node corresponds to a query, and the node's children correspond to the next query when the answer to the question is yes or no. For leaf nodes, the output corresponds to a permutation π that describes how the input sequence was scrambled from the fully ordered list of items. (The inverse of this permutation, π^{-1}, re-orders the input sequence.)

One can show that comparison sorts must use $\Omega(n\log(n))$ comparisons through a simple argument: for an algorithm to be correct, it must be able to output every possible permutation of n elements; otherwise, the algorithm would fail for that particular permutation as input. So, its corresponding decision tree must have at least as many leaves as permutations: $n!$ leaves. Any binary tree with at least $n!$ leaves has depth at least $\log_2(n!) = \Omega(n\log_2(n))$, so this is a lower bound on the run time of a comparison sorting algorithm. In this case, the existence of numerous comparison-sorting algorithms having this time complexity, such as mergesort and heapsort, demonstrates that the bound is tight.[2]:91

This argument does not use anything about the type of query, so it in fact proves a lower bound for any sorting algorithm that can be modeled as a binary decision tree. In essence, this is a rephrasing of the information-theoretic argument that a correct sorting algorithm must learn at least $\log_2(n!)$ bits of information about the input sequence. As a result, this also works for randomized decision trees as well.

Other decision tree lower bounds do use that the query is a comparison. For example, consider the task of only using comparisons to find the smallest number among n numbers. Before the smallest number can be determined, every number except the smallest must "lose" (compare greater) in at least one comparison. So, it takes at least $n-1$ comparisons to find the minimum. (The information-theoretic argument here only gives a lower bound of $\log(n)$.) A similar argument works for general lower bounds for computing order statistics.[2]:214

Linear and algebraic decision trees

Linear decision trees generalize the above comparison decision trees to computing functions that take real vectors $x \in \mathbb{R}^n$ as input. The tests in linear decision trees are linear functions: for a particular choice of real numbers $a_0, \ldots, a_n$, output the sign of $a_0 + \sum_{i=1}^n a_i x_i$. (Algorithms in this model can only depend on the sign of the output.) Comparison trees are linear decision trees, because the comparison between x_i and x_j corresponds to the linear function $x_i - x_j$. From its definition, linear decision trees can only specify functions f whose fibers can be constructed by taking unions and intersections of half-spaces.

Algebraic decision trees are a generalization of linear decision trees that allow the test functions to be polynomials of degree d. Geometrically, the space is divided into semi-algebraic sets (a generalization of hyperplane).

These decision tree models, defined by Rabin[3] and Reingold,[4] are often used for proving lower bounds in computational geometry.[5] For example, Ben-Or showed that element uniqueness (the task of computing $f : \mathbb{R}^n \to \{0,1\}$, where $f(x)$ is 0 if and only if there exist distinct coordinates i, j such that $x_i = x_j$) requires an algebraic decision tree of depth $\Omega(n\log(n))$.[6] This was first showed for linear decision models by Dobkin and Lipton.[7] They also show a n^2 lower bound for linear decision trees on the knapsack problem, generalized to algebraic decision trees by Steele and Yao.[8]

Boolean decision tree complexities

For Boolean decision trees, the task is to compute the value of an n-bit Boolean function $f : \{0,1\}^n \to \{0,1\}$ for an input $x \in \{0,1\}^n$. The queries correspond to reading a bit of the input, x_i, and the output is $f(x)$. Each query may be dependent on previous queries. There are many types of computational models using decision trees that could be considered, admitting multiple complexity notions, called *complexity measures*.

Deterministic decision tree

If the output of a decision tree is $f(x)$, for all $x \in \{0,1\}^n$, the decision tree is said to "compute" f. The depth of a tree is the maximum number of queries that can happen before a leaf is reached and a result obtained. $D(f)$, the *deterministic decision tree* complexity of f is the smallest depth among all deterministic decision trees that compute f.

Randomized decision tree

One way to define a *randomized decision tree* is to add additional nodes to the tree, each controlled by a probability p_i. Another equivalent definition is to define it as a distribution over deterministic decision trees. Based on this second definition, the complexity of the randomized tree is defined as the largest depth among all the trees in the support of the underlying distribution. $R_2(f)$ is defined as the complexity of the lowest-depth randomized decision tree whose result is $f(x)$ with probability at least $2/3$ for all $x \in \{0,1\}^n$ (i.e., with bounded 2-sided error).

$R_2(f)$ is known as the Monte Carlo randomized decision-tree complexity, because the result is allowed to be incorrect with bounded two-sided error. The Las Vegas decision-tree complexity $R_0(f)$ measures the *expected* depth of a decision tree that must be correct (i.e., has zero-error). There is also a one-sided bounded-error version which is denoted by $R_1(f)$.

Nondeterministic decision tree

The nondeterministic decision tree complexity of a function is known more commonly as the certificate complexity of that function. It measures the number of input bits that a nondeterministic algorithm would need to look at in order to evaluate the function with certainty.

Formally, the certificate complexity of f at x is the size of the smallest subset of indices $S \subset [n]$ such that, for all $y \in \{0,1\}^n$, if $y_i = x_i$ for all $i \in S$, then $f(y) = f(x)$. The certificate complexity of f is the maximum certificate complexity over all x. The analogous notion where one only requires the verifier to be correct with 2/3 probability is denoted $RC(f)$.

Quantum decision tree

The quantum decision tree complexity $Q_2(f)$ is the depth of the lowest-depth quantum decision tree that gives the result $f(x)$ with probability at least $2/3$ for all $x \in \{0,1\}^n$. Another quantity, $Q_E(f)$, is defined as the depth of the lowest-depth quantum decision tree that gives the result $f(x)$ with probability 1 in all cases (i.e. computes f exactly). $Q_2(f)$ and $Q_E(f)$ are more commonly known as *quantum query complexities*, because the direct definition of a quantum decision tree is more complicated than in the classical case. Similar to the randomized case, we define $Q_0(f)$ and $Q_1(f)$.

These notions are typically bounded by the notions of degree and approximate degree. The *degree* of f, denoted $\deg(f)$, is the smallest degree of any polynomial p satisfying $f(x) = p(x)$ for all $x \in \{0,1\}^n$. The *approximate degree* of f, denoted $\widetilde{\deg}(f)$, is the smallest degree of any polynomial p satisfying $p(x) \in [0,1/3]$ whenever $f(x) = 0$ and $p(x) \in [2/3,1]$ whenever $f(x) = 1$.

Beals et al. established that $Q_0(f) \geq \deg(f)/2$ and $Q_2(f) \geq \widetilde{\deg}(f)/2$.[9]

Relationships between boolean function complexity measures

It follows immediately from the definitions that for all n-bit Boolean functions f, $Q_2(f) \leq R_2(f) \leq R_1(f) \leq R_0(f) \leq D(f) \leq n$, and $Q_2(f) \leq Q_0(f) \leq D(f) \leq n$. Finding the best upper bounds in the converse direction is a major goal in the field of query complexity.

All of these types of query complexity are polynomially related. Blum and Impagliazzo,[10] Hartmanis and Hemachandra,[11] and Tardos[12] independently discovered that $D(f) \leq R_0(f)^2$. Noam Nisan found that the Monte Carlo randomized decision tree complexity is also polynomially related to deterministic decision tree complexity: $D(f) = O(R_2(f)^3)$.[13] (Nisan also showed that $D(f) = O(R_1(f)^2)$.) A tighter relationship is known between the Monte Carlo and Las Vegas models: $R_0(f) = O(R_2(f)^2 \log R_2(f))$.[14] This relationship is optimal up to polylogarithmic factors.[15] As for quantum decision tree complexities, $D(f) = O(Q_2(f)^4)$, and this bound is tight.[16][15] Midrijanis showed that $D(f) = O(Q_0(f)^3)$,[17][18] improving a quartic bound due to Beals et al.[9]

It is important to note that these polynomial relationships are valid only for *total* Boolean functions. For partial Boolean functions, that have a domain a subset of $\{0,1\}^n$, an exponential separation between $Q_0(f)$ and $D(f)$ is possible; the first example of such a problem was discovered by Deutsch and Jozsa.

Sensitivity conjecture

For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the *sensitivity* of f is defined to be the maximum sensitivity of f over all x, where the sensitivity of f at x is the number of single-bit changes in x that change the value of $f(x)$. Sensitivity is related to the notion of total influence from the analysis of Boolean functions, which is equal to *average* sensitivity over all x.

The **sensitivity conjecture** is the conjecture that sensitivity is polynomially related to query complexity; that is, there exists exponent c, c' such that, for all f, $D(f) = O(s(f)^c)$ and $s(f) = O(D(f)^{c'})$. One can show through a simple argument that $s(f) \leq D(f)$, so the conjecture is specifically concerned about finding a lower bound for sensitivity. Since all of the previously-discussed complexity measures are polynomially related, the precise type of complexity measure is not relevant. However, this is typically phrased as the question of relating sensitivity with block sensitivity.

The *block sensitivity* of f, denoted $bs(f)$, is defined to be the maximum block sensitivity of f over all x. The block sensitivity of f at x is the maximum number t of disjoint subsets $S_1, \ldots, S_t \subset [n]$ such that, for any of the subsets S_i, flipping the bits of x corresponding to S_i changes the value of $f(x)$.[13]

Since block sensitivity takes a maximum over more choices of subsets, $s(f) \leq bs(f)$. Further, block sensitivity is polynomially related to the previously discussed complexity measures; for example, Nisan's paper introducing block-sensitivity showed that $bs(f) \leq D(f) = O(bs(f)^4)$.[13] So, one could rephrase the sensitivity conjecture as showing that, for some c, $bs(f) = O(s(f)^c)$. In 1992, Nisan and Szegedy conjectured that $c = 2$ suffices.[19] This would be tight, as Rubinstein in 1995 showed a quadratic separation between sensitivity and block sensitivity.[20]

In July 2019, 27 years after the conjecture was initially posed, Hao Huang from Emory University proved the sensitivity conjecture, showing that $bs(f) = O(s(f)^4)$.[21] This proof is notably succinct, proving this statement in two pages when prior progress towards the sensitivity conjecture had been limited.[22][23]

Summary of known results

Best-known separations for complexity measures as of October 2020[16]

	D	R_0	R_2	C	RC	bs	s	Q_0	deg	Q	$\widetilde{\deg}$
D		2	2, 3	2	2, 3	2, 3	3, 6	2, 3	2, 3	4	4
R_0	1		2	2	2, 3	2, 3	3, 6	2, 3	2, 3	3, 4	4
R	1	1		2	2, 3	2, 3	3, 6	1.5, 3	2, 3	3, 4	4
C	1	1	1, 2		2	2	2.22, 5	1.15, 3	1.63, 3	2, 4	2, 4
RC	1	1	1	1		1.5, 2	2, 4	1.15, 2	1.63, 2	2	2
bs	1	1	1	1	1		2, 4	1.15, 2	1.63, 2	2	2
s	1	1	1	1	1	1		1.15, 2	1.63, 2	2	2
Q_0	1	1.33, 2	1.33, 3	2	2, 3	2, 3	3, 6		2, 3	2, 4	4
deg	1	1.33, 2	1.33, 2	2	2	2	2	1		2	2
Q	1	1	1	2	2, 3	2, 3	3, 6	1	2, 3		4
$\widetilde{\deg}$	1	1	1	2	2	2	2	1	1	1	

This table summarizes results on separations between Boolean function complexity measures. The complexity measures are, in order, deterministic, zero-error randomized, two-sided-error randomized, certificate, randomized certificate, block sensitivity, sensitivity, exact quantum, degree, quantum, and approximate degree complexities.

The number in the A-th row and B-th column denotes bounds on the exponent c, which is the infimum of all k satisfying $A(f) = O(B(f)^k)$ for all boolean functions f. For example, the entry in the D-th row and s-th column is "3, 6", so $D(f) = O(\mathrm{s}(f)^{6+o(1)})$ for all f, and there exists a function g such that $D(g) = \Omega(\mathrm{s}(g)^{3-o(1)})$.

See also

- Comparison sort
- Decision tree
- Aanderaa–Karp–Rosenberg conjecture
- Minimum spanning tree#Decision trees

References

1. Jr, Lester R. Ford; Johnson, Selmer M. (1959-05-01). "A Tournament Problem" (https://doi.org/10.1080/00029890.1959.11989306). *The American Mathematical Monthly*. **66** (5): 387–389. doi:10.1080/00029890.1959.11989306 (https://doi.org/10.1080%2F00029890.1959.11989306). ISSN 0002-9890 (https://www.worldcat.org/issn/0002-9890).
2. *Introduction to algorithms* (https://www.worldcat.org/oclc/676697295). Cormen, Thomas H. (Third ed.). Cambridge, Mass.: MIT Press. 2009. ISBN 978-0-262-27083-0. OCLC 676697295 (https://www.worldcat.org/oclc/676697295).
3. Rabin, Michael O. (1972-12-01). "Proving simultaneous positivity of linear forms" (http://www.sciencedirect.com/science/article/pii/S0022000072800345). *Journal of Computer and System Sciences*. **6** (6): 639–650. doi:10.1016/S0022-0000(72)80034-5 (https://doi.org/10.1016%2FS0022-0000%2872%2980034-5). ISSN 0022-0000 (https://www.worldcat.org/issn/0022-0000).
4. Reingold, Edward M. (1972-10-01). "On the Optimality of Some Set Algorithms" (https://doi.org/10.1145/321724.321730). *Journal of the ACM*. **19** (4): 649–659. doi:10.1145/321724.321730 (https://doi.org/10.1145%2F321724.321730). ISSN 0004-5411 (https://www.worldcat.org/issn/0004-5411). S2CID 18605212 (https://api.semanticscholar.org/CorpusID:18605212).
5. Preparata, Franco P. (1985). *Computational geometry : an introduction* (https://www.worldcat.org/oclc/11970840). Shamos, Michael Ian. New York: Springer-Verlag. ISBN 0-387-96131-3. OCLC 11970840 (https://www.worldcat.org/oclc/11970840).
6. Ben-Or, Michael (1983-12-01). "Lower bounds for algebraic computation trees" (https://doi.org/10.1145/800061.808735). *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*. STOC '83. New York, NY, USA: Association for Computing Machinery: 80–86. doi:10.1145/800061.808735 (https://doi.org/10.1145%2F800061.808735). ISBN 978-0-89791-099-6. S2CID 1499957 (https://api.semanticscholar.org/CorpusID:1499957).
7. Dobkin, David; Lipton, Richard J. (1976-06-01). "Multidimensional Searching Problems" (https://epubs.siam.org/doi/10.1137/0205015). *SIAM Journal on Computing*. **5** (2): 181–186. doi:10.1137/0205015 (https://doi.org/10.1137%2F0205015). ISSN 0097-5397 (https://www.worldcat.org/issn/0097-5397).
8. Michael Steele, J; Yao, Andrew C (1982-03-01). "Lower bounds for algebraic decision trees" (https://dx.doi.org/10.1016%2F0196-6774%2882%2990002-5). *Journal of Algorithms*. **3** (1): 1–8. doi:10.1016/0196-6774(82)90002-5 (https://doi.org/10.1016%2F0196-6774%2882%2990002-5). ISSN 0196-6774 (https://www.worldcat.org/issn/0196-6774).
9. Beals, R.; Buhrman, H.; Cleve, R.; Mosca, M.; de Wolf, R. (2001). "Quantum lower bounds by polynomials". *Journal of the ACM*. **48** (4): 778–797. arXiv:quant-ph/9802049 (https://arxiv.org/abs/quant-ph/9802049). doi:10.1145/502090.502097 (https://doi.org/10.1145%2F502090.502097). S2CID 1078168 (https://api.semanticscholar.org/CorpusID:1078168).

10. Blum, M.; Impagliazzo, R. (1987). "Generic oracles and oracle classes". *Proceedings of 18th IEEE FOCS*. pp. 118–126.
11. Hartmanis, J.; Hemachandra, L. (1987), "One-way functions, robustness, and non-isomorphism of NP-complete sets", *Technical Report DCS TR86-796, Cornell University*
12. Tardos, G. (1989). "Query complexity, or why is it difficult to separate $NP^A \cap coNP^A$ from P^A by random oracles *A*?". *Combinatorica*. **9** (4): 385–392. doi:10.1007/BF02125350 (https://doi.org/10.1007%2FBF02125350). S2CID 45372592 (https://api.semanticscholar.org/CorpusID:45372592).
13. Nisan, N. (1989). "CREW PRAMs and decision trees". *Proceedings of 21st ACM STOC*. pp. 327–335.
14. Kulkarni, R. and Tal, A. On Fractional Block Sensitivity. Electronic Colloquium on Computational Complexity (ECCC). Vol. 20. 2013.
15. Ambainis, Andris; Balodis, Kaspars; Belovs, Aleksandrs; Lee, Troy; Santha, Miklos; Smotrovs, Juris (2017-09-04). "Separations in Query Complexity Based on Pointer Functions" (https://doi.org/10.1145/3106234). *Journal of the ACM*. **64** (5): 32:1–32:24. arXiv:1506.04719 (https://arxiv.org/abs/1506.04719). doi:10.1145/3106234 (https://doi.org/10.1145%2F3106234). ISSN 0004-5411 (https://www.worldcat.org/issn/0004-5411). S2CID 10214557 (https://api.semanticscholar.org/CorpusID:10214557).
16. Aaronson, Scott; Ben-David, Shalev; Kothari, Robin; Rao, Shravas; Tal, Avishay (2020-10-23). "Degree vs. Approximate Degree and Quantum Implications of Huang's Sensitivity Theorem". arXiv:2010.12629 (https://arxiv.org/abs/2010.12629) [quant-ph (https://arxiv.org/archive/quant-ph)].
17. Midrijanis, Gatis (2004), "Exact quantum query complexity for total Boolean functions", arXiv:quant-ph/0403168 (https://arxiv.org/abs/quant-ph/0403168)
18. Midrijanis, Gatis (2005), "On Randomized and Quantum Query Complexities", arXiv:quant-ph/0501142 (https://arxiv.org/abs/quant-ph/0501142)
19. Nisan, Noam; Szegedy, Mario (1992-07-01). "On the degree of Boolean functions as real polynomials" (https://doi.org/10.1145/129712.129757). *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*. STOC '92. Victoria, British Columbia, Canada: Association for Computing Machinery: 462–467. doi:10.1145/129712.129757 (https://doi.org/10.1145%2F129712.129757). ISBN 978-0-89791-511-3. S2CID 6919144 (https://api.semanticscholar.org/CorpusID:6919144).
20. Rubinstein, David (1995-06-01). "Sensitivity vs. block sensitivity of Boolean functions" (https://doi.org/10.1007/BF01200762). *Combinatorica*. **15** (2): 297–299. doi:10.1007/BF01200762 (https://doi.org/10.1007%2FBF01200762). ISSN 1439-6912 (https://www.worldcat.org/issn/1439-6912). S2CID 41010711 (https://api.semanticscholar.org/CorpusID:41010711).
21. Huang, Hao (2019). "Induced subgraphs of hypercubes and a proof of the Sensitivity Conjecture". *Annals of Mathematics*. **190** (3): 949–955. arXiv:1907.00847 (https://arxiv.org/abs/1907.00847). doi:10.4007/annals.2019.190.3.6 (https://doi.org/10.4007%2Fannals.2019.190.3.6). ISSN 0003-486X (https://www.worldcat.org/issn/0003-486X). JSTOR 10.4007/annals.2019.190.3.6 (https://www.jstor.org/stable/10.4007/annals.2019.190.3.6). S2CID 195767594 (https://api.semanticscholar.org/CorpusID:195767594).
22. Klarreich, Erica. "Decades-Old Computer Science Conjecture Solved in Two Pages" (https://www.quantamagazine.org/mathematician-solves-computer-science-conjecture-in-two-pages-20190725/). *Quanta Magazine*. Retrieved 2019-07-26.
23. Hatami, Pooya; Kulkarni, Raghav; Pankratov, Denis (2011-06-22). "Variations on the Sensitivity Conjecture" (http://www.theoryofcomputing.org/articles/gs004). *Theory of Computing*. **1**: 1–27. doi:10.4086/toc.gs.2011.004 (https://doi.org/10.4086%2Ftoc.gs.2011.004). ISSN 1557-2862 (https://www.worldcat.org/issn/1557-2862). S2CID 6918061 (https://api.semanticscholar.org/CorpusID:6918061).

Surveys

- Buhrman, Harry; de Wolf, Ronald (2002), "Complexity Measures and Decision Tree Complexity: A Survey" (http://homepages.cwi.nl/~rdewolf/publ/qc/dectree.pdf) (PDF), *Theoretical Computer Science*, **288** (1): 21–43, doi:10.1016/S0304-3975(01)00144-X (https://doi.org/10.1016%2FS0304-3975%2801%2900144-X)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Decision_tree_model&oldid=1032311247"

This page was last edited on 6 July 2021, at 17:30 (UTC).