Computability and Complexity - Assignment 4

Mikkel Willén
mw@di.ku.dk
Collaborators: Caroline Amalie Kierkegaard, qlj556

April 12, 2024

Task 1

To prove that any (non-constant) monotone Boolean function can be computed by a monotone Boolean circuit, we will show that for any monotone Boolean function, there exists a composition of $\wedge$- and $\vee$-operations that correctly computes the function.

The proof is by induction on the number of variables n in the Boolean function.

Base Case:
For $n = 1$, the function is either the identity function or the constant 1 function, both of which are trivially computed by a single gate(an $\vee$-gate for constant 1, and simply wiring for identity).

Inductive step:
Assume any monotone Boolean function of $k < n$ variables can be computed by a monotone circuit. For a function of n variables, consider decompositions based on the value of one variables x_n leading to two functions of $n - 1$ variables, one for $x_n = 0$ and one for $x_n = 1$. These two functions are also monotone and thus by the inductive hypothesis can be computed by monotone circuits.

Task 2

To prove that if a language L is downward self-reducible, then $L \in \mathbf{PSPACE}$, we approach this problem step-by-step. We do the proof by induction on the length of the strings.

Base case:
For strings of a certain minimal length k, deciding whether $x \in L_k$ can be done without any oracle, as L_0 and L_1, can be hard-coded into the algorithm or decided in constant space.

Inductive step:
Assume we can decide membership in L_k using polynomial space for strings of length at most k. We need to show that we can also decide membership for strings of length $k + 1$ in polynomial space.

Given an input string x of length $k + 1$, the downward self-reducibility property allows us to use an algorithm A with oracle access to L_k to decide if $x \in L$. Instead of an oracle, we can recursively invoke the polynomial space procedure for strings of length at most k, which, by our inductive hypothesis, we assume to be decidable within polynomial space.

The recursive approach might seem to require more than polynomial space due to the depth of recursion. However, because the space used at each level of the recursion can be reused after each oracle call is resolved, the total space required is proportional to the depth of the recursion. The depth of the recursion is linear in the length of the input string x, and at each level, the space used is polynomial in $|x|$. Therefore the overall space complexity remains polynomial in $|x|$. thus fitting the definition of **PSPACE**.

Task 3

To demonstrate the quotient

$$\frac{|R_{t+s} \times \{0,1\}^l|}{|R_t|}$$

is small enough to support the claim of the switching lemma as presented, we start by finding the sizes. The size of R_t is the number of ways to choose t variables out of n, which is $\binom{n}{t}$. Similarly, $|R_{t+s}| = \binom{n}{t+s}$. There are 2^l possible strings in $\{0,1\}^l$, as each position in the string can be either 0 or 1. The size of the Cartesian product $R_{t+s} \times \{0,1\}^l$ is $\binom{n}{t+s} \times 2^l$. Thus the quotient we are interested in is

$$\frac{|R_{t+s} \times \{0,1\}^l|}{|R_t|} = \frac{\binom{n}{t+s} \times 2^l}{\binom{n}{t}}$$

Now to demonstrate that for $t > \frac{n}{2}$, it holds that

$$\binom{n}{t+s} \leq \binom{n}{t} \left(\frac{e(n-t)}{n}\right)^s$$

starting from the relation

$$\binom{n}{t+s} = \binom{n}{t} \cdot \frac{\binom{n-t}{s}}{\binom{t+s}{s}}$$

which follows from

$$\begin{aligned}
\binom{n}{t+s} &= \frac{n!}{(t+s)!(n-t-s)!} \\
&= \frac{n!}{(n-t-s)!} \cdot \frac{1}{(t+s)!} \\
&= \frac{n!}{t!(n-t)!} \cdot \frac{(n-t)!}{s!(n-t-s!} \cdot \frac{t!s!}{(t+s)!} \\
&= \frac{n!}{t!(n-t)!} \cdot \frac{\frac{(n-t)!}{s!(n-t-s)!}}{\frac{(t+s)!}{t!s!}} \\
&= \binom{n}{t} \cdot \frac{\binom{n-t}{s}}{\binom{t+s}{s}}
\end{aligned}$$

Now we can apply the inequality $\binom{n-t}{s} \leq \left(\frac{e(n-t)}{s}\right)^s$ (which follows from the upper bound on the binomial coefficient, $\binom{n}{k} \leq \left(\frac{en}{k}\right)^k$) and observe that $\binom{t+s}{s} \geq 1$, as it counts the ways to choose s items out of $t+s$, which simplifies the expression further, hence we have

$$\binom{n}{t+s} \leq \binom{n}{t} \left(\frac{e(n-t)}{s}\right)^s$$

To reach the final form as claimed, notice that $s \leq n$, thus replacing s with n in the denominator provides the bound

$$\binom{n}{t+s} \leq \binom{n}{t} \left(\frac{e(n-t)}{n}\right)^s$$

Given that $l = O(s \log k)$ we can express 2^l as $n^{O(s)}$ given that l scales with $s \log k$.

Combining the bounds we found, we get

$$\frac{|R_{t+s} \times \{0,1\}^l|}{|R_t|} \leq \left(\frac{e(n-t)}{n}\right)^s \times n^{O(s)} = n^{-\Omega(s)} \times n^{O(s)}$$

The specific negative exponent in $n^{-\Omega(s)}$ and the positive one in $n^{O(s)}$ depends on the constants hidden in the O and Ω notations, thus when the constants and the specific bounds are chosen, the overall probability diminishes exponentially with s, showing that the likelihood of encountering a "bad" restriction is indeed $n^{-\Omega(s)}$.

Task 4

The transformation of a given bounded-depth polynomial-size circuit C into an equivalent circuit C' that satisfies the conditions listed requires a systematic approach that preserves the computational power while conforming to the structural constraints. Here is how each of the modifications can be achieved.

We start by ensuring all gates have fan-out 1. This is achieved by duplicating gates to ensure that each output is used by exactly one gate in the next level. If a gate has a fan-out greater than 1, we create enough copies of this gate and its subtree so that each copy provides output to exactly one gate that requires it in the subsequent level. This duplication increases the size of the circuit but does not affect its depth. since each gate is duplicated a number of times that is at most linear in the size of the circuit, the size of C' grows at most polynomially with respect to the size of C.

Now we will show how to move $\neg$-gates to the input level. De Morgan's laws are utilized to push $\neg$-gates towards the inputs. This process involves flipping $\wedge$ to $\vee$ and vice versa, down the circuit whenever a $\neg$-gate is pushed one level lower. This operation does not increase the depth of the circuit and only introduces a constant factor to the size since each $\neg$-gate push down potentially doubles the number of gates at that level, because of the duplication needed to maintain the tree structure, leading to a polynomial increase in size.

Next is to have alternating $\wedge$- and $\vee$-gates at each level. This can be ensured by inserting dummy layers if necessary. If two consecutive layers of the same type are encountered, and identity gate layer (e.g $\wedge$ with a single input) can be inserted between them to maintain the alternation without affecting the logical function computed by the circuit. This process might at most double the depth of the circuit, thereby adding a constant to the depth since the original depth d becomes $2d$, which is still a constant for bounded-depth circuits.

Lastly, we want to bound the fan-in of $\wedge$ gates at the bottom level. If the fan-in of some $\wedge$-gates at the bottom is more than the allowed constant K, we can use a tree structure to break down these gates into multiple levels of $\wedge$-gates, each with a fan-in of at most K. This might increase the depth of the circuit by a logarithm of the maximum fan-in of these gates, which is a constant increase for fixed K and does not affect the polynomial size of the circuit.

Now let us analyse the size and depth in total. We denote the size of C as S and the depth as d.

The depth increases by at most a constant factor due to the insertion of dummy layers and the restructuring to reduce fan-in at the bottom level. Therefore, the depth of C' is $O(d)$, where d is the original depth of C.

The size might increase polynomially due to duplications for ensuring fan-out of 1 and for pushing $\neg$-gates down. Specifically, the increase is polynomial in S, making the size of C' $O(S^k)$ for some constant k, which depends on the operations performed, but remains polynomial in S.

Thus, C' maintains the property of being a bounded-depth polynomial-size circuit, effectively computing the same function as C under the new structural constraints.

Task 5

To tackle this problem, we start by looking at the stuff we have. We have a Boolean function $f : \{0,1\}^n \to \{0,1\}$ represented by a k-DNF formula F. A restriction $\rho \in R_t$ sets exactly t of the n variables of f to either 0 or 1, reducing the function f to $f \upharpoonright_\rho$, which depends only on the remaining $n-t$ variables. We now need to prove, that the probability of $f \upharpoonright_\rho$ not being a constant function is $O\left(\frac{(n-t)k}{n}\right)$ when ρ is chosen uniformly at random from R_t.

A "bad" restriction ρ is one for which $f \upharpoonright_\rho$ is not constant. We will denote the set of all such bad restrictions by B. We will construct an injective (one-to-one) mapping from B to $r_{t+1} \times A$, where A is a set of size $O(k)$.

A restriction ρ is "bad" if, after applying ρ, there exists at least one term in F that is not eliminated and F is not reduced to a constant. Each term in F has at most k literals. For a term to survive under ρ, none of its variables can be set to 0 by ρ.

Now lets consider a bad restriction $\rho \in B$. this means $f \upharpoonright_\rho$ is not constant, which implies there exists at least one term in the k-DNF that is not contradicted by ρ, but does not evaluate to 1 under all setting of the remaining variables. For each bad restriction ρ, we can identify a specific variable x_i among the remaining $n-t$ variables that, if fixed, would make $f \upharpoonright_\rho$ constant. We can then form a new restriction ρ' by adding this variables to ρ. This maps ρ to a pair $(\rho, x_i$. The set A here represents the possible variables x_i that we can fix to ensure the function becomes constant. Since each term has at most k literals, there are at most k such variables to consider for any term, making $|A| = O(k)$.

Since we have an injective mapping from B to $R_{t+1} \times A$, the number of bad restrictions is at most the size of $R_{t+1} \times A$. The total number of restrictions in R_t is $\binom{n}{t}$, and for R_{t+1}, it is $\binom{n}{t+1}$. The size of A is $O(k)$, and thus the probability that a randomly chosen ρ from R_t is bad is therefore

$$\frac{|B|}{|R_t|} \leq \frac{\binom{n}{t+1} \cdot O(k)}{\binom{n}{t}} = O\left(\frac{(n-t)k}{n}\right)$$

The inequality follows from the fact that choosing an additional variable to fix, increases the number of ways to choose variables by a factor of $\frac{n-t}{t+1}$, and since we are multiplying by $O(k)$ for the set A, the final expression simplifies to $O\left(\frac{(n-t)k}{n}\right)$.

Task 6

To conduct a proof, that the VC dimension of the concept class of half-spaces in $\mathbb{R}^d$ is at most $d+1$, we will systematically demonstrate that no set of $d+2$ points in general position in $\mathbb{R}^d$ can be shattered by the class of half-spaces, establishing the the VC dimension is at most $d+1$.

Suppose we have $d+2$ points in $\mathbb{R}^d$. By the pigeonhole principle and the properties of linear algebra, these points cannot all be linearly independent, as there are more points than dimensions. Using Radon's theorem, which states that any $d+2$ points in $\mathbb{R}^d$ can be partitioned into two non-empty subsets whose convex hulls intersect, we understand that there exists no single linear separator (i.e. half-space) that can classify both subsets into separate classes without error if the labeling is chosen adversarially.

Let us assume, for the sake of contradiction, that $d+2$ points can be shattered. Consider a specific partition of the $d+2$ points into two subsets S_1 and S_2 as per Radon's theorem. We assign binary labels such that all points in S_1 have label 0 and all points in S_2 have label 1. Since the convex hulls of S_1 and S_2 intersect, there is no half-space that can completely separate all points in S_1 from all points in S_2 based on the given labels. This leads to a contradiction, because, by our assumption, there must be such a half-space if the set of $d+2$ points were to be shattered.

Because any set of $d+2$ points in $\mathbb{R}^d$ fails to be shattered due to the inability to find a half-space that satisfies every possible labeling, we conclude that the maximum number of points that can be shattered, and thus the VC dimension, is at most $d+1$.

Task 7

We will construct a proof that there is a constant $C > 0$ such that for all integers n, the sample complexity of PAC learning the class $\{f_{n,i} : i \in \{0, 1, \ldots, n+1\}\}$ is at most C. this class consists of step functions or thresholds functions on a domain of size n.

For the class $\{f_{n,i}\}$, each function $f_{n,i}$ is defined as

$$f_{n,i}(x) = \begin{cases} 0 & \text{if } x < i \\ 1 & \text{if } x \geq i \end{cases}$$

The VC dimension for this class is 1. This is because we can choose any single point in $[n]$, and $\{f_{n,i}\}$ can realize both possible labelings on this point, either 0 or 1 by setting i appropriately. However, no set of two points can be completely shattered, because once the threshold i is set to shatter one point, it automatically determines the labels for all other points based on their position relative to i.

By theorem 1 from NotesLec20-22.pdf, we have that the sample complexity of PAC learning H is equal, up to universal constants, to the VC dimension of H. This theorem indicates that if the VC dimension is d, then the sample complexity m of PAC learning H is

$$m \leq O(d)$$

Given that the VC dimension $d = 1$ for $\{f_{n,i}\}$, we establish that the sample complexity is a constant independent of n.

From the formal application of VC theory, it is evident that the sample complexity for learning the class $\{f_{n,i} : i \in \{0, 1, \ldots, n+1\}\}$ via the PAC model is at most a constant C. This constant does not depend on n due to the consistent VC dimension of 1 across all n.

Task 8

To prove from first principles that there is a constant $C > 0$ such that the sample complexity of PAC learning a class $H \subseteq \{0,1\}^n$ of size $|H| = m$ is at most $C \log_2 m$, we will focus on the PAC learning model, the VC dimension, and how they determine the sample complexity.

By the Sauer-Shelah-Perles lemma, the size of H is bounded by

$$|H| \leq \sum_{i=0}^{d} \binom{n}{i}$$

This expression indicates that H grows as function of d. For large n, this is $O(n^d)$, implying a dependency of $|H|$ on d. Since $|H| = m$, we invert the relationship to estimate d. As $H|$ grows combinatorially with d, we can approximate d as

$$d \approx \log m$$

This is derived from the observation that the number of hypotheses, i.e. the different ways d points can be labeled, is exponential in d and must at least cover all hypotheses in H.

Give that the sample complexity is $O(d)$ and $d \approx \log m$, the sample complexity N of learning H is thus

$$N = O(\log m)$$

and thus there exists a constant $C > 0$ such that

$$N \leq C \log_2 m$$

Task 9

To prove from first principles that the sample complexity of PAC learning a concept class $\mathcal{C}$ consisting of all functions from $\{0,1\}^d$ to $\{0,1\}$ is at least $\frac{d}{4}$, we look at the relationship between the VC dimension of the concept class and its sample complexity.

The VC dimension of a concept class $\mathcal{C}$, is the maximum number of points that can be shattered by $\mathcal{C}$. A class $\mathcal{C}$ shatters a set of points if, for every possible labeling of these points, there exists at least one function within the class that can replicate this labeling exactly. In our case, $\mathcal{C}$ contains all possible functions from $\{0,1\}^d$ to $\{0,1\}$, and thus it can shatter any set of up to d points, where each point corresponds to an input of distinct binary strings of length d, as each point can independently be assigned either 0 or 1, reflecting the complete function mapping capability of the set $\{0,1\}^d$ to $\{0,1\}$.

The sample complexity for PAC learning is at least linearly proportional to the VC dimension. A concept class is PAC learnable if for any $\epsilon > 0$ and $\delta > 0$, there exists a learning algorithm that, with probability at least $1-\delta$, produces a hypothesis that has an error of at most ϵ using a polynomial, in terms of ϵ, δ and d, number of samples.

Assume, for contradiction, that the sample complexity of PAC learning the concept class $\mathcal{C}$ is less than $\frac{d}{4}$. We denote this assumed sample complexity as m, where $m < \frac{d}{4}$.

According to the fundamental theorem of statistical learning, the sample complexity m if learning $\mathcal{C}$ is proportional to the VC dimension, denoted as d, and it is bounded by $\Omega(d)$, up to logarithmic factors of ϵ and δ.

Each example provides at most 1 bit of information, and thus m examples provide at most m bits of information. However, to distinguish between the 2^{2^d} different functions in $\mathcal{C}$, we would require at least 2^d bits of information to uniquely identify one function. Assuming $m < \frac{d}{4}$ we can calculate

$$m < \frac{d}{4} \Rightarrow 4m < d$$

and thus

$$2^{4m} < 2^d$$

where 2^{4m} represents the maximum number of different hypotheses or functions that can be distinguished based on the m bits of information. Since $2^{4m} < 2^d$, it is evident that m samples are insufficient to cover all possible functions, leading to an inability to guarantee learning within any non-trivial bounds of ϵ and δ.

Thus the assumption that $m < \frac{d}{4}$ leads to the conclusion that the number of samples is insufficient to distinguish between a significant fraction of possible functions in $\mathcal{C}$, and thus, by assuming a lower sample complexity than $\frac{d}{4}$ and showing how this leads to an inability to meet the PAC criteria of learning effectively, we have demonstrated a contradiction. Therefore, the sample complexity of PAC learning for the concept class $\mathcal{C}$ must be at least $\frac{d}{4}$, as any lesser number of samples fails to provide the necessary informational coverage to ensure learning as defined in PAC.