Bandit Algorithm (Online Machine Learning)

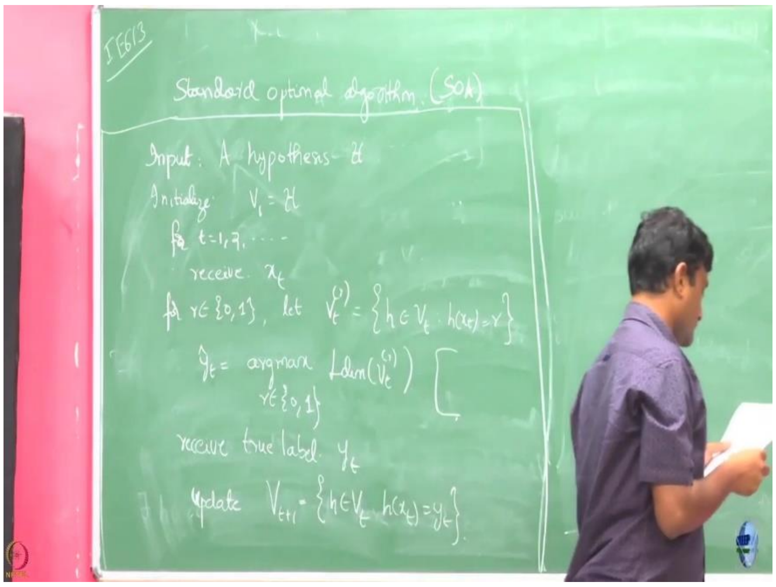
Prof. Manjesh Hanawal
Industrial Engineering and Operations Research
Indian Institute of Technology, Bombay

Lecture - 08
Classification in unrealizability case

Now, we know that the minimum number of errors that can be incurred is L dimension of the hypothesis class. Now, can we come up with an algorithm which also ensure that I do not make no more than this mistakes that is L dimension of h. Is the question clear?

I know that that is unavoidable, maybe in the worst case I have to make L dimension of mistakes. Can I now aim for an algorithm which actually makes no more than this mistake? If I can do that, then that is the optimal algorithm right. It is only going to make the mistake which is unavoidable and it is not going to make any mistakes ok. Now, how to come up with an algorithm, such there exists an algorithm.

(Refer Slide Time: 01:29)



IE613

Standard optimal algorithm (SOA)

```
Input: A hypothesis $\mathcal{H}$
Initialize  $V_1 = \mathcal{H}$
for $t = 1, 2, \ldots$
    receive $x_t$
    for $r \in \{0,1\}$, let $V_t^{(r)} = \{h \in V_t : h(x_t) = r\}$
        $\hat{y}_t = \underset{r \in \{0,1\}}{\operatorname{argmax}} \; Ldim(V_t^{(r)})$
    receive true label $y_t$
    update $V_{t+1} = \{h \in V_t : h(x_t) = y_t\}$
```

So, we will just discuss 1 algorithm which indeed does that job called standard optimal algorithm. So, we just notice that like when I computed the L dimension I computed for the simple case right, where there are only d points there and there are d hypothesis for which it was almost easy to compute.

But I kind of avoided computing L dimension for the case, where it has large number of hypothesis right. So, it is because it needs too bit work out; it is not like as simple as this. I mean finally, when you work out, you will feel that that is not much.

So, what is in general, the case is computation of L dimension is not easy. If I give you arbitrary hypothesis class, it may be pretty computationally involved. How to; so, how. So, I defined the maximal depth d, if there exists binary graph which will be shattered, then I call it as L dimension for that is the definition. But how to compute it right? Like right when you and I computing the simple example, you are thinking why not something if I go further down, take graph of bit more depth will let be shattered or not.

So, we have to exclude those cases right like we have to understand what is L dimension ok, only I can shatter up this point and not beyond this point. So, that all these things could be pretty computationally involved, but when we are going to write on this, we do not care about computational aspects ok. We just care about as long as they exist that is fine and as long as our definitions are consistent and make sense, fine; computationally maybe to compute simple hypothesis, it may be taking on several GPOs, it may be taking 4-5 days, weeks, months, years; I do not care as long as it is well defined and it exists we should be fine.

Now, so this one standard optimal algorithm which we will argue that. So, you see that this algorithm is almost same as halving algorithm, except for the fact that here you are going to compute the L dimension of the hypothesis class corresponding to the one which predicted label 0 and label 1 and then, make your prediction based on that.

Let us understand this. So, at any given time, you are maintaining this hypothesis class and you are going to partition your hypothesis at time t into two parts; one, set of all hypothesis which are assigning label, let us say r equals to 0 and another set of hypothesis which.

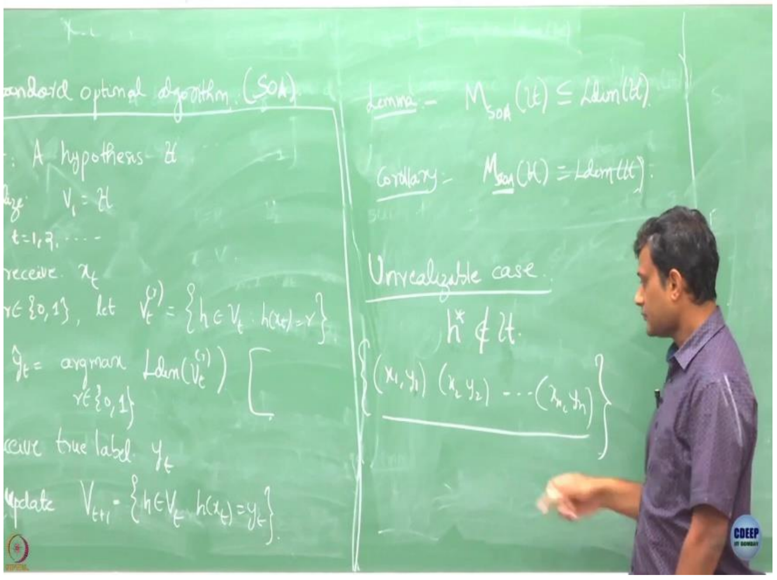
Student: 1.

Is to 1. You have two hypothesis class there. So, one set saying 0, one set saying 1. In the halving algorithm, what we did? We just set a label which corresponding to the one which has the largest number of hypothesis in this right. But here, instead of that, we compute the L dimension of the hypothesis class of both my hypothesis class and then, take the one which has the largest L dimensions here and then, go into predict according to this.

So, in all this case, it may happen that sometimes the two hypothesis class I have here, they met we may end up with the same L dimensions for them in which case, we are going to make the break the try arbitrarily. We are going to say if they are both of them same L dimension, you are going to just say either 0 or 1 ok. Either you toss a coin or say that are its up to your description whatever you want to do.

And then, you receive y_t and then you are going to retain only those which are making a correct prediction on your correct predictions on your given observed point; rest, you are going to throw away. You can show that this algorithm is optimal, in the sense I am just going to state this.

(Refer Slide Time: 08:34)



andard optimal algorithm (SOA)

: A hypothesis $\mathcal{H}$

ize $V_1 = \mathcal{H}$

$t = 1, 2, \ldots$

receive x_t

$r \in \{0,1\}$, let $V_t^{(r)} = \{h \in V_t : h(x_t) = r\}$

$\hat{y}_t = \underset{r \in \{0,1\}}{\operatorname{argmax}} \; Ldim(V_t^{(r)})$

ceive true label y_t

pdate $V_{t+1} = \{h \in V_t : h(x_t) = y_t\}$

Lemma:- $M_{SOA}(\mathcal{H}) \leq Ldim(\mathcal{H})$

Corollary:- $M_{SOA}(\mathcal{H}) = Ldim(\mathcal{H})$

Unrealizable case

$h^* \notin \mathcal{H}$

$\{(x_1, y_1)\ (x_2, y_2) \ldots (x_n, y_n)\}$

That is even this algorithm it is guaranteed that it is not going to make no more than L dimension number of errors that means, what I know that this SOA is also lower bounded by this. So, it is going to make the minimal number of mistakes ok.

So, I am just; so, as a corollary to this as corollary, I can based on my lower bound I can actually say that this SOA is actually equals to L dimension of H for this particular algorithm ok. I will just the proof is also straightforward, but I will skip that part.

So, finally, we are completing the loop for the case of realizability condition, when the reliability assumption holds right. We have first showed that for the realizability assumption, what kind of bounds we can get. Then, we asked for what is the best bound, we can get. Then, we establish that what is the minimum bound, we can expect right that we shown through a lower bound. Then, we are showing that that lower bound is achievable through this one.

So, this is the best like any algorithm, this is the best. It can guaranteed; it cannot guarantee anything more than this, when the realizability assumption holds ok. Now, why should realizability assumption hold? Ok, yeah. So, that is what like I am I have skipped the proof of this.

We are just saying that if you are going to if you are going to choose this $\hat{y_t}$ based on the 1 which has the largest L dimension, you will be ending up ensuring that you will be not making this many number of mistakes, fine. What happens if there is no fixed hypothesis in my class, according to which the labels are generated? Ok. Now, what will be my strategy and what will be my evaluation criteria? Ok. So, first if there is no hypothesis in my hypothesis class according to which my true labels are generated, can you guarantee of finite bound on any of your algorithm?

Like whatever algorithm hypothesis you are going to choose, it is not necessary that the labels are going to come from that, it could be coming from labels could be coming from some hypothesis which is outside your hypothesis class. So, because of that there is no way like you can guarantee that you will stop making mistake after some point. You may still want to guarantee that you want to be still want to minimize the number of mistakes you are going to make. So, in that case now we have to see what kind of evaluation criteria, we are going to set up and how we can minimize that; how we can do best on that.

Now, we are looking to unrealizable case. So, by just unrealizable, I am saying that there a hypothesis generates the labels need not be in the my hypothesis class H. So, that is this h^* which regenerates my label, need not much hypothesis class H.

So, then in this case, now the first question is how we are going to set up our evaluation criteria? How I am going to evaluate myself? How good I am doing? All I can do is while learning, I can only use hypothesis that are coming from this hypothesis class ok; but the true labels could be generated from your hypothesis which is outside, then how I am going to measure my performance against?

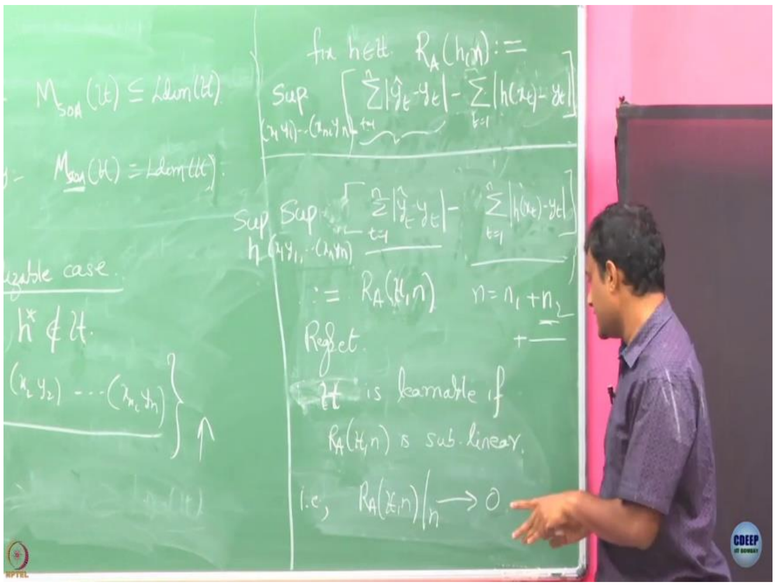
So, earlier when h^* was included in my hypothesis class H, I aim to quickly identify that and always play that. Identify quickly that and always play that right; if you can quickly identify, always play that you are making no mistakes. Now, anyway that h^* is not there in my hypothesis class, now what I would like to do is what is that single hypothesis in my hypothesis class, if I keep to playing throughout that would have given me the smallest number of mistakes. I do not know a priori which is the one which is going to give the minimum number of mistakes a priori; but I want to do as close as to that ok. So, let us put it.

Suppose, I kept on telling you in round 1, this was x_1 and this was corresponding label y_1 in the x_2, y_2 like this and these labels are generated by this h^*; but this h^* does not belong my capital H. Now, all you can use this hypothesis from this hypothesis class H right. Which one among this you would have liked to apply on this, let us say you have after observing all of this?

You will take first take one hypothesis from this and apply it on all x_1, x_2s thing and see at how many point it made mistakes? That is the number of mistakes, you get for the first hypothesis and you can do it for all the hypothesis and you will see some someone which has made the smallest number of mistakes. You would have like to ideally start applying that one right from the beginning right because that is the one which is. So, here now you have the advantage of hindsight. You have observed all the things and at this point, you are saying what is the best I should have done.

Now, I am going to take that as my evaluation criteria. I am going to take that hypothesis which has done best on this hypothesis class is my benchmark. But I a priori do not know that right because I have not seen this. I am in an online setting; I am not in a batch setting. If I start from if I start making predictions how much errors I am going to make compared to that hypothesis which is the best in hindsight ok, that we are going to express as follows.

(Refer Slide Time: 16:05)



$M_{SOA}(\mathcal{H}) \leq Ldim(\mathcal{H})$

$M_{SOA}(\mathcal{H}) = Ldim(\mathcal{H})$

izable case.

$h^* \notin \mathcal{H}$.

$(x_2, y_2) \cdots (x_n, y_n)$

fix $h \in \mathcal{H}$ $R_A(h, n) :=$

$\sup_{(x_1 y_1) \cdots (x_n y_n)} \left[\sum_{t=1}^{n} |\hat{y}_t - y_t| - \sum_{t=1}^{n} |h(x_t) - y_t| \right]$

$\sup_{h} \sup_{(x_1 y_1), \cdots (x_n y_n)} \left[\sum_{t=1}^{n} |\hat{y}_t - y_t| - \sum_{t=1}^{n} |h(x_t) - y_t| \right]$

$:= R_A(\mathcal{H}, n)$ $n = n_1 + n_2$

Regret.

$\mathcal{H}$ is learnable if

$R_A(\mathcal{H}, n)$ is sub linear.

i.e, $R_A(\mathcal{H}, n)/n \rightarrow 0$

Fix first hypothesis. So, what is the first part here? $\hat{y}_t$ is what you predicted. y t is what? Been revealed. So, $|\hat{y}_t - y_t|$ is the mistake you are going to make right; there is a binary setting. So, over t rounds, so this is the mistake you have made.

Suppose, you happen to apply the same hypothesis h in every round; by applying this hypothesis h, this is the number of mistakes you have made ok. Now, this is basically you are comparing yourself against the fixed hypothesis h ok. Now, what I will do? I will take the worst of things. I am going to because I do not want to be dependent on a sequence right, like I do not care about which particular sequence I have been observing. I want to do this guarantee for all possible sequences.

So, that is why I am going to take this supremum over all of this ok. What is this now? If I have to play a single hypothesis throughout my interaction with the environment, this is how the that is going to give me this much of errors, where all possible sequence and this is what I would have incurred according to applying my own algorithm. So, $\hat{y}_t$ is given by your own algorithm. This is what you are going to incur if by your algorithm, this is what you are going to incur by playing a fixed hypothesis.

Student: Which gives the minimum number of mistakes?

No, I do not know, whatever. I do not know; I am not saying that gives me the minimum number of mistakes. It is a just one hypothesis from that.

Student: Belonging to H.

Belonging to H and let me call this is as $R_A(h, n)$. So, this R your algorithm A which is going to predict $\hat{y_t}$ in round t and in n rounds, this is what you are going to incur.

So, you think it like this, suppose you give me a sequence that you are you have been faced. So, let us say when you are going to face an environment, this is the sequence you observed. In that, this is the number of mistakes you incurred; this is the number of mistakes incurred by hypothesis h and this is the difference between you and the hypothesis h. Now, I am looking at the maximum difference of this.

Student: What does the $\hat{y_t}$ that we have got?

Yeah.

Student: That is also we have got from some hypothesis.

No, this is by your algorithm you are going to use one of the hypothesis in each round right, you are going to, but what you are going to do this your $\hat{y_t}$ is coming from this right in your SOA algorithm, it is coming like this. In halving algorithm, it is coming in a different way. Wherever it is, I am just going to that is why I am just writing it has $\hat{y_t}$ and I have now looked it a worst case how is?

Student: It is dependent on?

This one, yeah. Because I am looking at a number of rounds right, I have been I said that my interaction is for n number of rounds and my sequence length is also n here that is what the x_1, y_1 all the way up to x_n, y_n; I have looked at n interactions. If we are going to change n, the summation will also be larger ok.

Now, what I want to now look into this is I am going to think, I am going to now call it as the regret I have for not using hypothesis h all the time. This is what you get, the number of mistakes you incurred. Had you played h? This is what you would have gotten. You are compare yourself with this. So, you are basically saying how well, I am doing or how I am doing compared to the case where had I played h throughout. My benchmark here is playing the same hypothesis throughout.

Now, as you said this hypothesis is here need not be the best one ok, I do not know which one in my hypothesis class does the best job ok. I want to now compare my performance against that best hypothesis. So, how to take that into account? Now, I am going to define my regret as supremum over of this (Refer Time: 22:57). So, this is the regret I am going to incur if I have been playing h right.

If I have played a good h, this quantity is going to be smaller right because if I am using a good h, it is going to make a smaller number of mistakes. So, that is why I am taking supremum over all h ok. So, alternatively, you can just before writing this step, I am going to do this. So, what I am doing? I am looking instead of this particular h, I am looking for a h which minimizes this quantity. In a way, I am looking at my best hypothesis which gives me the smallest value right.

So, if I right here this h is now a function of this this sequence right the best one, but I wanted to it too. Now, define it in this fashion. So, this is minus of in right; if I pull this outside this, it will becomes sup over h. But now, if I write sup over h here, this h is still a function of the sequence, the best sup here ok. So, I but I do not want it to be depend on h. So, the way we are defining it is, by just defining it as like this.

So, fine whatever you can ponder on this finally. But are you convinced just the benchmark, we are setting is the single best hypothesis against that I would have like to use compare to how much I am going to incur, when I am using my algorithm? I am just comparing whatever algorithm I am going to use, how it fares against a hypothesis, which does the which gives me smallest number of mistakes. I do not know what it is, but a priori; but I want to perform as close as possible to that one.

And this is characterized this is capturing that and we are going to call this as. Now, because we are taken sup over h, now this is only function of n here. Suppose, my realizability condition holds ok, what would be this term here? It is going to be 0 right because there exists one h^* which is always going to make a prior prediction and this term is going to be 0.

But now, I do not know; I have removed that condition right because of that this guy need not be know necessarily 0. The even the best guy I have in my hypothesis, it is going to possibly make mistakes; but fine that is what I can afford too and I am going to compare my performance against the best I could I can afford or I have access to and this we are going to call it as regret.

In this setup, we will define keep defining different notions of regret throughout. But in this case, this is how we are defining our regret. Did I make any such assumption in defining this? I did not make any such assumptions here or if the finite hypothesis class is not finite, does anything break down here in my definitions; is everything still make sense?

As long as these things are consistent and not violating any of our consistency checks, that is fine. So, here nothing is being violated ok. Now, find this is what your you have defined your regret to be; this is your performance criteria. This is what you are going to set your criteria evaluation criteria as. Now, what you want? What you want this regret to be? Small, large? Small. How small?

Student: 0.

0. Can you guarantee 0?

Student: Yeah.

Why; yeah?

Student: (Refer Time: 28:39).

No, only it can be 0 is.

Student: (Refer Time: 28:46).

This quantity is.

Student: (Refer Time: 28:49).

For the best hypothesis I have, I am doing similar to that. Now, we get to the notion of learnability here. Remember how did you define learnability in the realizable case, we said that as long as the number of mistakes I can bound by some finite number; then, we said that hypothesis class is learnable by algorithm A. What is the notion of learnability here? Ok.

We say that; so, actually I should also write this as H. This is now you are learning this hypothesis class right yeah. So, I am just also it is important to mention what is the hypothesis class, you were learning. So, this is $R_A(H, n)$.

Now, we are going to say that. So, we are going to say that this hypothesis class H is learnable, if my regret is sub linear. And what is the definition of sub linear? If you are going to divide this quantity, the regret $\frac{R_A(H,n)}{n} \to 0 \ as \ n \to \infty$. What is that ratio telling $\frac{R_A(H,n)}{n}$? So, R_A was regret over n round right, when you divide it by n that is regret per round.

If you are learning; that means, your if you are figuring out which is the best hypothesis in your hypothesis class, you should be eventually make lesser and lesser mistakes right and you should be start doing as good the best hypothesis class in your hypothesis class would have done. If you do this, you should be incurring kind of the same amount of mistakes that the best hypothesis class in your best hypothesis class has made.

Because of that, maybe as for some n this could be some finite number, but if you are going to start looking at that difference beyond some number that should be smaller right ok. Let us say this is for n right; let us split it into two parts. I am going to now take this n as $n_1 + n_2$; first n_1 rounds and subsequent into n_2 round.

Maybe the initial n rounds you do not have any clue, what is happening; you will be some making mistakes. But you are kind of figuring out what is happening, but later, in the next n_2 rounds, you have kind of figured out what is happening, you expect yourself to make lesser mistakes right and maybe like after further rounds, you accept to make lesser and lesser mistakes. Because of that if you look into this summation here, maybe for the first entrance this is some number; but in the next n_2 rounds you expect this deference to shrink and also further shrink right as n goes.

So, that is why if you are really doing figuring your algorithm is smart and it is figuring out what is happening, it is eventually going to as n increases later; in the subsequent part, it is not going to incur too much of penalty here. It will be doing as good as the best hypothesis class in your algorithm would have done.

So, because of that, we if this condition holds like you will see in this case, we were on an average when n is large, you were doing as good as your best hypothesis in your hypothesis class would have done. Is that do you understand like by letting this n go to infinity, this going to 0, it exactly means that.

So, this is why we call if this condition holds, this hypothesis class is learnable. Because if you give me sufficient number of rounds, I am kind of figuring out what is the best hypothesis and I am incurring lesser and lesser mistakes. So, you will see like as I said you will be seeing different notions of regret as we go along this course.

Now, we are our next step is to see what kind of bounds we can give on this, what kind of guarantees we can give and always, we will ask the question is this the best guarantee I can give ok, like the way we did it for the realizable case. So, note that this, if I can come up with an algorithm a such that this condition holds, then it is called learnable, fine.