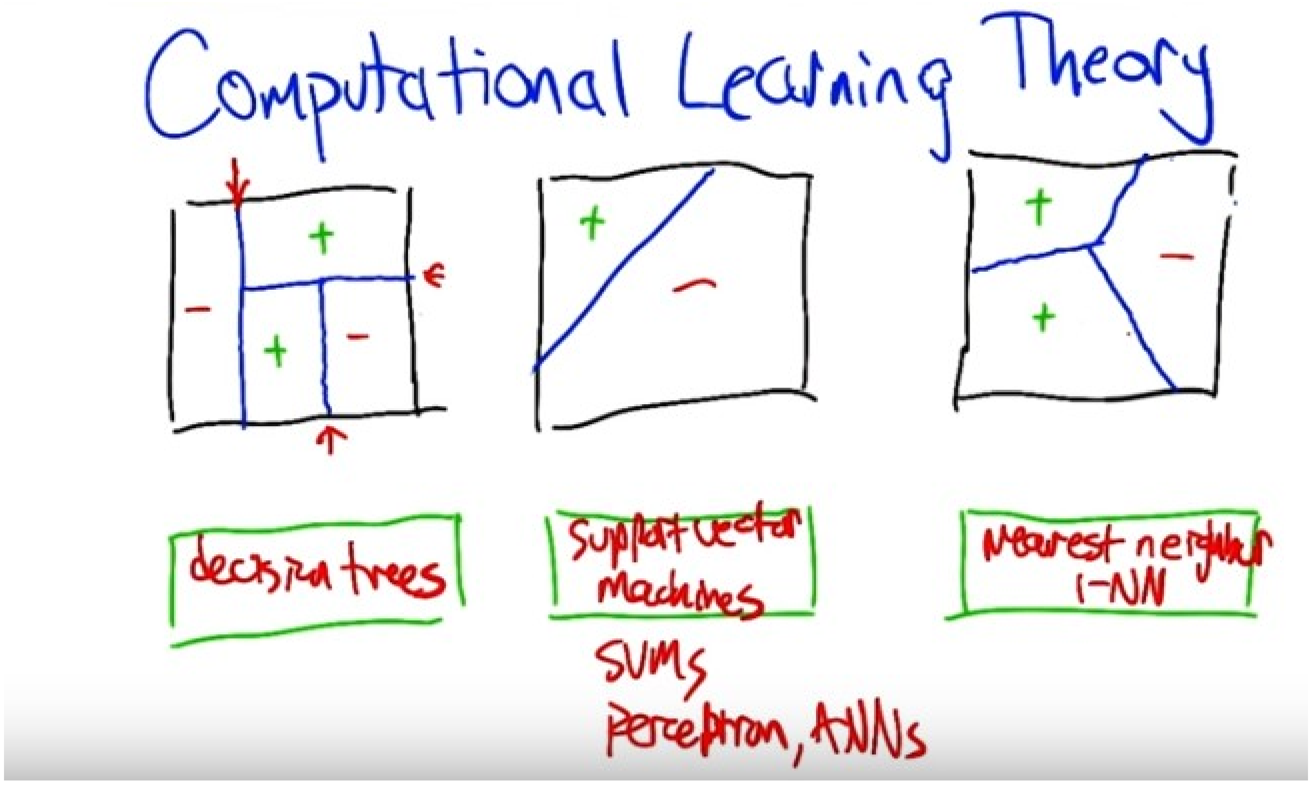
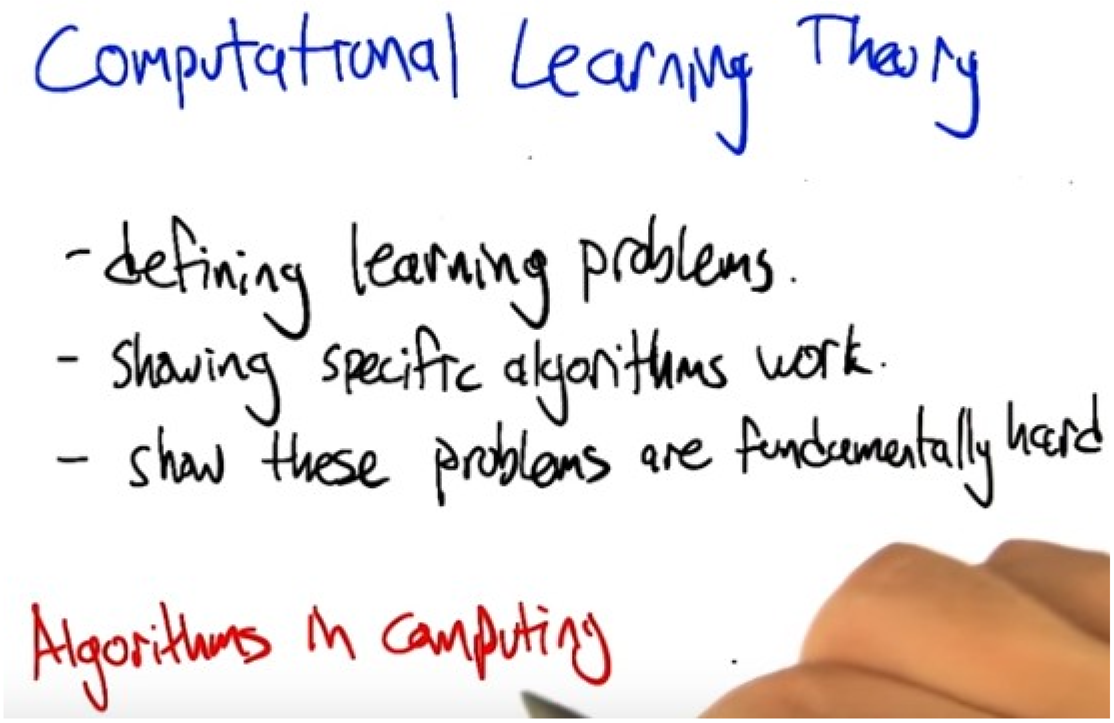
本 pdf 文件 和 下一個 pdf 文件(SL8) 之目的都是想回答這麼一個問題:
How many data do we need to do a successful learning?
本 pdf 文件是 討論 finite hypothesis space 之情況,
下一個 pdf 文件是 討論 infinite hypothesis space 之情況

(本段就是個 quiz, 所以圖在後面)
1. Hey, Charles. >> Oh, hi Michael. >> It's funny running into to you here. >> It is. It's always funny running in to you over the interwebs. >> So, today, I have the pleasure of telling you about computation learning theory. >> Hmm, it's my favorite kind of theory. >> [LAUGH] Well, sure. Now how do you like theory in general? >> I am happy that there are peolpe that do theory. >> There we go. Alright. And now you are about to be one of those people, at least for the next you know, hour. >> No, no. I am not doing theory. You're doing theory. I'm listening to you do theory. >> No. I am coaching you through it. Don't you understand? It's about, it's about the learning process. Learning about learning. >> Fine. >> So let's start out with a quiz. No wait. So let me let me at least set the stage as to why, what you know, what it is that we're talking about today and how it's different from what we've talked about, in the previous days. So mostly what we've been talking about up to this point, is, algorithms for doing learning. Alright we talked about what, what learning, the machine learning field was like. And we talked about a number of specific algorithms for building classifiers, and building regression. And there's a problem with that. Specifically, I have this, this feeling that it's very, very necessary to always make sure you know what problem you're solving, before you start proposing algorithms for solving it. And we haven't really nailed down what exactly that problem is. >> Hm. >> And that makes things hard. It makes it hard to know, for example, whether or not one algorithm's better than another. >> One algorithm's better than another, if it works better. >> If it works better? Yeah, well, you know, that's not wrong. But I think it still is not a very helpful definition for designing better algorithms. >> Okay. >> So so we're going to talk about this computational learning theory. And I want to say that it's not about particular algorithms, it's about some other stuff. But I thought it would be helpful if we started out by saying. well, what is it that we talked about so far? So one of the things we talked about so far is algorithms for doing for learning classifiers. So, I wanted to draw a picture of that, and then I realized maybe that would be actually a useful quiz. So, if you can image each of three boxes is the output of a classifier in a two dimensional space. So it's a two dimensional input space. We've ran a leaner, and when it spits out a classifier, this is how it classifies the the regions of the space. So I, I use blue lines to separate the square into regions. The two dimensional space into regions, and then I labeled each region with a minus or a plus, so you can tell what the classifier was doing. >> Okay. So I'm wondering if you could fill in underneath each one, what learning algorithm was probably used to, to find this classifier? >> maybe. >> You want to give it a shot? It's not, it's not that important to this lecture, but I thought it might be helpful in just kind of setting the stage. >> Okay. I'll give it a try.



2. All right, Charles welcome back, so let's let's see what you've come up with. >> Okay, so I would choose for the second one support vector machines, and the reason why is because there seems to be a single line that seems to be in the middle between where the plus and the minus are, so it looks like it's trying to maximize the margin. >> Yeah, though, those aren't training points. I'm just labeling the regions. >> Oh never mind. Well then I totally misunderstood. [LAUGH] to do. I stand by it. >> Yeah, I think it, I think that's a perfectly fine answer because the sup, support basic vector machine is going to find a linear separator. And so the output, the classifier that comes out of it is going to separate any kind of region into pluses and minuses, with some lines separating them out. That's also true of perceptrons, it's also true of certain kinds of you know, simple neural nets. So all those seem like they would be reasonable answers for the middle one. >> Oh okay well, then given that, I'm going to say the third one is a nearest neighbors method. >> good. And like what nearest neighbor do you think? >> K nearest neighbor? [LAUGH] >> Sure, I was thinking one nearest neighbor. >> Yeah I was too actually, believe it or not. >> One nearest neighbor. So we, I don't think we got a chance to draw one of these diagrams when we were talking about nearest neighbors. But this is called a, >> Voronoi diagram? >> Voronoi diagram! Right! Because what, what's happening is, it's breaking up the plane into the regions that are closest to some particular points. >> Right. >> So you give it a set of points, it breaks up the plane into the things closest to that. And in this particular case our We had, we probably had three points. And one was labeled plus, and another was labeled plus. One was labeled minus, and so it break the space up into these sections that way. So this, this should. Voronoi diagrams can be very aesthetically pleasing I don't think I drew this one particularly well. >> I think it's perfect. My eyes are

closed. I think it's perfect. Given that, that I'm going to say the first one is decision trees. >> Very good. Because decision trees split things you know, if you're talking about in, in a, in a two dimensional space it's going to split at the top node in one dimension. And here it looks like that's this split. Then maybe, looks like on the left side that's a minus. On the right side there's another split. On the left side of that it's a plus. On the right side there's another split into pluses and minuses. You get these sort of interesting nested rectangles. They also can be very pretty. >> Right. >> Very Mondrianish. Yeah, very good. So this is not what we're going to talk about today. >> Okay, good.


Computational Learning Theory

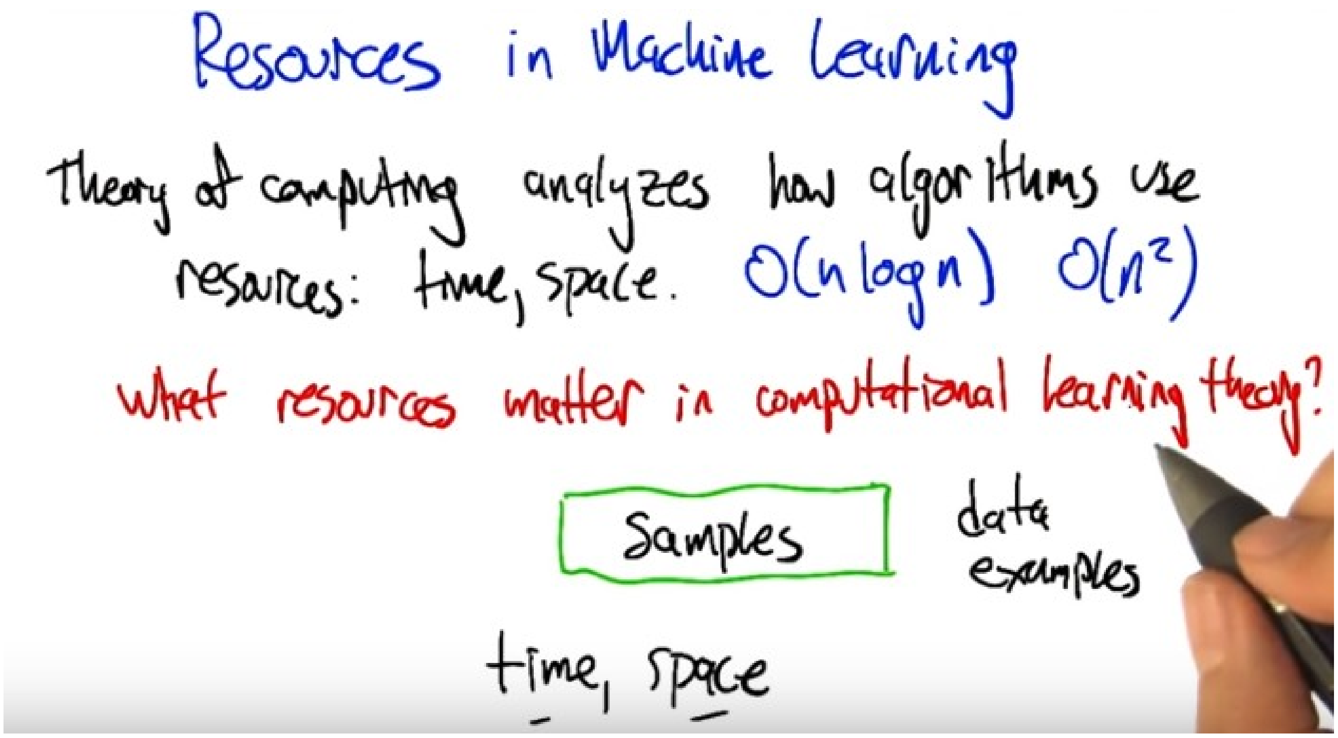
- defining learning problems.
- Showing specific algorithms work.
- show these problems are fundamentally hard

Algorithms in computing

3. >> All right, so that quiz maybe was ill-placed, in that it was about what this is not about. What this is about is computational learning theory. And computational learning theory really gets us a formal way of addressing three really important questions. One is, what's a learning problem? Let's, let's define very carefully what it is that we want a learning algorithm to do. If we can do that, we can actually show that specific algorithms either work or don't work, with regard to the definition of the problem. And maybe we can even come up with algorithms that solve those problems better. So that's kind of on the upper-bound side. And then on the lower-bound side we can also show, for example, in some cases that some problems are just fundamentally hard. So you, you define a particular learning problem and you discover. Wait, the algorithms that I'm thinking of don't seem to work. You might actually be able to show that no algorithms, say, no algorithms in some particular class are ever going to be able to solve them because, that problem is not solvable by problems in that class. So those problems are fundamentally hard. So, answering these kinds of questions require that you be fairly careful about defining things and using mathematical reasoning to, to determine what's going on. So we're going to focus mostly on that, talk about some algorithms that are not necessarily practical. You wouldn't necessarily want to use them, but they do help illuminate what the fundamental learning questions are and, and why certain algorithms are effective and ineffective. >> Okay, so Michael, so can I ask you a question then? >> Sure, please. >> So, it sounds to me like you've just justified this in the same way that a computing theoretician might try to justify theory. Are they related? >> Right. That's a very good observation. In fact, the, the kinds of analyses and tools that are used for analyzing learning questions are also the same kinds of tools and mechanisms that are used in the context of a-, analyzing algorithms

in computing. So yeah, that's exactly right. >> Okay, great. >> In fact, let's let's, let's leverage that analogy. And we'll do it in the context of a quiz.

4. >> So often in the theory of computation, we analyze algorithms in terms of their use of resources. And it's usually time and space. So like, when we talk about an algorithm running in say n log n time, we're saying something about this, the time that it takes. So if you say that it uses n-squared space, we're talking about the, the amount of memory that it takes up as the function of the growth of the inputs. So what sort of resources do you suppose might matter in the context of computational learning theory? So we're just, we're trying to select among algorithms. We want algorithms that use their resources well. What source of resources would we analyze? If there's multiple possible answers, just fill in one of them. We'll just see if it's any of the ones on our list.

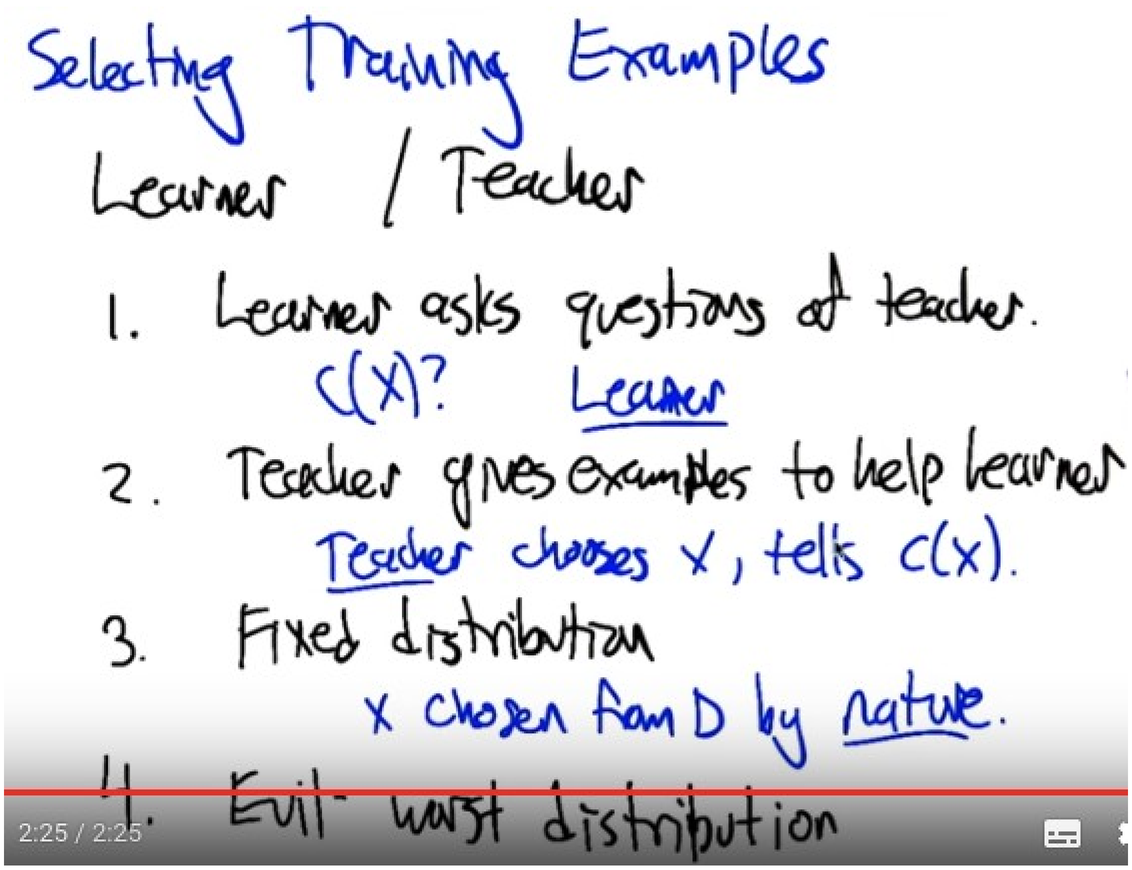

Resources in Machine Learning
Theory of computing analyzes how algorithms use resources: time, space. $O(n \log n)$ $O(n^2)$
what resources matter in computational learning theory?
Samples
data
examples
time, space

5. All right. So Charles, what do you think would some reasonable resources to try to manage in a learning algorithm? >> Okay, well I was thinking of three. Because three is my favorite number. Two of them are what you already have written up there. Time and space. After all, at the end of the day, it's still an algorithm. We need to be able to analyze algorithms in terms of time and space. >> That's right, so if in particular we have a learning algorithm and they, they do more or less the same things, but one runs in n-squared time and the other runs in exponential time, we'd really like to have the one that runs in the shorter amount of time. Or, in particular, if there's a, if we define a learning problem and we say well, We could do this computation, but its MP hard. Then maybe that's a problematic way of defining the problem. So yeah, time definitely matters. Space for the same reason. And those are the same things that happen in in, or that are relevant in regular algorithms. What about anything specific to the machine learning setting? >> Okay, so that was my third one. So The only thing that matters in machine learning or the most important thing in machine learning is data. So I would think that another resource that we care about is the data and in particular the set of training samples that we have. >>

Yes I like the, I like the word samples. Though data is probably pretty good. Thing to stick in there as well or examples. Those should all be okay. >> Yes. Indeed. Yeah. We, we want to know, can we learn well with a small amount of samples. In particular if, our learning algorithm works great in terms of time and space, but in order to run it you actually have to give Examples of every possible input, then that's not going to be a very useful algorithm. So, the fewer samples that it can use, the more that it's generalizing effectively, and the better it is at learning. >> Oh, that makes sense.

6. >> All right. So we should take a moment to define inductive learning. So inductive learning is learning from examples. It's the kind of learning problem that we've been looking at the whole time, But we haven't been very precise about all the various quantities that we want to be able to talk about. So the number of properties that we actually need to be thinking about when we go and define what an inductive learning problem is, and measure what an inductive learning algorithm does. So one of them is just a simple thing like, what's the probability that the training is actually is going work. You know, whatever it is that it's trying to produce, it may be that in some cases, because the data is really noisy or just you know, got a, got a bad set of data, it might not actually work. So, we generally talk about a quantity like $1 - \delta$ as the probability of success. δ here obviously is a probability of failure and this is just 1 minus that. There's also issues like the number of examples to train on. How many, how much data does the does the algorithm get to see? . Is there a letter you like for that, Charles? >> no. >> Okay. >> Is there a letter you like? >> I don't know, sometimes I like M for number of samples. But I thought maybe you went, you would want N because, you know, things tend to grow with N. >> Yeah. I did want N, but then I thought well, we can't use N because we use N for everything else. >> [LAUGH] Fair enough. There's also something that we really haven't talked about yet, but you could imagine that the complexity of the hypothesis class might matter. Why, why do you think that could come into play, Charles? >> Well, if you don't have a very complex hypothesis class, then there's some things, well, do you mean the complexity of the class or the complexity of the hypotheses in the class? >> That's a good question. Do we mean the complexity of the class or the complexity of the hypotheses in the class? Well it depends on how we measure complexity. But the complexity of the class could be like the sum of the complexities of all the hypotheses in the class, so it could be both. >> Hm. Well if, if you mean, you know, a hypothesis class is complex if it has very complex hypotheses, then you can say, well, if you have a hypothesis class that can't represent much, then it will be hard for you to, well, represent much. It will be hard for you to learn anything complicated. So that could matter. >> Sure. That's right. Now, could you see a downside to having a complexity class, I'm sorry, a hypothesis class that is very, very complex? >> You mean like my daughter? sure. I think you could, it would be much easier to overfit. >> Right. So getting something that actually works well might be challenging. You might need a lot more data to kind of nail down what your're, what you're really talking about. So it's a bit of a double edged sword. All right. So then, there, another issue is, well, you know, it may be easy to learn if you don't have to learn very well. [LAUGH] So the, the accuracy to which the target concept is approximated, often written as epsilon, is another thing that's going to be important in understanding the complexity of a learning algorithm. And and so those, those are kind of the main complexity related quantities that I, that I wanted to talk about. But there's also some choices as to how the learning problem is actually framed. There's the manner in which training examples are presented. And there's the manner in which training examples are selected for presentation. And we're going to talk about both of these. Let me just first say that when I talk about the manner in which training examples are presented, there's, there's two that I think are really, really important to look at. One is batch and that's mostly what we've looked at so far, that there's a training set that's fixed and handed over to the algorithm in a big bolus, right. A big group. >> A big batch. >> A big batch, exactly. Or it could also be presented on line, so one at a time. So we say to the training algorithm, or the learning algorithm, here's an example. And then it has to predict what the label is. And then the algorithm can say, oh here's what the right label is. Let's try again. Here's another example. And it can go back and forth like that.

>> Mm-hm. >> We haven't really talked about algorithms that work that way, but it is useful in the context of computational learning theory to have both kinds of algorithms. They have different sorts of behavior. All right. So let's talk about the manner in which training examples are selected.



Selecting Training Examples

Learner / Teacher

1. Learner asks questions of teacher.
   c(x)? <u>Learner</u>
2. Teacher gives examples to help learner
   <u>Teacher</u> chooses x, tells c(x).
3. Fixed distribution
   x chosen from D by <u>nature</u>.
4. Evil- worst distribution

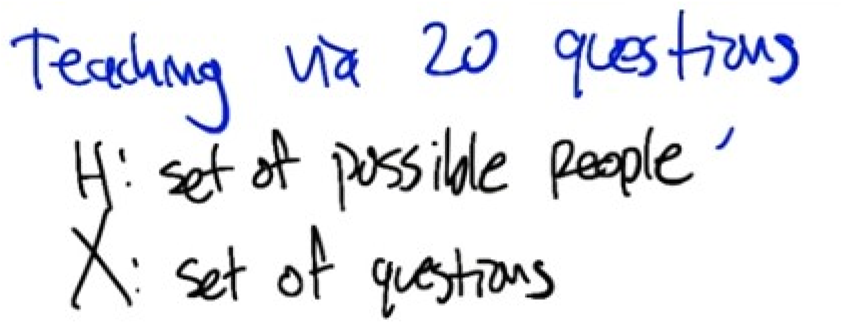
2:25 / 2:25

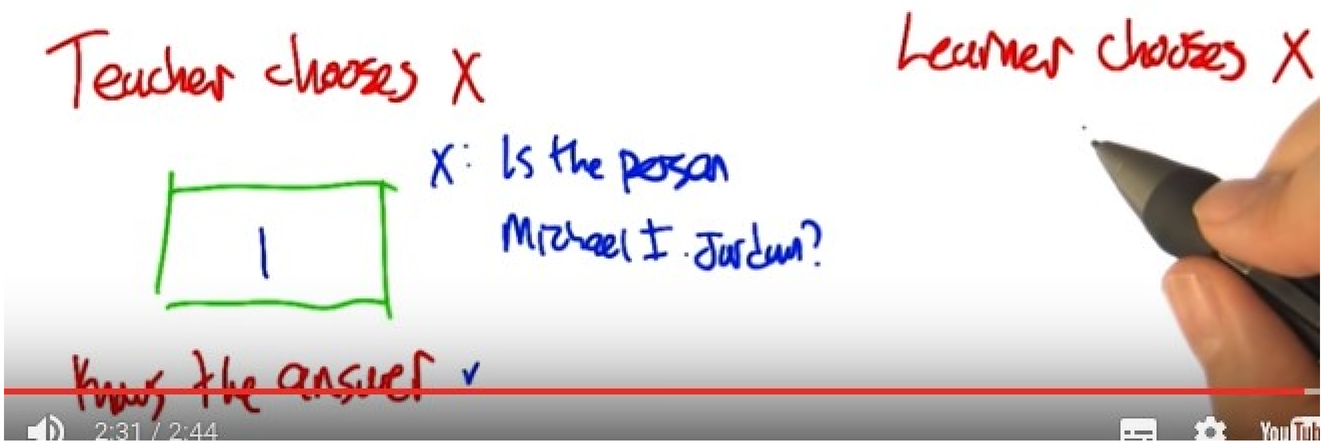
上圖中 x chosen from D by nature 中的 D 是 distribution 的意思.

7. >> All right, so turns out it, it matters how we select training examples when when a learner is needing to learn. So let's at least articulate some various ways that training examples could be selected. And then for each one we might end up with a different answer as to how much training data is going to be needed, if the training examples are selected in that particular way. So, so what do you think, Charles? Are there, there any ways you can think of for selecting training examples? >> Well, so, I, I, I so here's how I think about it. So, you keep using the word learner, and when there's a learner there's often a teacher. So, I'm trying to think about this in the different ways that learners and teachers can interact. So, I'm going to try to think about my, my experience as a professor. So, there are a couple. Here's one, one is: sometimes the learner asks questions of the teacher. >> I see, so in this case, the learner would be selecting the training examples. The learner would say, I, you know, here's, here's an input x, would you please tell me c of x? >> Right, that's exactly what I mean, right, right. Then there's another case. So the learner asks questions of the teacher. But sometimes the teacher just goes ahead and gives x, c(x) pairs. >> And let's say it's, it actually is trying to help the learner. >> Sure. >> So it's a, friendly teacher. >> Okay, I've had some of those. >> Good, all right so, the, it could be the, the learner asking to try to get data. It could be the teacher asking to try to lead the learner to something

good. And, anything else? >> Well, so, I like this, this way of thinking about it. Because it sort of makes sense in my world as a professor, but it doesn't actually sound like either of things is what we've been looking at so far. What I think we've been looking at so far is that somehow the Xs and the CXs, the training examples, just come out of nowhere. They come from some underlying distribution. It's just nature. It's just coming at us from some process that we don't know what that process is. Yeah, I like that. So in some sense there's three individuals that could be asking for examples. There's the teacher, there's the learner, and there's, like, the world around them. And, and, the things could come from any of those. I guess another possibility would be that questions are asked specifically in an evil way. All right, so all of these different kinds of distributions or, or ways of selecting training examples are actually going to come into play in the stuff that we're talking about. Let's let's use the first couple to get a sense of why these things might be different from each other. So let's go back to this notion of 20 questions. >> Okay.

8. >> Right, so let's go back to, to 20 questions that we talked about a number of lectures ago. And we're going to consider the following analogy. That in, in some sense, what is a, what is a learner? What is an inductive learner trying to do? It's trying to find the best hypothesis in some class H. And the hypotheses all map input to yes or no. Let's say that what we're trying to do in twenty questions is, deal with our hypothesis class H, which is maybe a set of possible people. And so, these are the possible answers to the 20 questions problem. And, X is the set of questions that you can actually ask. So X, they're kind of like training examples. That's how the learner's getting information about what the right answer is. But no question, in and of itself, might actually be sufficiently revealing of what the hypothesis is. All right does that, does that analogy make sense to you, Charles? >> Makes perfect sense to me. >> All right, good. So, now lets think about two different cases. One where the teacher is going to try to select which questions to ask. So the teacher's going to say to the student, here's the question you should ask me next. To try to figure out which person it is, as quickly as possible. And then we'll, next we'll look at what happens if the learner's the one asking those questions. So in some sense, it doesn't seem like it should make a difference, because, you know, in either case, the learner is going to ask the question and the teacher is going to answer truthfully. Just in one case, the learner has to come up with the questions, and in the other case the teacher has to come up with the questions. So why, why are these different from each other? What does the teacher have that the learner doesn't have? >> Well, the teacher actually knows the answer. >> Indeed. Alright, so let's let's just do a quick quiz and say, how many questions are necessary for a smart learner to figure out the right person, assuming that the teacher is the one who gets to feed the learner good questions? >> Okay. >> Go.

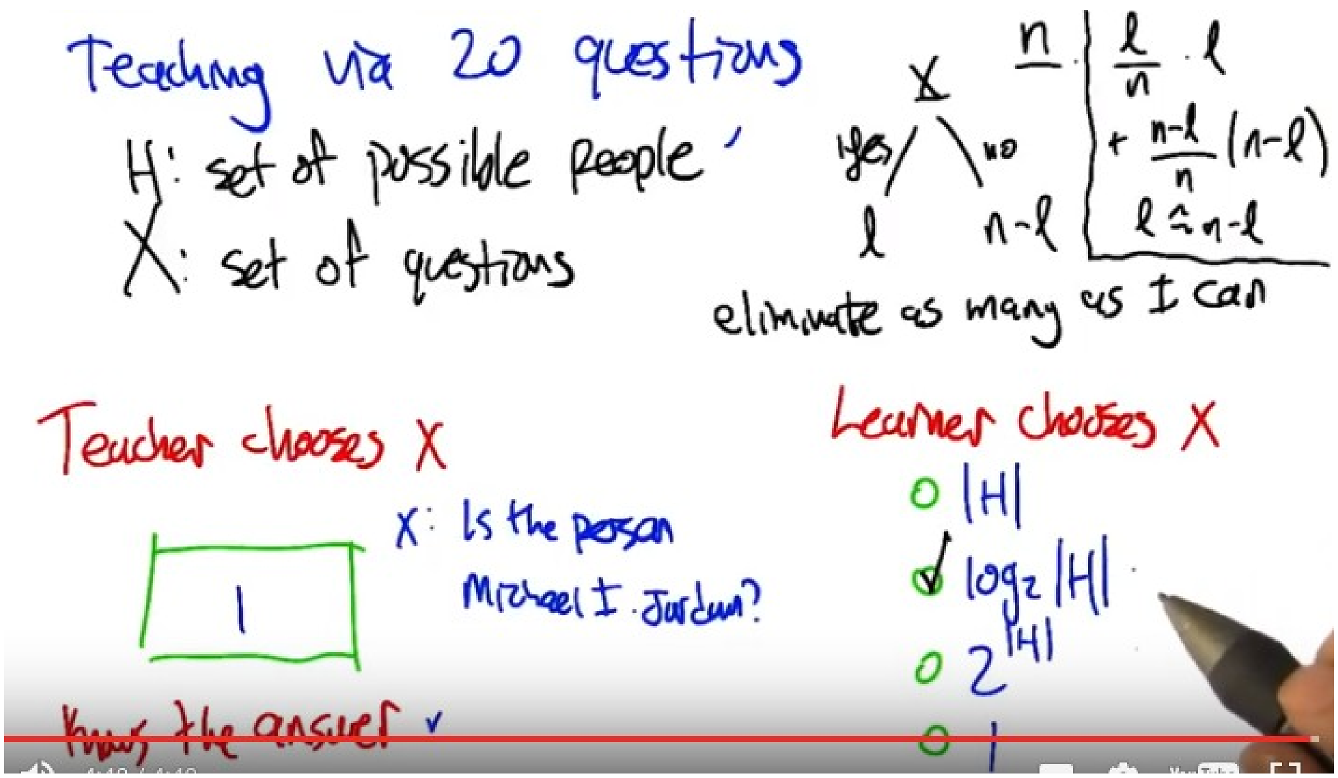




9. Alright we're back. So let's, let's, let's think this through, so the teacher gets to suggests to the student any, any question. The teacher knows the answer, and the teacher would like the learner to get the answer as quickly as possible. Not requiring the full 20 questions if, if possible. So what do you think, what do you think the teacher could do as a good strategy here? >> For 20 questions? >> Yeah. >> So, if, if we think of the set of possible people as being these sort of hypotheses. And you know which of the hypothesis is correct and which is wrong, then presuming the teacher can keep all this in her head, I guess the right thing to do would be to always pick an X, such that it gets rid of as many of the hypotheses is possible. But what, what, do I mean by that? I mean, it gives the most information. >> Well, but there's a, there's a very strong kind of information that you could get in this particular case. So, so, so let's imagine that this set of questions is, is this sort of, you can ask any question in the universe. >> So, I give you some question, I oof. >> Let's, let's try this because remember we did this before, you thought of a person, and you had me ask questions. So I want you to think of a person. >> Okay. >> Alright, and I want you to tell me what question I should ask, that's going to figure out who the person is as quickly as possible. >> Okay. Is this person as good in his field as his namesake (From wiki: A namesake is a person named after another. Namesake may also refer to a thing, such as a company, place, ship, building, or concept, named after a person.) is in his field? >> Is it Michael Jordan? >> Yes. >> Is it the Michael Jordan who's a professor at Berkley? >> Yes. >> Okay. >> You could have asked if I say, is it Michael I Jordan? >> Yeah, I don't think I knew the initials of the two

Michael Jordans. >> Yeah. Michael Jordan, for, for people who don't know is a professor at Berkley and one of the giants in machine learning. And he is literally the Michael Jordan of machinery. >> [LAUGH] Alright, but, but, but you could've done this in one question, right? If you had to ask me, if the first question was, is the person, and then the actual person. You knew that the answer to that would be yes. And so I would've gotten it in one step. >> Hmm, that's true. >> So, given a, you know a, a sufficiently, helpful teacher. You can, you can do this in one. >> Well, that make sense, right? So basically, if my questions include, is this the correct hypothesis, then you should always just ask that question. >> Right. >> But that seems, that seems like cheating. >> [LAUGH] Yeah, I would, I. You're right and all I was really trying to do is show that, it's a very different problem, when the, the question has to come from the learner's side. That kind of question from the learner's side would be pure folly. Because it would, it, it, the learner would not know what person to pick. If the person started, if the learner started to ask questions like, is the person Michael I Jordan? The answer's very likely to be no. In which case that's not very helpful. Now, a really helpful teacher could potentially change the target. But that's, that's definitely cheating. >> Mmm. OK, fair enough.

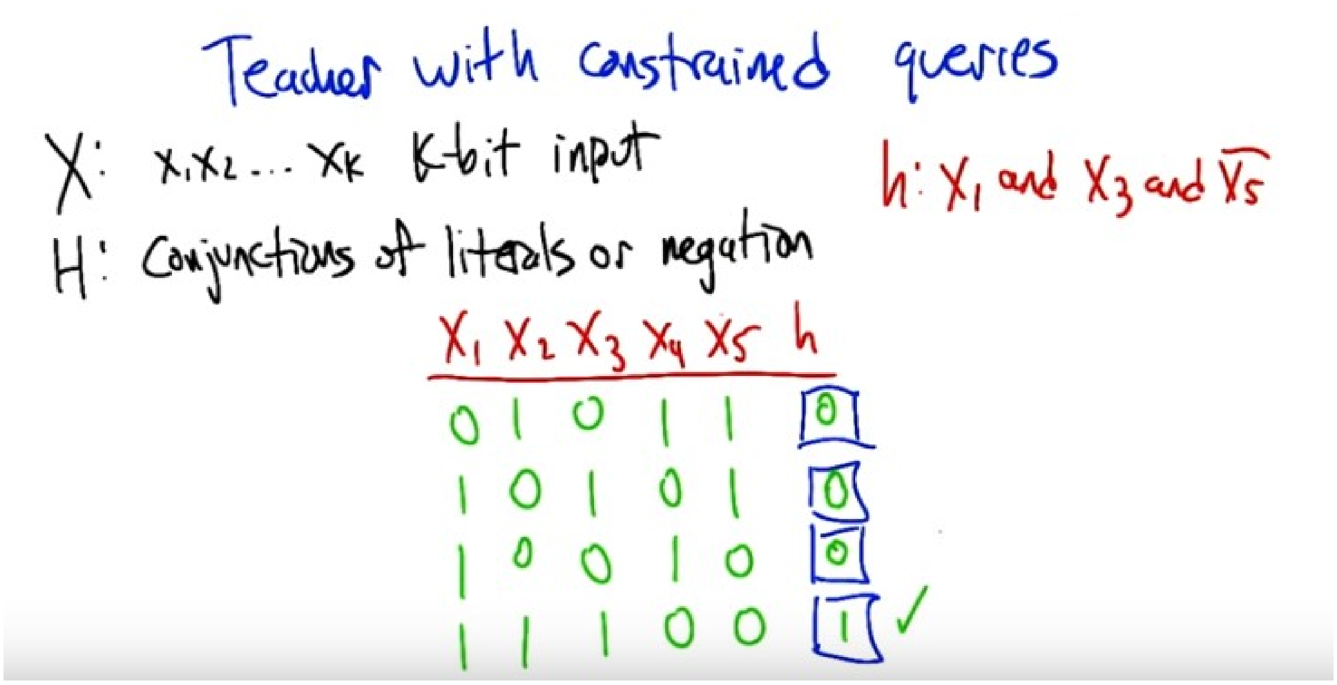
10. All right. So the learner, if the learner is choosing which questions to ask the learner is in a distinctly more difficult situation because it doesn't actually know what the right answer is. So whereas the teacher could ask questions there were maximally informative. That is to say, whittle down the hypothesis set to one in a single question. The learner doesn't, doesn't have that ability. The learner can't know which, what whether the question that's being asked is going to have a yes or no answer. So again, assuming that the learner can ask any question it wants, but doesn't know what the right answer is. How many questions is it going to need to ask before it can it has whittled down the set of possible people to just the one that is the right answer? So I gave four helpful formulae here. All in terms of the number of people that we have to choose from. Is, do we have to ask questions that is the size of the set of people? Is it going to be more like the log of size of the set of people? Since there's, you know, lots of difference ways that the person can be chosen. Do we have to look at two to the number of people? Or again can the learner be as powerful as the teacher, and get the whole thing in just a single question? Alright, so, is, is, is this a little clearer Charles? Can you, can you work this one through? >> Yeah. I think I can work this one through.


Teaching via 20 questions
H: set of possible people
X: set of questions
x
yes/ \no
l   n-l
$\frac{l}{n} \cdot l + \frac{n-l}{n}(n-l)$
$l \hat{=} n-l$
eliminate as many as I can
Teacher chooses X
X: Is the person Michael I. Jordan?
1
knows the answer
Learner chooses X
○ |H|
✓ $\log_2 |H|$
○ $2^{|H|}$
○ 1

右上角框中的那幾個式子不重要, 可以不看.

11. Well so, there's a couple a ways to think about it, but let me just kind of go top down. So, here's what I'm thinking. I think the same principle applies, where you sort of want to ask the question that gives you the maximum amount of information. And from the teacher's point of view, I might be able to pick a question that gives you the answer right away, depending upon what h is and what x is. But from the learner's point of view, I sort of can't. So what does the learner know? >> So, when we ask a question, x, it's going to either have a yes answer. Or a no answer. And let's say that at any given point in time, we have n possible people that we're trying to reason about. And then after we choose x, after we choose the question, if the answer is yes then we're going to have, I don't know, some other number of people. Let's say l. And if the answer is no then it's going to be n minus l, right? And so you said that in the case of the teacher, pick a question so that this yes is going to whittle it down so that this is just going to be the one right answer. >> That's exactly right. That, that's assuming that x is set up in such a way that you could ask that kind of question. >> Basically a question could be of the form, is it this or this or this or this or this? Any pattern of answers that we want, we can construct the corresponding question. >> Okay, that makes sense to me and I don't think it changes, my thought process, so, so, that's good. So I always want to ask the question that's going to give me maximum information. I'm the learner. I don't know, everything like the teacher does, but what I do know are all of my hypotheses. Alright. So I know how each hypothesis responds to each of the questions. So, in general, it seems to me that I should try to eliminate as many hypotheses as I can, okay? That makes sense? >> Yeah, I'll write that down. I like that. >> Now, you could say, that's what the teacher did. But, let's take what the teacher did, the teacher knew that there was a hypothesis where it would whittle it down that, since the answer was yes, it would whittle it down to one. But if the answer had been no, it would have only gotten rid of one. So since the teacher knew the answer the teacher could pick that question. But as a learner, I don't know the answer, so finding questions that whittle it down to one if the answer is yes aren't very helpful because the answer might be no. >> Alright so how many. Let's see so under the assumption the target hypothesis was chosen from H say, uniformly at random. >> Right. >> What's

the expected number of hypotheses we can eliminate with a question that has this form? >> Well, it's binary. So if you assume that the hypothesis covers every, the hypothesis base covers everything. Then the best you can do is eliminate about half of them, on average. >> Well, so, but in particular, if you have a question like this, x. That, you now, you ask it when there's n possible people. And then, after you ask it if the answer is yes, you're at l possible people. And otherwise, you're at n minus l. Which branch are we going to go down? >> Well, we don't know. It could be either one. >> ha. But do, can we put a probability on it? >> Yes if it covers everything. It's about a half and a half. You said it was uniform, right? Uniformly chosen? >> The, the right answer, the, the target person is uniformly chosen. But this question isn't, isn't necessarily of that form, right? This, this question could be very skewed, it might only have one thing on the left. And n minus 1 on the right like you said. >> Right, well then I, that's not a very good question to ask. >> No, that's right. But I wanted, I want to be able to distinguish good from bad questions by saying we want questions that, that, that are going to leave us with the smallest expected size set. >> Oh, well then since all possibilities are there, the smallest the, the best you can hope for is about a half. >> Yeah, and in particular, eh, with this, this particular question's going to do, is it's going to have a probability of n over l of going down the left branch and getting an l. And it's going to have a probability of n minus l over n of going down the right branch and getting n minus l. And so that could be one way of actually scoring this, this quesiton The question that has the best score, as you pointed out, is going to be one where l is about the same as n minus l, in fact, half would be a really good choice. >> Right, I thought that's what I said. >> Yeah. >> Oh, okay, good. >> But, but I just wanted to write down the formula. >> Okay, that makes sense, that makes sense. Okay, no, it's always good to write down the formula. Okay, so you want to pick questions that roughly split things in half. And if you can do that every single time, then every single time, you will split the set in half. So you'll start out with n, then n over 2, then n over 4, then n over 8, then n over, and eventually you'll get to n over n which will give 1. So you'll only have one, possible answer. And so that'll take you a logarithm amount of time. >> Right, so the number of times you can divide a number in half before you get down to one is exactly the log base two. So if we start with size of h hypothesis, it will take us like log base two to whittle it down to a single question. This is a lot larger. I mean this is a nice small number, but it's a lot larger than one, right? You know, one, one is really great. This is you know, exponentially, no, this is much, much worse than one. But but it's still really good. You can, you know, it's not considering all the hypothesis. It's very cleverly whittling it down so that it only has to look at the log of that. >> Right.



Teacher with constrained queries

X: $x_1 x_2 \ldots x_k$ k-bit input

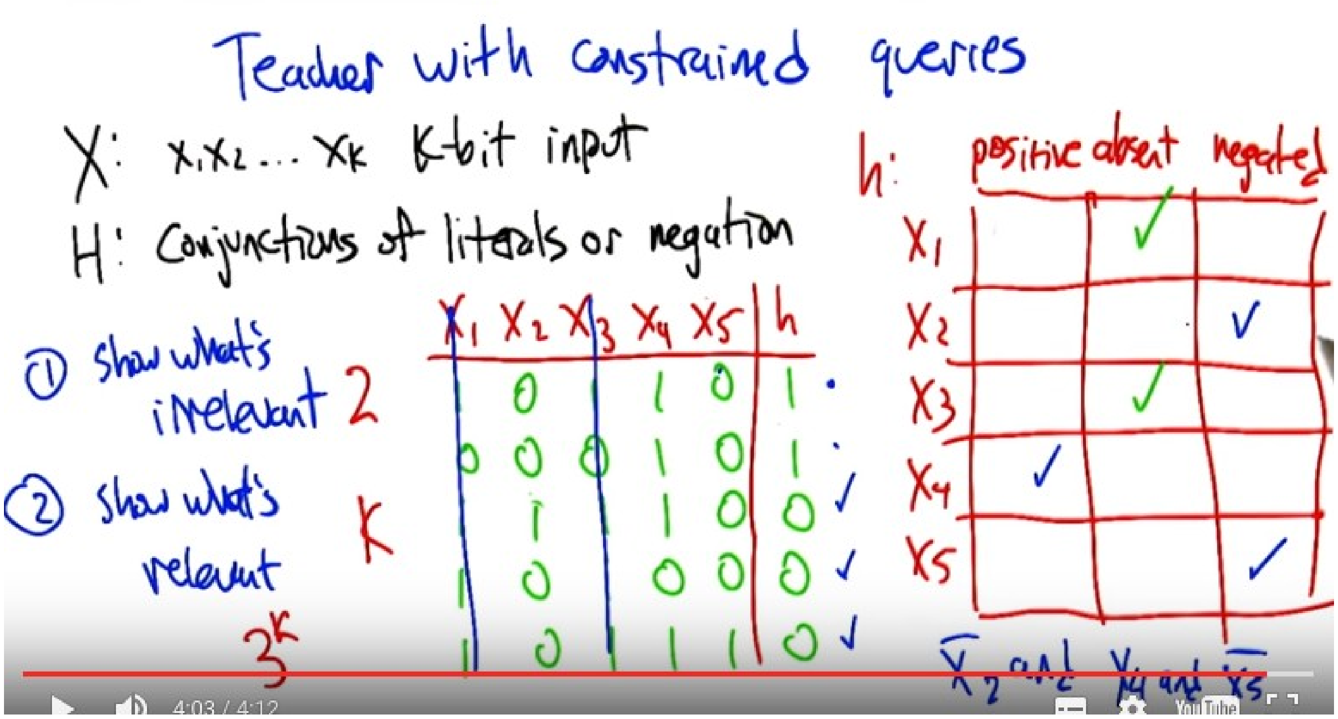
H: Conjunctions of literals or negation

h: X_1 and X_3 and $\overline{X_5}$

X_1	X_2	X_3	X_4	X_5	h
0	1	0	1	1	$\boxed{0}$
1	0	1	0	1	$\boxed{0}$
1	0	0	1	0	$\boxed{0}$
1	1	1	0	0	$\boxed{1}$ ✓

「x1 and x3 and not x5」就是一個 hypothesis.

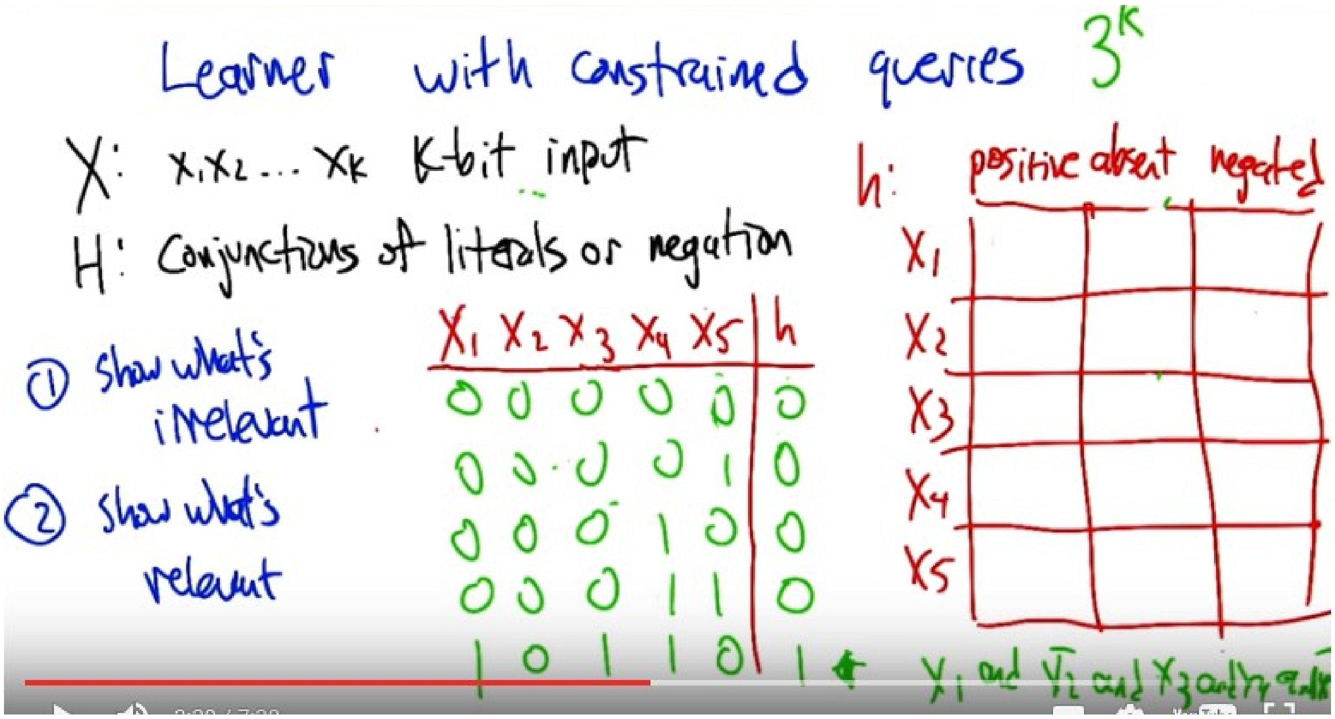
12. >> All right. Now, you might get a misleading impression from what I just presented there, because I did this sort of odd trick that's not really very kosher. Specifically, when I gave the example of a teacher asking questions to to lead the learner to a particular hypothesis, I allowed the teacher to ask any possible question in the universe of possible questions. Right, so the teacher could construct a question, and in this particular case, that, that specifically had a yes answer for the target hypothesis. And a no answer everywhere else. And that just simply doesn't happen in any realistic learning questions. So we need to think a little bit more about the question of what happens when the teacher is constrained with respect to the kinds of queries that it can suggest. So I'm going to, I'm going to set up classic computational learning theory hypothesis class, and we'll go through some examples about what happens when the teacher's constrained in this way. So let's imagine that our input space is basically k-bit inputs, so x1 through xk, and the hypothesis class [INAUDIBLE] and literals or their negation. So here's a, here's a concrete example. Here's a hypothesis that is in this set. X1 and x3 and not x5. We're going to connect together some of these variables, not necessarily all of them. And some of them can be negated. And, it's all the connectors have to be ands (注意沒有 or). All right, so Charles, let's make sure that that you understand this. So let's let's look at, some inputs. Alright. So let's say our input is this. X1 is 0. x2 is 1. x3 is 0. x4 is 1. x5 is 1. What is this formula going to evaluate in this case? >> Okay so, a 0 is false and 1 is true I assume, because, that's what we do in these things. X2 and x4 don't matter. I just have to look at x1, x3, and x5. So, x1 had better be true, and it isn't, so I already know the answer. >> It's false. Very good. All right, let's let's try a couple more of these. >> Okay. So, I would do the same thing I did before. I would look at x1, x3, and x5. So in this case, x1, cleverly, I've noticed this 1 so, that doesn't give me the answer right away because you're an evil teacher. x3 is also 1 but x5 is 1, and according to the hypothesis it has to be 0 for this to be true, so 0. I can do the same kind of thing for the third line, so x1 is true like it needs to be, but x3 is false instead of being true so also 0. And then in the bottom, let's see. X1 is true, which it needs to be. X3 is true, which it needs to be, and X5 is false, which it needs to be, so the output is 1. >> Very good. >> All right. I got it.


Teacher with constrained queries
X: x1x2...xk k-bit input
H: Conjunctions of literals or negation
① Show what's irrelevant 2
② Show what's relevant k
3^k
x1 x2 x3 x4 x5 h
h: positive absent negated
X1
X2
X3
X4
X5
X̄2 and X4 and X̄5
4:03 / 4:12

13. >> All right, so here, here is, here's a table of examples with the input pattern and the actual output pattern of the hypothesis we're looking for. But I'm not going to tell you the hypothesis, you have to figure it out. So the way you're going to do that is, by figuring out for each variable, does it appear in the conjunction in its positive form, in its negative form or it's not there at all? For example if you want to say the hypothesis is X1 and not X5, you would write something like X1 and not X5, and the other ones are all absent. And just to be clear Charles, I, I've picked this particular set of examples so that, you particularly, Charles Isbell PhD, would have the best chance of figuring out the hypothesis with the smallest number of examples. >> I appreciate that. >> Go.

14. >> So, what, how do you start with this now, Charles? >> Okay, so the first thing I'm going to do is, I'm going to look at the first two examples (即 rows). And the reason I'm doing that is because I know they both generate true. And so I'm going to look for variables that are inconsistent. So if I look at X1, for example, it's a one in the first case, and a zero in the second case. So it can't be the case that it's required in order for this to be true. And the same would be true for x3. So I'm going to say that neither x1 nor x3 matter (注意題目要求連接的只能是 and, 不能有 or. 如果可以有 or, 則 x1 可以存在, 比如 (1 or x1), 不管 x1 為 1 還是 0, 此式都為 1). By contrast, x2, x4, and x5 all have the same values in both of those cases. >> So we don't know much about them quite yet. >> No >> But let's so let's, that seems very well reasoned. So we know that x1 can't be part of this formula and x3 can't be a part of this formula. So, let's just sort of imagine that they're not there cause they don't really give us any information any more. >> Beautiful. >> Alright, So what's left? >> So what's left is now to make certain that x2, x4 and x5 are necessary, and particularly necessary with the, the values that they have. So I guess all I can really do is see if there's anything else to eliminate. If I were just looking at the first two (rows), I would think that the answer was 'not X2 and X4 and not X5'. >> Alright. so, hang on, not X2, X4, not X5. >> Right. So, that's what I currently think it is based upon what I just saw. >> And that would be, that's consistent with the first two examples. >> Right. And so, now I want to make certain is consistent with the next three examples. This is the easiest way for me to think about this, anyway. So, let's see. Not X2, X4, so that should be false, which it is. They're all false, so let's see not X2. But x4, up, that should be false, which it is. And then I do that same thing. But wait, why isn't that

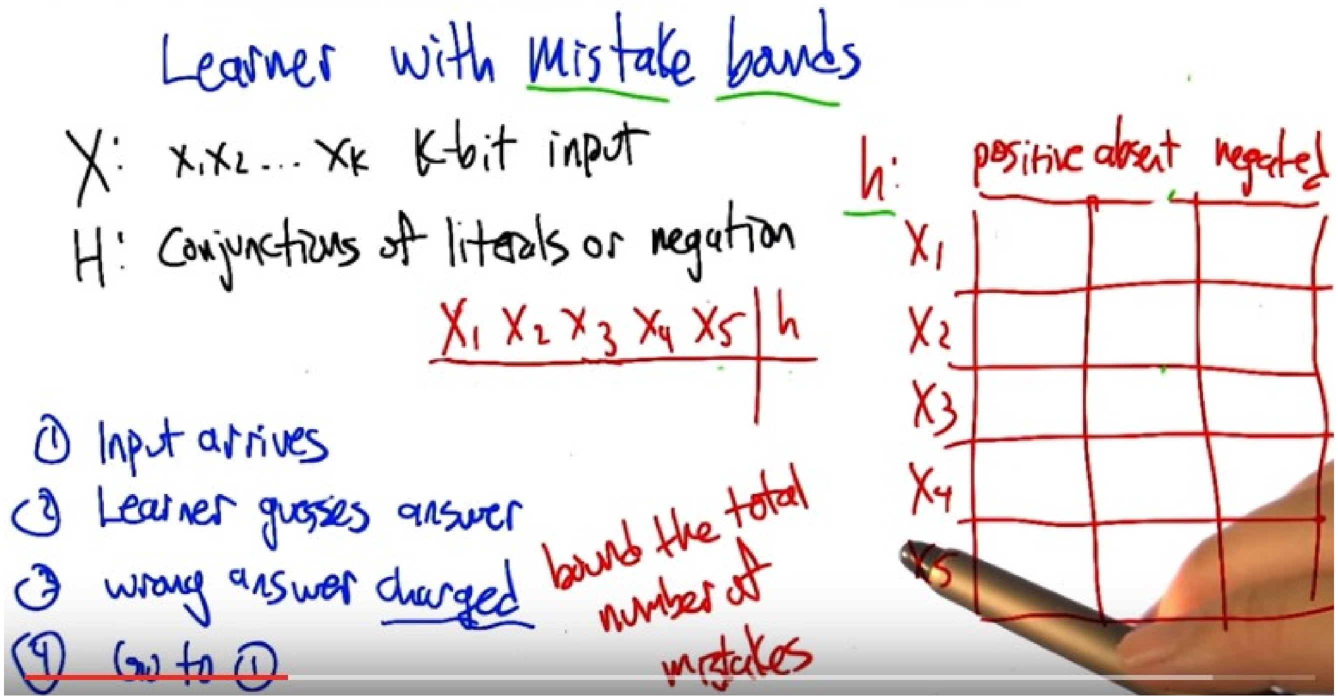
the answer? That can't be the answer. >> Is the answer you got it. >> Huh I got it right. >> So the thing to notice is that in these first two examples we have X2 is false X4 is true and X5 is false and that's enough to make the conjunction true but making, flipping any one of those bits (x2, x4, x5) is enough to make it false so what I showed in the in the remaining examples is that just by turning this X2 into an X1 (意思是 turning this x2 into 1) leaving everything else the same we lose it (意思是 h 為 0). Similarly if we flip the X4 to zero and leave everything else the same, we lose it. Similarly if we flip x5 to one, and leave everything else the same, we lose it. So that means that each of these (x2, x4, x5) is necessary to make the conjunction. They (x2, x4, x5) are all actually in there. >> That's just what I was thinking. So, in other words, you gave me some positive examples to eliminate things that were necessary, and then you gave me negative examples to validate that each of the variables that I saw so far were necessary because getting rid of any one of them, gave me the wrong answer. >> Exactly. Let's, let's even write down those two steps. So the first thing was show what's irrelevant (x1, x3). And how many questions How many queries might we have needed to show that? >> Well, one per variable. >> Well actually we only need two because what I did is I, I used, all the relevant ones (x2, x4, x5) I kept the same and all the irrelevant ones (x1, x3) I flipped from one to the other. I just have to show you that it's still, the output is still one even though they have two different values. >> Oh, no, no. When I said all of them, you know, k of them was because I didn't know that, what if all of them were irrelevant >> Then it would still be two. because then I could just show you the all zeroes, and the all ones. >> You're right. You're right. You just need, oh that's right. That's exactly right. >> Alright. And then I have to show you that the, that each, the remaining variables (x2, x4, x5) is relevant by flipping it and showing you that the answer is zero. And how many questions did I need to do for that? >> Three. >> Yeah, three in this case cause there were three variables that were used in the formula. What's the most it could be? >> Well k, cause all of them could be relevant (k 為 x 的個數, 注意圖中寫了的是 k-bit input). >> Yeah, so it's you know, it's kind of interesting that, that in fact the total number of hypothesis here is 3^k (下句即說原因). Because you know, you can see it right off this table that for each of the variables, it's either positive, absent or negated. But the number of questions that a smart teacher had to ask was more like, k+2. >> Huh. >> Which is pretty powerful. >> Right, so. The smart teacher can help me do this in linear term. So what if I were, I, I didn't have the teacher who could give me these examples and I always had to ask? >> That's a good question, let's let's do that.


Learner with constrained queries 3^k
X: $x_1 x_2 \ldots x_k$ k-bit input
H: Conjunctions of literals or negation
① Show what's irrelevant
② Show what's relevant
x_1 x_2 x_3 x_4 x_5 | h
0 0 0 0 0 | 0
0 0 0 0 1 | 0
0 0 0 1 0 | 0
0 0 0 1 1 | 0
1 0 1 1 0 | 1
h: positive absent negated
x_1
x_2
x_3
x_4
x_5
x_1 and $\bar{x}_2$ and x_3 and x_4 ...

本段的意思是: 對於 learner, 如果沒有 constraint, 則可以問 (log_2 3^k) = (k log_2 3) 個問題即可得出 hypothesis; 如果有 constraint, 則要難得多, 很難通過只問 (log_2 3^k) 個問題就得出 hypothesis.

15. Alright, so, you asked unfortunately what happens when the, the learner is now a part of this. Now the learner doesn't have that advantage that the teacher had of knowing what the actual answer was and therefore being able to show specifically what's irrelevant and show what's relevant. So, what could the learner do to try to learn about this? So again, remember that there are 3^k possible hypotheses, and if it could use the 20 questions trick, it could do this in (log_2 3^k), which is the same as (k log_2 3). Which is you know, worse than what we had. It's this is, this is larger than 1. But it's still linear, it's still linear in k. So, but can we actually do that? >> I'm going to say yes. >> I don't think we can, so can you help me figure out how that would go? >> Oh, I was just going to assert it, then hope you would tell me. so, how would we do that? Well, we, we, the trick we did before is, we, we tried to find a specific question we could ask, such that we would eliminate half the hypotheses. >> Indeed. But it's not clear how you could even ask such a question. Yeah, so, so just to do this as a thought exercise, I have a hypothesis in mind. >> Okay. >> And you can ask me anything you want, and I will tell you true or false. But you're going to have a very painful time finding it. >> Yeah, but that's just because I'm human. Okay, so I need to find a question where, of all the hypoth, so I have all the possible 3 to the K hypotheses. I want to try to come up with something that's going to eliminate a third of them which is just going to be hard for me to do because I could write the program to do this. >> I'm not sure you could. I think, at the moment, there's well, because I didn't choose my hypothesis at random. I chose a specific hypothesis. Though I guess I could have chosen at random from a subset, and you would have still had a hard time finding it. But let's, just as an exercise. Throw out, give me a, give me a x1 to x5, and I'll tell you what the output is. >> Okay, 00001. Or actually, you know what? All zeros. >> Okay, all zeroes, the output is zero. >> Oh, that's what I should, that's not what I should have done. I should have. No, no. >> That's okay, I won't count that one. >> [LAUGH] Can I just give you like, maybe 3 to the k of them and you'll not count any of them until I get it right? >> Well, that's the problem, right? Well, not 3 to the k, but if you, if you, you know, make 2 to the k guesses, do, you'll be okay. But you'll also have looked at all possible inputs. So that's not really that interesting. But in

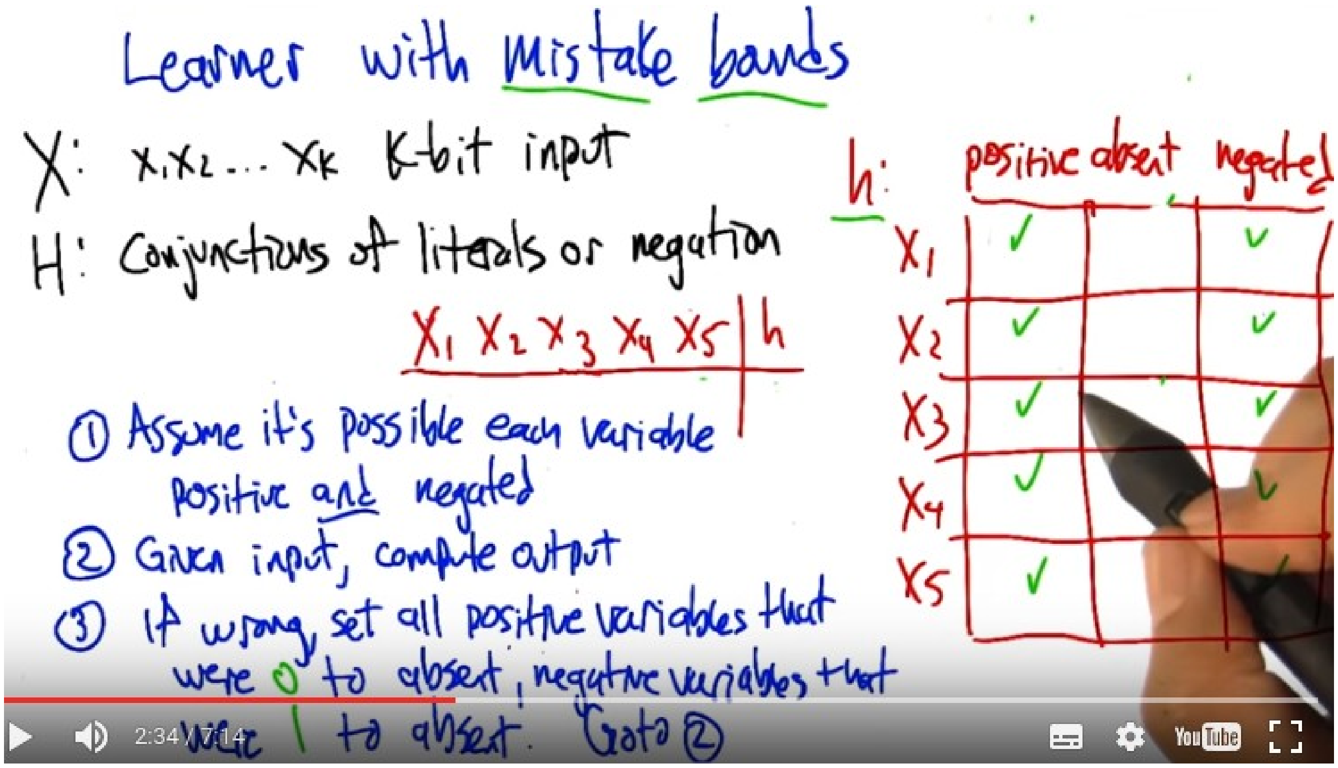
particular, the example that I'm thinking of, you're going to have to guess almost this many just to get a positive example. So almost everything that you throw in is giving almost no information. Because saying no doesn't really tell you very much. >> Yeah that's what I was thinking. Well, what I was thinking was I need to find one where the answer is yes (即 output 為 1, 此時基本上可以定下 hypothesis 是甚麼, 見下面藍字即知, 不重要). >> Exactly, and I made it so that it's going to take you exponential time just to find one. Once you've found that one, then you're, then you're home free but it's going to take you, you know, you essentially have to enumerate all possibilities before you find one. >> Okay, 0 0 0 0 1, okay? 0 0 0 1 0. And you're going to tell me that 0, 0, 0, >> There's only one pattern that gives a one. >> Right. Exactly. And you're going to, because every single one of them is relevant. And I'm going to have to look. >> Two of them are negated. This is the only pattern that gives you a one. Now once you have found that and you know that that's the only one, now it's easy. You can just read off the equation. So, what's the equation? >> x1 and not x2 and x3 and x4 and not x5. >> And that is the, that's the equation and you are not, you're not, there's no as a learner you are not going to be able to find that, right? Because it's just a needle in a haystack until you hit it. >> Yeah, so it's, it's going to take me exponential time, but it, but remember we're not worried about time. We're worried about sample complexity. So remember the cheat that we have here. The cheat that we have here is that I know all the hypotheses and what they say. >> It doesn't help you. >> Yeah it does, because the hypothesis, cause every hypoth, well no, that's not true. I'm thinking the wrong thing. I'm sorry. I'm cheating, you're right. I'm cheating. I'm, I'm acting as if we have the example you had before. >> So this constrained-ness is really, it's very frustrating, right? Because the question that you really want to be able to ask, you can't really ask, right? You want to be able to ask a question that, that takes the hypothesis class and split it in half and. Well maybe you can, maybe you can nearly do that. But it's still going to be, oh no sorry, that would make it linear. I'm sorry, let me say that again. You'd like to be able to ask a question that, that splits this hypothesis class in half, but unfortunately almost all of your questions give very little information. Just knocks out a couple of the possible hypotheses, and so it ends up being 2 to the k kind of time, not time but samples before you can get a handle on what the hypothesis is. So, it is harder for the learner too. >> Right, so when the learner does it you have no reason to believe one hypothesis over the other. You've got all of them. And so in order to figure it out, no it kind of has to be that way because otherwise it is still linear. So, this is bothering me, because if what you said is true, then why does 20 questions work? Why do i ever get log, log 2. >> Right. So we'd like to be able to ask questions. So I, so here, let's play this game now. You think of a, a formula. And I'm going to. >> Oh, wait, you. I know the, the answer is, is that the 20 questions is still the optimal thing to do, given that you know nothing. So that, that log base 2 is kind of an expected answer. But sometimes you'll do much worse, and sometimes you'll do better. >> No, in this particular case, if I could ask you more general questions. I can do this in, in with the, you know, linear in K. So the questions that I'd like to ask you are things like, is X1 in the formula, yes or no? [LAUGH] Is X1 positive in the formula? Is X1 negative in the formula? I can just fill in these boxes by asking the right questions. >> Right. >> But, but those questions are not in our constrained set. And it's the constrained set that matters here. And our constrained set is, in this particular example just really harsh. >> So, and there's no way to approximate that, right? So I can't say, okay, so the first question I want to ask is x1 positive, negative, or absent? So, if I looked at all the hyp, if I looked at all the hypotheses I could do that by asking, now it's very hard to do, because there's no direct way to ask that question. The only way to ask that question is, I have to try. Well, I have to try all possible exponential cases to know. >> Yeah, 'because we're constrained to only ask queries that are data points (下句馬上講意思), right? So give me the label(即 output) for this data point(如 x1=1, x2=0, x3=1,...). And that's not really the same as is the hypothesis you're thinking of having this particular property. >> But as soon as I get a one, I know something. >> Soon as you get a one, you're in a much happier place. So, in fact, if we didn't, if we had conjunctions of literals without negations >> Mm-hm. >> We'd be in a much better situation, because then you could, your first question can be, you know, one one one one one one. You know the answer has to be one, or the formula's empty. So then you're, you're basically off and running, but the fact that there can be negation in there means that most queries really give you useless information. >> So, so Michael, okay, so you've depressed me. You've basically said this is really hard to do, to learn because I think that we've convinced ourselves, at least you've convinced me that until I get a one, until I, I, I get a positive result, I can't really know anything. And eventually I will get one if I can just do an exponential number of samples, but then my sample complexity is exponential, and I'm sad. So what you're basically saying is, I'm sad sample complexity makes me a bad person, and there's really nothing I can do to learn anything or get anything good out of my learning process. >> That seems like a very sad way of saying it. >> Yes, is there a happy way of saying it? >> There isn't a happy way of saying that. But there is, there are other questions that have happier answers. >> Okay, like what?


Learner with mistake bounds
X: x1 x2 ... xk k-bit input
H: Conjunctions of literals or negation
x1 x2 x3 x4 x5 | h
① Input arrives
② Learner guesses answer
③ wrong answer charged
④ Go to ①
bound the total number of mistakes
h: positive absent negated
x1
x2
x3
x4
x5

左下角的 wrong answer charged 中的 charged 應該是懲罰的意思, 即答錯了的話, 要記數.

16. So maybe, maybe this will make you feel better Charles. So there's we can actually, you know, when all else fails, change the problem. So let's say that instead of trying to learn the way we were describing it before, we're going to change the rules. We're going to say we're going to use a learning formalism that's sometimes referred to as mistake bounds. So here's how the things work in mistake bounds, the learner is sitting around and input arrives. And then the learner gets to guess an answer for that input. So the learner the, maybe the learner chose the input or maybe it came from a, a helpful teacher or maybe a malicious teacher, turns out it's not going to matter. But the input's going to show up. The learner's going to guess the answer, so it doesn't have to now guess the hypothesis and get that right. It just has to get he output correct for this input (意思是: learner 每次猜一個 input 對應的 output 是多少, 猜了很多次後, learner 通過這些信息猜出 hypothesis). If the learner is wrong, then we're going to charge it a point and tell it, that it was wrong and then it goes up to (step) one and we repeat this and so, this is going to run forever and what we're going to do is bound the total number of mistakes made, into infinity. Right? So, were going to keep playing this game forever. And we want to say it'll never make tot total number of mistakes will never be larger than a certain amount. >> So this is giving the learner some new powers, right? So the learner now is guessing answers. And all we have to guarantee is that

if is guess is wrong, it better learn a heck of a lot from that. Hmm. >> Otherwise if it even if it doesn't know much if it guesses right that's fine. It doesn't have to learn. >> Okay. I see. So, so in the case before so long as I had been guessing false I would have been okay even if I didn't know what the hypothesis was. And then the moment I got a one, I got a true, I could actually learn something. Then I should learn something and try to do better guesses from that point on. Okay. >> Outstanding, yes, exactly so.



Learner with mistake bounds

X: $x_1 x_2 \ldots x_k$ k-bit input

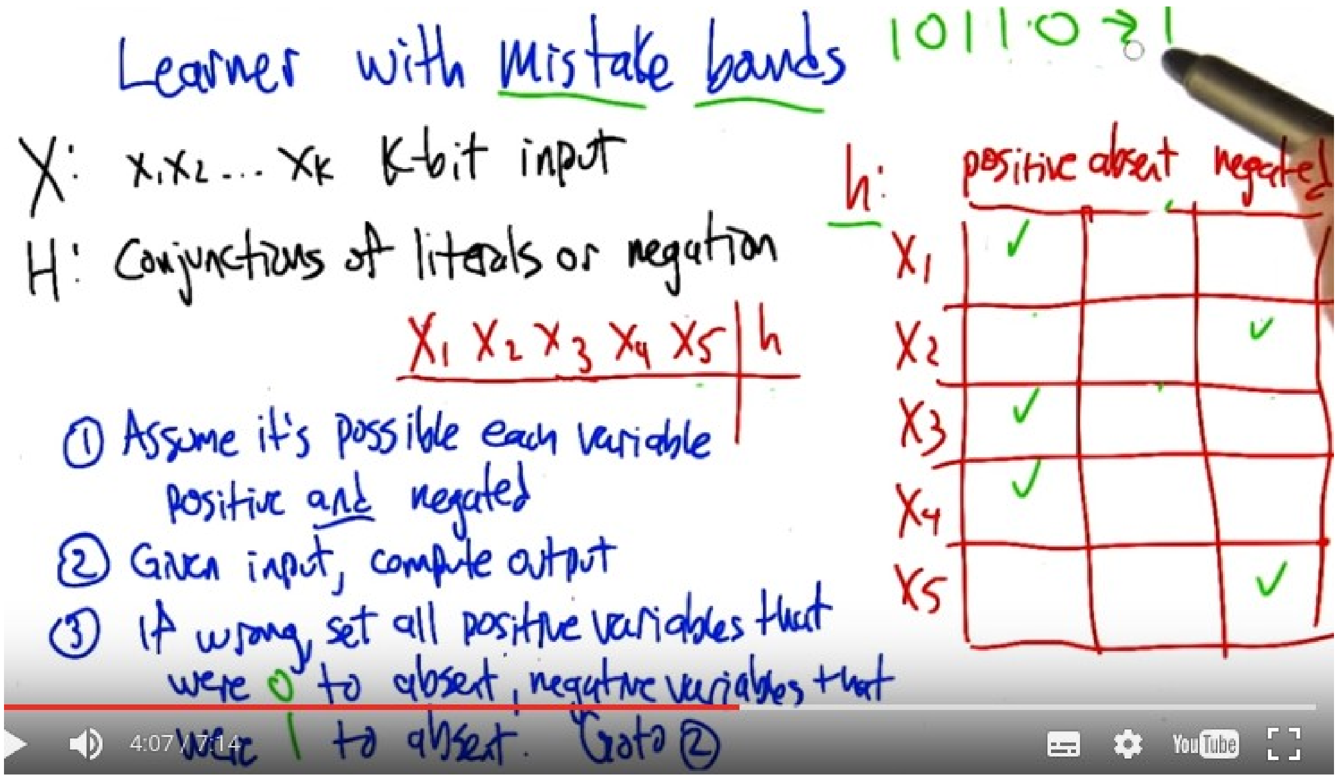
H: Conjunctions of literals or negation

X_1	X_2	X_3	X_4	X_5	h

1. Assume it's possible each variable positive and negated
2. Given input, compute output
3. If wrong, set all positive variables that were 0 to absent, negative variables that were 1 to absent. Goto 2

h:	positive	absent	negated
X_1	✓		✓
X_2	✓		✓
X_3	✓		✓
X_4	✓		✓
X_5	✓		✓

So lets turn that into an algorithm. So here's an algorithm that's really simple and actually can learn very effectively for these mistake bound problems. So it, it works like this. It starts off in this weird state where it imagines in the formula that every variable is present. In, in both it's positive and negated form. Right, which is kind of weird. And so what that, that would mean is the formula is x1 and not x1. X2 and not x2. X3 and not x3. X4 and not x4. X5 and not x5. So any input that it gets, it;s always going to produce the same answer, right? What, what is that answer. >> False. False, right, so it's going to keep saying false, for a long time. Until at some point, it actually could be right. Each time it's right, it's actually not getting charged any mistakes for that. It's just that at some point, it's going to get an input that the correct answer is true. It's going to say false, and it's going to have made a mistake.



Learner with mistake bands 10110 → 1

X: $x_1 x_2 \ldots x_k$ k-bit input

H: Conjunctions of literals or negation

X_1	X_2	X_3	X_4	X_5	h

① Assume it's possible each variable positive and negated

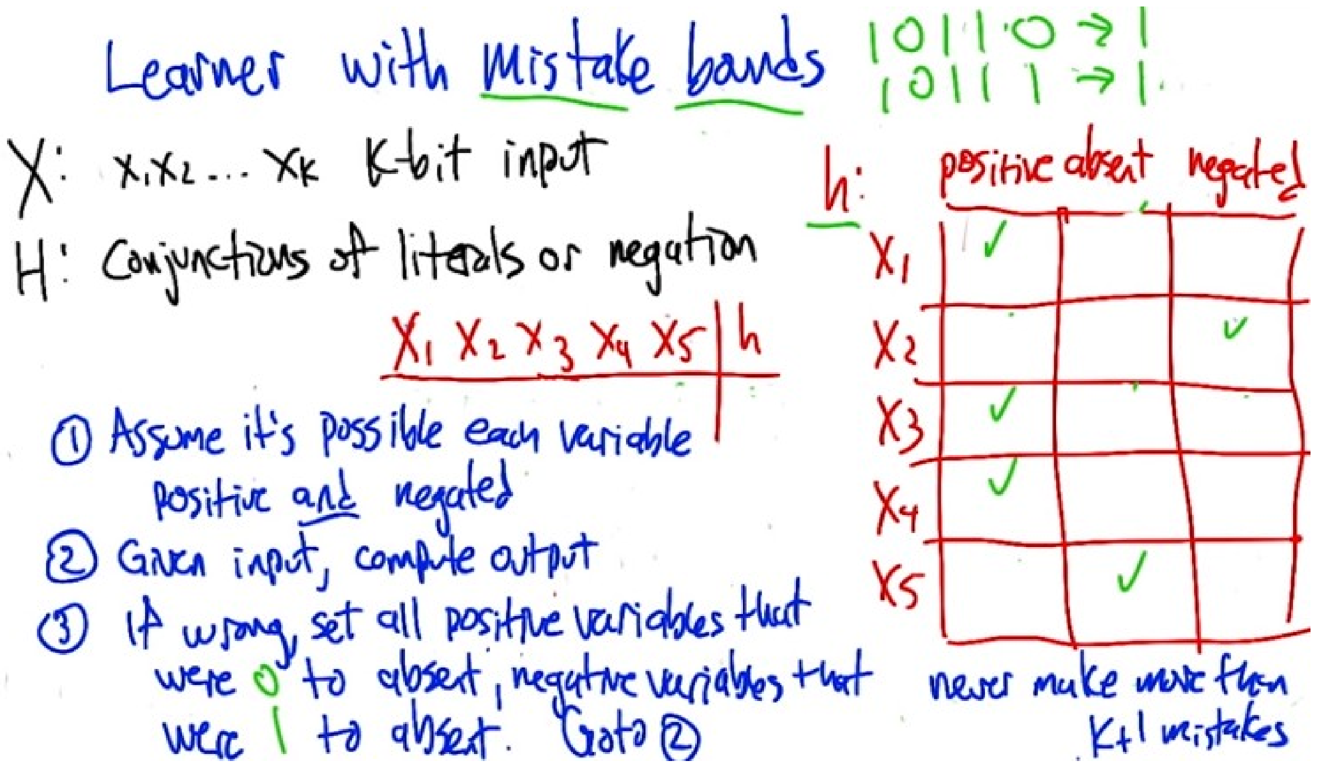
② Given input, compute output

③ If wrong, set all positive variables that were 0 to absent, negative variables that were 1 to absent. Goto ②

h:

	positive	absent	negated
X_1	✓		
X_2			✓
X_3	✓		
X_4	✓		
X_5			✓

So let's say that this here, here's that example. X1 is true. X3 and X4 are true, and the other two are false (即輸入為 10110), and the learner said false, but the answer was actually true. So if the answer to this one is true, what do we know? >> We know that one zero. Wait we know that 0100 1, which is the opposite of what you're saying could not be a part of the formula. So we could remove that from our formula. >> okay. That's one way to think of it. Couldn't quite think of it that way. >> But am I right? Yeah, if this is what you meant. so... >> [LAUGH] >> Uh...the first variable, the 「x1 not (即 not x1)」 cannot be in the formula. Cause if it was, there's no way that this would've been able to produce true. >> Right. >> So we can erase x1 not. We can erase(即 make) x2 in a positive form, because of the second bit. X3, x4, x5 都類似弄. We can, the third bit says that X3, if it's in there, it can't be negated. We don't know if it's in there, but we know it can't be negated. X4 Is the same. And x5, if it's in there. >> Right you get rid of the positive. Yeah, that's what I meant. That's why I wrote down the opposite of everything that was up there. >> Yeah, and that is what we're left with. Okay, it's not exactly what the algorithm says, but it, it produced the right answer in this case. Alright, so now, now what is it going to do? Now it's going to continue to say no, unless it sees. This particular bit pattern, right, this is the only thing that will ever predict true on and it will always be right,when it does that because we know that is the correct answer for that.



Learner with mistake bounds

$10110 \rightarrow 1$

$10111 \rightarrow 1.$

X: $x_1 x_2 \ldots x_k$ k-bit input

H: Conjunctions of literals or negation

$x_1\ x_2\ x_3\ x_4\ x_5 \mid h$

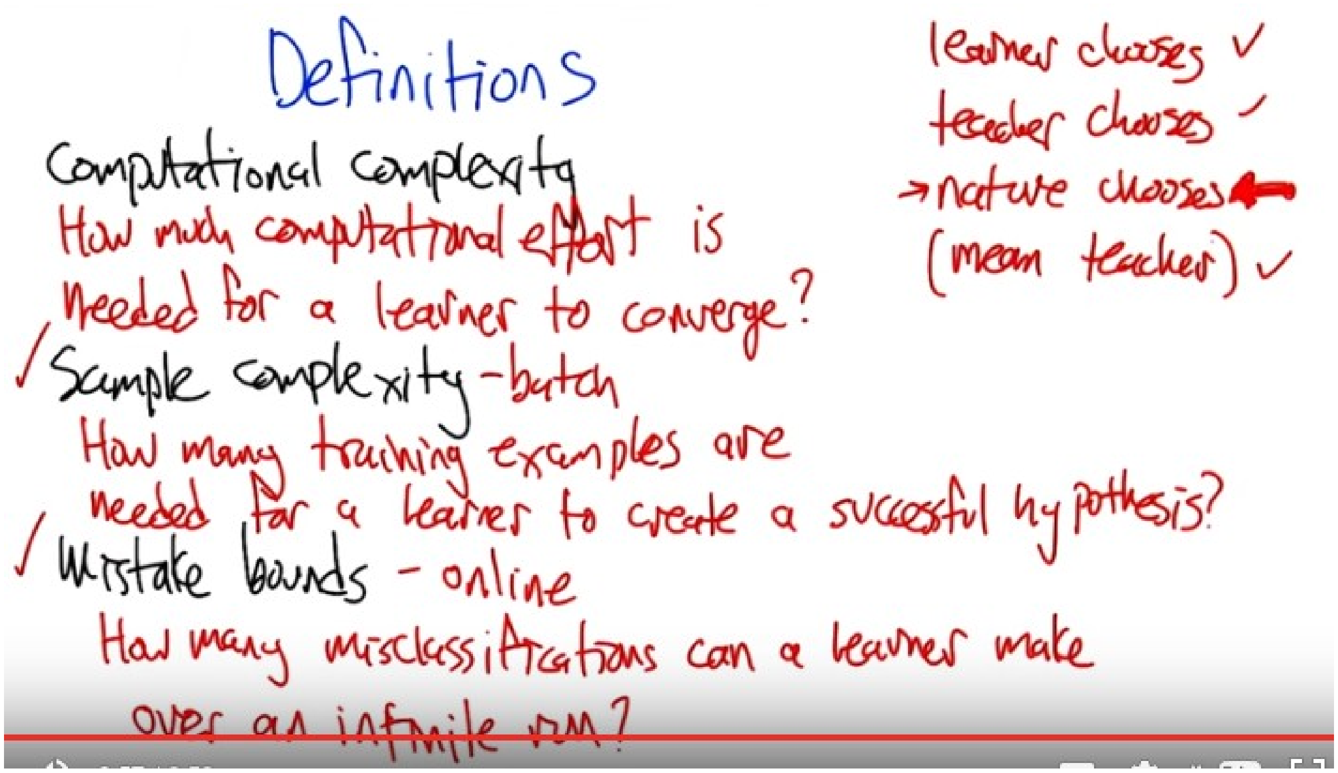
1. Assume it's possible each variable positive and negated
2. Given input, compute output
3. If wrong, set all positive variables that were 0 to absent, negative variables that were 1 to absent. Goto 2

h:

	positive	absent	negated
x_1	✓		
x_2			✓
x_3	✓		
x_4	✓		
x_5		✓	

never make more than K+1 mistakes

But let's so, so it's going to guess no everywhere else and so let's say it gets something wrong again and let's say it gets 10111 wrong. So in this particular case, it's going to guess no (即 false), and we say, I'm sorry, the answer is yes (即 true). All right, so now what does it know from that? >> Well, I'm just reading your algorithm now. And I'm just going to do what number three says. >> All right. That's a good idea. It says if we're wrong, which we are in this case, set all the positive variables that were zero to absent. All the positive values that were zero, there are none of those and said all the negative variables that were one, to absent, alright. So then that was x5, x5 is there in its negated form(上一節的圖中), but it's actually a one in the input, so we're going to turn that away to absent. Alright. >> Mm-hm. >>So and that's the same thing that you did when we were looking at the problem before, you said (右邊剩餘部份的意思即 前面講到過的 若某一個 x_i 在某一個 input 中為 1, 而在另一個 input 中為 0, 且這兩個 input 的 output 都是 1, 則這個 x_i 不能出現在 hypothesis 中) if you have two answers, where the, two inputs where that output is both true, any bits that are different in those two patterns, must be not part of the formula. >> Right. >> Alright, so now we've definitely, X5 is not in the formula and that's actually correct there's, there's at no point in the future where we have to revisit that. In this other cases we are not quite so sure. It could be that they're, in there or not in there. And, so each time that we get one wrong, we're going to move something from negated to absent. And when we do that, that thing is always going to be correct. So at most we can move K things from negated or positive to absent (因為後面每一個二進製數, 都至少有一位跟最開始那個二製數不同(否則兩個二進製數就相等了), 而不同的那位是必須 absent 的). >> Oh! So if I. . Think about, oh, so even if we may have to see in fact, every, even if we may see an exponential number or examples. We will never make more than k+1 mistakes (加 1 是指最開始范的那個錯. 另外注意, 是范 k+1 次錯, 而不是試 k+1 次, 因為有時候是不范錯的). >> Perfect. >> And so that's exactly what I wanted to do before, right? So, if, if I'm a teacher, if I'm a good teacher in this case, then I can basically, and I knew that you started out assuming that everything was false. >> That you know, all variables were there in both their positive or negative form, I can just give you one example that is true. And that would let you eliminate half of the formula right away, and then I could just keep giving you examples that are true but only with one variable difference each time. And then eventually you would learn it. So, then the total number of samples you would need would also be k+1 if I know how you're starting out as a learner. Does that make sense? >> If we charge things by mistakes. >> No. But even if we don't charge things by mistake. If I'm a teacher who's trying to give you the best examples and I know that as the learner you're starting out with a formula that is x one and not x one, x two and not x two. Like you said before. Then I could just give you the first example that is true. That'll eliminate half of those literals. And then only give you true examples from that point on (意思是 teacher 每次都故意讓 learner 范錯), where only change one of the variables that are left, and you'll know that you can make them absent to get rid of it, and so, just as you can only make k+1 mistakes, I could give you exactly the right k+1 examples. If I know how you're starting out as a learner. If I don't know that, then I have to do the k+2 that you showed before (此句話應該不重要).


Definitions

Computational complexity
How much computational effort is needed for a learner to converge?

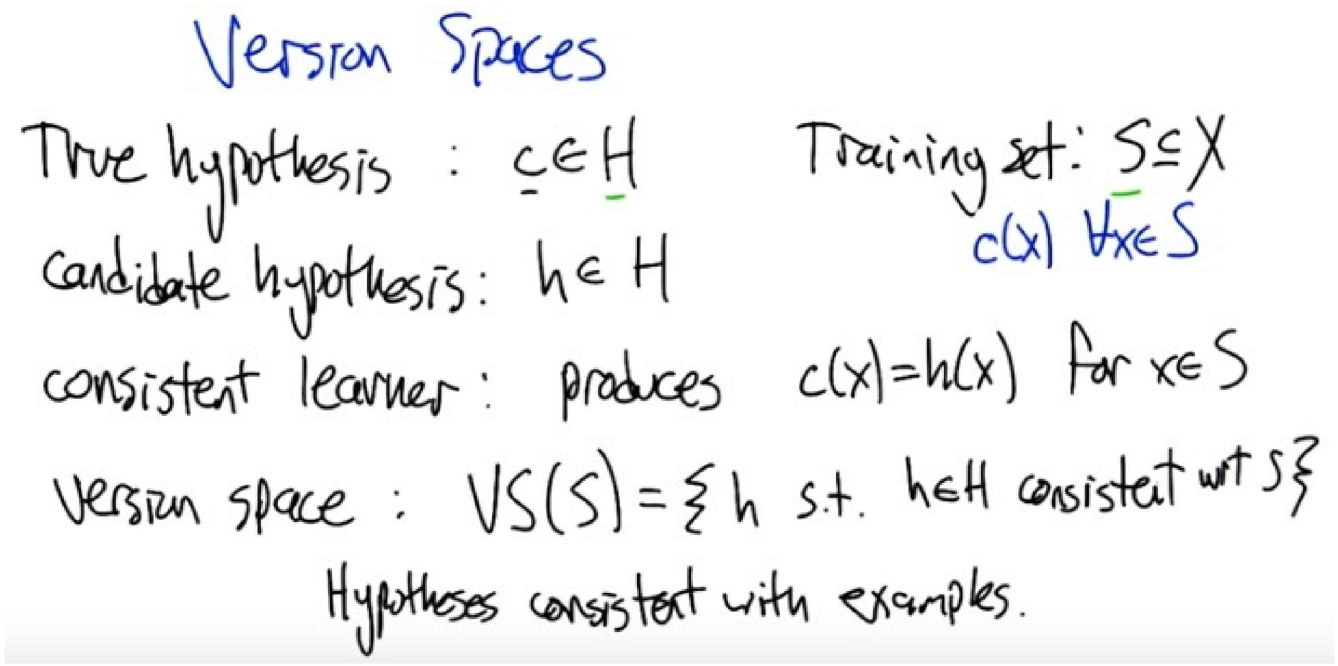
✓ Sample complexity - batch
How many training examples are needed for a learner to create a successful hypothesis?

✓ Mistake bounds - online
How many misclassifications can a learner make over an infinite run?

learner chooses ✓
teacher chooses ✓
→ nature chooses
(mean teacher) ✓

17. Alright, so, remember, Charles, we were talking about three different kinds of ways of choosing the inputs to the learner. Okay? So what were they again? We just looked at two of them. >> So the learner chooses examples. The teacher, hopefully a nice one, chooses examples, and then there was the case with the examples are given to us by nature. And I guess there was a fourth one, which is. That a mean teacher gives it to us but I, you know I don't think that's all that interesting. I tend to think of nature as a mean teacher. >> Well not only that but in the, at least in the mistake bound setting that we just looked at again the, the learner was robust against any choice of where the inputs were coming from. So mean teacher it would've it would've done just as well. >> That's a fine point. >> It doesn't matter when it makes its mistakes, it's only going to make a fixed number of them no matter what. >> Okay. >> So, we've kind of dealt with three of these but we haven't really talked about the, the nature chooses case yet. And that's in some ways the most interesting and relevant and in other ways the most complicated because you have to really take into consideration this space of possible distributions. I think now though we're in a pretty good situation in terms of being able to define some important terms and we're going to use those terms to get a handle on this question of what happens when nature chooses. >> Okay, that sounds reasonable. >> So computational complexity, we talked about. The, the, how much

computational effort is going to be needed for a learner to convert to the answer. Sample complexity in the case of a batch, that is to say, we have a training set. Is how large is that training set need to be for the learner to be able to create successful hypothesis. And that's in the batch setting. In the online setting, we have this notion of a mistake bound. How many misclassifications can the learner make over an infinite run? >> Mind if I ask you a question? You said something I thought pretty interesting. How, for computational complexity, you said how much computational effort is needed for a learner to converge To the right answer. Is that the, is that a requirement when you talk about computational complexity? Or is just that you need to know how much computational effort is needed for a learner to converge to something? >> Well, so if it's converging to something and we don't care what it's converging to, then it's really easy to have an algorithm with low computational complexity, right? It's just like return garbage. It runs in constant time. >> Mm hm. >> So, yeah, it's in the context of actually solving the problem, that computational complexity is most interesting. >> Well, I was going to say, what if a set of hypothesis that I'm willing to entertain, doesn't include the true concept. >> good. Right. So in some sense maybe in that case what we're trying to find is the best hypothesis in the hypothesis class. But we can still ask about computational complexity in that case. So, I was saying successful hypothesis here. By that I meant, you know, whatever it is that the problem demands that we return and if it's a hypothesis class that's very limited we might just return the best thing in that class. >> Okay. >> But the important thing here is that we're not going to talk about computational complexity, for the most part. We're going to focus on sample complexity for the time being. And that's really the relevant concept when we're talking about the idea of nature choosing. >> I believe you



Version Spaces

True hypothesis : $c \in H$ Training set: $S \subseteq X$

$c(x)\ \forall x \in S$

candidate hypothesis: $h \in H$

consistent learner: produces $c(x)=h(x)$ for $x \in S$

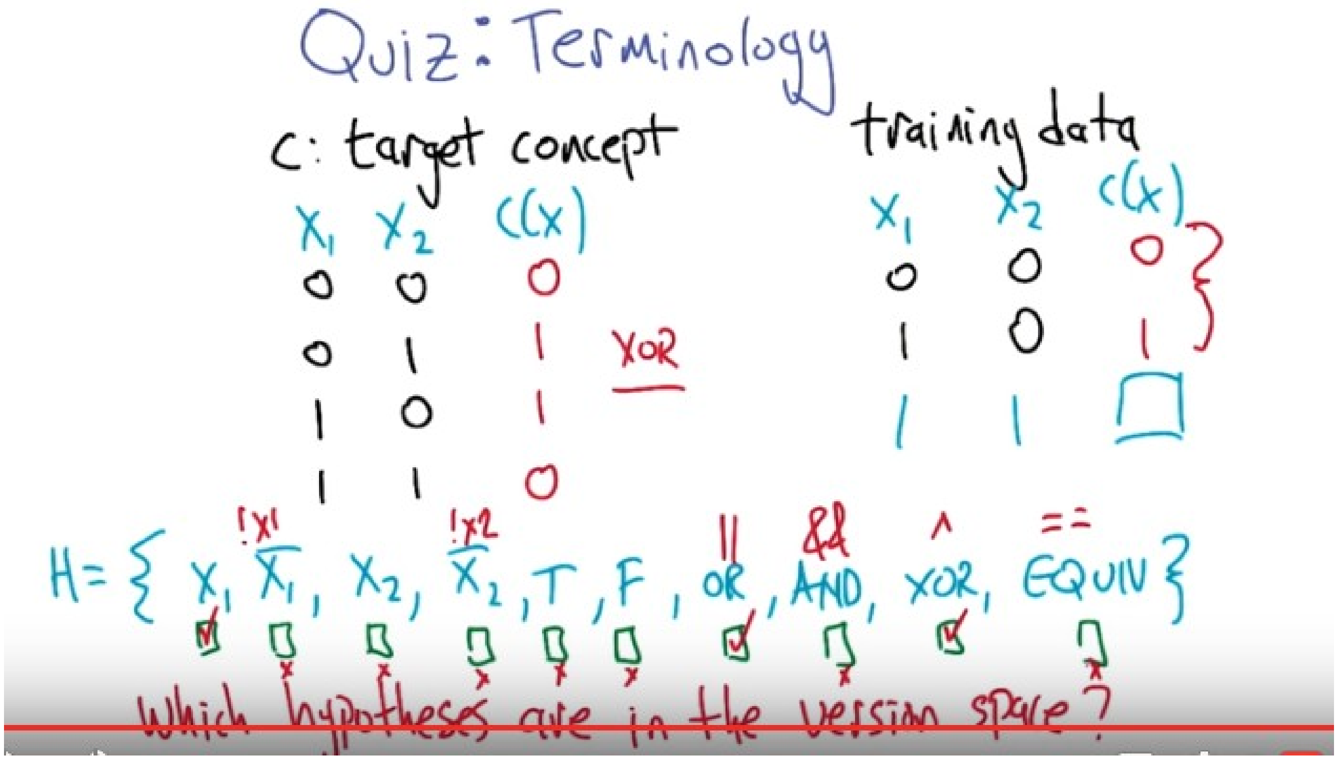
version space : $VS(S) = \{ h$ s.t. $h \in H$ consistent wrt $S \}$

Hypotheses consistent with examples.

18. All right. We're going to do a couple more definitions. This notion of a version space turns out to be really important in understanding how to analyze these algorithms. So, imagine we've got some hypothesis, space H, capital H. And a true hypothesis that we're trying to learn, C of H. This is also sometimes called a concept. And we're trying to learn from a training set S, which is a subset of the possible inputs. And what our training set consists of is those examples along with the true class for all of those X's. So, at any given time a learner might have some candidate hypothesis, little H and big H, and a learner who produces a candidate hypothesis little H, such that C of X is equal to H of X for all X of the training set, is called a consistent learner. Right, so what would be another way of describing hat a consistent learner is? >> A consistent learner can actually learn the hypothesis, or the true concept.

>> Well it produces, right, I mean of all the possible things it could return, it returns the one that matches the data that it's seen so far. >> Right, so it's consistent with the data (即 training set, 因為圖中為 consistent with S). That makes sense. >> Yeah. And the version space is essentially the space of all the hypotheses that are like that, that are consistent with the data (即 training set. 後面第 28 段已驗證, 那裡會深刻理解甚麼是 version space). So we'll say the version space for a given set of data S is going to be the set of hypotheses, that are in the hypothesis set, such that they're consistent with respect to the samples that they're given. >> Okay that makes sense. >> All right. Good. So I think what we'll do next is a quiz just to make sure that we kind of get this concept and how it works. And we'll do that next time. >> Okay sure. I like quizzes.

19. Alright, here's the quiz. So just to practice this concept of what a version space is let's go through an example. So here we go. Here's the actual target concept C and, for all possible, it's, mapping from two input bits to an output bit. And the two inputs, X1 and X2 can take on all four possible different values and the outputs are zero One, one, zero, which is to say the XOR function, okay so that's what we're trying to learn. >> Okay. >> But the training data that we have only has a couple examples in it. It says well, one training example says if the inputs are zero and zero. The output is zero. And if the inputs are one and zero, the output is one. And, ultimately, we're going to ask questions like, well, what happens if the input is one and one? What's the output? Now, since we know that it's XOR, we know that the answer is zero but that's kind of using information unfairly. So here's what we're going to do. Here's a set of, a hypothesis set. These are set of functions. That says, well the output could be just copy X1, negate X1, copy X2, negate X2, ignore the inputs in return true, ignore the inputs in return false, take the OR of the inputs, take the AND of the inputs, take the XOR of the inputs and then return whether or not the inputs are equal to each other. And, what I'd like you to do is, some of these are in the version space and some of them are not for this training set that I marked here. So, can you check off which ones are in the version space? >> I think I can. >> Awesome. Let's do it.



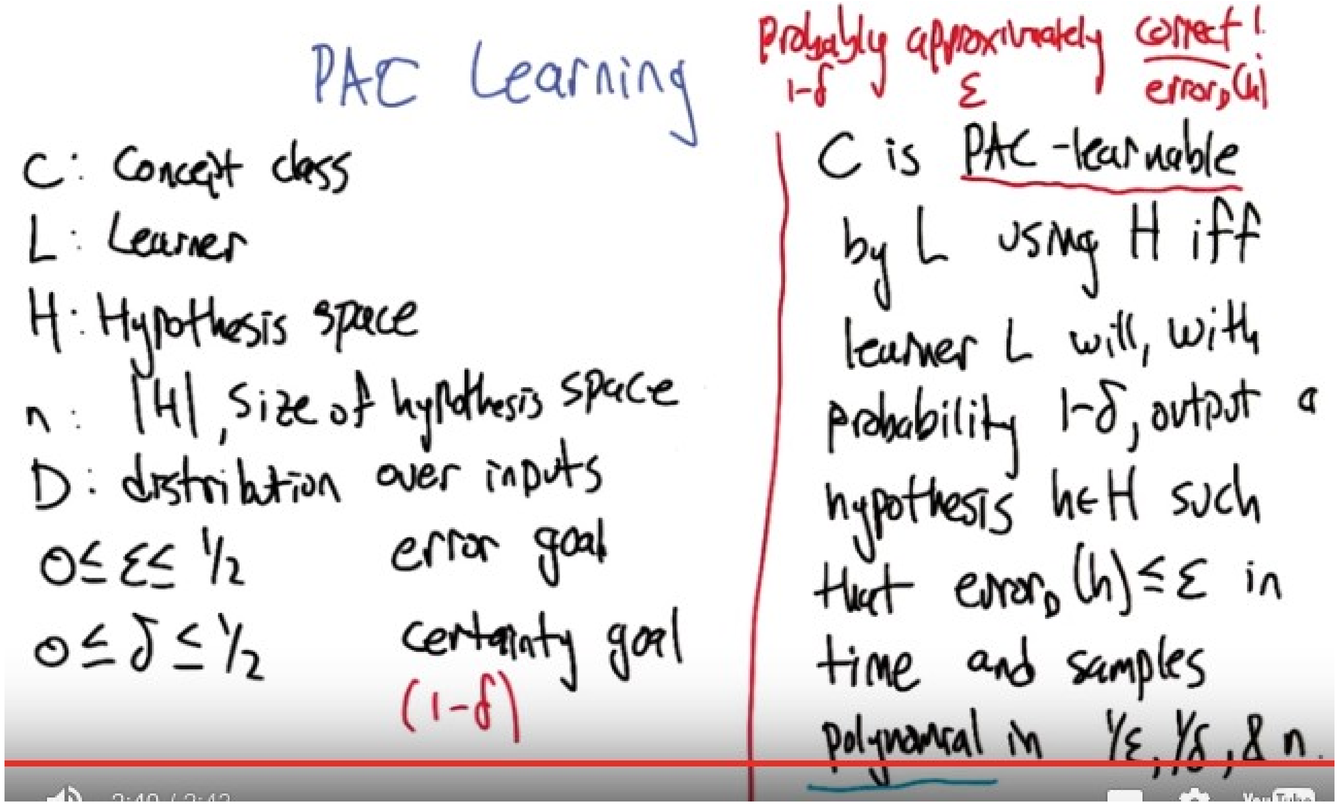
x_1	x_2	$c(x)$
0	0	0
0	1	1
1	0	1
1	1	0

x_1	x_2	$c(x)$
0	0	0
1	0	1
1	1	$\square$

$H = \{ x_1, \overline{x_1}, x_2, \overline{x_2}, T, F, OR, AND, XOR, EQUIV \}$

20. >> All right Charles, what do you think? >> Okay, so being in the version space just means that you're consistent with the, data that you see. Right? >> Good. Mm-hm. >> Okay, so we should be able to very quickly go through this. So X1, just copy X1 over. Well, if we look at the training data, in the first case, X1 is zero and the output is zero. Second case, X1 is one and the output is one. So that is, in fact, consistent with just always copying X1. >> So that is in the version space, right? >> Right. And without even having to think about it, since x1 is in the version space, doing the opposite of x1 can't possibly be in the version space. >> Agreed. >> So let's skip that. >> Looking at x2, we can see that in the first case, you go x2 0, c of x is 0. Yeah, so yeah that's looking good. That's consistent so far. But then on the next row you get 0 and then you get the, opposite of 0. You get 1, the compliment of 0. So, that definitely is inconsistent with just copying over X2, so you can't have X2 and by very similar reasoning you can't have the opposite of X2. >> Agreed. >> Okay, so, true is clearly not consistent because we got a 0 once and a 1 once. >> Mm-hm. >> And by the same argument false is not consistent because we got a zero once and a one once. Now or, let's see or. >> But in case it's not clear, I probably could have been clearer about this, but here, zeroes and falses are the same thing and ones and trues are the same thing. >> Right. Just like in C. >> [LAUGH] >> Okay. So I just assume everything you do is written in C, Michael, so, it just works that way. In fact there's a C right up there at the top of the, at the top of the slide. >> Yeah so I feel like I should do this. Now I understand what's going on. >> Right, now it's in C. >> Much better. Okay, so, or. Or means if either one of them is true, then you say yes and, huh! That's actually consistent with or. >> Yep. >> Hm. But it is not consistent with and, because one and zero would be zero, not one. Second case, XOR, well, I already know it's consistent with XOR, because I happen to know that that's the target concept. >> Yeah. >> And an equiv would be not consistent. Though interestingly, not equivalent would be consistent. >> Yes, because not equivalent is XOR. >> Oh, yeah, that's right. >> All right, so that's, yeah, that's it, X1, OR, AND, XOR. >> Excellent. >> Cool.

21. Alright, the next topic we're going to get into is to nail down a concept called PAC learning. So to do that, we're going to delve into what the error of a hypothesis is. And there's two kinds of errors. There's the training error and the true error. So the training error is on the training set. What is the fraction of those examples that are classifieds by some given hypotheses H. >> Okay. >> Now H could be different from the target concept. The target concept ought to have a training error of zero. But some other hypothesis h might have some error. Okay? >> Sure, that makes sense. >> Okay, so the true error is actually the fractions of examples that would be misclassified on a sample drawn from D (D 就是 distribution, 即 input 出現的概率分佈) in essentially the infinite limit. So what is the, essentially the probability that a sample drawn from D Would be mis-classified by some hypothesis, h. >> Okay. >> And we can actually write that mathematically. Error with respect to some distribution D of some hypothesis h, is the probability that if we draw the input X from that distribution that you're going to get a mismatch between the true label, according to the concept, and the hypothesis that we're currently evaluating. >> Right, so this sort of captures the notion that it's okay for you to misclassify examples you will never ever ever see. >> Yes, that's right. So we're only being penalized proportional to the probability of getting that thing wrong. So, for examples that we never see, that's fine. For examples that we see really really rarely. We get only a tiny little bit of contribution to the overall error. >> Okay, I can follow that.



PAC Learning

C: Concept class

L: Learner

H: Hypothesis space

n: |H|, size of hypothesis space

D: distribution over inputs

$0 \le \varepsilon \le 1/2$ error goal

$0 \le \delta \le 1/2$ certainty goal $(1-\delta)$

Probably approximately correct!

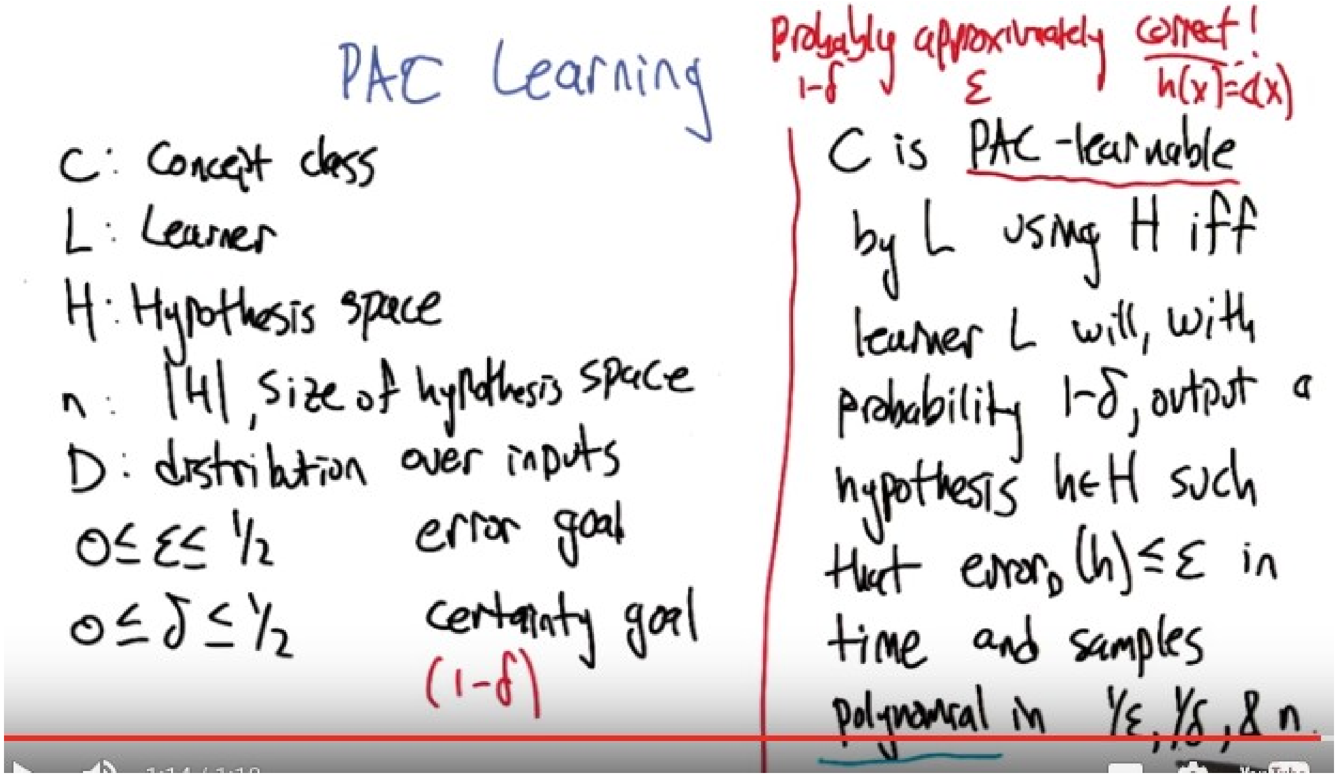
$1-\delta$ ε $\text{error}_D(h)$

C is PAC-learnable by L using H iff learner L will, with probability $1-\delta$, output a hypothesis $h \in H$ such that $\text{error}_D(h) \le \varepsilon$ in time and samples polynomial in $1/\varepsilon$, $1/\delta$, & n.

29 段面會證明這樣一個結論(我的表述): If we want to PAC learn a hypothesis set which has size n, the numer of samples we need is polynomial in 1/ε, 1/δ, & n, 即 m >= (1/ε) (ln|H| + ln(1/δ)), m 為 number of samples, n 即|H|(即 size of the hypothesis space). 其中 PAC learn a hypothesis set 的意思是: 該此 hypothesis set 中的每一個 hypothesis 都在 1-δ 的概率下 范錯率在 ε 內.

22. Alright, with just a couple more definitions we'll be able to nail down what PAC Learning is. So let's consider a, concept class C. That is to say that the set from which the concept that we're trying to learn comes from. L is our learner. H is the hypothesis space, so it's the set of, of mappings that the learner is going to consider. N is going to be the size of that space. So kind of the space of the hypotheses that are being considered. D is that distribution of inputs like we looked at before. Then we got Greek letters epsilon and delta, where ε is our error goal. Which is to say we would like the error in the hypothesis that we. Produce to be no bigger than ε. It could be smaller, that'd be great. It could be zero, that'd be awesome. But it can't be bigger than epsilon. But, we could get really unlucky and actually not meet our error goal. And so δ is what allows us to set a, a kind of uncertainty goal or certainty goal, which is to say, that with probability 1-δ, the algorithm has to work. >> And by work, you mean has to be able to produce a true error less than or equal to ε. >> Exactly. >> So why can't we always force epsilon to be zero, and, you know, delta to be zero? >> Right, to be absolutely sure that we get zero error. So the, part of it is because we are sampling training examples from this distribution D. and it's always possible that we, for example, unless well, as long as there's probability mass on more than one example, there's always a probability that we draw a fine sample that only ever gives up one example over and over again. That's fair. So we just have to be really careful about that. So, so if we're that unlucky it's very unlikely to happen but when that happens our error is going to be very high. But that's okay because it won't happen very often. >> Okay sure,sure. So basically you can't force it to

be zero under all circumstances either epsilon or delta you can't force them to be zero under all circumstances. So you have to allow somewhere to. Exactly, it gives us the wiggle room we need. Thats actually where this name PAC comes from. It stands for probably appoximatly correct. I see. So we would like to be correct, but then we are just going to keep adding weasel words until we can actually achieve it. We would like to be correct, but we can't be exactly correct, so lets be approximatley correct. But we can't be approximately correct all the time, so let's only probably be approximately correct. >> So using your, your, your words here you could have called that one minus delta epsilon correct. >> Yes, right. That's exactly right. That's what the Greek letters are playing, and correct is the, this you know, error Sub D of H equals 0. >> Yeah, so 1 9 is delta upsilon error sub D of H equals 0 doesn't roll off the tongue quite as well as probably approximately correct. >> I would agree with that. >> Okay fair enough.



PAC Learning

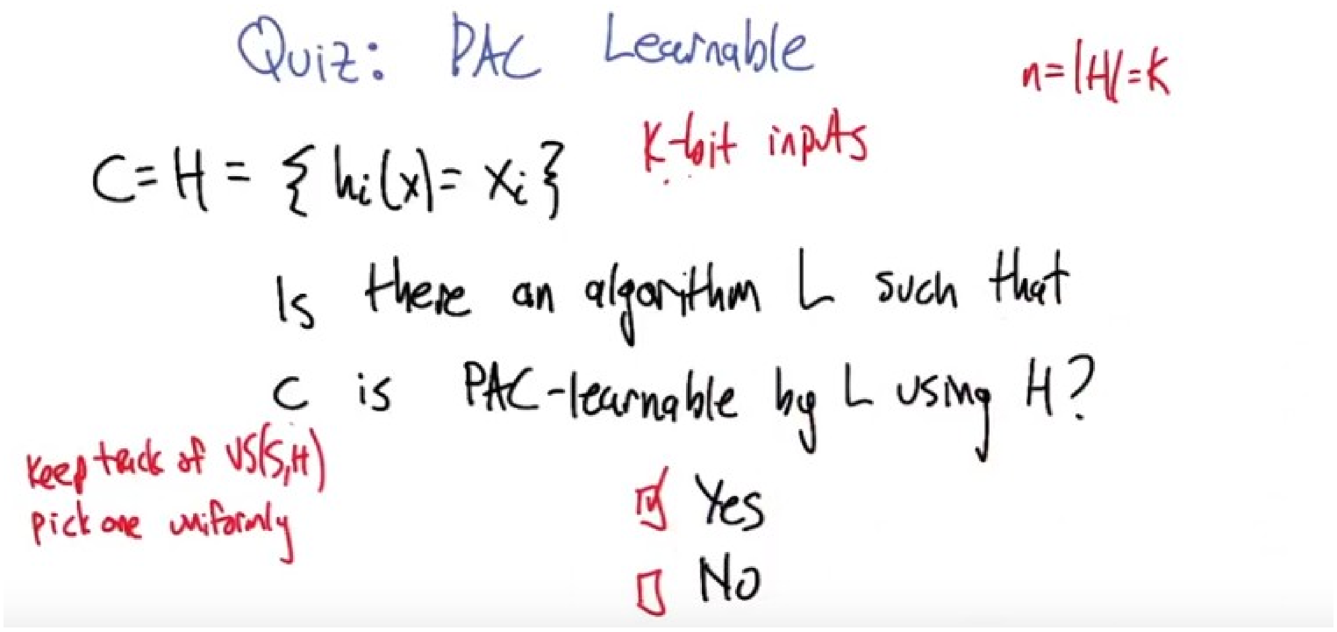
C: Concept class
L: Learner
H: Hypothesis space
n: |H|, size of hypothesis space
D: distribution over inputs
$0 \le \varepsilon \le 1/2$ error goal
$0 \le \delta \le 1/2$ certainty goal $(1-\delta)$

Probably approximately correct!
$1-\delta$ ε $h(x)=c(x)$

C is PAC-learnable by L using H iff learner L will, with probability $1-\delta$, output a hypothesis $h \in H$ such that $\text{error}_D(h) \le \varepsilon$ in time and samples polynomial in $1/\varepsilon$, $1/\delta$, & n.

iff: if and only if

23. Alright, now we can actually dive in and give a definition for PAC-learnable. So, a concept class, C, is PAC-learnable by some learning algorithm, L, using its own representation of hypothesis, H, if and only if that learner will, with high probability, at least 1 minus Delta, ouput a hypothesis h, little h, from its set of hypothesis. That has error less than or equal to epsilon, so it's very accurate. And it needs to be the case that the time that it takes to do this. And the number of samples that it needs draws from this distribution D is relatively small. In fact, bounded by polynomial in 1/ε, 1/δ and the size of the hypothesis space n. >> Okay, so in other words you're saying something is PAC-learnable if you can learn to get low error, at least with some high confidence you can be fairly confident that you will have a low error in time that's sort of polynomial in all the parameters. >> Exactly, and you can see here that this, this ε and δ actually giving us a lot of wiggle room, if you really want to have perfect error. Or perfect certainty. Then these (1/ε 等) things go to infinity. So, you just, you need, you need to look at all the possible data. >> Mm. >> So, yeah this is really going to only give us partial guarantees. Okay. Sure. Okay, I think I understand that. .



Quiz: PAC Learnable

$C = H = \{ h_i(x) = x_i \}$ K-bit inputs

$n = |H| = k$

Is there an algorithm L such that c is PAC-learnable by L using H?

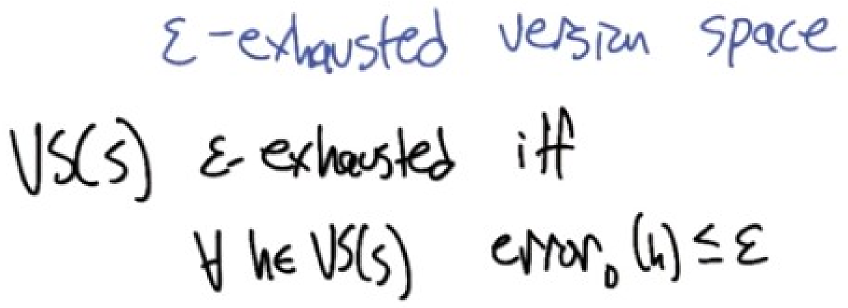
Keep track of VS(S,H)
pick one uniformly

[✓ Yes]
No

24. >> Let's just do a little quiz. So, here is our concept class and our hypothesis class, which are the same in this case, which is the set of functions h_i, that return, for an input x (x 表示{x1, x2, ...}), it returns the i-th bit of that input (即 xi). >> Mm-hm. >> All right. So if we're talking about K bit inputs then there's going to be a hypothesis in this set for each of those K bits, and that hypothesis returns the corresponding bit. And so, we're given a training set with examples like that. And then we need to produce outputs that are consistent with that. And so what I'd like you to try to figure out is, tell me, do you think that there is an algorithm L, such that C is PAC learnable by L using H? So if you can give me a learning algorithm that would do a good job, and would, with high probability, be able to figure this out, then you should go for it. >> Okay.

25. Okay, so what'd you think? Was that enough information to kind of chew on it a little bit? >> Maybe. [LAUGH] So, let's see, I have, I guess, k hypotheses I have to chose from. I know that basically you always return the output of one. I don't get to chose what those examples are, they're drawn from some distribution. >> Right. >> And I want to know whether I'm going to need to see, another of examples that are, polynomial, in the error that I'm interested in, with the certainty that I'm interested in. And I gotta be able to come up with an algorithim, a learning algorithm that will do that. >> Exactly. So what do you think? >> I want to say the answer is yes. >> So how would you argue that though? Do you have a particular algarythm in mind? >> Well I actually did, I was going to keep the all the hypothesis that are consistent with the data that I've seen. >> Okay, alright and what is that called? >> The version space. >> Right. Okay. >> So keep track of that and then, whenever I stop getting samples, I have to pick one of those. So I'm just going to, pick one of them uniformly (意思是在 version space 中隨便返回一個 hypothesis) (甚麼情況下可以這樣做？見下段). >> Okay. Well that seems good so far. You could pick one uniformly. So what makes you think that that's actually going to. Be

able to return the right answer, or a, a close enough answer with not that much data. How do we actually bound the number of samples that we need to make it so that when we pick uniformly we actually get a good answer? >> Well, it's a little hard, I mean my, my intuition is that, given what we don't know what the concept is only the class that it comes from If I do anything other than choose uniformly, then I can be very unlucky. I basically... uniformly basically means in this case I don't know anything else to do. So there's sort of no better algorithm than the one that we just came up with, which is find all the hypotheses that are consistent... And you have no reason to choose one over the other because you don't have anymore data. So you should just close your eyes and pick one. And if you do anything else, then in the absence of some specific domain knowledge that tells you to do otherwise, you know, you can just basically end up being very unlucky. >> So, I, I agree with you and this is a good algorithm (即 pick up uniformly). But what we lack at the moment is an argument as to why the number of samples that it needs, isn't exponential. Right? Cause there's an exponentially large, you know, 2^k different possible inputs. If we had to see all of them to be able to guess right, that would be too many. >> Mm hm. >> So, we really want it to be, you know, a polynomial in K, not an exponential in K. So I'm going to say that we don't have enough background yet to be able to answer this definitively. The, yes is the right answer. But we're going to need to dive in a little bit more deeply, and develop some more concepts to be able to argue why that's the right answer.


ε-exhausted version space

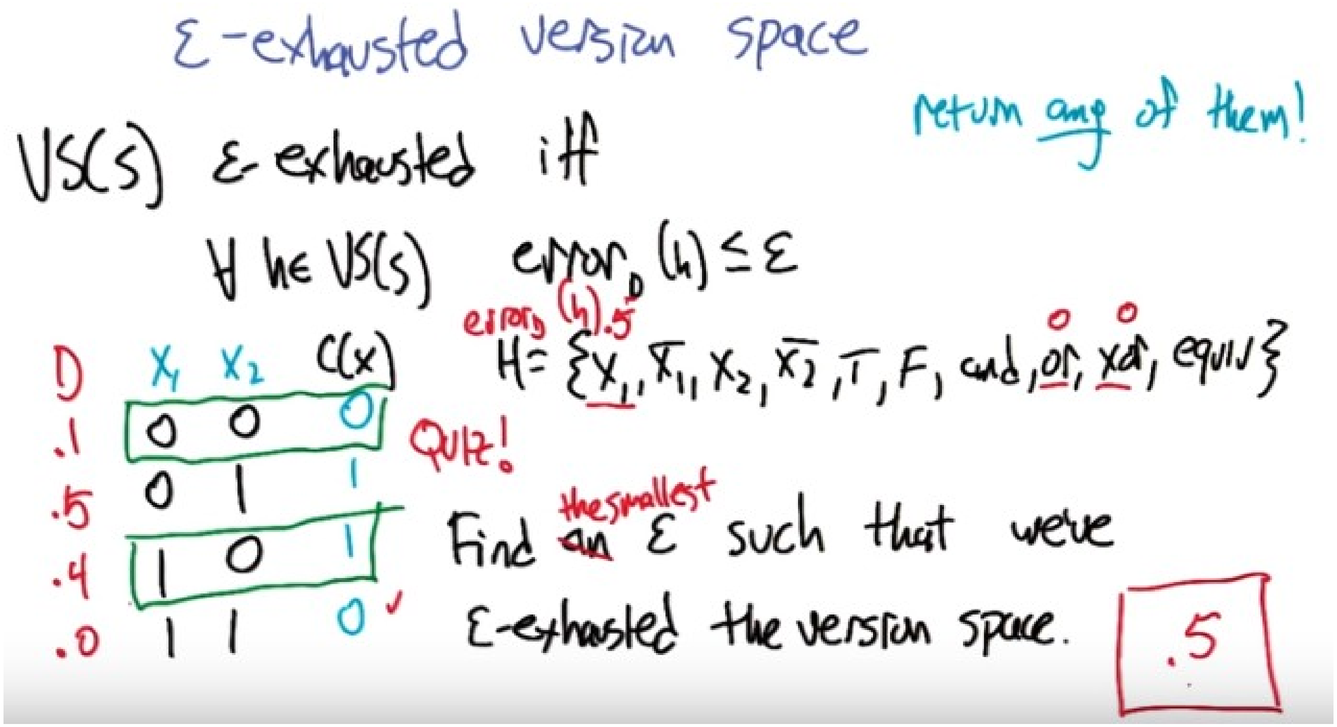
VS(s) ε-exhausted iff

$\forall h \in VS(s) \quad error_D(h) \le \varepsilon$

26. So this is going to be a key concept for being able to develop and answer two questions like that and it's the notion of epsilon exhaustion. Which sounds kind of tiring. >> I know I'm epsilon exhausted right now. >> Yea, me too. So what we mean by this is, well here's the definition. So, a version space. Of, version space that is derived from a particular sample, it's considered ε exhausted if and only if for all the hypotheses that are in that version space they have low error. So if we can do this then your algorithm (即上段中的 pick up uniformly) going to work. Your algorithm says at that point choose any of the hypotheses in your hypothesis set. You are going to be fine, you are going to have low error. >> Sure. >> If you don't do this, your algorithm is going to be in trouble because you are uniformly at random choose one of the hypothesis and it has a chance of being wrong. It could be a fairly high chance if there is, say there is only two hypothesis left in the version space, one has high error and one has low error. We really have to make sure that the only things left in the version space have low error >> Okay, that makes sense, that makes sense. >> Alright, so we're going to have to develop a little bit of theory to figure out when that occurs but this is a really key concept. >> Okay, so, I guess if was trying to put this into English just to make certain I understand it, what you're saying is, something is ε exhausted, a version space is ε exhausted exactly in the case when everything that you might possibly choose has an error less than ε. >> Sure. >> Period. And so if there's anything in there that has error greater than epsilon, then it's not epsilon exhausted. >> Right. It's still epsilon energized. >> Right. That makes a lot of sense, epsilon energized. That's a pretty good name for a band. OK. >> [LAUGH]

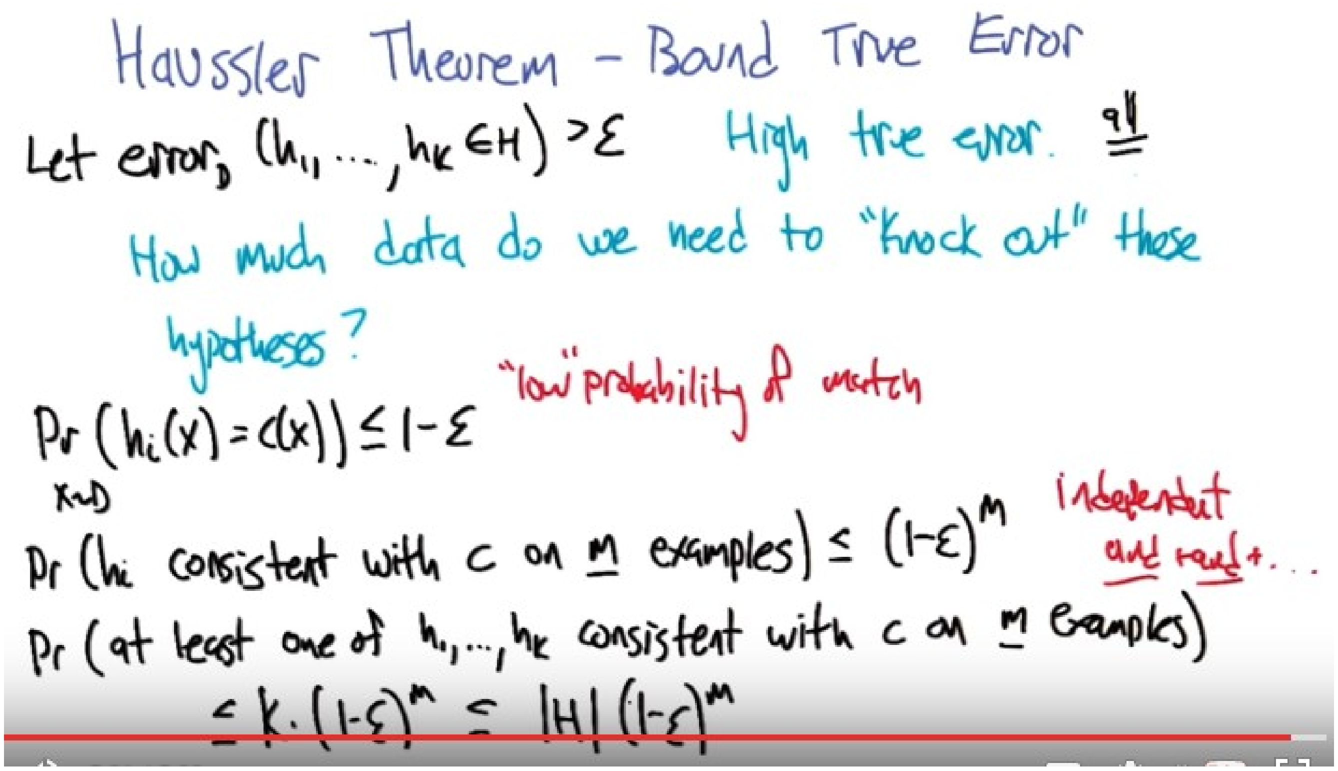
27. Alright, and just to make sure this ε exhaustion conpcet is clear, let's actually use it in a quiz. So, here's our example from before. We've got our target concept, which is xor, we've got our training data

which is these, with the things that are in the green boxes here, 0 0 output 0 1 0 output 1. And this time, I'll actually write down a distribtiuon D that gives the probability of drawing each of these different input examples. All right? So now what we'd like to do is find an epsilon such that that this training set that we've gotten has epsilon exhausted the version space. >> Okay. I got it. I think I got that. >> You think you, you think you can work that one through? >> Yeah. >> Anything else that we'd have to tell you? Remember again that These are the hypothesis in the hypothesis set. >> huh. Yeah. Yeah. I got it. >> Alright. Let's, let's, let's see what you got. >> Okay.


ε-exhausted version space
return any of them!
VS(S) ε-exhausted iff
∀ h∈VS(S) error_D(h) ≤ ε
error_D(h) .5
H = {x1, x̄1, x2, x̄2, T, F, and, or, xor, equiv}
0 0
D x1 x2 c(x)
.1 0 0 0
.5 0 1 1
.4 1 0 1
.0 1 1 0 ✓
Quiz!
Find ~~an~~ the smallest ε such that we've ε-exhausted the version space.
.5

28. Okay, so one. >> See, now that was mean. So right, one is an epsilon, no because ε has to be less than or equal to half (見 22 段的圖), we already said that, but you're right. In a sense setting epsilon to one, is always a valid answer to the question, find an epsilon set that we've epsilon exhausted the version space. Because all it's saying is, that there is nothing left in the set that has an error greater than one. And since the error is defined to be the probability, it can't be greater than one. So that was kind of you know, kind of rude. >> All right, I, I thought I just answered the question. But I guess you want me to give you [CROSSTALK] >> Yeah but you, but you prob-, what you probably should have pointed out is that I left out the word smallest. >> Oh! Yes, yes. Okay. Well, so, I don't know the answer to that but, I think I could walk through it very quickly. >> Okay. >> Okay, so you saying the ones that are in green (即框中的那兩行) are the training examples that we see (training example 是用來定出 version space 的(稍後會講. 深刻理解甚麼是 version space). 但是算每個 hypothesis 的 error 時, 那四行 input 都要算在內), right? >> Right. >> So we should be able to use that (即 training examples) to figure out what the version space actually is. >> Right, which we did in a previous question. >> Right, although I don't remember [LAUGH] what the answer was. >> I'll remind you. >> I'll remind you,It's okay. So it was x1 >> Mh-hm. >> Right, because the x1 matches, it was, or and xor. >> Mh-hm. >> I think that was it. >> Yeah, I think that's right. Okay, so, then what we can do is given that those are the three things that we've done. We could actually compute, what the error is according to this distribution for each of those three. >> Yes, exactly so. >> So let's, let's start with X1. So which one, x1, so all three of those are going to get the first one and the third one correct, right? >> All of them are going to get the first one and the third one correct. Yes, by design. >> By design. >> Right, to be in the version

space. >> So now we can ask which ones will get the second one wrong? The fourth one doesn't matter because it has zero probability of showing up. >> That's right. So, it doesn't matter if you get this one right or wrong, it's not going to contribute to this true error measure. >> Okay. So let's look at x one. So x1 will in fact get the second one (即第二行) wrong, because the output is not the same as the value for x1. >> Good. And so what;s the probablity that x one, this hypothesis x one, is going to give a wrong answer on a randomly drawn input? >> Well, half the time it will get the second answer, and so the error is, in fact, one half. >> Yes. Exaclty. Good. All right. Let's move on to the or. >> Okay. So, we can do an easy one actually. We can do x or. Since we know x or is the right answer, we know It will has a probablity of being wrong of zero. >> Oh, good point. >> Okay. And so for or we can do the same thing. So is, we know it's going to get the first and the third ones right. So now we can ask whether it's going to get the second one, right. And zero or one is in fact true. Or one. So in fact it also has an error of zero. >> Okay. >> Which is kind of interesting. So and so, even though the function is xor, if we can get to the point where we have or or xor left, we actually will get zero true error. >> That's right. >> But in the meantime, because x1 has still survived the two examples that we have. Epsilon is therefore 0.5. >> Right, in particular, we're saying that, this is, if, if epsilon were smaller than 0.5, then it wouldn't be epsilon exhausted because you'd have a hypothesis that has error that's too high. >> Right. >> So this is the smallest epsilon that we can use. And in fact, we let you through if if it was anything 0.5 to, to one, but this is the, this is the value that I was really hoping you'd be able to reason out. >> Okay, well that all made sense. >> Good, nice work. >> Thanks.



Haussler Theorem - Bound True Error

Let $\text{error}_D(h_1, \dots, h_k \in H) > \varepsilon$ High true error. $\underline{\underline{\text{all}}}$

How much data do we need to "knock out" these hypotheses?

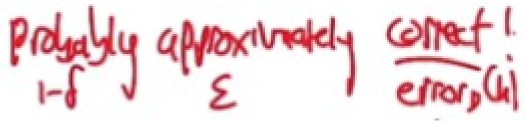
$\Pr_{x \sim D}(h_i(x) = c(x)) \le 1 - \varepsilon$ "low" probability of match

$\Pr(h_i$ consistent with c on $\underline{m}$ examples$) \le (1-\varepsilon)^m$ independent and rand+...

$\Pr($at least one of $h_1, \dots, h_k$ consistent with c on $\underline{m}$ examples$)$

$\le k \cdot (1-\varepsilon)^m \le |H|(1-\varepsilon)^m$

上圖中為何有 Pr(...) <= 1- ε, 尚不清楚(注意這是 probability). 以下來自前面的圖, 由它可知 ε 的意思:



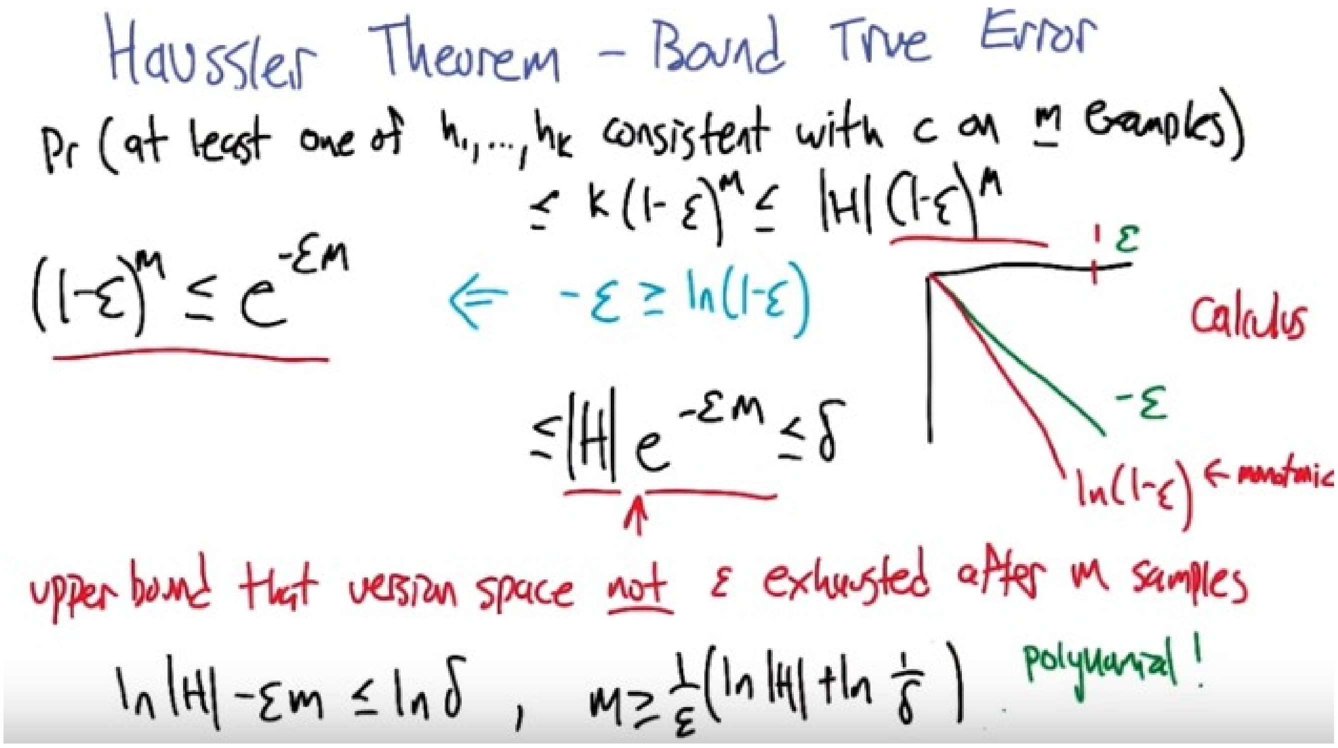
Probably approximately correct!

$1-\delta$ ε $\text{error}_D(h)$

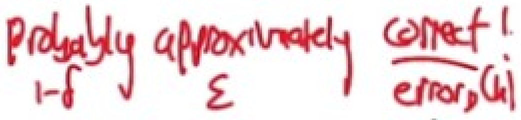
本段(有兩個圖)之目的在於證明了這樣一個結論(我的表述): If we want to PAC learn a hypothesis set which has size n, the numer of samples we need to is polynomial in 1/ε, 1/δ, & n, 即 m >= (1/ε) (ln|H| + ln(1/δ)), m 為 number of samples, n 即 |H|(即 size of the hypothesis space). 其中 PAC learn a hypothesis set 的意思是: 該此 hypothesis set 中的每一個 hypothesis 都在 1-δ 的概率下 范錯率在 ε 內.

29. >> Alright, so now we're ready to dive in and actually work out some math that turned out not to be so bad, but it was, it ends up being okay specifically because we were very carefully about setting up all the definitions to lead us to this moment in time. It is our destiny, Charles. >> Oh good. I love destiny. Is this destiny's theorem? >> No, actually turns out it's Haussler's Theorem. >> Is Haussler like German for destiny? It might be, but I don't think it is. So, Haussler is the name of a person in this particular case. And what he worked out is a way of bounding the true error as a function of the number of training examples that are drawn. >> Oh. Nice. >> So, let's consider. From the hypothesis set that we have, all the hypotheses can be categorized as to whether or not they have high true error or low true error. >> Sure. >> Right so, let's let h1 through hk be the ones that have a priority of high true error. What we'd like to do is make sure that as we're drawing data sets that we have knocked all these guys out. We've gotten enough examples that actually allow us to verify that they have high error. So they have high error on the, on the training set. So they have high training error. Alright, so how many, how much data do we need to establish that these guys are actually bad? Alright, so let's take a look at the probability that if we draw an X, an input from this distribution D. That, for any of these hypotheses. Hi, and this set of bad hypotheses. That it will match the true concept, right? So that H sub I of X is equal to C of X. And we know that that's less than or equal to one minus epsilon. It's unlikely that they match. Because it's likely, or relatively likely, that they mismatch. Right? That's what this exactly means. This, this error being greater than epsilon. >> Oh, I see, so if I have an error of greater than epsilon, that means that the probability that I'm wrong is greater than epsilon, which means the probability that I'm right is one minus epsilon, less than one minus epsilon. Okay, that makes sense. >> Yeah, so it's sort of a relatively low probability of match. >> Well, if epsilon is high. >> Low relative to one minus epsilon. >> Right, okay. >> So this is, this is a fact about the hypothesis in, in the abstract. For any given sample set we've got a set of m examples, and what we'd like to know is, since we're trying to knock it out, what's the probability that even after we've drawn m examples that this hypothesis, h of i, remains consistent with c. Right, even though the data doesn't really match all that well, we've drawn M examples, and it still looks like it matches. It's still in the version space. So the probability that that happens if it were the case that everything was independent is going to be one minus epsilon raised to the M power. Right. Because it's less than one minus epsilon to be wrong once Well, which is to say, that your consistent ones. To continue to be consistent, we keep having to have this probability come true. So, it's going to be, we're going to just keep multiplying it in again and again and again. So one minus epsilon raised the the m power. >> Right. So that makes sense because you're basically saying it's Consistent with the first example and the second example and the third example and dot dot dot nth example. So, and with independent variables is just multiplication so it's one minus epsilon times, one minus epsilon times, one minus epsilon n times. Okay, I see that. >> Great. Alright, so can we use that to figure out what's the probability that at least one of these h1 through hk's is consistent with c on m examples. We have to knock them all out. That's...that's really what the goal is. To knock out all the ones that have high true error We failed at that. If one of them still slips through, one of them still looks consistent and remains in the version space. So what's the probability that at least one of these remains consistent. That this still has happened. >> I think I know that. So just like you did add before and did multiplication... Another way of writing at least one of is to say or. So h1 or h2 or h3 or h4... or hk is consistent. And just like and is multiplication, or is addition. That's true. >> And there are k different ones of these, so I have to say this one minus epsilon to the m plus one minus epsilon to the m plus one minus epsilon to the mk times So that would be k*(1-ε)^m (為何不是 1-ε^m? 因為要都對). >> Great. So that is a bound on the probability that at least one of these bad hypothesis is going to remain in the version space even after m examples. But how many bad examples, how many

bad hypothesis are there? What's an upper bound on that? >> Well, there might be one or there might be two. There's actually k of them, but we know that k itself is bound by the total number of hypotheses. >> Yeah, that's right. So it has to be, you know, the number of bad hypotheses. We, we assume if c is equal to h anyway that there's at least one that is not bad, but, you know, almost h of them can be bad. So that, that should give us a bound. Alright, so now we're going to to work on this expression a little bit more to put it in a more convenient form. >> Okay.


Haussler Theorem - Bound True Error
Pr (at least one of h_1,...,h_k consistent with c on m examples)
≤ k(1-ε)^m ≤ |H|(1-ε)^m
(1-ε)^m ≤ e^{-εm}
⇐ -ε ≥ ln(1-ε)
≤ |H| e^{-εm} ≤ δ
upper bound that version space not ε exhausted after m samples
ln|H| - εm ≤ ln δ , m ≥ 1/ε (ln|H| + ln 1/δ)
polynomial!
ε
-ε
ln(1-ε) ← monotonic
Calculus

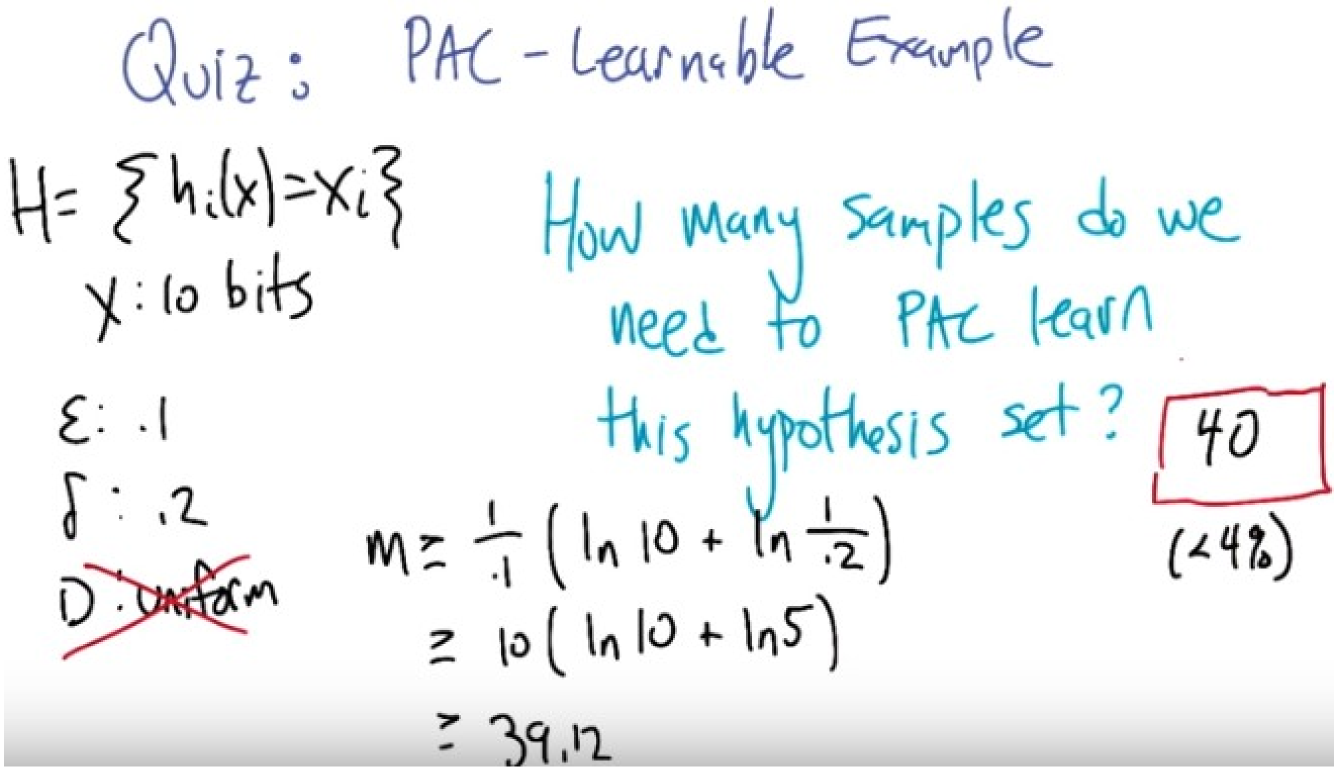
上圖中間為何是<=δ, 尚不清楚. 以下來自前面的圖, 由它可知 δ 的意思:



30. All right, so it turns out there is going to be an useful step here. Which is, we are going to take advantage of the fact that minus epsilon is greater than or equal to the natural law of one minus epsilon; which maybe it's not so obvious but if you plot it you could see that it's true. If this is the epsilon axis then minus epsilon looks like a straight line going down like that. >> Sure it's got slope minus one. >> Yep, and the log of one minus epsilon looks like this, it starts off, they'll they're totally lined up at zero,

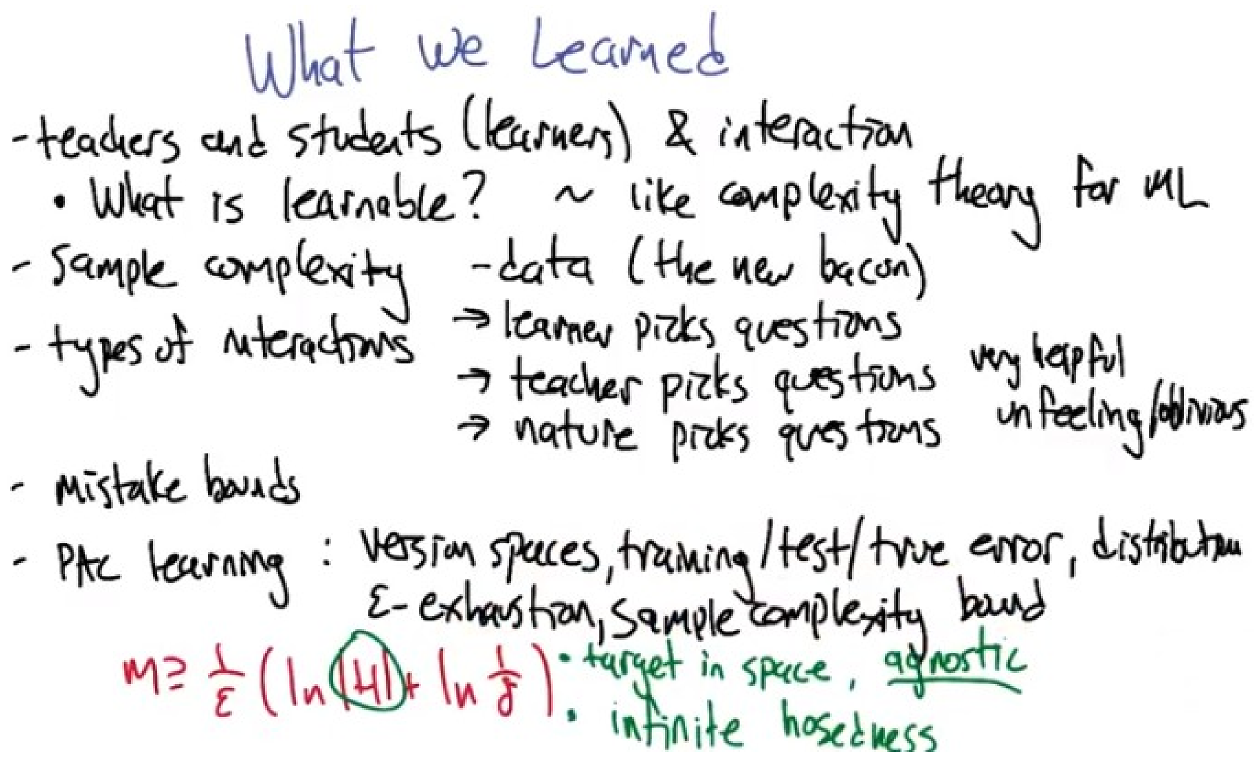
epsilon zero. >> Sure because one minus zero is one then absolute log of one is zero. >> Exactly, and then what happens is it starts to fall away from it, the slope is actually, I mean you could, you can test this by taking the derivative of it but the slope is if it's you know it's monotonically I'm changing [LAUGH] so that it falls away from the line and always stays below it,okay, so if we believe that, which, you know I'm going to just say calculus. >> Well, you can kind of see that, right? Because when epsilon is one, that would be the natural log of zero and the only way you can raise e to a power and get zero, is by having effectively, negative infinity. >> Yeah, so right, by then, it's definitely below and but, it stay below all along, I mean, because, just that is not enough, because. >> Well, it has to stay below all along because natural long is the natural log is a monotonic function. Alright, so it can't like, get bigger and then get smaller again, so, yeah, okay I buy that. >> Good, alright, so if that's the case,if we accept this line, then, it's also going to be the case, that one minus epsilon to the m, is than or equal e to the minus epsilon m, so why is that? So, if you multiply both sides by m, and then take each of the both sides, you get exactly this expression. >> Sure. Alright? So now that we've gotten that, we can use it here in our derivation and rewrite that as, the size of the hypothesis space times e to the minus epsilon m. Alright that gives us another upper bound on the quantities that we had before and this is much more convienent to work with. The epsilon which had been kind of trapped in the parentheses with the y minus now comes up to the exponent where we can work with it better. >> Sure. >> Alright, so what this is is an upper bound that the virgin space is not epsilon exhausted after m samples. >> Mm-hm. >> And that is what we would like delta to, we would like delta to be a bound on that. >> Mm-hm. >> Right, so if δ is the failure probability essentially. So the failure probability aught to be bigger than or equal to this expression here, alright? >> Okay. >> So now, the last thing we need to do, is we can just re-write this in terms of M. >> Mm. So if we do that, let's see what happens. All right, when we're done rewriting that what we find is the sample size m needs to be at least as large as one over epsilon times the quantity the log of the size of the hypothesis space plus the log of one over delta. >> Okay and that is polynomial in 1/ε, 1/δ, and the size of the hypothesis base. >> Indeed! >> Nice. >> So that's pretty cool. >> That is pretty cool. >> So, right, it tells us if you know the size of your hypothesis base, and you know what your ε and δ targets are, you know, sample a bunch, and then you'll be okay. >> That's pretty good!

31. This puts us in a good position to be able to go back to that question we looked at before, so lets look at it a little bit differently this time. If our hypothesis space is the set of functions that take 10 bit inputs and there is a hypothesis corrisponding to returning each of those 10 bits, seperate hypthesis, one returns the first bit, one returns the second bit. And so on, and now we have a target of Epsilon equals point 1, so we would like to, return a hypothesis whose error is less than or equal to point 1, and we want to be pretty sure of it, the error, or failure probability needs to be less than or equal to point 2, and lets just say for concreteness, that our distribution of our imput is uniform. >> Ok. >> So given this setup, how many samples do we need to pack learn this hypothesis set. >> Okay. >> And remember the algorithm that we're going to use is we're going to draw sample of the size that we want. Then we are going to be confident that we've epsilon exhausted that, that, version space. And so anything they left in the version space should have low error. And that procedure should fail with probability, no more than .2. >> Right. So it's just exceeds probability .8. >> Yeah. >> Or. Better. >> Okay, think we can work that through? >> I think we can. >> Alright let's do it. >> Okay. Cool.


Quiz: PAC-Learnable Example
$H = \{h_i(x) = x_i\}$
X: 10 bits
ε: .1
δ: .2
D: ~~uniform~~
How many samples do we need to PAC learn this hypothesis set?
$m \geq \frac{1}{.1}\left(\ln 10 + \ln \frac{1}{.2}\right)$
$\geq 10(\ln 10 + \ln 5)$
≥ 39.12
$\boxed{40}$
(<4%)

32. >> All right Charles, what do you think? >> I don't know but I know how to do it. >> All right. >> I'm just going to substitute any equation we had before. >> Yeah, that's what I was thinking, uh-huh. >> Alright, so M is greater than or equal to, 1 over epsilon times the natural log of the size of the hypothesis space. >> Mm, which is what, that is not one of our variables here. >> ten. >> Yeah. Right, so it's not 2 to the 10. Even though the input space is 2 to the 10. The number of hypotheses. There's one hypothesis corresponding to each of the bit positions. So, good? >> Right. Plus, the natural log of 1 over delta. So that would be greater than or equal to ten times the natural log of ten. >> [LAUGH] >> Plus the natural log of, five. So, let's see. The natural log of ten is something like three point something, the natural log of five is something like, two point something. We add those up, multiply by ten you're going to end up with 39.12. >> Good, so, we need, you know, 40 samples? >> Yeah. That sounds about right. >> That actually doesn't sound too bad. Well, you know, it's not learning a very hard problem, but it's, you know, a pretty big input space. So let's see. What, how big is the input space? It's like 2^10, which is. >> 1,024. So how much of 1024 is 40? It's, it's, you know, less than 4%. Hm, that's not bad. >> Before we leave this quiz, let me point out one more thing: that this bound is actually agnostic to the distribution from which samples came, so this idea that it's from a uniform distribution is actually not being directly used here. So so this is pretty cool. It actually doesn't

matter, we only need 40 samples no matter what the distribution is. It's not like some distributions are harder or easier ,because we are measuring the true error on the same distribution that we used to, to create the training set. So if it's a really hard distribution and some tough examples never appear, then we're unlikely to see them in the training set, but they're not going to contribute very much to the true error. >> Well that makes sense. So the distri, oh right. So in some sense, I mean, I guess the equation doesn't show this, but in some sense, the distribution is, cancels out between the training and the true error >> Yeah, that's one way to think about it. >> Well, I like that. So 40 is pretty good to get 10% error. If we wanted to get say, only 1% error, Then we would go from 40 to 400. >> Mm. >> Right? >> That's a good point, yeah. >> And it's, it's, it's one decimal point even. And so, that would be about 40% of the data. >> Yeah, that's true. Yeah, if we want to go a little bit beyond that we may need all the data multiple times. >> Mm-hm. >> Yeah, but this example doesn't look so bad. So let's just move on before we think about it too hard. >> Okay. That seems fair, I like that.



What we Learned

- teachers and students (learners) & interaction
  - What is learnable? ~ like complexity theory for ML
- Sample complexity -data (the new bacon)
- types of interactions → learner picks questions
  → teacher picks questions very helpful
  → nature picks questions unfeeling/oblivious
- mistake bounds
- PAC learning : version spaces, training/test/true error, distribution
  ε-exhaustion, sample complexity bound

$m \geq \frac{1}{\varepsilon}\left(\ln|H| + \ln\frac{1}{\delta}\right)$ • target in space, agnostic
• infinite hosedness

33. >> So actually that was all we were going to talk about in this lesson about computational learning theory. So let's just recap where we went so far. >> Okay. So what? You want me to do it? >> Yeah, that's been our technique all along. >> Fine. So here you're the teacher and I'm the student. I get that. Which is actually one of the things that we talked about. Aha. >> We talked about what it would mean to be a learnerversus being a teacher. And how teachers and learners interact to make learning happen faster or not. >> Okay. >> But that was actually in a larger context which I thought was kind of cool which was this sort of notion of trying to understand what is actually learnable. Right and I think the comparison that made sense to me was that we were trying to do the equivalent to what we do in computer science with complexity theory and algorithms. While here we were bouncing up from a specific algorithm like decision trees or. KNN and asking a question about how fundamentally hard is the problem of learning. >> Good. >> And you know, like that. And we focused on a particular measure of difficulty, which I guess drove everything else, so we talked about which was sampled before it was excepted. Okay, how many examples, how many samples do we need in order to learn some concept? >> Good yeah, that's a really powerful idea because it's a different resource than what's

normally studied in computer science, things like time and space. This is now how much data do we need? >> Right, which makes sense because we do machine learning and machine learning people, what we care about is data. >> I, I saw a t-shirt recently that says data is the new bacon. >> Mm. So you're saying data is delicious? >> Yeah, I think we like data a lot. >> I love data. Okay, so that ties us back into a discussion about teachers and students because what we talked about was how If the relationship between the teacher and the student was one way versus another way, we might get different answers about sample complexity. So in particular, we talked about what would happen in a world where the learner had to ask all the questions. >> And that's powerful because the learner knows what the learner doesn't know but the learner doesn't know what the learner needs to know. So that is somewhat powerful. But it may be useful for the teacher to be more involved. >> Right, so that's the other thing, where the teacher, gets to actually pick the questions. >> Great. >> And then the third sort of case was where. The teacher didn't really pick the questions or the teacher didn't have an intent to pick the questions, but the teacher was, in fact, nature, so like a fixed distribution. >> Yeah, good. >> Right. And some of those are, you know, easier to deal with than others, like the teacher, since the teacher knows the answer, can ask exactly the right set of questions and get you there very quickly versus, say, when the teacher is just nature. And, you know, you get it according to whatever distribution there happens to be. >> Sort of oblivious, maybe, is a better word. >> I think unfeeling. >> Nature just doesn't care about me. >> I think nature cares about you just as much as nature cares about everyone else. [LAUGH] >> Yeah, that's exactly what I was afraid of. >> Yeah. Okay so let's see, what else did we cover? So we talked about mistake bounds as a different way of measuring things. >> Hm. You know how many mistakes do you make as opposed to how many samples do you need. That was kind of neat. >> Yeah. >> I know there's some tie-in there. And then the bit that I like a lot is that we started talking about version spaces and PAC learn ability. And what really worked for me with that was this distinction between training error which we talked about a lot. Test error which is how we've been thinking about all of the assignments we've been doing, and true error. And true error in particular got connected back to, to this notion of nature. >> Right, the distribution d. >> Right. And then you introduce the notion of epsilon exhaustion of version spaces, and it gave us an actual sample complexity bound For the case of distributions in nature. >> And the sample complexity bound is pretty cool, because it depends polynomially on the size of the hypothesis space, and the target error bound and the, the failure probability. >> Hm. So actually that reminds me, I had two questions about this one. >> Uh-huh? >> So, the first question was, that equation, m greater than or equal to one over epsilon times the quantity, natural log size of hypothesis space plus natural log of one over delta, close quantity. [LAUGH] >> Assumed that our target concept was in our hypothesis space, didn't it? >> Yes, that's true. >> So, whatever happens if it isn't? >> Then we have a learning scenario that's referred to in the literature as agnostic, that the learner doesn't have to have a hypothesis that is in the target space. And, instead, needs to find the one that fits nearly the best of all the ones in there. So it doesn't have to actually match the true concept. It has to, it has to get close to the best in its own collection. >> Okay, well, so, do we get the bounds? >> It's very similar bound. I think, I think maybe there's an extra epsilon, there's an extra squared on the epsilon. >> Hm. Okay, okay. >> And I think there's maybe slightly different constants in here. So it's, it's a very similar form. It's still polynomial. It is worse though because it, the learner has kind of less strength to depend on. >> Okay, that's fair. Okay, so then my second question was, I just realize staring at this now since you wrote it in red, that the bound depends upon the size of the hypothesis space. >> Indeed. >> So what happens if we have an infinite hypothesis space? >> Well, according to this bound, The technical term is your hosed. >> Oh, is that what the h stand for? >> Yes. >> Hm, so n would be greater than one over epsilon times the natural log of infinite which I'm pretty sure is infinite. >> Yeah, even with the, even once you multiply it by one over epsilon. So yeah, you know, this is a really important issue, and I think it really deserves its own lessons. So let's, let's put this off to lesson eight. You're right that the infinite hypothesis spaces come up all the time. They're really important. They almost everything we've talked about so far in the class, like actually learning algorithms, deal with infinite hypothesis bases, we would really like our bounds to deal with them as well. >> Yeah, I would like that. >> So, I know, anything else to talk here, or should we say, without further ado, let's move on to the next lesson? >> I think we should say, without further ado, let's move on to the next lesson. >> Alright then, without further ado, let's move on to the next lesson. >> Okay, well then I will see you next time, Michael. Sure Dan, thanks, thanks for listening >> Oh well, you know, I, I enjoy doing it so much. >> [LAUGH] >> Bye.