CS 229r Information Theory in Computer Science March 29, 2016

Lecture 17

Lecturer: Themis and Ali *Scribe: Guannan Qu*

1 Introduction

In this lecture we will cover Lovasz Local Lemma. We will give its definition and application, and give a constructive proof (which uses information theory) for it. Lovasz Local Lemma was first discovered in [2] but the proof is not constructive. The first constructive proof was given in [1], and the version we are going to cover in this lecture is derived from [3].

In many combinatorial problems, the objective is to prove the existence of some combinatorial object, and this object can be specified as an avoidance of bad events. An usual approach is to use a probabilistic argument, i.e. proving the object (understood as event) occurs with non-zero probability. Examples include graph coloring, where the bad events are two neighboring nodes being monochromatic; and k-SAT, where the bad events are that the clauses are not satisfied.

Formally, in a probability space, let the bad events be $A_1, A_2, \dots A_n$, and we would like to prove

$$\mathbb{P}(\cap_i \bar{A}_i) > 0$$

One naive way to do this is using union bound

$$\mathbb{P}(\cup A_i) \leq \sum_i \mathbb{P}(A_i)$$

And we need to show the right hand side is strictly less than 1. However, this bound can be very loose, especially when the number of events are large, or the probability of a certain event is large, in which case the right hand side of the above can be larger than 1. Another approach is that, when the A_i's are mutually independent, we have

$$\mathbb{P}(\cap_i \bar{A}_i) = \Pi_i(1 - \mathbb{P}(A_i))$$

so we only need to show $\mathbb{P}(A_i) < 1, \forall i$. But usually, the A_i's are not mutually independent and the above argument does not work. What Lovasz Local Lemma does is that it relaxes the mutually independence requirement to some 'limited dependence requirement', but still get a similar result as the mutually independent case. To formally state the Lovasz Local Lemma we need the following definitions.

Definition 1 *Given events $A_1, \dots, A_n$ and a undirected graph $G = (V, E)$ where the vertices V are the set of events, we say G is the dependece graph for the events if $\forall i$, A_i is independent from events $\{A_j : j \neq i$ and A_j is NOT a neighbor of $A_i\}$. We also define $\Gamma(A_i)$ to be the the set of neighbors of A_i in G, and $\Gamma^+(A_i) = \{A_i\} \cup \Gamma(A_i)$*

We now formally state the Lovasz Local Lemma.

Theorem 2 (General Lovasz Local Lemma) *Supppose $A_1, \dots, A_n$ are bad events with dependence graph D. If $\exists x_1, \dots, x_n \in (0, 1)$ such that $\forall i$,*

$$\mathbb{P}(A_i) \leq x_i \Pi_{j \in \Gamma(A_i)}(1 - x_j)$$

then $\mathbb{P}(\cap_i \bar{A}_i) \geq \Pi_{j=1}^n (1 - x_j) > 0$

There is also a symmetric version of Lovasz Local Lemma.

Theorem 3 (Symmetric Lovasz Local Lemma) *Given events $A_1, \dots, A_n$, and each A_i is independent from all but at most d other events (in other words, the events have a dependence graph with maximum degree upper bounded by d). If $\exists p > 0$ s.t. $\forall i, \mathbb{P}(A_i) \leq p$ and $p(d+1) \leq \frac{1}{e}$, then $\mathbb{P}(\cap_i \bar{A}_i) > 0$*

Remark The Symmetric Lovasz Local Lemma is an easy corollary of the General Lovasz Local Lemma. Think of $x_i = \frac{1}{d+1} \in (0,1)$, then $\forall i$,

$$\mathbb{P}(A_i) \leq p \leq \frac{1}{d+1}\frac{1}{e} < \frac{1}{d+1}(1-\frac{1}{d+1})^d \leq x_i \Pi_{j\in\Gamma(A_i)}(1-x_j)$$

where we have used $|\Gamma(A_i)| \leq d$. Then by the general Lovasz Local Lemma, we have $\mathbb{P}(\cap_i \bar{A}_i) \geq (\frac{d}{d+1})^n > 0$

2 Applications

We now give two applications of Lovasz Local Lemma.

Hypergraph Coloring. Given hypergraph $H = (V, E)$, where each edge $e \in E$ is a subset of vertices. We want to assign colors to each vertex, i.e. find a map $c : V \to \{red, blue\}$, such that no edge is monochromatic. In general deciding if there exists a coloring of a hypergraph is NP-hard. However if the hypergraph satisfies certain conditions, we can guarantee a coloring exists. In details, we assume in the hypergraph, each edge has at least k vertices, and no edge intersects more than d other edges. Then we can show if $e(d+1) \leq 2^{k-1}$, there exists a coloring. To see this, we randomly color all the nodes uniformly. Define event A_i to be the event edge i is monochromatic. Then $\mathbb{P}(A_i) \leq p = \frac{1}{2^k} + \frac{1}{2^k} = \frac{1}{2^{k-1}}$, and each A_i is independent from all but at most d other events. Since $ep(d+1) \leq 1$, we can apply the Symmetric Lovasz Local Lemma and conclude $\cap \bar{A}_i$ happens with positive probability.

k-SAT. Suppose there are n boolean variables, $x_1, \ldots, x_n$, and m clauses $C_1, \ldots, C_m$, where each clause is a disjunction of k literals (a literal is a x_i or its negation $\bar{x}_i$). For example, if $k = 3$, a clause could be $C_i = x_1 \vee x_3 \vee \bar{x}_4$. The objective of k-SAT is to decide whether there exists an assignment of $x_1, \ldots, x_n$ such that all the clauses are satisfied. This is in general a NP-hard problem (when $k \geq 3$) but we can use Lovasz Local Lemma to show that when the clauses satisfy certain conditions, a satisfying assignment exists. Given a k-SAT instance, the condition is that no variables appear in more than $2^{k-2}/k$ clauses. To see this, we randomly assign values to $x_1, \ldots, x_n$. Let A_i be the event that clause C_i is not satisfied. Notice each C_i have k variables, we have $\mathbb{P}(A_i) \leq p = \frac{1}{2^k}$. Since each variable appears in no more than $2^{k-2}/k$ clauses, A_i is independent from all but at most $d = 2^{k-2}$ other events. We can check $p(d+1) < \frac{1}{e}$, therefore we can avoid all events with positive probability. Hence a satisfying assignment exists.

3 A Constructive Proof of Lovasz Local Lemma

In this section, we will give an efficient randomized algorithm that, when the conditions in Lovasz Local Lemma hold, can output a sample in the probability space that avoids all the bad events A_i in polynomial time. We will state and analyze the algorithm in the context of the k-SAT problem.

Theorem 4 *There exists an algorithm, such that given a k-SAT instance, that is, a set of n variables $x_1, \ldots, x_n$ and m clauses $\mathcal{C} = \{C_1, \ldots, C_m\}$ where each clause depends on exactly k variables, when each variable x_i appears in less than $\frac{2^{k-3}}{k}$ clauses,[1] the algorithm will output a satisfying assignment for this k-SAT instance in poly(n, m) time.*[2]

To introduce the algorithm, we define a undirected graph $G = (V, E)$ whose nodes are the clauses $\mathcal{C}$, and two clauses C_i and C_j are connected if and only if they share a common variable. We define $vbl(C_i)$ to be the set of variables in C_i. We define $\Gamma(C_i)$ to be the set of neighbors of C_i in G, and we define $\Gamma^+(C_i) = \Gamma(C_i) \cup \{C_i\}$. We also define d to be the maximum degree of the graph, and we have $d < 2^{k-3}$. The algorithm is given in Algorithm 1.

As we can see, the algorithm contains a main function 'Solve' and a recursive function 'Fix'. We now analyze the algorithm. We have the following two claims.

[1] The optimal quantity here should be $\frac{1}{e}\frac{2^k}{k}$. To simplify the proof we prove a suboptimal quantiy $\frac{2^{k-3}}{k}$.

[2] Here we need to assume the following two operations can be conducted efficiently. First, we can efficiently sample from $\{0,1\}^n$ uniformly. Second, given an assignment σ and a Clause C_i, we can efficiently check if C_i is violated by σ

Algorithm 1 Lovasz Local Lemma Search (k-SAT)

1: **function** SOLVE($x, \mathcal{C}$) ▷ x is the set of variables, $\mathcal{C}$ is the set of clauses
2: $\sigma \leftarrow$ a uniformly random assignment in $\{0,1\}^n$
3: **while** $\exists C_i$ such that C_i is violated by σ **do**
4: $\sigma \leftarrow \text{Fix}(\mathcal{C}, \sigma, C_i)$
5: **return** σ
6: **function** FIX($\mathcal{C}, \sigma, C_i$)
7: $\sigma' \leftarrow$ resample the variables in $vbl(C_i)$, while other variables same as σ
8: **while** $\exists C_j \in \Gamma^+(C_i)$ such that C_j is violated by σ' **do**
9: $\sigma' \leftarrow \text{Fix}(\mathcal{C}, \sigma', C_j)$
10: **return** σ'

Claim 5 *Assuming 'Fix' can terminate, 'Solve' calls 'Fix' for at most m times and outputs a satisfying assignment.*

Proof Let D_i be the set of clauses that lie within the same connected component as C_i in G. Assume $\text{Fix}(\mathcal{C}, \sigma, C_i)$ terminates. Then the returned assignment must satisfy all clauses in D_i, while the variables that is not involved in any clause in D_i remain the same as σ. Therefore, after each call of 'Fix' by 'Solve', the number of violated clauses decreases by at least 1. Hence 'Solve' calls 'Fix' for at most m times, and will output a satisfying assignment. ■

Claim 6 *'Fix' terminates after $poly(n, m)$ resampling procedures.*

Proof We can understand the operation of 'Fix' in the following way. Suppose the input assignment to 'Fix' is σ_0, and then 'Fix' will repeatedly calls itself and conduct many resamplings (Line 7 in Algorithm 1). We let the t-th resampled assignment be σ_t, and in the t-th resampling, let the clause being resampled be C_{ℓ_t}, and let the $vbl(C_{\ell_t})$ part of σ_t be r_t (which is a k-bit random string). We define $Log_t = C_{\ell_1} C_{\ell_2} \dots C_{\ell_t}$, which is a record of the clauses being resampled. We also define $R_t = r_1 r_2 \dots r_t$ which is the concatenation of all the random bits used for resampling. R_t has length kt.

We claim that we can recover R_t from Log_t and σ_t. Given σ_t and Log_t, we know that σ_t is constructed from σ_{t-1}, but with the variables in $vbl(C_{\ell_t})$ being replaced by r_t. Hence, r_t are the values of the $vbl(C_{\ell_t})$ part in assignment σ_t. In this way we can recover r_t. Next, notice that σ_{t-1} violates C_{ℓ_t}, therefore, the $vbl(C_{\ell_t})$ part of σ_{t-1} can be uniquely determined, while σ_{t-1}'s other variables are the same as σ_t. Hence we can also recover σ_{t-1}. Then repeat this procedure, we can recover all $r_{t-1}, \dots, r_1$, and hence we can recover R_t.

Then, we can apply the data processing inequality,

$$H(R_t) \le H(Log_t, \sigma_t) \le H(Log_t) + H(\sigma_t)$$

Since σ_t have n bits, $H(\sigma_t) \le n$. Since R_t is a kt-bit uniformly random string, $H(R_t) = kt$. Therefore, we have

$$H(Log_t) \ge kt - n$$

The next step is to upper bound $H(Log_t)$. We provide the following encoding of Log_t. First, we encode C_{ℓ_1} using $\log m$ bits. Then, notice $C_{\ell_1} \dots C_{\ell_t}$ can be understood as the recursive tree τ of the algorithm. τ is rooted at C_{ℓ_1}, and C_{ℓ_i} is a parent of C_{ℓ_j} if and only if the run of 'Fix' on C_{ℓ_i} calls 'Fix' on C_{ℓ_j}. By the nature of the algorithm, if C_{ℓ_j} is a child of C_{ℓ_i}, then $C_{\ell_j} \in \Gamma^+(C_{\ell_i})$. Using this special structure we can embed τ in a infinite tree T, where T is rooted at C_{ℓ_1}, and is constructed in the following way: each node C_ℓ has exactly $d+1$ children, each representing a element in $\Gamma^+(C_\ell)$.[3] Then τ corresponds to a subtree of T rooted at the root of T with size t. By some counting arguments, T has no more than $(4(d+1))^t$ such

[3]It might be that $|\Gamma(C_\ell)| < d$, but it doesn't matter since we can create some 'fake' neighbors of C_ℓ.

subtrees, and hence τ can be encoded using $t \log 4(d+1) < t(k-1)$ bits, and hence Log_t can be encoded using $\log m + t(k-1)$ bits. Therefore,

$$\log m + t(k-1) \geq H(Log_t) \geq kt - n$$

This gives $t \leq n + \log m$, i.e. the 'Fix' must terminate after no more than $n + \log m$ resamplings.
■

It is easily seen that the two claims lead to Theorem 4, and we can conclude the algorithm terminates after no more than $m(n + \log m)$ resampling procedures.

References

[1] József Beck. An algorithmic approach to the lovász local lemma. i. *Random Structures & Algorithms*, 2(4):343–365, 1991.

[2] Paul Erdos and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and finite sets*, 10(2):609–627, 1975.

[3] Robin A Moser and Gábor Tardos. A constructive proof of the general lovász local lemma. *Journal of the ACM (JACM)*, 57(2):11, 2010.