CSS.201.1 Algorithms

Instructor: Umang Bhaskar

TIFR 2024, Aug-Dec

Scribe: Soham Chatterjee

sohamchatterjee999@gmail.com
Website: sohamch08.github.io

Contents

Chapter 1

Derandomization

In this section we will see a derandomization technique called Conditional Expectation. With this technique we will show derandomization of some randomized algorithms in the following sections.

1.1 Conditional Expectation

Let $\mathscr{A}$ be a randomized algorithm which is successful with probability at least $\frac{2}{3}$. Suppose $\mathscr{A}$ uses m random bits and suppose the random bits are $R_1, \ldots, R_m$. Then we have

$$\mathop{\mathbb{P}}_{R_1,\ldots,R_m}[\mathscr{A}(x, R_1, \ldots, R_m) = \text{Correct}] \geq \frac{2}{3}$$

We want to derandomize $\mathscr{A}$.

Now think of $\mathscr{A}$ as a binary tree which, given x, branches on the sampled value of each random bit R_i where it goes to left child if the random bit takes value 0 and goes to right child if the random bit takes value 1. Every path in this tree from root to leaf corresponds to different possible random strings and the leaf nodes corresponds to the output of the algorithm with the corresponding random string. Since $\mathscr{A}$ succeeds with probability at least $\frac{2}{3}$ means that at least $\frac{2}{3}$ of the leaves are good outputs for the input x.

Idea. *To derandomize $\mathscr{A}$ we need to find a deterministic algorithm that traverses from the root to a leaf which at any branch at level i chooses a direction which leads to a good output.*

Now suppose $r_1, \ldots, r_m \in \{0, 1\}$ denote the values taken by the random variables $R_1, \ldots, R_m$. Now let $P(r_1, \ldots, r_i)$ denote the fraction of the leaves of the subtree below the node obtained by following the path $r_1, \ldots, r_i$. Formally,

$$P(r_1, \ldots, r_i) = \mathbb{P}[\mathscr{A}(x, R_1, \ldots, R_m) \mid R_1 = r_1, \ldots, R_i = r_i] = \frac{1}{2}P(r_1, \ldots, r_i, 0) + \frac{1}{2}P(r_1, \ldots, r_i, 1)$$

From the last equality it is clear that there is a choice r_{i+1} such that $P(r_1, \ldots, r_{i+1}) \geq P(r_1, \ldots, r_i)$. Therefore to find a good path in the tree it suffices at each branch to pick such an $r \in \{0, 1\}$. Then we would have

$$P(r_1, \ldots, r_m) \geq P(r_1, \ldots, r_{m-1}) \geq \cdots \geq P(r_1) \geq \mathbb{P}[\mathscr{A}(x, R_1, \ldots, R_m) = \text{Correct}] \geq \frac{2}{3}$$

Since $P(r_1, \ldots, r_m)$ is either 0 or 1 it must be 1.

1.2 Max-SAT

Max-SAT

Input: SAT formula φ with n variables and m clauses and non negative weights w_c on clauses.

Question: Given a SAT formula φ with n variables and m clauses and non negative weights w_c on clauses find an assignment that maximizes weight of satisfied clauses.

We will first show a randomized algorithm for this problem. Then we will use conditional expectation to derandomize the algorithm.

1.2.1 Randomized Algorithm

First lets see what is the expected weight of satisfied clauses. Let Y_c be the indicator random variable if clause C is satisfied. Suppose there are k variables in C. Then we have $\mathbb{E}[Y_c] = 1 - \frac{1}{2^k} \geq \frac{1}{2}$. Therefore expected weight of satisfied clauses is

$$\mathbb{E}\left[\sum_C w_c Y_c\right] = \sum_C w_c \mathbb{E}[Y_c] \geq \frac{1}{2}\sum_C w_c$$

Let OPT be the optimal Max-SAT solution for the given formula. Then we have $\sum\limits_C w_c \geq \text{OPT}$. Therefore

$$\mathbb{E}\left[\sum_C w_c Y_c\right] \geq \frac{1}{2}\text{OPT}$$

Hence we have the following randomized algorithm:

Algorithm 1: 2-Approximate Max-SAT

Input: SAT formula φ with n variables and m clauses and non negative weights w_c on clauses.
Output: Find an assignment that maximizes weight of satisfied clauses.

```
1 begin
2     for i ∈ [n] do
3         x_i ⟵ Pick a value from {0, 1} uniformly at random
4     return x
```

By the above discussion we have an assignment with an expected weight of satisfied clauses at least half the maximum.

1.2.2 Derandomization

Now we want to derandomize the algorithm using conditional expectation. Let $X_1, \ldots, X_n$ denote the random variable for each variables and $x_1, \ldots, x_n \in \{0, 1\}$ denote the value the random variables took. A key step will be evaluate the conditional probabilities:

$$\mathbb{E}\left[\sum_C w_c Y_c \mid X_1 = x_1, \ldots, X_i = x_i\right] = \sum_C w_c \mathbb{P}[Y_c = 1 \mid X_1 = x_1, \ldots, X_i = x_i] \quad \forall\, i \in [n]$$

Hence we have to find the value of $\mathbb{P}[Y_c = 1 \mid X_1 = x_1, \ldots, X_i = x_i]$, $\forall\, i \in [n]$. Now if the clause C is already satisfied by the setting $x_1, \ldots, x_i$ then $Y_C = 1$. Else if C has r variables from $x_{i+1}, \ldots, x_n$ then

$$\mathbb{P}[Y_c = 1 \mid X_1 = x_1, \ldots, X_i = x_i] = 1 - \frac{1}{2^r}$$

. Now if at height i, we find $\mathbb{E}\left[\sum_C w_c Y_c \mid X_1 = x_1, \ldots, X_i = 0\right]$ and $\mathbb{E}\left[\sum_C w_c Y_c \mid X_1 = x_1, \ldots, X_i = 1\right]$ and which ever gives the higher value we will set the assignment for X_i to be that one. Thus we can derandomize the algorithm.

1.3 Set Balancing

Chapter 2

Bibliography

[Ide16] Martin Idel. A review of matrix scaling and sinkhorn's normal form for matrices and positive maps. September 2016.

[LSW98] Nathan Linial, Alex Samorodnitsky, and Avi Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98*, STOC '98, pages 644–652. ACM Press, 1998.