An Algorithmist's Toolkit — Brown CSCI 2952T – Spring 2026

Lecture 4: Probabilistic Method, Union Bounds and the LLL

February 11, 2026 — Lecturer: D Ellis Hershkowitz — Scribe: Renxuan Yao, Joshua Kou

1 Overview

Today we explore the **Probabilistic Method**, specifically applied to the Satisfiability problem (k-SAT). The lecture is structured into two main parts:

1. Using the Probabilistic Method to show a solution exists via:
   - The Union Bound
   - Independence
   - The Lovász Local Lemma (LLL)
2. From Probabilistic Method to Algorithm
   - Bounding bad settings via compressions
   - Compressions from recursion trees

2 Recall: The Probabilistic Method Framework

(to show $(*)$ is possible)

(a) Define a Random Process.

(b) Define "Bad Events" $B_1, B_2, \ldots$ such that if none of them occur $(\bar{B}_1 \cap \bar{B}_2 \cap \ldots)$, then $(*)$ holds.

(c) Show Probability is Positive: We must prove:

$$Pr\left(\bigcap_i \overline{B}_i\right) > 0$$

3 Definitions: k-SAT

Definition 1 (Literal)**.** *A literal is a boolean variable or its negation* (x *or* $\bar{x}$)*.*

- Example: x, $\bar{x}$.

Definition 2 (Clause)**.** *A clause is the logical "OR"* ($\vee$) *of distinct literals.*

- A k-clause contains exactly k literals.
- Example: $x_1 \vee \bar{x}_2 \vee x_3$.

Definition 3 (k-SAT Formula)**.** *A k-SAT formula is the logical "AND"* ($\wedge$) *of m clauses, where each clause is a k-clause involving variables $x_1, \ldots, x_n$.*

- Example (2-SAT):
$$(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2)$$

Definition 4 (Satisfiable)**.** *There exists a truth assignment making the formula true.*

- The example in **Definition 3** is satisfiable by setting $x_1 = \text{True}, x_2 = \text{True}$. However, adding $(\bar{x}_1 \vee \bar{x}_2)$ renders it unsatisfiable.

4 Proving Satisfiability

Intuition:

- Many variables + Few clauses = Easy to satisfy.
- Many clauses + Few variables = Hard to satisfy.

4.1 Method 1: The Union Bound

Fact 1. *Any k-SAT Clause w/ $m \leq \frac{2^k}{e} - 1$ clauses is Satisfiable.*

The Random Process: Independently assign each variable x_i:

$$x_i = \begin{cases} \text{True} & \text{w/ prob } 0.5 \\ \text{False} & \text{w/ prob } 0.5 \end{cases}$$

Bad Events: Let B_i be the event that the i-th clause is **not** satisfied. For a k-clause, there is only 1 specific assignment of its k literals that makes it false (e.g., all literals must evaluate to False).

$$Pr(B_i) = \frac{1}{2^k}$$

Analysis: By the Union Bound, the probability that *at least one* clause is unsatisfied is:

$$Pr\left(\bigcup_i B_i\right) \leq \sum_{i=1}^{m} Pr(B_i) = \sum_{i=1}^{m} \frac{1}{2^k} = \frac{m}{2^k} = \frac{2^k}{e}\frac{1}{2^k} = \frac{1}{e} < 1$$

For the formula to be satisfiable, we need the probability of the "bad events" to be strictly less than 1 (so the complement event has probability > 0).

$$Pr\left(\bigcup B_i\right) < 1 \implies Pr\left(\bigcap \overline{B}_i\right) > 0$$

Note: In class we used $m \leq 2^k - 1$, which can be proved with the same process.

4.2 Method 2: Independence

Definition 5 (Overlap)**.** *A k-SAT formula has overlap α if each clause shares variables with at most α other clauses.*

- Example: $(x_1 \vee x_2) \wedge (x_3 \vee x_4) \rightarrow$ overlap 0
- Example: $(x_1 \vee \bar{x}_2) \wedge (x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\bar{x}_3 \vee x_4) \rightarrow$ overlap 3

Fact 2. *Any k-SAT Clause w/ $\alpha = 0$ is Satisfiable.*

Even though we know a k-SAT formula with no overlap ($\alpha = 0$) is always satisfiable, we cannot prove this using the Union Bound. We can distinguish between the *algebraic* failure and the *moral* (intuitive) reason for this failure.

4.2.1 Algebraically

Suppose we have $m = 2^k$ clauses. We know that for any single clause i, the probability it is unsatisfied is $Pr(B_i) \leq (1/2)^k$.

If we apply the Union Bound:

$$Pr\left(\bigcup_i B_i\right) \leq \sum_{i=1}^{m} Pr(B_i) = \sum_{i=1}^{2^k} \frac{1}{2^k} = \frac{1}{2^k} \cdot 2^k = 1$$

Since the upper bound is **not strictly less than 1** ($\not< 1$), the Probabilistic Method yields no conclusion. We cannot guarantee that a "good" assignment exists, even though we know one must.

4.2.2 Morally: Disjointness vs. Independence

The Union Bound sums the probabilities of events, effectively treating them as if they were disjoint (non-overlapping) in the worst case.

- **Union Bound View:**

  $$\text{Sum of Areas} \approx \text{Total Area}$$

  This is a good upper bound if the events B_i are **(mostly) disjoint**.

- **Independence View:** If events are independent (and have non-zero probability), they **are not disjoint**.

  $$A \perp B \implies Pr(A \cap B) = Pr(A)Pr(B) > 0 \implies A \cap B \neq \emptyset$$

- **The Conflict:** In our case, the bad events B_i are **independent**.

  $$\text{Independent} \implies \text{Not Disjoint} \implies \text{Union Bound is Bad}$$

4.2.3 The Correct Approach: Independence

Since the events are independent, we should calculate the probability of the "good" event directly using products rather than sums:

$$Pr\left(\overline{B}_i\right) = 1 - \frac{1}{2^k} > 0 \quad \forall i$$

$$Pr\left(\bigcap_i \overline{B}_i\right) = \prod_i Pr\left(\overline{B}_i\right) = \prod_i \left(1 - \frac{1}{2^k}\right) > 0$$

This product is strictly positive regardless of the number of clauses m.

4.3 Method 3: The Lovász Local Lemma (LLL)

Fact 3. *Any k-SAT formula with* $\alpha \leq \frac{2^k}{e} - 1$ *is satisfiable.*

The LLL allows us to prove existence when events are "mostly" independent (small dependencies).

Definition 6 (Mutual Independence)**.** *Event* A *is mutually independent of a set of events* $\mathcal{B} = \{B_1, \dots\}$ *if* A *is independent of any boolean combination of events in* $\mathcal{B}$*.*

Definition 7 (Dependency Graph)**.** *Let* $\mathcal{B} = \{B_1, \dots, B_m\}$ *be a set of events.* $G = (\mathcal{B}, E)$ *is a dependency graph if for every* $B \in \mathcal{B}$*,* B *is mutually independent of all events in* $\mathcal{B} \setminus (\Gamma(B) \cup \{B\})$*, where* $\Gamma(B)$ *are the neighbors of* B*.*

In k-SAT, B_i and B_j are connected in the dependency graph if their clauses share variables.

Theorem 1 (Symmetric LLL)**.** *Given events* $\mathcal{B}$ *with dependency graph* G *of maximum degree* Δ*. If there exists a* P *such that:*

1. $Pr(B) \leq P$ *for all* $B \in \mathcal{B}$
2. $e \cdot P \cdot (\Delta + 1) \leq 1$

Then:

$$Pr\left(\bigcap_i \overline{B}_i\right) > 0$$

4.3.1 Proof of Fact 3 using LLL

Any k-SAT formula where each clause shares variables with at most $\Delta \leq \frac{2^k}{e} - 1$ other clauses is satisfiable.

Proof. Let the bad events $B = \{B_1, \dots, B_m\}$ correspond to unsatisfied clauses.

- We know $Pr(B_i) \leq \frac{1}{2^k} = P$.
- The overlap corresponds to the degree in the dependency graph. Let $\Delta = \alpha$.
- The LLL condition requires $e \cdot P \cdot (\Delta + 1) \leq 1$.

Substituting values:

$$e \cdot \frac{1}{2^k} \cdot (\Delta + 1) \leq 1 \implies \Delta + 1 \leq \frac{2^k}{e} \implies \Delta \leq \frac{2^k}{e} - 1$$

Since the condition holds, $Pr(\cap \overline{B}_i) > 0$, and a satisfying assignment exists. □

5 LLL as a Union Bound Generalization

Suppose n events $B_1, B_2, \ldots, B_n$ w/ $\Pr(B_i) \leq p \quad \forall i$

UB: If $p \cdot n < 1$ then $\Pr(\bigcup_i B_i) \leq p \cdot n < 1$ so $\Pr(\bigcap_i \overline{B_i}) > 0$

LLL on Complete graph: If $e \cdot p \cdot (\Delta + 1) = e \cdot p \cdot n \leq 1$ then $\Pr(\bigcap_i \overline{B_i}) > 0$

6 From Probabilistic Method to Algorithms

Suppose the number of clauses is $\leq \frac{2^k}{e} - 1$. This implies that the probability that a single random assignment does **not** satisfy the formula is $\leq \frac{1}{e}$.

6.1 Boosting

Algorithm 1 (for Union Bound conditions)

Assign $x_1, x_2, \ldots, x_n$ Uniformly At Random (UAR).
while there exists an unsatisfied clause **do**
    Resample **all** variables $x_1, \ldots, x_n$.
end while
return $x_1, \ldots, x_n$

Analysis: We calculate the probability that the algorithm finishes within r iterations:

$$Pr(\leq r \text{ iterations}) = 1 - Pr(> r \text{ iterations}) \geq 1 - \left(\frac{1}{e}\right)^r$$

So, if we set $r = \ln n$:

$$Pr(\leq \ln n \text{ iterations}) \geq 1 - \frac{1}{n}$$

6.2 Moser-Tardos Algorithm(MT)

Algorithm 2 (for LLL conditions)

```
Assign x_1, x_2, ..., x_n Uniformly At Random.
while there exists an unsatisfied clause c do
    Fix(c)
end while
return x_1, ..., x_n
procedure Fix(c)
    Resample each variable x ∈ c (independently).
    for each clause c' sharing variables with c (including c itself) do
        if c' is unsatisfied then
            Fix(c')                                        ▷ Recursive repair
        end if
    end for
end procedure
```

7 Moser-Tardos algorithm

Theorem 2. *If $\alpha \leq 2^{k-C}$ for a sufficiently large constant C, then Moser-Tardos finds a satisfying assignment in $O(m)$ fixes in expectation.*

We model randomness as picking a random bit string b. Let $\mathsf{MT}(b)$ refer to calling the Moser-Tardos function with random bits taken from b.

For $s \in \mathbb{N}$, let n_s be the number of random bits used after s fixes. n bits are needed for the initial assignment of $x_1, \ldots, x_n$, and for each fix we need k bits to replace the k variables in the clause. Hence

$$n_s = n + sk$$

Let B_s be the set of "bad" strings, namely the set of bit strings b such that $\mathsf{MT}(b)$ does not find satisfying assignments within s fixes.

Let X be the number of fixes required in MT. To show that $\mathbb{E}[X] = O(m)$, it suffices to show that

$$\frac{|B_s|}{2^{n_s}} < 2^{m-3s}.$$

To see this, we note that

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{s=0}^{\infty} s \cdot \Pr[X = s] \\
&\leq m + \sum_{s \geq m} s \cdot \Pr[X = s] \\
&\leq m + \sum_{s \geq m} s \cdot \Pr[X \geq s] \\
&= m + \sum_{s \geq m} s \cdot \frac{|B_s|}{2^{n_s}} \\
&\leq m + \sum_{s \geq m} s \cdot 2^{m-3s} \\
&= m + O(1)
\end{aligned}$$

Given $B \subseteq \{0,1\}^n$, a **compression** of B is an injective function $\mathcal{C} : B \hookrightarrow \{0,1\}^{n'}$. The **advantage** of $\mathcal{C}$ is defined to be $n - n'$. Under the lens of compressions, we translate our condition that

$$\frac{|B_s|}{2^{n_s}} < 2^{m-3s}$$

to finding a compression $\mathcal{C}_s : B_s \to n'_s$ of advantage $3s - m$ for all s. Indeed, if we have an injective function into a set of size $2^{n'_s}$, then

$$\frac{|B_s|}{2^{n_s}} \leq \frac{2^{n'_s}}{2^{n_s}} = 2^{n'_s - n_s} = 2^{m-3s}.$$

Observation 3. *If we are fixing a clause c, then we learn all k bits of c. However, the number of possibilities for the next clause c' to fix is $\alpha \leq 2^{k-c}$ with $k - c$ bits.*

Recall that we want an advantage of $3s - m$. The **recursion tree** of $\mathsf{MT}(b)$ is the ordered tree with root $\mathsf{MT}(b)$ and a node $\mathsf{Fix}(C)$ for each call of Fix, and edges such that for two nodes u, v, we have that v is a child of u if u calls v. Let R_s be the set of all recursions trees of $\mathsf{MT}(b)$ for $b \in B_s$.

Lemma 4. *Given a recursion tree of $\mathsf{MT}(b)$ and the values of $x_1, \ldots, x_n$ at the end of step s, we can recover the input bit string b. In other words, there exists an injective function $f_s : B_s \to R_s \times \{0,1\}^n$.*

Proof. By example in class. □

Lemma 5. *A tree $T \in R_s$ can be represented with only $m + s(k - c + O(1)$ bits. In other words, there exists an injective function $g_s : R_s \to \{0,1\}^{m+s(k-c+O(1))}$.*

Proof. For each clause c_i, let $N[c_i]$ denote the closed neighborhood of c_i in the overlap graph, namely the clauses adjacent to c_i along with c_i itself. For all clauses $c_j \in N[c_i]$, define the index of c_j with respect to c_i to be l if j has the lth smallest index among $N[c_i]$. Now to describe $T \in R_s$:

1. for all i, use 1 bit to indicate whether $\mathsf{Fix}(c_i)$ is a child of $\mathsf{MT}(b)$.
2. for each $\mathsf{Fix}(c_i)$:
   (a) the index of each child of c_i with respect to c
   (b) $O(1)$ space for overhead (e.g. parenthesis, commas, etc. to specify grouping) □

Now back to the original problem. As a summary, f_s is an injection from the set of "bad" strings $B_s \subset \{0,1\}^{n_s}$ into the set of recursion trees and final values $R_s \times \{0,1\}^n$. Moreover, g_s is an injection from R_s into $\{0,1\}^{m+s(k-c+O(1))}$. Hence the composition $\mathcal{C}_s := g_s \circ f_s$ has advantage

$$n_s - n'_s = (n + sk) - (m + s(k - C + O(1) + n) = -m + s(C - O(1)) \geq 3s - m$$

for sufficiently large C, which proves the theorem.