VC-Dimension and Set Cover

1 Network Failure Detection

Recall that we call a set $S \subseteq V$ separable by k edges iff there exists a set $F \subseteq E$ with $|F| \leq k$ of edges such that S and $\bar{S}$ are not connected in $(V, E \setminus F)$. In other words, the adversary could disconnect S from $\bar{S}$ by deleting at most k edges. Let $\mathcal{S}_k$ be the set system of all $S \subseteq V$ that are separable by k edges. We still need to show the following lemma:

Lemma 1. *$(V, \mathcal{S}_k)$ has VC-dimension at most $2k+1$.*

Proof. Because we are getting an upper bound, we need to show that no set $A \subseteq V$ with $|A| \geq 2k+2$ is shattered by $(V, \mathcal{S}_k)$. That is, for every such A, there exists $A' \subseteq A$ such that $A' \neq A \cap S$ for all $S \in \mathcal{S}_k$ (*i.e.,* there exists an $A' \subseteq A$ such that $|e_G(A, A \setminus A')| \geq k+1$).

Lemma 2. *Let G be an undirected, connected graph and $A \subseteq V$ a node set of size $|A| = 2k$. Then we can write $A = \{v_1, v_2, \ldots, v_{2k-1}, v_{2k}\}$ such that for each i, there is a path P_i in G from v_{2i-1} to v_{2i}, and the P_i are edge-disjoint.*

By Lemma 2, we can pair up the $2k+2$ nodes in A and get edge-disjoint paths between the pairs. Define A' to be exactly one node from each pair (*e.g.,* all the even-indexed nodes). To separate each pair, we must cut at least one edge from each path P_i. Because the P_i are edge-disjoint, we need to cut at least $k+1$ edges. □

We now prove Lemma 2:

Proof. Because more edges help us, we prove the hard case of a spanning tree of G. Then, we repeatedly delete all leaves that are not in A. If there is a pair $u, v \in A$ such that the unique $u-v$ path does not contain any degree 3 (or higher) node, pair up u, v and remove them. Otherwise, pair up leaves u, v such that the unique $u-v$ path has only one node of degree 3 (or higher) and delete them. We can show that such a pair always exists, and this produces edge-disjoint paths. □

2 Set Cover/Hitting Set

2.1 Definitions

In the set cover problem, we are given a ground set X and a collection $\mathcal{C}$ of subsets $S_i \subseteq X$. The goal is to find a (small) subcollection $\mathcal{D} \subseteq \mathcal{C}$ such that

$$\bigcup_{S \in \mathcal{D}} S = X.$$

In the hitting set problem, we are given a ground set X and a collection $\mathcal{C}$ of subsets $S_i \subseteq X$. The goal is to find a (small) set $H \subseteq X$ such that

$$H \cap S \neq \emptyset \text{ for all } S \in \mathcal{C}.$$

A reduction from set cover to hitting set is to define $X' := \mathcal{C}$, $T_i := \{S \in \mathcal{C} : i \in S\}$, and $\mathcal{C}' := \{T_i : i \in X\}$. Then x' hits a set $T_i \in \mathcal{C}'$ iff $x' \in T_i$ iff $i \in S$ iff S covers i. So hitting all of $\mathcal{C}'$ is equivalent to covering all of X.

For set cover, the greedy algorithm is an $\mathcal{O}(\log n)$ approximation and this is optimal unless $\boldsymbol{P} = \boldsymbol{NP}$. But maybe we get a better approximation for instances that are "less complex". In particular, if $(X, \mathcal{C})$ has low VC-dimension.

Here, we will focus on constant VC-dimension (independent of n) and get an algorithm that finds a solution of cost at most $\mathcal{O}(OPT \cdot \log OPT)$.

2.2 Set Cover is still NP-hard

Could set cover be polytime solvable for constant VC-dimension? We will show that even for VC-dimension 2, set cover is still $\boldsymbol{NP}$-hard because vertex cover instances also have VC-dimension 2.

In the vertex cover problem, we are given an undirected graph $G = (V, E)$. The goal is to find a (small) set $S \subseteq V$ such that each $e \in E$ has at least one endpoint in S. To prove VC-dimension, let $X = E$ and T_v be the set of edges incident of v. Then our range space is $R = \{T_v : v \in V\}$. To show that this set system has VC-dimension at most 2, we need to show that no 3-edge set $E' := \{e_1, e_2, e_3\}$ is shattered by R. We need to find $E'' \subseteq E'$ such that for all v, $E'' \neq E' \cap T_v$.

If the edges in E' don't all have a vertex in common, then there is no v with $E' = E' \cap T_v$. In other words, we cannot obtain all of E' as an intersection. If all edges in E' share a vertex v, then any set E'' with $|E''| = 2$ cannot be obtained, as $E'' = E' \cap T_u$ because the other endpoint of the edges in E'' (besides v) is different.

Lemma 3. *If a set cover instance $(X, \mathcal{C})$ has VC-dimension d, the corresponding hitting set instance $(X', \mathcal{C}')$ has VC-dimension at most $2^{d+1} - 1$. If d is a constant, then this is still a constant.*

Proof. Later. □

2.3 Generalized ε-nets

Recall that an ε-net is a set N that hits all "large" sets (*i.e.,* those with probability mass at least ε). And, a hitting set is a set S that hits all sets. If we adjusted ε or the probabilities so that all sets are large, then an ε-net is exactly a hitting set.

Definition 4. To avoid some renormalization, we will slightly generalize the notion of an ε-net. Consider non-negative weights w on $\Omega = X$. We remove the assumption that $\sum_x w(x) = 1$ (so this is no longer a probability distribution). Then, N is a weighted ε-net *w.r.t.* w if

$$N \cap S \neq \emptyset \ \forall S : w(S) \geq \varepsilon w(X).$$

Notice that if we renormalized and divided everything by $w(X)$, we would obtain the previous definition.

2.4 Initial Approach

To leverage the ε-net/hitting set insight, we assume that we know the size OPT of the smallest hitting set and also the actual hitting set H^*. Now, define the following weights:

$$w^*(x) = \begin{cases} 1 & x \in H^* \\ 0 & o.w. \end{cases}.$$

Also, let $\varepsilon = 1/OPT$. Then, the total weight $w^*(X) = OPT$ and for any set $S \in \mathcal{C}$, we have $w^*(S) = |S \cap H^*|$ and this is at least 1 because H^* is a hitting set. So,

$$w^*(S) \geq 1 = \frac{1}{OPT} \cdot OPT = \varepsilon w^*(X).$$

Therefore any weighted ε-net *w.r.t.* w^* is a hitting set.

So, if we knew w^* we could use the ε-net theorem to sample an ε-net of size

$$\Theta(\frac{d}{\varepsilon} \log \frac{d}{\varepsilon} + \frac{1}{\varepsilon} \log \frac{1}{\delta}) = \Theta(OPT \cdot \log OPT).$$

The sample would succeed with probability of at least $1-\delta$. Therefore we have an $\mathcal{O}(\log OPT)$ approximation.

2.5 Multiplicative Weights Update

Obviously we don't know w^* or OPT. For now, continue to assume we know OPT, and we will try to "learn" w^*.

Algorithm 1 Las Vegas Multiplicative Weights Update

```
1: Set w(x) = 1 for all x ∈ X, and set ε = OPT/2.
2: Find a weighted ε-net H for (X, C, w) using random sampling. We can check that it is an ε-net and
   repeat until we get one.
3: if H is a hitting set then
4:     Return H.
5: else
6:     Let S be an arbitrary set in C not hit by H.
7:     Double the weights of all x ∈ S and repeat.
8: end if
```

Theorem 5. *Algorithm 1 terminates in $\mathcal{O}(n)$ iterations and returns a hitting set of size $\mathcal{O}(OPT \cdot \log OPT)$.*

Proof. If the algorithm terminates, it returns a hitting set of size

$$\Theta(\frac{d}{\varepsilon} \log \frac{d}{\varepsilon} + \frac{1}{\varepsilon} \log \frac{1}{\delta}) = \Theta(OPT \cdot \log OPT).$$

The runtime analysis is similar to other multiplicative weights analyses. We keep track of (1) the total weight on X, and (2) the weight on H^*. We show that (2) grows faster than (1). Thus, if the algorithm runs too long, (2) will exceed (1) which is a contradiction.

(1) The total weight on X starts at $w(X) = |X| = n$. In each iteration, we double the weights on a subset S of X where $S \in \mathcal{C}$. This increases $w(X)$ by $w(S)$. Since H is an ε-net and S was not hit, we know $w(S) < \varepsilon w(X)$. After k iterations, $w(X) \leq n(1+\varepsilon)^k$.

(2) The weight on H^* starts at $w(H^*) = |H^*| = OPT$. In each iteration, $H^* \cap S$ is nonempty because H^* is a hitting set. So at least one element of H^* has its weight doubled. To lower-bound the weight of H^*, notice that by convexity of $x \mapsto 2^x$, the weight is smallest if the lowest weight is always doubled. After k iterations, $w(H^*) \geq 2^{k/OPT} \cdot OPT$.

Clearly, after each iteration, $w(X) \geq w(H^*)$, so

$$(1+\varepsilon)^k \cdot n \geq 2^{k/OPT} \cdot OPT.$$

Taking logs and solving for k we find

$$k \leq 4OPT \log \frac{n}{OPT} = \mathcal{O}(n).$$

□