Lowerbounds

12th of October, 2017

- Note this is Chapter 11 in the lecturer's notes

Creating a Decision Tree

We'll follow the same process as before, but add additional info.

At the root of the tree we record the possible inputs of the given size n. We do this so that each time we branch, we can assign every input case to one of the two branches. When we get to the leaves, we can see which cases are left at each leaf.

For each comparison L[i] vs. L[j] in the algorithm, we assign one branch to the case where L[i] < L[j] and one to the case where L[i] > L[j].

Each node also records the result of the swap based on the comparison for the branch leading to that node. E.g. if we start with (a, b, c) and we take the first branch a > b, the node at the end of that branch will contain (b, a, c), as b is less than a, so we would've swapped a and b as a result of the comparison.

Decision trees are related to machine learning – we're grouping inputs according to shared characteristics.

Verification of Ω(nlogn) lowerbound for worst-case time

The intuition is that to have a lot of leaves, you need some long paths.

The length of the longest path in a binary tree T is at least log2(N), where N is the number of leaves in T.

For a complete binary tree, there are two branches at every node, giving a total of 2^d leaves, where the depth of the tree is d. Every path is the same length, which is d, which is equal to log2(2^d) – so it's true for a complete binary tree.

Take an incomplete binary tree, with longest path d'. This tree has the same longest path as the tree you would get from completing this tree. By the previous point, we know the longest path of the complete tree is >= log2(N), where N is the number of leaves in the complete tree. However, completing the tree requires adding leaves, and so N > N'.

Kraft Inequality

Prefix Codes and Binary Trees

Prefix codes are a finite set of binary numbers, chosen so that no number is a prefix of another.

{0, 10, 101, 111} is not a prefix code, because 10 can be found as a prefix of 101.

{0, 10, 110, 111} is a prefix code.

Prefix codes are used to encode characters in files – each character is replaced by a binary number of the code. By using prefix codes, you can decode the file again:

```
A: 0
B: 10
C: 110
D: 111

100111 -> BAD
```

There is only one way of interpreting 100111. Looking at the first 1, you know you need to keep reading to find out what the character is. Once you reach the first 0, you know that letter has to be B, since nothing else starts with 1,0.

Relationship to Binary Trees

Each unique binary tree is equivalent to a unique prefix code.

Every leaf in a binary tree has a unique path, and you can represent that path by the branches you took at each internal node, recording (e.g.) a 0 if you went left and a 1 if you went right.

If a path is a prefix of another (e.g. 010 and 01), then the shorter path must lead to an internal node, rather than a leaf.

So, by assigning each value to a leaf in a binary tree, the paths to those leaves form a prefix code for the values.

Proof

We'll prove the Kraft Inequality for prefix codes, and since every prefix code can be represented as a binary tree and vice-versa, we'll take this to mean it holds for binary trees.

Note that the path length of a binary tree corresponds to the length of the code word for that value at the path's leaf.

Step 1: Order the codes by length, in increasing order.

- Now l1 <= l2 <= lk

Step 2: Note that there are 2^li binary numbers for each length (if no prefix constraints).

Step 3: Given l1, a code word of a length n, how many possible values are there for l2, a code word of length m (> n)?

[...]

Ω(nlogn) for Average-case Time as Well

[...]