Computational Geometry

Prof. Sandeep Sen

Department of Computer Science and engineering

Indian Institute of Technology, Delhi

Module No. # 13

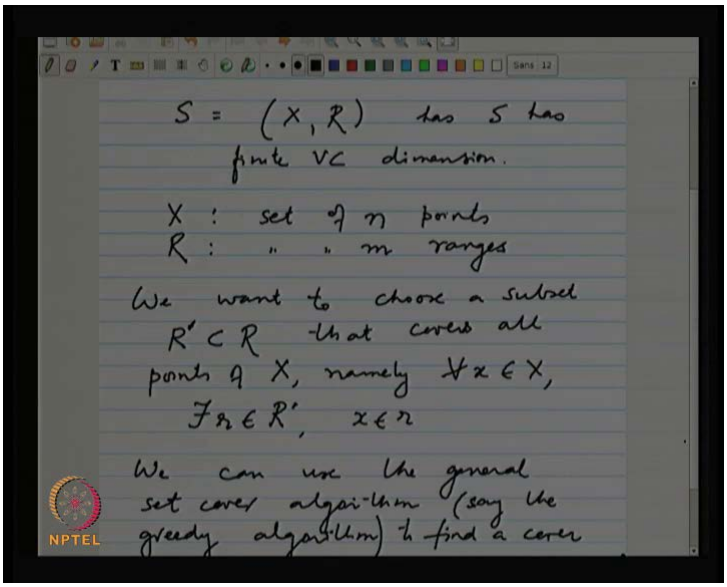
E-nets, VC Dimension & Applications

Lecture No. # 03

Geometric set Cover

We begin with a problem for which we have been doing the buildup exercise for last few lectures. So, that is geometric set cover problem and particular will assume that we have a configuration space, where we have some kind of handle on the VC dimension.

(Refer Slide Time: 00:52)



More generally, now we have this range space S equal to X, R and S has finite VC dimension. So, X is a set of n points.

(Audio not available: 1:17-1:49)

We want to cover, want to choose a subset, say R prime of R that cover all points of X, namely of all X, the axis and R are prime such that X (No audio available: 2:59-3:10).

There is no difference in set cover problem, except that now we are in a geometric domain or we have a set system that has finite VC dimension. Not absolutely in general, we have some handle on the VC dimension; we are saying that the VC dimension is finite.

Now, the general set cover problem is one of the most classic hard problems right. It is NP hard problem and we have to live with some kind of approximation and the classic greedy algorithm for the set cover gives you log n approximation. So, the question here is that, well of course, we can use any kind of because the generic set cover will also work here. We can use the general set cover algorithm say, the greedy algorithm to find a cover of size, that is k times log of n, where k is a size of the optimum cover.

What we want to do better? So, the other thing I should point out is that in general, in the general frame work, the set cover is a hard problem. It is probably a NP hard problem, but when I pose this problem in this context, we have to actually formally show that this is also a hard problem. So, there are proofs for let us say, some so for so I can give you an example that if you consider the covering problem, where you have a set of points and the ranges are circular disk, something like this.

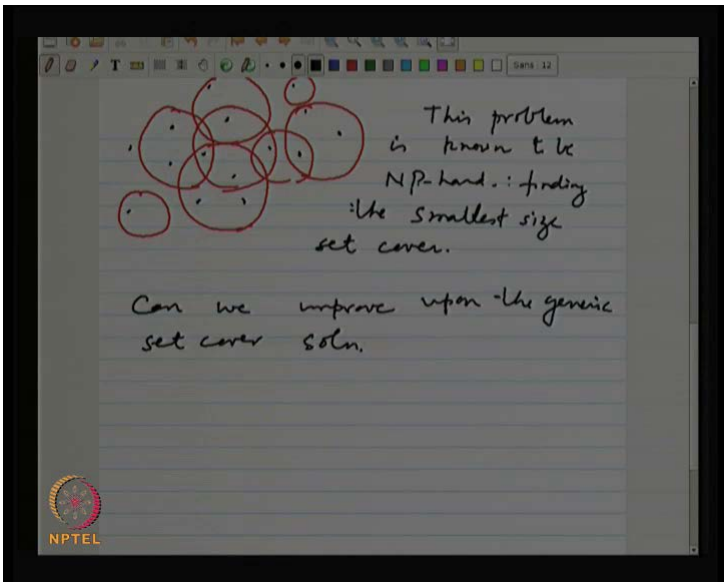
(No audio available: 5:39-5:58)

So, ranges are disks and points are points in the plain, so there is a proof that this problem is a hard problem, if you want to find the optimum size of the set cover. So, this problem is known to be NP hand finding.

(No audio available: 6:30-6:48)

So, again the only thing that we can hope for is to if we want to live in the polynomial time world, then we have to be we have to find way to approximate this solution. If we run the generic set cover algorithm, then it can be approximated within login factor of the optimum size set cover.

(Refer Slide Time: 05:32)



The question now is that given that we have this VC dimension constrain, can we do anything better right? Can we improve upon the generic?

(No audio available: 7:28-7:34)

It is also now known that, that login factor approximation is kind of the best that one can do in polynomial dynamic. I mean of course, you know modular the fact, whether if p is not equal to NP, so modular p not equal to NP. That is the best example one can do.

So, if we are hoping about something better than login approximation, then clearly this problem is at least you know in the current state of knowledge, the problem will be easier than the generic set cover, right. If we get provably something in approximation factor better than login, then it proves that this problem is you know actually easier to some extent than the generic set cover problem.

(No audio available: 8:20-8:27)

But, that is a very special set cover right. This is a yeah vertex cover is a, so set cover the generic cover problem is the login solution and that also lower bound, but vertex cover is a special case of a set cover problem, which is also a hard problem, but it is a factor to a approximation right.

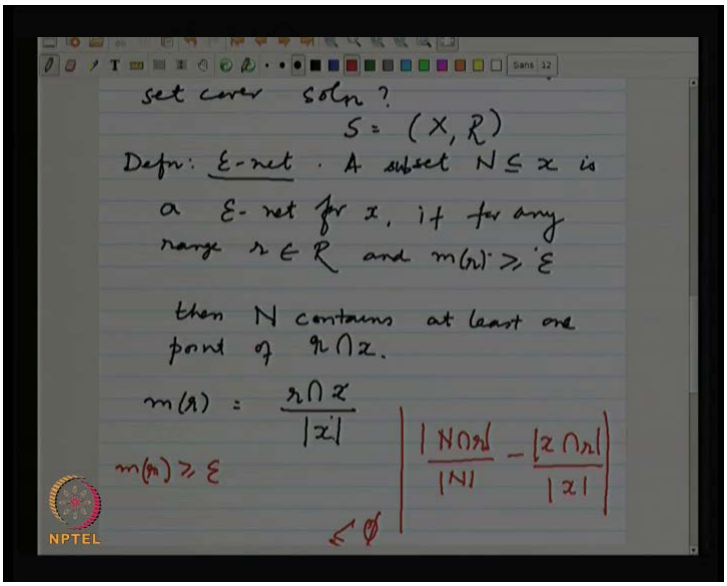
So, this is also the sets are given to us, you know the elements, the sets. The only difference between the generic set cover and this is that this is a bounded VC dimension. So, yes I mean, to some extent what you are saying is right. I mean we can relate, if we can relate to the fact that vertex cover has a better approximation factor because this is special case of a set cover, this is also a special case of a set cover problem because it has a bounded VC dimension. So, we want to see if we can do anything with anything better than the generic set cover because we have a bounded VC dimension.

(Audio not available: 9:18-9:24)

No, so we are not imposing. Yeah yeah sure. As such there are all kinds of these special cases, where you have thing. So, this can be viewed upon a special case, but this special case you know is rather a still quiet generic because it manages to cover the whole domain of geometric configuration and many other things, right. So, this is a very, this is one of the most relaxed sort of version of the set cover, where you can, you are hoping to get a solution provably better than what we get for a set cover. There is still, this still contains large number naturally occurring problem in the geometric domain.

So, the answer to this is clearly yes. Yes, because we are building up to it. So, how should we go about this? So, before I present the algorithm, let me go back to the definition of the epsilon net, so that we will require for the algorithm right. So, I did given intuitive definition of epsilon net, that it is essentially some kind of a cover for large sets. Epsilon net is essentially a set cover that covers only the large sets, so here is a formal definition.

(Refer Slide Time: 10:37)



(No audio available: 10:36-12:06)

So, let me remind you that m's of X is all the (Audio not available: 12:13-12:21).

So, let me not define this. Sorry. Then let us use this.

(No audio available: 12:29-13:33)

So, m's of power is the ratio of the points lying inside the range of the sample divided by the total size of sample.

(Audio not available: 13:40-14:28)

Ya right. So, this is the, so there is some range R that contains at least epsilon fraction of the points. This is relatively large range. The numbers of points of X inside this range are divided by total number of points. This m saying is greater than equal to epsilon. So, this range is a fairly heavy range. It contains a lot of points of X. So, moment we have such a range that, then we require the subset N must hit at least 1 point of this range.

(Audio not available: 15:04-15:11)

Capital X is the range space, so range spaces oh sorry sorry I do not, should not use capital X, I should use small x. Range space is x, so capital X is some subset of N points.

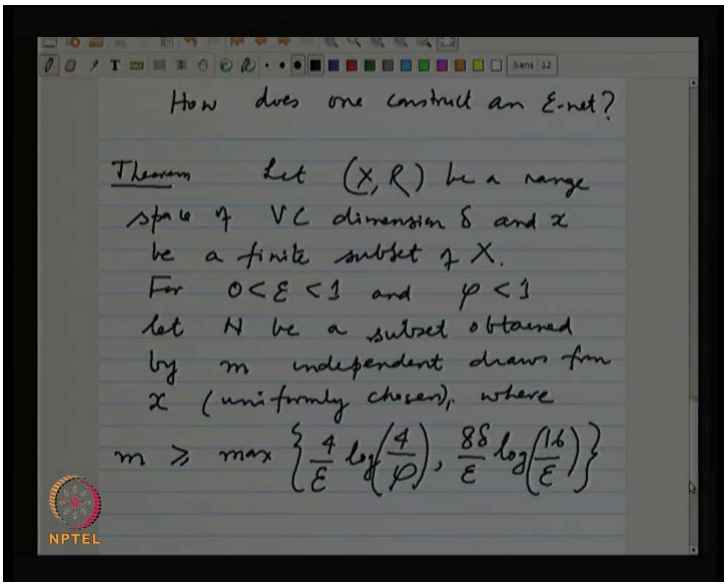
(No audio available: 15:48-15:56)

N contain at least one point of r intersection x.

(No audio available: 16:12-16:32)

No no, what I am saying, I am (()). Finally, we got it right. So, let me go through again. So, a subset N is an epsilon net for X if for any range that contains large fraction of the points of X, N must contain at least one such point.

(Refer Slide Time: 17:35)



(No audio available: 17:00-17:24)

So, how does one construct an epsilon net?

(No audio available: 17:28-17:45)

So, turns out that the easiest way of computing epsilon net is that you do some random sampling. You select the sample of a certain size and you can prove that if you choose a large enough samples, if you satisfy this problem, which in some sense gives an existential proof that there is an epsilon net of this size. See, if I choose all the points, of course I am going to hit everything. The point is, how small make this subset N? How can I choose this capital N as small as possible and still accomplish the same? I am trying to hit everything, so I want to find this small hitting set in some scenes right. It has to hit all the large subsets.

So, if I pick up all the point, of course, that is a trivial solution, but I will not be satisfied with that. So, you will want to select a relatively small subset N and if we pick a you know this subset by random sampling, and I pick a sample of certain size that I will just state and you know what the bounds are and we show that, that is N epsilon net with a non 0 probability. There is some positive possibility. It means that there must be some subset of that size, which fulfills the condition of being an epsilon net. So, it is an existential proof. So, it is a, so that sampling is a randomized algorithm because it is an algorithm I just have to sample and the sample satisfy the property that are constructive, but if someone is interested to find out such a subset deterministically, then that is different bounding. So, all you know is that if I pick a subset of this size randomly, then it satisfies the property.

Now, what do we have to do, so that we can choose a subset deterministically? That could be the other challenging problem right and it turns out that if I suppose, I pick, I manage to show, let us say there is some, if I pick 100 points, then it is a it is a epsilon net at probability at least half. Then, from that argument, that existential argument one can use something called general (()) technique. So, there is some technique by which you can (()) certain algorithm, not all algorithms, but it so happens for epsilon nets and well, actually not just epsilon net, something called epsilon sample.

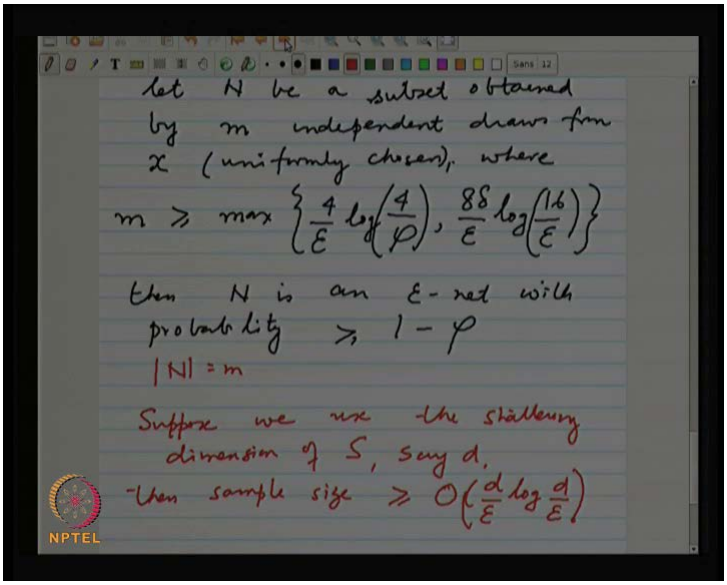
So, here what we are doing? We are saying that this one as long as is a large subset, we will pick up at least 1 point of this. In an epsilon in a in a in a, I do not know if you will recollect the definition of epsilon sample, we want that ratio, so the number of points that we pick up from the subset. Here, we are saying one is sufficient, but if we require that that ratio should not be ratio when we are talking about points in n and the ratio when we are talking in X, they are not be very different. If we if we if we want that to be very close to each other, suppose I want this condition, let me write it down.

Suppose I want this condition that N intersection r divided by N. This is the number of sample hitting that range R minus the number of points in X that intersects that and the ratio, this one. So, if I want this ratio to be bounded, let us say I want this whole thing, take that absolute value of that and suppose, I can bound this by some epsilon. I do not have the rights, so I will just write it here. So, suppose this is this less than some p specify epsilon. Let me use, maybe something else, phi and this will again hold only for the large subsets.

So, we are saying this will hold only for m's of R greater than epsilon. Again, for the large subsets, I want this condition to hold such that, the difference between these ratios is no more than some p specified phi. So, this is the most stringent condition. It is not that just one point that we want, I want even the difference to the ratio to be bounded. Then, it is called an epsilon sample, all right. So, there is a way that you can, so this random sample you know gives us some properties of an epsilon sample and from there, it is possible to de-randomize to get a epsilon sample of, let us say, size which is no more than the constant times what I get by randomize algorithm.

Suppose, the randomize argument shows that if I pick 100 points, it is sufficient. Then, starting from that point that is existential proof, we can convert that and as is in very clever searching methods in polynomial time to obtain a deterministic sample, it is a deterministic sample of size. Let us say of no more than 2 times 100, so that those are generic techniques that are called de-randomization. So, it is possible to de-randomize some of these algorithms, this epsilon net algorithm to actually construct epsilon net deterministically of almost the same bound as what you get by randomization, but this is rather neat and simple. So, we will not get into that. That will be required complete different techniques. So, I will just state the bounds and know what is the sample is required that will guarantee that. It is not a lower bound, it is an upward bound.

(Refer Slide Time: 26:30)



(No audio available: 24:01-25:32)

So, this is the bound.

(No audio available: 25:34-25:53)

Essentially, it is little messy expression, but it is fallen out for some kind of proof.

(No audio available: 25:57-26:29)

So, what we are doing? We have this finite set of point small x, right. We are drawing; we are creating a random sample by sampling each points of x uniformly. So, I will create a sample you know if I have 100 points, you know I want to choose 50 points. What I am going to do is, I will choose one point at random and then, I will choose a next point at random so and so forth. Independent draw essentially means that you know these are, so I have little bit of doubt whether it should be with the replacement or without replacement. Let us say, this is without replacement. So, I draw one point and then, I draw another point and so on so forth. So, I create this random sample of this size m, small m and small m is larger of these two terms.

What is the significance? So, we have a bounded VC dimension delta. So, this sample size turns out to be a function of the VC dimension delta. Here, you can see in the second term, there is 8 delta and there is also, it must be a function of epsilon because we are constructing an epsilon net. So, for it must hit all sets of size of at least epsilon net, right. So, it is a function of both epsilon and delta. So, the larger the VC dimension, the more points we need in the sample. The smaller the epsilon, the more points we need. That is how we can reconcile these two and then, there is a dependence on phi which is nothing, but the success probability.

So, if we can choose phi 2 equal to half, let say. So, we are satisfied as long as there is a positive probability that it is going to be epsilon net. So, if you can set phi to be equal to half and then, you know, then you may not even want phi in this expression, but this one say actually this one is very general expression. If I want a higher probability, then of course, phi is the failure probability. So, it is inversely proportional to phi. So, this is calculation requires a proof, but you know we are not going to go through this proof. This proof is not that hard, but this proof uses one very clever observation which will require sometime for us to digest.

Now, what I can do is, I can post this proof or I can you know we can make a Xerox copy of this and leave somewhere, so you can access that. In fact, I could do the same thing with lectures that I have been doing in the last few weeks. So, it is from an unpublished manuscript. I do not, I cannot make it public, I cannot post it on the web, but I have the author's permission that I can sort of make the printed copies available to you. So, I will do that. It is about 70-80 pages. It is not very trivial, but anyway I will leave it, so that if anyone is interested in the proof of this. This proof is available in many places, if you just search the internet, you will find the proof of VC dimension. The proof is not trivial, but the proof is not too hard. It is along the same lines that we have been arguing before, but it requires one clever trick, so that you know I do not want to, that is to get into this today.

This was given by the original paper (()) case. So, the original paper contains this form actually. So, the objective of the original paper was to show that there is an epsilon net of this size right and the de-randomization etcetera has been done much later. So, this was in the early 70s, de-randomization happened sometimes in late 80s. De-randomization is due to (( )).

So, just stare at this and then, we will take off into the algorithm. So, there is let us say, some bond that if you pick a sample of this size, then it is going to be a epsilon net with some, let us say probability at least half. In other words, you know and suppose, I can verify this, suppose I can verify that whether or not a sample is an epsilon net and if I do not succeed, what do I do. I resample again. So, if it succeeds with probability half, then the expectation I need to only resample twice before I actually get an epsilon net. This is the way, this through randomized algorithm, it will work by repetition.

(No audio available: 30:52-31:08)

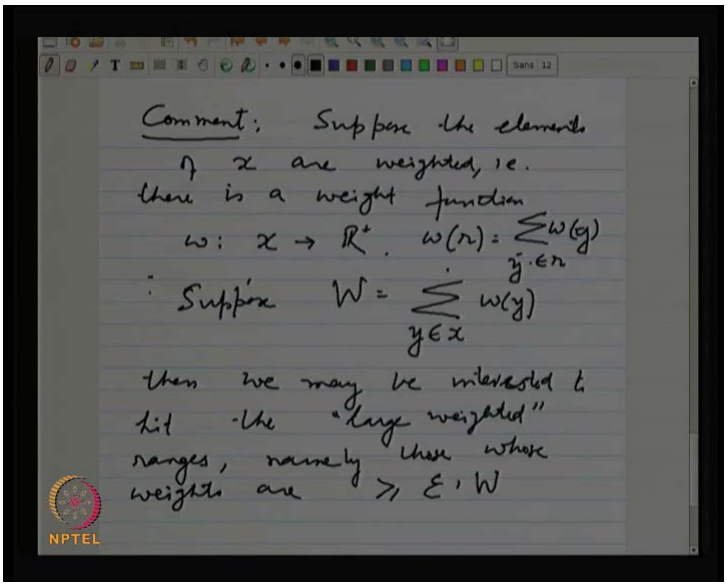
No, the capital N is the subset that is constructed and m is the small m is the size of the sample. So, have I used l to be a subset of 10 by N independent row rates of size of m is n, right? I am constructing a sample this way. So, as long as larger than this size, as long as the sample is larger than this size, then it satisfies the property of an epsilon net. There is one more comment, let me make. So, here we are talking about VC dimension.

(No audio available: 31:44-31:54)

No, capital X may or may not be finite right. Capital X could be just you know space of the plain basically or the real line, whatever. So, the definition of VC dimension as such is not limited to finite spaces. You know it applies to (( )) places, continuous spaces, but when you are defining, so that is beauty of it actually. So, we are defining it for a continuous space, but the definition is with respect of the shattering function is respect to a finite point set. So, the definition of the VC dimension is with respect to a finite point set and then, the shattering function, the shattering set. So, somehow, this continuous space can be captured by some you know finite point set essentially.

So, one more comment on this is if instead of VC dimension, so here we are using in the formula VC dimension delta, right. Suppose, instead of VC dimension, we were given the shattering dimension. Suppose we use the shattering dimension of S, say d. Then, again we can get some kind of a, there is a relationship between shattering dimension and VC dimension. So, we could write a formula and if you simplify that, it turns out the sample size, you know can be relatively that the formula is relatively simple. So, then sample size is less than equal to, think it is order delta, sorry d epsilon net log of E. So, here d is the shattering dimension. So, if you use the shattering dimension instead of the VC dimension, then the size of the sample turns out to be about this much.

(Refer Slide Time: 35:25)



Now, we will go to the algorithm.

(No audio available: 34:43-35:17)

May be I should make one more comment on to it before I go to algorithm.

(No audio available: 35:20-35:26)

So, this is an epsilon net, the formula for epsilon net when the elements are unweighted. Suppose, let us make this comment, some other comment basically. Suppose, the elements of x are weighted, that is weight function, some real numbers and the weight of a subset of R equals the weight of summation of, where all x, weight of x. So, in this weighted scenario, so here we were interested to hit every large subset in terms of cardinality. In the weighted scenario, we will be interested to hit those ranges that have a certain minimum weight associated with it. Again maybe, so if capital W is a total weight.

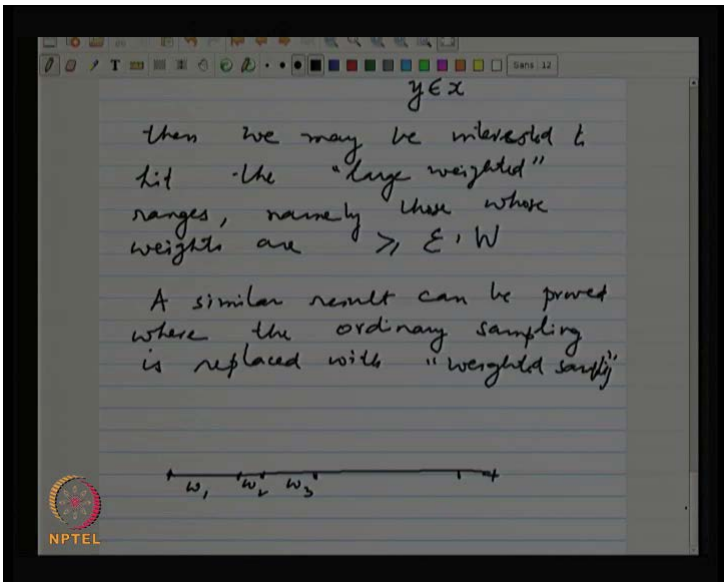
(No audio available: 37:25-37:39)

Suppose I am using the x in too many ways, so the sum of the weights of all the elements in x, right. Then, we may be interested to hit the large weighted ranges, namely those whose weights are epsilon W. So, I will like to hit those ranges that have weights greater than at least epsilon fraction of that, all right. Then, again the same sampling theorem works.

(No audio available: 38:51-39:03)

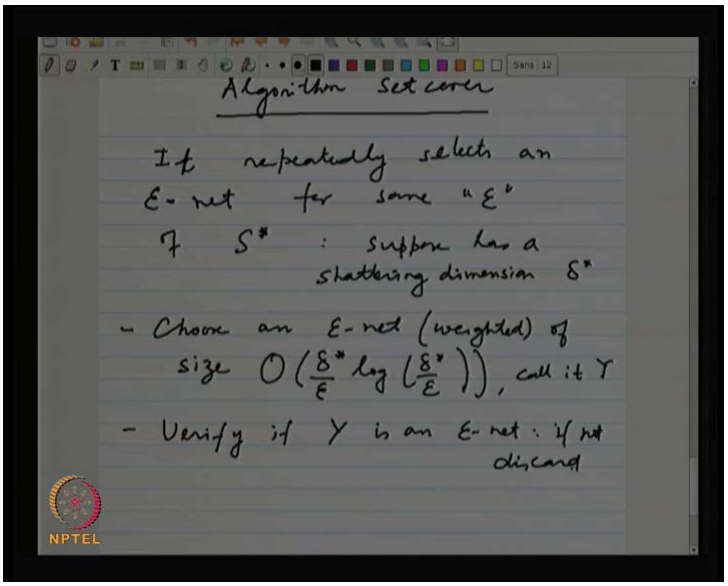
A similar result can be proved, where the ordinary sampling is replaced with weighted sampling. What is the weighted thing? If the weight of a particular element is larger, then the probability of choosing that is small.

(Refer Slide Time: 39:02)



So, we normalize with the respect to the weights and that is the probability of sampling and how do you do this kind of weighted sampling? You know one simple way to do the weighted sampling, suppose I can do uniform sampling in this. So, I take the interval and suppose, I have different, I want to sample elements of different weights, then you just have those you know intervals of different weights. Then, when you sample uniformly in this interval, then the chance of picking an element from a larger interval, sub intervals is more. So, this is the simplest way of doing a weighted sampling. So, you can do weighted sampling also fairly easily. Now, let us proceed to the algorithm.

(Refer Slide Time: 40:44)



(No audio available: 40:33-40:43)

So, what does the algorithm do? It repeatedly selects an epsilon net for some epsilon. I am not defined what epsilon is going to be. What is the problem? The problem is that we have, this is a setting right. We have a set of N points, we have a set of m ranges and we want to pick a subset of the ranges, such that all points are covered. So, what we are going to choose, whatever we are choosing, we are actually choosing the ranges right. So, we are sampling actually from not the range space, but the dual range space right. We are not selecting the points, we are selecting the ranges. We want to choose the ranges. So, although the problem is posed in the terms of S, we are actually going to choose the subset or to choose the sampling in S star.

(No audio available: 42:03-42:14)

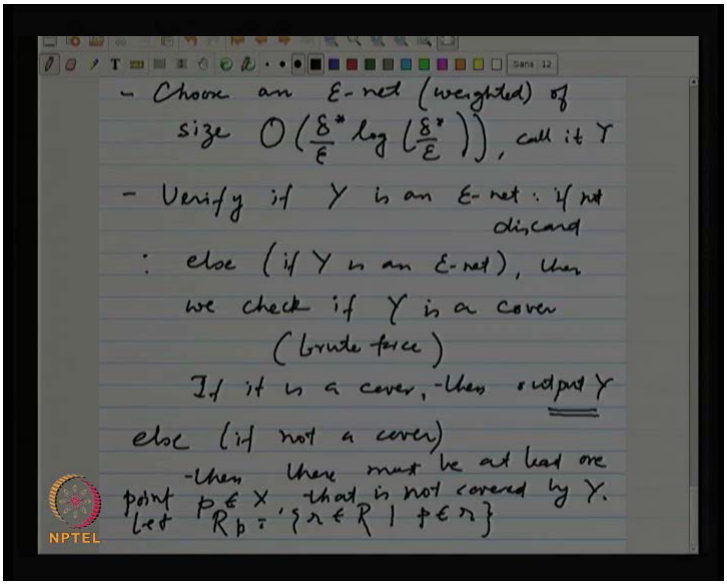
So, the epsilon net is chosen from S star, now because S has a bounded VC dimension. S star also has a bonded VC dimension, all those things. So, we will actually assume, actually S star suppose has S star has a, so I will use the shattering dimension because the sample size is more easily expressed with shattering dimension. So, suppose this is a shattering dimension. Once that has a bounded VC dimension, everything is bounded. That is what we have observed yesterday.

So, suppose the shattering dimension of the dual range space is delta star. So, what kind of epsilon are we going to choose? So, the algorithm is the following, very simple algorithm. Yes it can be used in many other contexts. So, I choose some epsilon sample, where epsilon I am not defined yet what the value is. We will do some back calculation to find out what epsilon should be. I choose an epsilon sample and then, I verify whether or not this is a cover.

If I choose a sample of a certain size, it is going to be an epsilon net with the probability at least half. If it turns out that it is not an epsilon net, we will discard that sample. We will not do anything with it. If it is an epsilon sample, we will first figure out if all the points are covered. We are sampling what? We are sampling the ranges. We will check actually if all the n points are covered by the epsilon net that we have by random sampling. So, the epsilon net first consider random sampling, but we just actually, let us say a brute force check, if every point is actually covered by the epsilon net.

Now, if it is covered, then we have a cover and the cover will be of the size of the epsilon net. So, we output that cover, right. So, let me see what I am saying. So, choose an epsilon net and let me point out that, it will be weighted epsilon net. We will see why it is weighted? Initially, all weights are 1. So, what is the formula for the sample size? It is delta star over epsilon log of delta star over epsilon. So, this is chosen epsilon net weighted of size or call it say, some Y. So, we have chosen Y, we have chosen by random sampling. We verify first of all, if Y is an epsilon net. If not, discard. So, I am not interested if it is not an epsilon net.

(Refer Slide Time: 45:40)



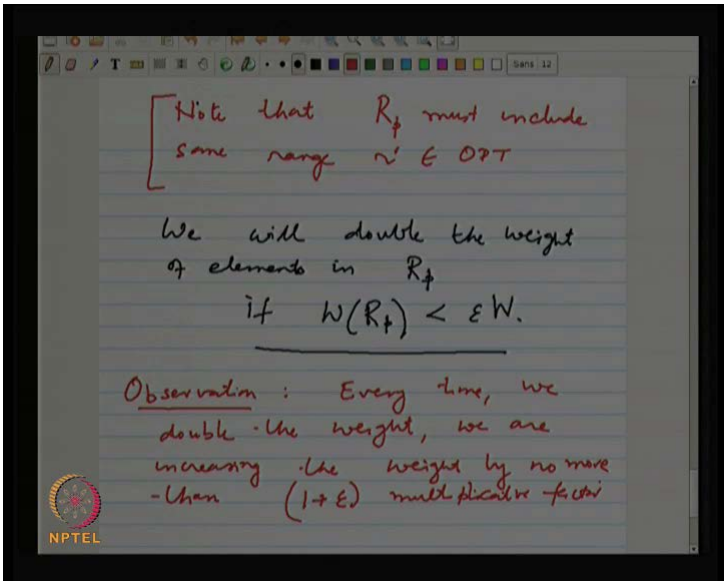
Now, if it is, so if else, then we check if Y is a cover and this cover we check with the brute force ok, whatever time it takes. There n points you know and this cover size and suppose, I can do the test in constant time or something, it will take some n times of the size of the epsilon net brute force or if you have some clever method, use that. Feel free, so it is a cover. So, if it is a cover, then output Y.

How would a cover, so what kind of size Y has, we know this. So basically, now we get something of the size of the epsilon net, that is the size of my cover and see this one does not have anything other than that the property of the range space, right. Delta star is the property of the range space and epsilon is what we have not defined yet, so I will define epsilon also, but this is free of those n. So, there is no dependence on n really. It only depends on the property of range space, although I have not defined epsilon yet. Epsilon

will have some kind of dependence on the size of the optimal set. We will soon get to that.

So, then output Y and you are done. Else, if it is not a cover, what do we do? Then, there must be at least one point p in x that is not covered. At least one point is not covered. What we do is you look at all those ranges. Let R p is equal to all those ranges, such that p belongs to R, right. So, if I have chosen even one of these ranges, the point would have been covered, but we have not chosen anything, otherwise there is no problem with this. I mean the output Y is a set cover.

(Refer Slide Time: 48:54)



So, there is some points p is uncovered and we are defining the corresponding R's of p and there must also observed that one of the, so note that R p must include some range R prime in the optimum solution also. Well, the optimum solution must cover every point, right. So, there must be some range R prime that must be in R p. Why I am saying that? This is not a part of algorithm. I am just brining something in notice.

Now, what we want to do? We have missed out this point p and we do not want anything else, but just you know (()) sample. You know we will try another sample next time, but then next time also, we will miss p or we will miss something else. So, what do we do? Exactly great. So, what we will do is, basically just increase the weight of all the ranges in R's of p, which includes R prime also, right. So, there is some unknown optimum set

cover which we do not know, but when we increase this weight, we will double the weight of elements in R p right.

So, we will double these weights. There is a certain condition here, let me say what it is. If weight of R p is less than, strictly less than, let us say epsilon W. Can you tell me why I have put down this? So, intuitively we must, we are increasing the weight of all the ranges in R p with that includes R prime also. So, next time we do random sample, we miss some point, again some subset of this optimal cover, the way it will double. So, every time we do not get a cover, we are going to increase, double the weight of some range in the optimum cover and if we keep doing it many times, the optimum cover is increasing its weight and it is going to increase its weight very rapidly. So, at some point, we should be able to basically get a cover. That is a point and the cover is of the size of the epsilon net.

So, we repeatedly do this by reweighting, but I have also added this condition. Can you tell me why?

(No audio available: 51:49-51:58)

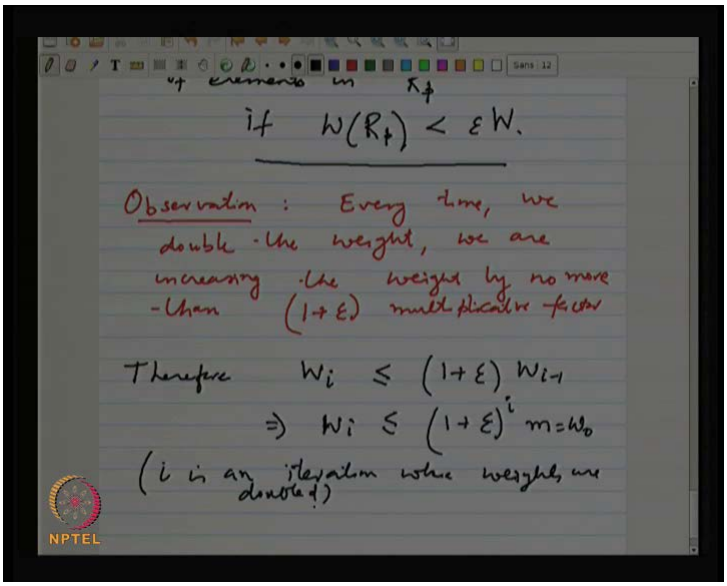
No. See we have retained this epsilon net. Why? Only if it is a yeah only if it is epsilon value. We have retained the subset, why? Only if it is an epsilon net, right. So, if it is an epsilon net, then it suppose to hit all subsets of which has weight at least epsilon times W. Now, if R p had weight epsilon time W, then it would have hit it. Then, there would not have any need for it. So, if we are assuming that the sample that we are retaining is an epsilon net, why is an epsilon net? Then, the weight of that must really be less than epsilon times w.

So observation, every time we double the weight, we are increasing the weight by no more than (Refer slide time: 53:36).

So, this is one kind of bound that we have basically factor of 1 plus w epsilon. We will have to probably wind up here, all right. This weight is being doubled, right. This weight is being doubled only when the, so because it is in epsilon net, the weight of this epsilon net is no more the epsilon W. So, when we double it, it means that we have increased the total weight of the entire system by no more than epsilon W, which means total weight has not increased more than 1 plus epsilon time what was the previous weight was, right.

The weight of the subset is in epsilon W. I am doubling it. It means, that I am increasing the total weight of the system by epsilon w, which means that if you compare W i at the ith time that I double, is less than or equal to 1 plus epsilon time W i minus 1. That is all you are saying.

(Refer Slide Time: 54:55)



So, here is where I will end today. So, if therefore, W i is less than or equal to 1 plus epsilon times W i minus 1, which is basically saying that, you know the same thing as W i is lesser equal to 1 plus epsilon times to the power i of m. That is nothing, but w 0, so this one kind of bound. So, i, by the way is an iteration where weights are doubled, i that it is actually one of the iteration, which were successfully picked an epsilon net. If you do not pick it, then we will forget about it and because we are succeeding with a fairly high product about 50 percent times, so eventually when we count how many iteration's do we need to pick a set cover of this size, we will have to kind of double this in this expectation because in half of the iterations, we are successful anyways.

So, we are only counting the number of iteration where we double the weight, that is simplify the analysis and the total number of iteration will not be more than that 2 times than expectations, right. So, tomorrow I will complete the analysis. So, this is one way of bounding the weight. The other way of bounding the weight will be just looking at the optimum subset. The optimum subset has at least one subset, where the weight is doubling and because the weights are doubling in that subset, it cannot go on for too long

and we will find a lower bound of the weight picked up by that subset. So, then we will be able to just equate that, this must be greater than equal to that weight and that will give you the bound on i. So, we will end here today.