COMS E6998-9: Algorithms for Massive Data (Fall'25) Nov 12, 2025

Lecture 20: Approx Min VC cont'd & Distribution Testing

Instructor: *Alex Andoni* Scribes: *Derek Che*

1 Estimate size of minimum vertex cover (VC) continued

1.1 Review

We proved in the previous lecture that if we set $S = M$ for any maximal matching M[1] of target graph G, then S is a VC and $|VC^*| \leq |S| \leq 2|VC^*|$. Therefore, we can solve the 2-approx VC problem by finding size of maximal matching. We also proved that a matching in G is equivalent to an independent set in G' (G's line graph), so finding the size of maximal matching in G is the same as finding the size of maximal independent set in G'.

1.2 Estimate size of maximal *IS* in G with max degree d

Theorem 1 (Nguyen, Onak' 2008)**.** *We can build a local oracle that has access to G and takes R (randomness / random seed). When queried whether $v \in I$, the oracle outputs yes / no. $\forall R$, $\exists I = I(R, G)$ the maximal independent set in G associated with R. $\forall v \in V$, L.O.(v) has expected runtime $O(e^d)$ and returns if $v \in I$. Using this L.O., we can get $\pm\epsilon \cdot n$ additive approximation of the problem in $O(\frac{1}{\epsilon^2})$ time.*

An idealized algorithm for maximal *IS* would be as follows:

Algorithm 1 Idealized L.O.

```
for each vertex v_i with i ∈ [n] in some order do
    if v_i has no neighbor in I then
        add v_i to I
    else
        skip v_i
    end if
end for
```

But this algorithm is dependent on the order, and we want to break the long chain of dependence. Therefore, our L.O. is as follows:

[1]Maximal matching is simply a matching that cannot add any more edges. It is different from maximum matching, which is the global largest matching of a graph.

Algorithm 2 L.O.

```
order := random order
assign v → r_v, where r_v ∼ Unif([0, 1])
Input: v, access to G (the original graph if G is the line graph)
Output: v ∈ I? I = IS obtained greedily by order r_v
for w ∈ N_G(v) do
    if r_w < r_v then
        check recursively if w ∈ I
        if w ∈ I then, return NO
        end if
    end if
end for
return YES
```

Correctness: Correctness simply follows from the fact that I is the maximal IS using order r_v. The algorithm checks nodes with smaller r_v first. If a node v has a neighbor n with $r_n < r_v$ and n is in I, to maintain the independent set property, v must not be in I.

Runtime: Firstly, note that the algorithm must terminate because each recursive call starts with a node with lower r_v, so there are at most n recursive calls. Then, we can bound the expected runtime as stated in Theorem 1.

Claim 2. $\mathbb{E}[\#$ *of vertices visited*$] \leq \frac{e^d}{d}$.

Proof. Consider a path that the recursive L.O. takes $(v \to w_1 \to w_2 \to \cdots \to w_k)^2$. The necessary condition for this path to be possible is $r_v > r_{w_1} > \cdots > r_{w_k}$. Since all r are i.i.d from Unif([0, 1]), all orders are equally likely, so this particular order occurs with probability $\frac{1}{(1+k)!}$. Therefore, $\Pr[\text{follows } v \to w_1 \to \cdots \to w_k] \leq \frac{1}{(k+1)!}$. Intuitively, it is a $\leq$ because even if these $k+1$ r.v. have the order, there could be another node not in this path with r sandwiched between some r_{w_i} and $r_{w_{i-1}}$, and the path will not take place.

$$\begin{aligned}
\mathbb{E}[\#\text{ of visited vertices}] &\leq \sum_{k=1}^{\infty} [\#\text{ of paths of length } k] \cdot \Pr[\text{follow a path of length } k] \\
&\leq \sum_{k=1}^{\infty} d^k \cdot \frac{1}{(k+1)!} = \frac{1}{d} \sum_{k=0}^{\infty} \frac{d^k}{k!} = \frac{1}{d} e^d
\end{aligned}$$

□

Corollary 3. $\mathbb{E}[\#$*queries*$]$ *to solve the approx-VC∗ problem in G using this approx-IS L.O. in G' with only access to G is* $O(\frac{e^d}{d} d) = O(e^d)$.

Proof. This follows immediately from the fact introduced from the previous lecture that query access to G' can be easily derived from $O(d)$ queries to G. □

[2]The graph is undirected. The arrows are only to indicate the direction of recursion.

Corollary 4. *We can estimate size of I up to $\pm\epsilon \cdot n$ with probability $\geq 1-\delta$ using $O(\frac{1}{\epsilon^2})$ L.O. queries (total time $O(\frac{e^d}{\epsilon^2})$). Therefore, it can efficiently solve 2-approx VC problem with up to $\pm 2 \cdot \epsilon \cdot n$.*

Proof. Fix R of the local oracle, so it defines a single maximal I. We want an estimator $\widehat{|I|}$ such that

$$\Pr\left(|\widehat{|I|} - |I|| \leq \epsilon n\right) \geq 1-\delta.$$

Sample k vertices $v_1, \ldots, v_k$ independently and uniformly at random from V. For each i, query the local oracle and collect $X_i = \begin{cases} 1 & \text{if } v_i \in I, \\ 0 & \text{otherwise.} \end{cases}$ Then $X_i \in [0,1]$ and $\mathbb{E}[X_i] = \frac{|I|}{n} := \mu$. Let $\hat{\mu} := \frac{1}{k}\sum_{i=1}^k X_i$ and $\widehat{|I|} := n\hat{\mu}$. By Hoeffding's inequality for i.i.d. $[0,1]$-valued variables,

$$\Pr\left(|\hat{\mu} - \mu| > \epsilon\right) \leq 2e^{-2k\varepsilon^2}$$

To make this at most δ, it suffices to choose

$$k \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta} = O\left(\frac{1}{\varepsilon^2}\right)$$

Consequently, $\pm 2\epsilon n$ for min VC follows as $|VC^*| \leq |S| = 2|M| \leq 2|VC^*|$. □

1.3 More Results

[Yoshida, Yamamoto, Ito' 2009] improved the algorithm by a simple heuristic:

Algorithm 3 L.O. with heuristic

```
for w ∈ N_G(v) in increasing order of r_w do
    if r_w < r_v then
        check recursively if w ∈ I
        if w ∈ I then, return NO
        end if
    end if
end for
return YES
```

This L.O. has the result that

$$\mathbb{E}_{R, v\in V}[\# \text{ of recursive calls}] \leq 1 + \frac{m}{n},$$

where $\frac{m}{n} \leq d$.

2 Start on distribution testing

2.1 Overview of problems

The objective is some discrete distribution D over $[n]$. The goal is to test properties (e.g. is D uniform?) or quantities (e.g. mean of D?) of the distribution. The full input is D, which is a vector in $[0,1]^n$. But query access is in the form of samples $(X_1, \ldots, X_m \sim D)$.

2.2 Testing distribution for property

Let $\underline{\text{P}}$ be a property. We want to be able to

1. return YES, if $D \in \underline{\text{P}}$
2. return No, if D is ϵ-far from $\underline{\text{P}}$

2.2.1 Testing if D is uniform U_n

Definition 5. *The total variation distance between two distributions p, q over $[n]$ is defined as*

$$\|p - q\|_{TV} = \max_{T \subset [n]} |\Pr_{i \sim p}[i \in T] - \Pr_{i \sim q}[i \in T]|$$

Fact 6. $2\|p - q\|_{TV} = \|p - q\|_1$

Proof. Let $T = \{i : q_i > p_i\}$ and $\bar{T} = \{i : q_i \leq p_i\}$. Note that $\|p - q\|_1 = \sum_{i \in [n]} |p_i - q_i| = \sum_{i \in T}(q_i - p_i) + \sum_{i \in \bar{T}}(p_i - q_i)$ and $\sum_{i \in T}(q_i - p_i) - \sum_{i \in \bar{T}}(p_i - q_i) = \sum_{i \in [n]} q_i - \sum_{i \in [n]} p_i = 0$. This implies that $\sum_{i \in T}(q_i - p_i) = \sum_{i \in \bar{T}}(p_i - q_i)$. Also observe that $\arg\max_{T \subset [n]} |\Pr_{i \sim p}[i \in T] - \Pr_{i \sim q}[i \in T]| \in \{T, \bar{T}\}$ because the absolute value will be maximized iff we only take positive or negative terms. So $\|p - q\|_{TV} = \sum_{i \in T}(q_i - p_i) = \sum_{i \in \bar{T}}(p_i - q_i) = \frac{1}{2}\|p - q\|_1$. □

Claim 7. *Given D, we can test wether $D = U_n$ or ϵ-far from U_n (i.e. $\|D - U_n\|_1 \geq \epsilon$). Using $m = \Theta(\frac{\sqrt{n}}{\epsilon^2})$ is sufficient.*

Proof will be discussed in the next lecture.