Special Topics in Algorithms

Shashwat Garg

September 24, 2014

1 Preface

These notes are my own and made for self-use.

2 Some required tools

2.1 Generating random permutations

There are two methods of generating uniform random permutations of n numbers, such that each permutation has $1/n!$ probability of occurring.

2.1.1 Brute force

Choose a random number from the n elements. This will be the first element of the list. For the i^{th} element, choose a random number from 1 to $n-i+1$, let it be j and choose the j^{th} largest element from the remaining elements.

2.1.2 Fisher-Yates / Knuth shuffle

Traverse the set of elements from 1 to $n-1$. Swap the i^{th} element with a randomly chosen element from i to n, inclusive.

2.2 Union bound

For countable number of events $A_1, A_2, ...$

$$P(\cup_i A_i) \leq \sum_i P(A_i) \tag{1}$$

2.3 Chernoff's bound

Lemma 1. *Let $X = X_1 + ... + X_n$ be all random variables such that $P[X_i = 1] = P[X_i = -1] = 1/2$. Then, $P[X \geq \Delta] \leq e^{-\Delta^2/2n}$*

Proof. For any $s > 0$,

$$\begin{aligned} P[X \geq \Delta] &= P[e^{sX} \geq e^{s\Delta}] \\ &\leq E[e^{sX}]/e^{s\Delta} \text{(by Markov's inequality)} \\ &= \frac{(\frac{e^s+e^{-s}}{2})^n}{e^{s\Delta}} \leq \frac{e^{s^2 n/2}}{e^{s\Delta}} \text{(by Taylor's expansion)} \end{aligned}$$

Choosing $s = \Delta/n$ completes the proof. □

3 Randomised incremental construction

The general paradigm of this technique is that, suppose we we are given a set S of n objects and we want a construct a structure $F(S)$. We will randomly choose the objects one at a time, and maintain a partial structure $F(S_i)$ where S_i is the set of first i chosen objects. After all elements have been chosen, we will get the desired structure.

3.1 Quick sort [1]

We can visualise quick sort in the following way, which is equivalent to the standard way. Choose a random pivot and put all elements smaller than it to its left and elements bigger than it on its right. Choose another random pivot and so on. After choosing the i^{th} pivot, we will have partitioned the input elements in to $i+1$ intervals, each of which is unsorted, but sorted with respect to other intervals. Let $p_1, p_2, ...$ be the order of pivots chosen and $P_i = \{p_1, ..., p_i\}$. Upon adding p_{i+1}, suppose it lies in interval I and splits it into I_l and I_r. The number of comparisons needed upon adding p_{i+1} was $|I_l| + |I_r|$. As p_{i+1} was chosen randomly, this is a random value and we want to compute its expectation. To analyse this, we would use **backward analysis**. We will compute its expectation cost conditioned that we already know the set P_{i+1} but not P_i. Then, to go from P_{i+1} to P_i we need to randomly delete one of the $i+1$ points p and it incurs a cost of $\left|I_l^p\right| + |I_r^p|$. Thus, expected cost conditioned on P_{i+1} is $\frac{1}{i+1}\sum_{p \in P_{i+1}}(\left|I_l^p\right| + |I_r^p|) \leq \frac{2}{i+1}\sum_{allintervals}|I| = O(\frac{n}{i+1})$. As this is not dependent on P_{i+1}, the probability unconditioned on P_{i+1} is the same.

The total expected cost of quick sort is then $\sum_{1 \leq i \leq n} O(\frac{n}{i+1}) = O(nlog\, n)$

Notice the important role that backward analysis played and how simple it made the analysis. Also note that the expectation is over the random choices we make in the algorithm, and not over any expectation of how input is distributed. We will always analyse expectation over the choices in the algorithm, and not over how input is generated.

3.2 Randomised binary search trees

Suppose we construct a randomised binary tree in the way we did quick sort above i.e. whenever we pick a random pivot, we make it a root node of that interval and split that interval in to two parts. At each stage, the intervals (as in quick sort) correspond to the leaves of the tree. It can be easily seen that this gives a binary search tree, and the construction time is the same as the time taken for quick sort. Let us analyse the query or search time now. Fix a query point q. We will use backward analysis again, in a more clearer and stronger way. Suppose, given the order of pivots, we delete them one by one in the reverse order i.e. $p_n, ..., p_1$ in which they were picked, and we keep an eye on which interval q is currently in. Let V_i be a random variable which is equal to 1 if while deleting p_i, we merge the interval in which q is to another interval and 0 otherwise. Clearly, the searching time for q is equal to $V = \sum_i V_i$.

When deleting p_i, as each element is equally likely to be deleted, (we are using backward analysis, we are assuming we know the set P_{i+1} but not P_i, the probability that the point

deleted will merge the interval of q is at most $2/i$ since it is adjacent to at most two pivots. Thus, $E[V_i] = 2/i \Rightarrow E[V] = O(log\, n)$.

Thus, the binary search tree constructed randomly is quite efficient.

3.3 Linear programming [2]

Suppose we have m constraints in d dimensions. We will show a randomised algorithm for for finding the optimal vertex in expected time $O(d!m)$.

We choose a random hyperplane randomly from the ones available to us and insert them one by one. If the old optimum vertex v is contained in the hyperplane H we add, then the optimum stays the same and we don't need to recompute the optimum. Else, a new optimum has to be computed, but note that this new optimum lies on H and can be found by projecting every other plane on H and solving a $d-1$ dimensional problem there.

We will use backward analysis again. Suppose H was the last plane to be added, v is the optimum with all m constraints, and v' the optimum with the first $m-1$ constraints. What is the probability that $v \neq v'$. This will happen if H is one among the d planes that define v, hence the probability is d/m.

Thus, we can write a recurrence relation as:

$$T(d,m) = T(d,m-1) + O(d) + \frac{d}{m}(O(dm) + T(d-1,m-1)) \tag{2}$$

The first term on the right corresponds to solving the problem recursively for first $m-1$ random choices of hyperplane. The second term is to determine whether v' lies in H or not. The probability of the need to compute a new optimum is d/m, in which case we project all hyperplanes onto H in $O(dm)$ time and solve the $d-1$ problem on it in time $T(d-1,m-1)$.

The base cases for the recurrence are $T(1,m) = O(m)$ and $T(d,1) = O(d)$.

It can be inductively verified that $T(d,m) = O(d!m\sum_{1\leq i\leq d}\frac{i^2}{i!})$, and the sum converges even when taken to infinity, which gives $T(d,m) = O(d!m)$.

3.4 Find the closest pair of points

Read from [3]'s chapter 1. Note that backward analysis is used here too. In d dimensions, since a grid cell has 3^d neighbours, the running time becomes $O(3^d n)$.

3.5

4 Random sampling

Here, instead of adding one element at a time, we randomly choose many of them and add them together.

4.1 Sampling lemma

Given a set of n points on a line, we choose r points randomly. What is the expected size of the partition of the points induced by the r chosen points?

It has to be made clear how we choose the r random points. One way is to choose a r-subset of the n points, such that each r-subset has equal probability of getting picked. But

this is computationally expensive. Another method, called binomial or parallel sampling, is to choose each element independently with probability r/n. We might not get exactly r points, but the expected size is r.

We now analyse the expected size of the intervals. Fix a point x among the n points and let $I_{x,R}$ be the interval x lies in for the set R of random sample and also its size. Let $I^l_{x,R}, I^r_{x,R}$ be the points in $I_{x,R}$ to the left and right of x respectively.

$P[I^l_{x,R} \geq k] = (1 - r/n)^k$ since it means that the next k elements to the left of x are not chosen in the sample. If we put $k = a\frac{n}{r} ln\, n$, the probability becomes:

$$\begin{aligned} P[I^l_{x,R} \geq a\frac{n}{r} ln\, n] &= (1 - r/n)^{a\frac{n}{r} ln\, n} \\ &= ((1 - r/n)^{n/r})^{aln\, n} \\ &\leq e^{-aln\, n} = \frac{1}{n^a} \end{aligned} \tag{3}$$

Symmetrically, the same holds for $I^r_{x,R}$.

$P[I_{x,R} > 2an/r\, ln\, n] \leq P[I^l_{x,R} > an/r\, ln\, n] + P[I^r_{x,R} > an/r\, ln\, n]$ because of union bound and the fact that the condition on the left implies that at least one of the conditions on the right must be true. Thus, this probability is less than $2/n^a$.

Remember that this was for a fixed x.

By Union bound, P[for any x, its interval size $> an/r\, ln\, n] \leq \sum_x P$[this holds for a fixed x] $= 2n/n^a = O(1/n^{a-1})$. Thus, for a large enough a, it holds high probability that all interval sizes are at most $O(n/r\, ln\, n)$. This is off from the ideal partition by only a factor of $log\, n$.

Specifically, for $r = O(ln\, n)$ we get a subproblem size i.e. interval size of $O(n/2)$.

We can improve this to $an/r\, ln\, r$. Consider an interval σ and let $|\sigma|$ denote the number of elements in its interior i.e. not counting its endpoints. What is the probability that σ is one of the intervals that results after we have chosen the r-sample? It is $(r/n)^2(1 - r/n)^{|\sigma|}$. Thus,

$$\begin{aligned} &P[\sigma \text{ is picked for a fixed } \sigma] = (r/n)^2(1 - r/n)^{|\sigma|} \\ &P[\text{for a fixed } \sigma, \sigma \text{ is picked and it has size } > an/r\, ln\, r] \leq (r/n)^2(1 - r/n)^{an/r\, ln\, r} \\ &\leq (\frac{r}{n})^2 \frac{1}{r^a} \\ &P[\text{any interval has size} > an/r\, ln\, r] \leq (\frac{r}{n})^2 \frac{1}{r^a} n^2 = 1/r^{a-2} \end{aligned} \tag{4}$$

The last equation follows from union bound and the fact that there are no more than $\binom{n}{2}$ intervals. If we now take $a > 2$, we get a constant $(1 - \frac{1}{r^{a-2}})$ probability that none of the intervals has size larger than $an/r\, ln\, r$.

This is inferable also from ϵ-net theorem, which says that any range space has an ϵ-set of size $O(\frac{1}{\epsilon} log(\frac{1}{\epsilon}))$. Since. we want no interval to have size more than $an/r\, ln\, r$, $\epsilon = a/r\, ln\, r$. This gives $O(\frac{1}{\epsilon} log(\frac{1}{\epsilon})) = O(r)$. Thus, we get that $O(r)$ points are sufficient to reduce the subproblem size to $an/r\, ln\, r$.

4.2 Point location

Given a set of n lines in the plane, and a query point q, we want to find the region in the arrangements of the lines in which q lies. This can be solved by a variety of ways:

- Making vertical slabs: Make a vertical line at each intersection points of the lines. Define a slab as the region between two consecutive vertical lines. Then, the set of lines inside a slab never intersects inside the slab, and hence we can do binary search first to find which slab q lies in, and then another binary search within the slab. This approach uses $O(n^3)$ space and $O(n^2 log\, n)$ processing time, and has query time $O(log\, n)$.
- We can use persistent data structures to reduce storage space in the above approach, after observing that two consecutive slabs differ in only one swap of two consecutive lines. This has $O(n)$ space, $O(nlog\, n)$ preprocessing and $O(log\, n)$ query time.
- Use random sampling, which we have described below

We choose r random lines, for some constant r and triangulate their arrangement. As r lines can have $O(r^2)$ intersections, the number of triangles by Euler's equality is also $O(r^2)$. We search for the query point in these triangles. Suppose it belongs in Δ. Then, we can throw away all lines that do not intersect Δ. We recurse, this time, choosing r lines only from those lines which intersect Δ. We first show that with high probability, no triangle in the arrangement of r lines has more than $n/r\, log\, r$ lines intersecting it.

From the $O(n^2)$ intersection of all lines, we can make at most $O(n^6)$ triangles. Choose one of them and call it Δ. Δ will belong to the triangulation of r lines if and only if the 6 lines defining Δ (2 for each vertex) are included in r and all other lines intersecting Δ are not included in the sample. Let $I(\Delta)$ denote the set of lines intersecting Δ, not including the 6 lines which make it. Then, $P[\Delta$ is in the triangulation$] = (r/n)^6(1 - r/n)^{|I(\Delta)|}$.

$P[\Delta$ is in the triangulation and intersects more than $an/r\, log\, r$ lines$] \leq (r/n)^6(1-r/n)^{an/r\, log\, r} \leq (r/n)^6 1/r^a$.

Taking the union bound over all triangles,

P[some triangle of r-sample has more than $an/r\, log\, r$ lines$] \leq 1/r^{a-6}$

Taking $a > 6$, the probability that no triangle has more than $an/r\, log\, r$ lines is then at least $(1 - \frac{1}{r^{a-6}})$. We keep recomputing this structure until we get such a triangulation, and then recurse. Thus, the data structure is deterministic, only its construction involved is randomised. The expected number of recomputations at each stage is $O(1)$ and hence does not affect the running time.

We now bound the total running time to compute the entire data structure. Taking the random sample of lines takes $O(n)$ time, computing the triangulation takes $O(r^2)$ time. Computing which line intersects which triangle takes $O(nr)$ time. If we need to recompute, then it only costs us $O(1)$ slowdown. Thus, one level of the data structure (tree) can be found in $O(nr + r^2)$ time. At the next stage, we have $O(r^2)$ subproblems, each intersecting at most

$O(n/r\,log\,r)$ lines. Thus, $T(n) = r^2 * T(n/r\,log\,r) + O(nr + r^2)$.

$$
\begin{aligned}
&T(n) = r^2 T(n/r log\, r) + O(nr + r^2)\\
&T(n log\, r/r) = r^2 T(n(log\, r/r)^2) + O(n log\, r + r^2)\\
&...\\
&T(n(log\, r/r)^i) = r^2 T(n(log\, r/r)^{i+1}) + O(n(log\, r)^i/r^{i-1} + r^2)\\
&...\\
&T(n(log\, r/r)^{k-1}) = r^2 T(n(log\, r/r)^k) + O(n(log\, r)^{k-1}/r^{k-2} + r^2)
\end{aligned}
$$

The depth of these equations is $k = O(log_r n)$. Multiplying the first equation by 1, second by r^2, third by r^4 and so on, and adding them to cancel the $T(n)$ terms, we get

$$
\begin{aligned}
T(n) &= O(\sum_{i=0}^{k-1} n(log\, r)^i r^{i+1} + r^{2+2i})\\
&= O(nr \sum_{i=0}^{k-1} (r log\, r)^i + r^2 \sum_{i=0}^{k-1} r^{2i})\\
&= O(nr \frac{(r log\, r)^k - 1}{r log\, r - 1} + r^2 \frac{r^{2k} - 1}{r^2 - 1})\\
&= O(n^2 log^{k-1}\, r + n^2)\\
&= O(n^2 log^{k-1}\, r) = O(n^{2 + \frac{log log\, r}{log\, r}})
\end{aligned}
$$

Since we can make r as large as we like, the running time can we written as $O(n^{2+\epsilon})$ for any $\epsilon > 0$.

What is the query time? Since $r = O(1)$, we can go down a level in constant time. The number of level is $O(log\, n)$, and hence so is the query time.

5 General analysis of randomised incremental construction

In an arrangement of n lines, a given line can intersect at most n faces. Moreover, the sum of size of faces (size of a face=edges on its boundary) intersected by a line is also $O(n)$.

Let there be N objects, which together make up some configurations. For example, a set of lines (objects) making up triangles (configurations), or making trapezoids. Each configuration σ is defined by $d(\sigma)$, which is the set of objects defining it, for example, making its boundary, and $l(\sigma)$, which is the set of objects intersecting it.

$\pi(N)$ = set of all configurations defined by the N objects

$\pi^j(N)$ = set of a all configurations with $l(\sigma) = j$

If we sample N to get a set R, then $\pi^0(R)$ is the set of configurations which are not intersected by objects from R.

If we sample N with probability p, then $P[\sigma \in \pi^0(R)] = p^{d(\sigma)}(1-p)^{l(\sigma)}$

Lemma 2. *Suppose we sample N with probability $1/2$ and get R. Then, $\pi^a(N) = O(2^{a+d}E[\pi^0(R)])$, where d is an upper bound on $d(\sigma)$, and we exploit notation and use the same symbol for a set and its size.*

Proof.

$$\begin{aligned} E[\pi^0(R)] &= \sum_{\sigma \in \pi(N)} P[\sigma \in \pi^0(R)] \\ &\geq \sum_{\sigma \in \pi^a(N)} P[\sigma \in \pi^0(R)] \\ &= \pi^a(N)\frac{1}{2^a}\frac{1}{2^d} \end{aligned}$$

□

Note that this is a combinatorial bound on $\pi^a(N)$, even though the proof is probabilistic, since the RHS is a fixed quantity.

Define the c^{th} of a configuration σ as $\binom{l(\sigma)}{c} \approx l(\sigma)^c$

$T_c = \sum_{\sigma \in \pi^0(R)} \binom{l(\sigma)}{c}$ for a random sample R. For $c = 0$, this is equal to $\pi^0(R)$. For $c = 1$, this is equal to the sum of the subproblem sizes i.e. sum of the lines intersecting each conflict-free object in R.

Lemma 3. $E[T_c] \leq \frac{1}{p^c}E[\pi^c(R)]$ *where the expectation is over the random sample R.*

Proof.

$$\begin{aligned} E[T_c] &= E[\sum_{\sigma \in \pi(N)} P[\sigma \in \pi^0(R)]\binom{l(\sigma)}{c}] \\ &= \sum_{\sigma \in \pi(N)} \binom{l(\sigma)}{c} p^{d(\sigma)}(1-p)^{l(\sigma)} \\ &= (\frac{1-p}{p})^c \sum_{\sigma \in \pi(N)} \binom{l(\sigma)}{c} p^{d(\sigma)+c}(1-p)^{l(\sigma)-c} \\ &= (\frac{1-p}{p})^c E[\pi^c(R)] \leq \frac{1}{p^c}E[\pi^c(R)] \end{aligned}$$

□

For $c = 1$, when we sample each object with probability r/n, we can combine lemma 1 and 2 to get $O(n)$ as the sum of subproblem sizes.

We now do the general analysis of RIC. We order the objects randomly and add them one by one. Maintain a bipartite graph between objects and the configurations. An edge exists if an object intersects a configuration. When we add a new object, some new edges have to be created, and some old deleted. The time for RIC is the sum of edges deleted and created in each step. Since the former can be charged to the latter, we focus only on the number of edges created upon adding the i^{th} object. Number of new edges $\leq \sum_{\sigma \in \pi^0(S^{i+1}) \setminus \pi^0(S^i)} l(\sigma)$.

$P[\sigma \in \pi^0(S^{i+1}) \setminus \pi^0(S^i)] = \frac{d(\sigma)}{i+1}$ by backward analysis. Thus, total time for RIC is $\sum_{i=1}^n \sum_{\sigma \in \pi^0(S^{i+1})} d(\sigma)l(\sigma)/i = \sum_{i=1}^n \frac{d}{i}\frac{n}{i+1}E[\pi^1(S^{i+1})] \approx \sum_{i=1}^n \frac{d}{i}\frac{n}{i+1}E[\pi^0(S^{i+1})]$.

For problems where $E[\pi^0(R)] = O(R)$, such as intersection of half planes give r triangles, tetrahedrals, we get the total time to be $O(ndlog\,n) = O(nlog\,n)$ for constant d. For lines, there are $O(r^2)$ triangles, thus we get total time as $O(n^2)$.

Consider a trapezoidal map of a set of line segments. We make vertical segments from each end point of the line segments, and from their intersection points. We cut the vertical segment short wherever it meets existing line segments. What is $E[\pi^0(S^i)]$ for this? Let A be the number of intersection points. Then, $E[\pi^0(S^i)] = S^i + A.P[\text{an intersection point is in } S^i]$. The latter happens when both the line segments of the intersection point are in R, the probability of which is i^2/n^2. Thus, $E[\pi^0(S^i)] = i + A.i^2/n^2$. And the total time for RIC comes out to be $O(nlog\,n + A)$ for constant d. Note that this is output sensitive running time. A can be as big as $O(n^2)$, but our algorithm performs much better when A is small.

6 Randomised approximation algorithm of set cover

We make relaxed LP for set cover:

$$\begin{aligned} min. & \sum_{j=1}^{m} w_j x_j \\ s.t. & \sum_{j:e\in S_j} x_j \geq 1 \text{ for all } e \in X \\ & x_j \in [0,1] \text{ for all } j \end{aligned}$$

We solve this to get optimal fractional solution x^*, and round each variable to 1 with probability x_j^*. But this alone might not be a set cover. We repeat this $c\,ln\,n$ times, and set x_j to 0 only if it comes to be zero in all trials. We will show that with high probability, this is a set cover.

$P[e \in X \text{ is not covered}] = \Pi_{j:e\in S_j}(1 - x_j^*)^{c\,ln\,n} \leq \Pi_{j:e\in S_j} e^{-x_j^* c\,ln\,n} \leq 1/n^c$. Thus, probability that some element is not covered is $1/n^{c-1}$. Thus, for large enough c, it is a set cover with high probability.

Now we bound its cost. $E[\sum_{j=1}^m w_j X_j] = \sum_{j=1}^m w_j P[X_j = 1] \leq \sum_{j=1}^m w_j (c\,ln\,n) x_j^*$ by Union Bound. Thus, $E[\sum_{j=1}^m w_j X_j] \leq c\,ln\,n \sum_{j=1}^m w_j x_j^* \leq (c\,ln\,n)OPT$

7 ϵ-nets, ϵ-samples

7.1 VC-dimension

Given a range space $S = (X, R)$, for $A \subset$ X, define $R_{|A} = \{r \cap A | r \in R\}$. If $R_{|A}$ contains all subsets of A, then A is said to be shattered by R. The VC-dimension of S $dim_{VC}(S)$ is the maximum cardinality of a shattered subset of X.

- X = points on line, R = all intervals. $dim_{VC}(S) = 2$
- X = points on plane, R = all disks. $dim_{VC}(S) = 3$
- X = points on plane, R = all convex sets. $dim_{VC}(S) = \infty$

- $S^c = (X, R^c)$ where R^c consists of the complement of all ranges in R. Then, $dim_{VC}(S^c) = dim_{VC}(S)$
- $S = (\Re^d, halfplanes)$, $dim_{VC}(S) = d + 1$

Let us prove the last one i.e. find the VC dimension for half planes. A regular simplex with $d + 1$ vertices in $\Re^d$ can be shattered by half planes, so $dim_{VC}(S) \geq d + 1$. Given a set of $d + 2$ points in $\Re^d$, we know by Radon's Lemma that there exists a partition of it into two subsets A, B such that their convex hulls intersect. But then, we can't cover the points of A without including some point of B in it too. Thus, no set of $d+2$ points can be shattered.

The most useful thing about bounded VC-dimension range spaces are that the number of ranges is bounded by a polynomial in n, not exponential in n. Thus, this reduces the complexity of the range space a lot. Formally, for a range space $S = (X, R)$ with VC-dim=d, the number of ranges is bounded by n^d. Let's prove this.

Define $G_d(n) = \sum_{i=0}^{d} \binom{n}{i} \leq \sum_{i=0}^{d} \frac{n^i}{i!} \leq n^d$. $G_d(n)$ can be interpreted as the number of subsets of size of at most d from a set of size n. Thus, $G_d(n) = G_{d-1}(n-1) + G_d(n-1)$.

Fix an element $x \in X$. Let $R_x = \{r \in R : r \cup \{x\} \in R, r \setminus \{x\} \in R\}$, $R_{-x} = \{r \setminus \{x\} : r \in R\}$. Then, $|R| = |R_x| + |R_{-x}|$. We will induct on d, n. $(X \setminus \{x\}, R_x)$ has VC-dim $d-1$, hence $|R_x| \leq G_{d-1}(n-1)$. $(X \setminus \{x\}, R_{-x})$ has VC-dim at most d, hence $|R_{-x}| \leq G_d(n-1)$. Thus, $|R| \leq G_{d-1}(n-1) + G_d(n-1) = G_d(n)$.

7.2 Shattering dimension

Define shatter function for a range space $S = (X, R)$ by $\pi_S(m) = max_{B \subset X, |B|=m} \left|R_{|B}\right|$ i.e. the maximum number of sets induced by the ranges of a subset of size m of X.

The shattering dimension of S is the smallest d such that $\pi_S(m) = O(m^d)$ for all m.

Lemma 4. *If $S = (X, R)$ has VC-dim= d, then $\left|R_{|B}\right| \leq \pi_S(|B|) \leq G_d(|B|)$*

Proof. The first inequality is obvious since, by definition of shatter function, π_S is the maximum over $\left|R_{|B}\right|$. Also, $\left|R_{|B}\right| \leq G_d(|B|)$ by the fact that this is the maximum number of ranges for a range space of VC-dim d. Since this holds for all B, it also holds if we take the max over it, while keeping $|B|$ fixed, as this won't change the RHS. Thus, the second inequality also holds. □

This lemma shows that shattering dimension is bounded by VC-dimension since $G_d(n) \leq n^d$.

Let's compute the shattering dimension of disks in plane. Given a set P of n points, and a disk, we can shrink,expand, translate the disk till it has exactly three points on its boundary and no point which was earlier in its exterior is now in interior and vice-versa.Thus, the number of disks is bounded by $n^3 3^3$, since we choose the 3 points in n^3 ways, and for each of those points, decide whether it is in the interior or exterior or on the boundary of the disk. Thus, shattering dimension is 3.

In general, shattering dimensions are easier to work with than VC-dimensions. As done above for disks, we can do for all range spaces defined by a family of shapes and say that the shattering dimension is bounded by the number of points that determine a shape in the

family uniquely.

If shatter dimension is d, then VC-dim is $O(d\,ln\,d)$.

7.3 Dual range spaces

Let S be a range space and S^* its dual range space. If S has R ranges, and we construct an arrangement on them, then each point lying in the same face of the arrangement gives the same range in the dual space. Thus, the number of ranges in the dual range space is bounded by the complexity of the arrangement of ranges of S. This is $O(m^2)$ for m disks.

Dual shatter function $\pi^*_S(m) = \pi_{S^*}(m)$
Dual shattering dimension of S = shattering dimension of S^*

From what is discussed above, dual shattering dimension for disks on a plane is 2, while shattering and VC-dimension are 3.

Lemma 5. *If S has VC-dim=d, then VC-dim of $S^* \leq 2^d$.*

Lemma 6. *Let $S = (X, R)$ have VC-dim d and $S' = (X, R')$ have VC-dim d'. $Q = \{r \cup r' : r \in R, r' \in R'\}$. Then, (X, Q) has VC-dim $= O((d + d')log(d + d'))$.*

Proof. Let A be a subset of size n of X shattered by Q. Since $\left|R_{|A}\right| = G_d(n)$, $\left|R'_{|A}\right| = G_{d'}(n)$, $|Q_A|$ is bounded by $G_d(n)G_{d'}(n)$. Thus, $2^n \leq n^d n^{d'} = n^{d+d'}$. Thus, $n = O((d + d')log(d + d'))$. □

The same result holds if we take intersection instead of union since $r \cap r' = (r^c \cup r'^c)^c$.

7.4 $\epsilon-$nets and samples

$\epsilon-$sample: Let $S = (X, R)$ be a range space, and B a finite subset of X. $C \subseteq B$ is an $\epsilon-$sample of B if for all ranges $r \in R$

$$\left|\frac{|B \cap r|}{|B|} - \frac{|C \cap r|}{|C|}\right| \leq \epsilon$$

i.e. C approximates the fraction of points that B covers in all ranges.

ϵ-sample theorem : If S has VC-dim at most d, then given $\epsilon, \delta > 0$, a random subset C of B of size

$$\frac{c}{\epsilon^2}(d\,log\frac{d}{\epsilon} + log\frac{1}{\delta})$$

is an ϵ-sample of B with probability $1 - \delta$ for some constant c.

ϵ-net: $N \subseteq B$ is an ϵ-net of B if for all $r \in R$, $|B \cap r| \geq \epsilon\,|B| \implies N \cap r \neq \phi$.

ϵ-net theorem : If S has VC-dim at most d, then given $\epsilon, \delta > 0$, a random subset C of B of size

$$max(\frac{4}{\epsilon}log\frac{2}{\delta}, \frac{8d}{\epsilon}log\frac{8d}{\epsilon})$$

is an ϵ-net of B with probability $1 - \delta$.

7.5 Discrepancy

Consider a colouring $\chi : X \to \{+1, -1\}$. We define the imbalance, or discrepancy of χ over $r \in R$ to be $disc(r, \chi) = \left|\sum_{p \in r} \chi(p)\right|$. Discrepancy of a coloring is defined as $disc(\chi) = max_{r \in R} disc(r, \chi)$.

Discrepancy of a range space S is defined as the discrepancy of the best possible colouring i.e. $disc(S) = min_{\chi} disc(\chi)$.

The motivation behind this is that a good colouring can serve as a good ϵ-sample, though it may be of a large size.

Consider a perfect matching P of the points of X. For each pair in P, we colour it $\{-1, +1\}$ or $\{+1, -1\}$ with equal probability. We will now show that with high probability this colouring χ has low discrepancy.

Theorem 7. *Let $S = (X, R)$ be a range space with n elements and m ranges. Then, with probability $\geq 1/2$, the above procedure gives a colouring with discrepancy at most $\sqrt{n\, ln(4m)}$.*

Proof. Consider a range $r \in R$. All the pairs of P that are completely inside or completely outside r do not contribute to its discrepancy. Let n_r be the number of pairs crossing r. Notice that $n_r \leq n/2$. Make a random variable X_i for each of these pairs which is equal to the colour of the point which is inside r. Then, $\chi(r) = |\sum_{i=1}^{n_r} X_i|$

Let $\Delta = \sqrt{n\, ln(4m)}$.

$$\begin{aligned} P[\chi(r) \geq \Delta] &\leq 2P[\sum_{i=1}^{n_r} X_i \geq \Delta] \\ &\leq 2e^{-\Delta^2/2n_r} \leq \frac{1}{2m} \end{aligned}$$

The first inequality follows since $\sum_{i=1}^{n_r} X_i$ is symmetrical around zero. Thus, probability that some range has discrepancy more than Δ is less than half. □

There exists a better result, which ensures discrepancy of $O(\sqrt{n\, ln(m/n)})$. This is known as Spencer's bound, and is tight.

7.5.1 Building ϵ-sample via discrepancy

Consider a range space S of VC-dim d. Let n be a power of 2. We build a matching on it, and construct a discrepancy as just described. Let Q be set of points with colour -1. Then, $|Q| = n/2$. Now,

$$\begin{aligned} & |(X \setminus Q) \cap r - Q \cap r| \leq \sqrt{n\, ln(4m)} \\ \Rightarrow & |X \cap r - 2Q \cap r| \leq \sqrt{n\, ln(4m)} \\ \Rightarrow & \left|\frac{X \cap r}{X} - \frac{Q \cap r}{Q}\right| \leq \sqrt{ln(4m)/n} \leq c\sqrt{d\, ln(n)/n} = \tau(n) \text{ for some constant c} \end{aligned}$$

Thus, this coloring yields a $\tau(n)$-sample. But its size is too big, namely $n/2$. For $\epsilon > \tau(n)$, we can repeat the process to get a smaller size sample.

Lemma 8. *Let A be a ρ-sample of X and B be an ϵ-sample of A. Then, B is a $(\rho+\epsilon)$-sample of X.*

Let $P_0 = X, P_1 = Q$. We compute P_i by computing a low discrepancy as earlier on P_{i-1} and taking all the points with colour -1. $|P_i| = n/2^i = n_i$ and P_k is a $\sum_{i=0}^{k-1} \tau(n/2^i)$-sample of X.

We choose the largest k such that $\sum_{i=0}^{k-1} \tau(n/2^i) \leq \epsilon$.

$$\begin{aligned}
&\sum_{i=0}^{k-1} \tau(n/2^i) \leq \epsilon \\
\Rightarrow &\sum_{i=0}^{k-1} c\sqrt{\frac{d\, ln(n/2^i)}{n/2^i}} \leq c_1\sqrt{\frac{d\, ln(n/2^{k-1})}{n/2^{k-1}}} \leq \epsilon \\
\Rightarrow &n/2^{k-1} \geq O(\frac{d}{\epsilon^2} ln \frac{d}{\epsilon})
\end{aligned}$$

Thus, we get an ϵ-sample of size $O(\frac{d}{\epsilon^2} ln \frac{d}{\epsilon})$.

7.5.2 Building ϵ-net via discrepancy

For P_i as constructed earlier and a low discrepancy colouring on it, we can get, by slight modification of the discrepancy bound, that for a range $r \in R$

$$|P_{i-1} \cap r - 2P_i \cap r| \leq c\sqrt{d\,|P_{i-1} \cap r|\, ln(n_{i-1})}$$

Define $v_i = |P_i \cap r|$ which is the size of the range r in the set P_i. The above equation can be rewritten as

$$\left|2^{i-1} v_{i-1} - 2^i v_i\right| \leq c\, 2^{i-1} \sqrt{d\, v_{i-1} ln(n_{i-1})}$$

Lemma 9. *$1/2(v_0/2^k) \leq v_k \leq 2(v_0/2^k)$ for all k with $v_0/2^k \geq c_1 d\, ln(n_k))$ for some constant c_1.*

Consider a range r with $v_0 \geq \epsilon n$. We need to show that we have at least one point from each such range. To apply the above lemma, we would then need $\epsilon n/2^k \geq c_1 d\, ln(n_k)) \implies n_k = \Omega(\frac{d}{\epsilon} ln \frac{d}{\epsilon})$.

But then, $|P_k \cap r| = v_k \geq \frac{1}{2} \frac{\epsilon n}{2^k} = \frac{1}{2} \epsilon n_k = \Omega(d\, ln \frac{d}{\epsilon}) > 0$, thus P_k covers at least one point of r.

8 Hitting sets via ϵ-nets

We present the iterative reweighted scheme, also known as Bronnimann Goodrich technique, to compute hitting sets.

We assume that we have an algorithm that given ϵ, and a weighted range space, returns an ϵ-net of size $s(1/\epsilon)$. The ϵ-net is required to hit all ranges which covers points of total weight more than $\epsilon w(X)$.

Suppose we know h, the optimal hitting set size. This can be found out within a factor of 2 by doing a binary search. We start with the unweighted range space, and find a $1/2h$-net

of size $s(2h)$. If it is a hitting set, we stop. Else, find a range r that is not hit. We double the weight of all the points in r. We now find a $1/2h$-net again and repeat till we have found a hitting set.

Lemma 10. *The above algorithm stop after at most $4hlog(n/h)$ iterations.*

Proof. Each time we double the weight of a range r which is not covered by a $1/2h$-net, hence $w(r) \leq w(X)/2h$. Thus, after k iterations, $w_k(X) \leq (1 + 1/2h)^k n \leq ne^{k/2h}$

Consider an optimal hitting set H. Since it hits all the ranges, at least one of its elements must have got its weight doubled in each iteration. Let k_i be the number of times the weight of the i^{th} element of H was doubled till k iterations. Thus, $w_k(H) = \sum_{i=1}^{h} 2^{k_i}$ and $\sum_{i=1}^{h} k_i \geq k$. Using convexity of exponentials, $w_k(H) \geq h2^{k/h}$.

We get $h2^{k/h} \leq w_k(H) \leq w_k(X) \leq ne^{k/2h} \leq n2^{3k/4h}$. This implies that $k \leq 4hlog(n/h)$ □

But this algorithms stops only when we find a valid hitting set. Thus, we are guaranteed to find a hitting set after at most $4hlog(n/h)$ iterations of size $s(2c)$.

9 Johnson Lindenstrauss lemma

Theorem 11. *For any $0 > \epsilon < 1$ and any integer n, let k be an integer such that*

$$k \geq \frac{4}{\epsilon^2/2 - \epsilon^3/3} log\, n.$$

Then for any set V of n points in $\Re^d$, there is a map $f : \Re^d \to \Re^k$ such that for any $u, v \in V$,

$$(1 - \epsilon)||u - v||^2 \leq ||f(u) - f(v)||^2 \leq (1 + \epsilon)||u - v||^2.$$

This map can be found in randomised polynomial time. Note that k doesn't depend on d and just on the number of points. The bound on k is tight up to constant factors.

10 Derandomization

Las Vegas algorithms: They are always correct, but running time is a random variable. Defined by class $\mathbb{ZP}$.
Monte Carlo algorithms: Correct with probability $> 1/2$ and run in polynomial time. Defined by class $\mathbb{RP}$

Given a set X of n elements, and subsets $S_1, ..., S_k$ of X s.t. $|S_i| = r \leq n$ for all i, colour each element of X in $\{-1, +1\}$ such that no S_i is monochromatic.

Color each element randomly with probability $1/2$ for each colour. Then, $P[S_i$ is monochromatic$] \leq 2/2^r$. Thus, P[some S_i is monochromatic$] \leq 2k/2^r \leq 1/2$ for $k \leq 2^{r-2}$. Thus, for such a k more than half of the colourings are acceptable. Now we derandomize this algorithm by method of conditional expectations.

We colour the elements of X one by one, say in order $x_1, .., x_n$.
Define $f_i = E[\text{some } S_j \text{ is monochromatic } |x_1, ..., x_i]$. f_i is the probability that some subset will be monochromatic given the colourings of first i elements. Thus, it is the failure probability. As long as it is strictly less than 1, we know that there exists at least one good colouring which matches the colours of the first i elements.

We first compute E[some S_j is monochromatic $|x_1 = 0$] and E[some S_j is monochromatic $|x_1 = 1$]. Since there exist good colourings, at least one of them has to be less than 1. We choose that value of x_1 and continue. At the i^{th} stage, we compute E[some S_j is monochromatic $|x_1, ..., x_{i-1}, x_i = 0$] and E[some S_j is monochromatic $|x_1, ..., x_{i-1}, x_i = 1$]. Again, one has to be strictly less than 1, so we choose that value of x_i and continue. After the last step, the failure probability must either be 0 or 1, and since we chose the one strictly less than 1, it is 0, hence we get a good colouring.

We might ask how to compute the expected values? Calculation just like before will give an upper bound on the expected values, and if any of them is less than 1, we choose that. But what if we get both values as more than 1 since union bound is loose and only gives an upper bound. To solve this, note that we can calculate the expected probability exactly for each subset by calculating. Let this quantity be $f_i^{S_j}$. We now change the definition of f_i. Now, $f_i = \sum_j f_i^{S_j}$. We also change the algorithm to choose at each stage, that colour for x_i for which f_i is minimum.

Then, $f_i \leq f_{i-1} = 1/2(f_{i-1}|x_i = 0) + 1/2(f_{i-1}|x_i = 1)$. Thus, $f_i \leq f_{i-1} \leq ... \leq f_0 < 1$. At the end f_i must be 0 since all its terms are either zero or one then, and it is strictly less than 1, so they all must be zero. Thus, we get a good colouring.

References

[1] Ketan Mulmuley *Computational Goemetry: An Introduction through Randomized Algorithms*

[2] Raimund Seidel *Small dimensional linear programming and convex hulls made easy* 1991

[3] Sariel Har-Peled *Geometric approximation algorithms*

[4] Motwani, Raghavan *Randomized Algorithms*

[5] Bronnimann, Goodrich *Almost optimal set covers in finite VC-dimension*

[6] Dasgupta, Gupta *An elementary proof of a theorem of Johnson and Lindenstrauss*