

# CS 6301.008.18S Lecture—April 5, 2018

Main topics are `#eps-nets` and `#set_cover`.

## Samples and Nets Review

- Let's start with a review of eps-samples and nets.
- For simplicity, say we have a range space  $(P, R)$  where  $P$  is finite.
- Given  $Q$  in  $R$ ,  $Q$ 's *measure* is the fraction of  $P$  that it contains. We denote it as  $\mu(Q) = |Q \cap P| / |P|$ .
- Given a *sample*  $S$  subset  $P$ , the *estimate* of  $Q$  is  $\hat{\mu}(Q) = |Q \cap S| / |S|$ .
- Given  $\epsilon > 0$ , we'll say  $S$  is an eps-sample if for *any* range  $Q$  in  $R$ , we have  $|\mu(Q) - \hat{\mu}(Q)| \leq \epsilon$ .
- Given  $\epsilon > 0$ , we'll say  $S$  is an eps-net if for any range  $Q$  in  $R$  with  $\mu(Q) \geq \epsilon$ ,  $Q$  contains at least one point of  $S$ . In other words, the net catches every large range  $Q$ .
- Theorem: Let  $(X, R)$  be a range space of constant VC-dimension, and let  $P$  be any finite subset of  $X$ . A random sample  $S$  subset  $P$  of size  $O((1/\epsilon^2) \log(1/\epsilon))$  is an eps-sample with constant probability.
- Theorem: A random sample  $S$  subset  $P$  of size  $O((1/\epsilon) \log(1/\epsilon))$  is an eps-net with constant probability.
- eps-samples have more obvious and direct use cases. Say you want to do *approximate range counting*. You want to know approximately how many people live within a certain distance of each major city in Texas. You only need to figure out locations for an eps-sample of the population.
- Applications for eps-nets are less obvious, and that's what I want to discuss today.

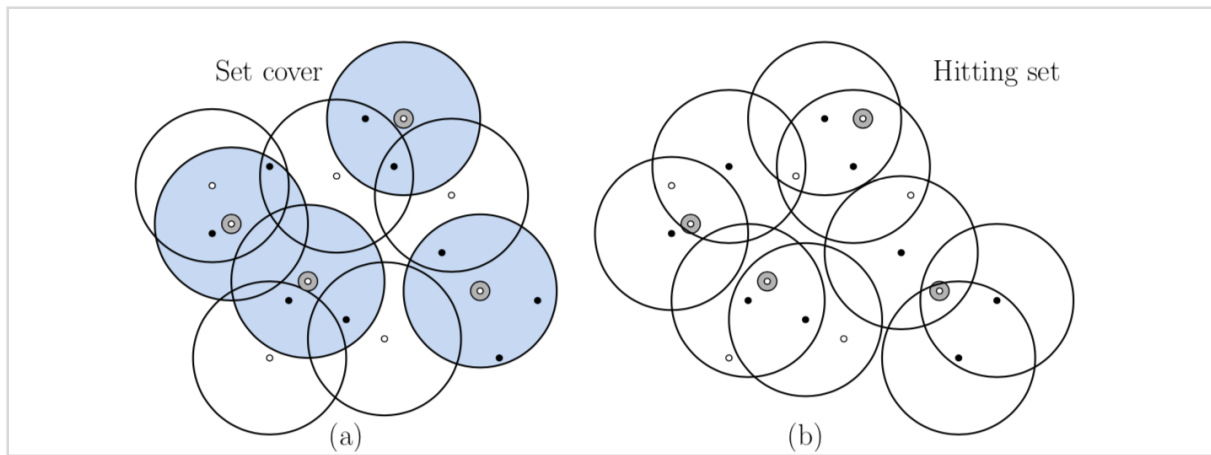
## Sampling from a Disk

- Say we have a very large point set  $X$  in the plane and there's this unknown disk  $D_{\text{unknown}}$ . We can check if a point of  $X$  is in the disk, but that's it. We want to make a guess on what  $D_{\text{unknown}}$  looks like.
- In fact, let's settle for being wrong on only an eps fraction of the points.
- Pick an  $\epsilon > 0$  and sample  $O(1/\epsilon \log(1/\epsilon))$  points at random from  $X$ . Test all the sample points, and find a disk enclosing only the samples that passed your test. Let that disk be  $D$ .
- Disks have finite VC-dimension, and you can argue that the symmetric difference of disks does as well.
- Our sample is an eps-net for  $X$  wrt the symmetric difference of disks with constant probability.

- If more than an  $\epsilon$ -fraction of  $X$  lied in exactly one of  $D$  or  $D_{\text{unknown}}$ , then we would have sampled a point in that symmetric difference, contradicting  $D$  enclosing exactly the samples that passed the test.

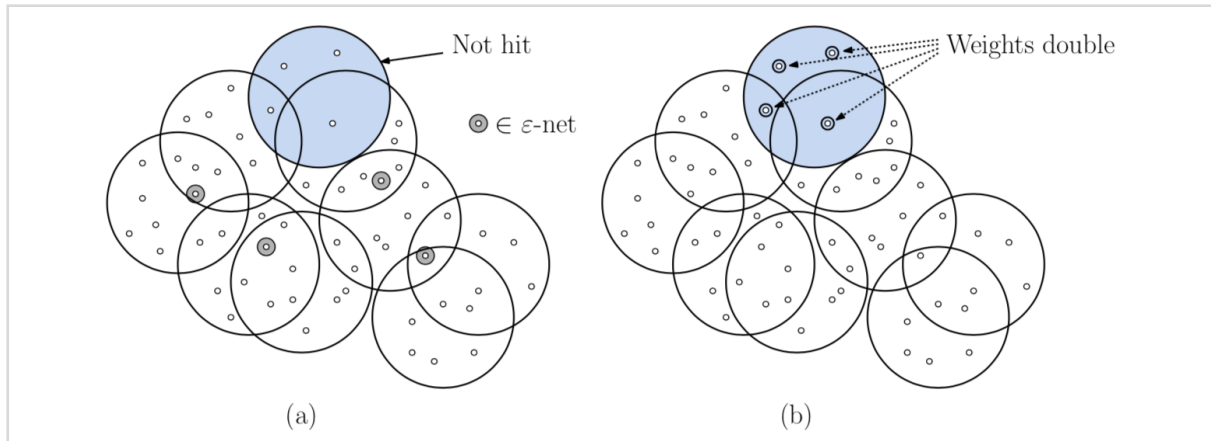
## Approximation Algorithm for Set Cover

- The second application will be an approximation algorithm for the set cover problem.
- In the set cover problem, you are given an  $n$ -element ground set  $X$  and a collection of subsets  $R$  over  $X$ .  $(X, R)$  is really just a range set.
- The goal is to find a minimum size collection of subsets from  $R$  whose union is  $X$ .
- The problem is NP-hard, but there's a pretty simple greedy algorithm that does well: iteratively include a single set that includes as many uncovered elements as possible.
- This algorithm is an  $O(\log n)$ -approximation, meaning you'll find a set cover with at most  $O(\log n)$  times as many subsets as is optimal.
- In general, you cannot do better! No polynomial time algorithm can provide better than an  $\Omega(\log n)$ -approximation for every instance of set cover unless  $P = NP$ .
- However, if  $(X, R)$  has constant VC-dimension, then you can get an approximation ratio of  $O(\log \text{OPT})$  where  $\text{OPT}$  is the size of the smallest set cover. That could be a lot less if  $\text{OPT} \ll n$ !
- I'll focus on the simpler case we saw last time of trying to place routers to cover a set of sites.
- We have an  $m$ -element point set  $P$  in  $\mathbb{R}^2$  (we'll see why I wrote  $m$  shortly) and an  $n$ -element point set  $T$  which represents possible locations of transmission towers.
- We'll make things even easier and focus on the decision version of set cover: given a value  $k$ , can we pick  $k$  points from  $T$  such that every point from  $P$  is in the disk of radius 1 centered at  $T$ ?
- We'll make an algorithm that finds a set of size  $O(k \log k)$  if such a set exists. It may report failure otherwise.
- To make things a bit easier, we'll actually look at a "dual" problem called *hitting set*. A point  $p$  in  $P$  lies in a disk of radius 1 centered at  $t$  in  $T$  if and only if  $t$  lies in a disk of radius 1 centered at  $p$ .
- So, we'll try to find a set of  $O(k \log k)$  points from  $T$  (which has size  $n$ ) that hit all  $m$  disks centered at points of  $P$ .



## Iterative Reweighting

- So, here's the algorithm for hitting set: again, we have a set  $T$  of  $n$  points and a set  $P$  of  $m$  points. We want to find a subset of  $T$  that hits all the unit disks centered at points of  $P$ .
- Given  $k$ , our algorithm will find a hitting set of size  $O(k \log k)$  if a hitting set of size  $k$  exists.
- We assign each point from  $T$  a weight initially set to 1.
- We're going to compute eps-nets on  $T$  where the range space is the set of unit disks. To define measures, we'll count each point  $t$  in  $T$  a number of times equal to its weight. So an eps-net has at least one point in each disk that contains at least an eps fraction of the *total weight*.
- Here are the details:
  1. Let  $\text{eps} = 1 / (4k)$ . Let  $k' = c k \log k$  for a suitably large constant  $c$ .
  2. Compute the weighted eps-net  $N$  of  $T$  of size  $k'$ .
    - To find it, just randomly sample  $k'$  points where a point is sampled with probability proportional to its weight. Keep trying until you finally do get an eps net.
  3. Check if there is a disk that it not hit by  $N$ . If so, double the weight of each point of  $T$  in that one disk and return to step 2.
    - $\lg$  means  $\log_2$
    - If you've done more than  $2k \lg(n/k)$  iterations, output failure; there is no hitting set of size  $k$ .
  4. If you reach this step,  $N$  must be a hitting set of size  $k' = O(k \log k)$ . Output  $N$ .



- So the algorithm is polynomial time (with high probability). You'll almost certainly find an  $\epsilon$ -sample within  $\log n$  attempts in each iteration, and there are a polynomial number of iterations.
- Now we need to argue that the algorithm will find a hitting set within  $2k \lg(n/k)$  iterations if one of size  $k$  exists.
- Here's the high level idea behind the argument before we get into the math.
- If  $N$  is not a hitting set, then we find that one disk that is not hit by  $N$ . But that set has an  $\epsilon$  fraction of the total weight. Doubling the weight of that set increases the total weight by a  $(1 + \epsilon)$  factor at most.
- On the other hand, the optimal hitting set does hit that disk we missed. So we'll double that weight of at least one of  $k$  points in the optimal hitting set.
- The weight of the hitting set eventually starts increasing at a faster rate than the weight of the whole point set. The hitting set is part of the point set, though, so we can only repeat the procedure  $O(k \log n)$  times.
- Now let's formalize the idea.
- Assume there is some optimal hitting set  $H$  of size  $k$ .
- Let  $W_i$  denote the total weight of all points in  $T$  after the  $i$ th iteration.
- $W_0 = n$ , because all points start with weight 1.
- If iteration  $i$  does not return a hitting set, then there is some disk not hit by the  $\epsilon$ -net  $N$ , meaning it has total weight at most  $\epsilon W_{i-1}$ . We double the weight of points in this disk, so
  - $W_i \leq W_{i-1} + \epsilon W_{i-1} = (1 + \epsilon)W_{i-1}$ .
- Because  $W_0 = n$ ,  $W_i \leq (1 + \epsilon)^i n \leq n * e^{\epsilon i}$  (for any  $x$ ,  $1 + x \leq e^x$ ).
- Optimal hitting set  $H$  hits every disk including that disk we missed, so at least one of its  $k$  points had its weight doubled.
- For  $1 \leq j \leq k$ , let  $t_i(j)$  denote the total number of times the  $j$ th optimal point has had its weight doubled by the end of iteration  $i$ .
- Let  $W_i(H)$  be the total weight of  $H$  after the  $i$ th iteration. We have
  - $W_i(H) = \sum_{j=1}^k 2^{t_i(j)}$ .
- At least one of these  $k$  points has its weight doubled each iteration, so  $\sum_{j=1}^k t_i(j) \geq$

i.

- Because  $2^x$  is convex,  $W_i(H)$  is minimized when those  $k$   $t_i(j)$ 's are as equal as possible subject to that bound on their sum. In other words,
  - $W_i(H) \geq \sum_{j=1}^k 2^{i/k} = k 2^{i/k}$ .
- Finally,  $W_i(H) \leq W_i$ , because  $H \subset T$ .
- So,  $k 2^{i/k} \leq n e^{\epsilon i}$ .
- Taking the  $\lg$  of both sides, we have  $\lg k + i/k$ 
  - $\leq \lg n + \epsilon i \lg e$
  - $= \lg n + (i/4k) \lg e$
  - $\leq \lg n + i/2k$  (because  $\lg e \approx 1.45 < 2$ ).
- A bit of algebra shows  $i/2k \leq \lg n - \lg k = \lg(n/k)$ , so  $i \leq 2k \lg(n/k)$ .
- Meaning we cannot have more than  $2k \lg(n/k)$  iterations if there is a hitting set  $H$  of size  $k$ .